รายงานโครงงาน กลุ่มการเรียนที่ 1 กลุ่มที่ 1

ประเภทธุรกิจ : ขายอุปกรณ์ช่าง

จัดทำโดย

นายธนวิชญ์	จึงกิจธนวัฒน์	รหัสนักศึกษา 6587017
นายศรุต	เฟื่องวุฒิ	รหัสนักศึกษา 6587035
นายจิตรกัณฐ์	ดำรงตระกูลวัฒน์	รหัสนักศึกษา 6587051
นายเจ้าทรัพย์	พงศ์ทวีไชย	รหัสนักศึกษา 6587068
นายสันติภาพ	โชติมานนท์	รหัสนักศึกษา 6587105

เสนอ

ดร. จิดาภา ไกรสังข์

ดร. วุฒิชาติ แสวงผล

รายงานเป็นส่วนหนึ่งของรายวิชา ทสวด 241

เทคโนโลยีด้านเว็บและการประยุกต์ใช้

ภาคเรียนที่ 1 ปีการศึกษา 2566

คณะเทคโนโลยีสารสนเทศและการสื่อสาร มหาวิทยาลัยมหิดล

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา เทคโนโลยีด้านเว็บและการประยุกต์ใช้ โดยมีวัตถุประสงค์เพื่ออธิบายรายละเอียดของโครงงานเพื่อการเรียนรู้แบบจำลองและการออกแบบเว็บแอปพลิเคชัน ประกอบด้วยข้อมูลองค์รวมของโครงงาน แผนผังหน้าต่าง ๆ ของเว็บแอปพลิเคชัน รายละเอียดหน้าเว็บต่าง ๆ ของผู้ดูแลระบบ (Admin) รายละเอียดของเว็บเซอร์วิสและโค้ด ตลอดจนผลการทดสอบของเว็บเซอร์วิสทั้งหมด โดยใช้โปรแกรม Postman

สำหรับโครงงานเพื่อการเรียนรู้แบบจำลองและการออกแบบเว็บแอปพลิเคชันได้รับจากชั้นเรียนรายวิชา ทสวด 241 เทคโนโลยีด้านเว็บและการประยุกต์ใช้ โดยใช้ประเภทธุรกิจ ขายอุปกรณ์ช่าง

นักศึกษาคณะเทคโนโลยีสารสนเทศและการสื่อสาร
สาขาวิทยาการและเทคโนโลยีดิจิทัล
รหัสนักศึกษา 65 กลุ่มการเรียนที่ 1 กลุ่มที่ 1

สารบัญ

สารบัญภาพ

1. ข้อมูลองค์รวมของโครงงาน

Web Application นี้เป็นการบริการขายสินค้าเกี่ยวกับอุปกรณ์ช่าง ซึ่งสามารถทำการแก้ไขรายการสินค้าได้โดย ผู้ดูแลระบบ (Admin)

- **ผู้ดูแลระบบ (Admin)** สามารถทำการแก้ไขรายการสินค้าได้ในทุก ๆ ขั้นตอน ได้แก่
  สร้าง (Create) รายการสินค้าใหม่
  ลบ (Delete) รายการสินค้าที่มีอยู่
  แก้ไข (Edit) ข้อมูลของรายการสินค้าที่มีอยู่
  ค้นหา (Search) รายการสินค้าตามเงื่อนไขที่ต้องการ
- **ผู้ใช้งาน (User)** สามารถทำการค้นหาข้อมูลของสินค้าได้เท่านั้น โดยสามารถค้นหาได้ตามเงื่อนไขที่ต้องการ เช่น ชื่อสินค้า หมวดหมู่สินค้า ราคาสินค้า เป็นต้น

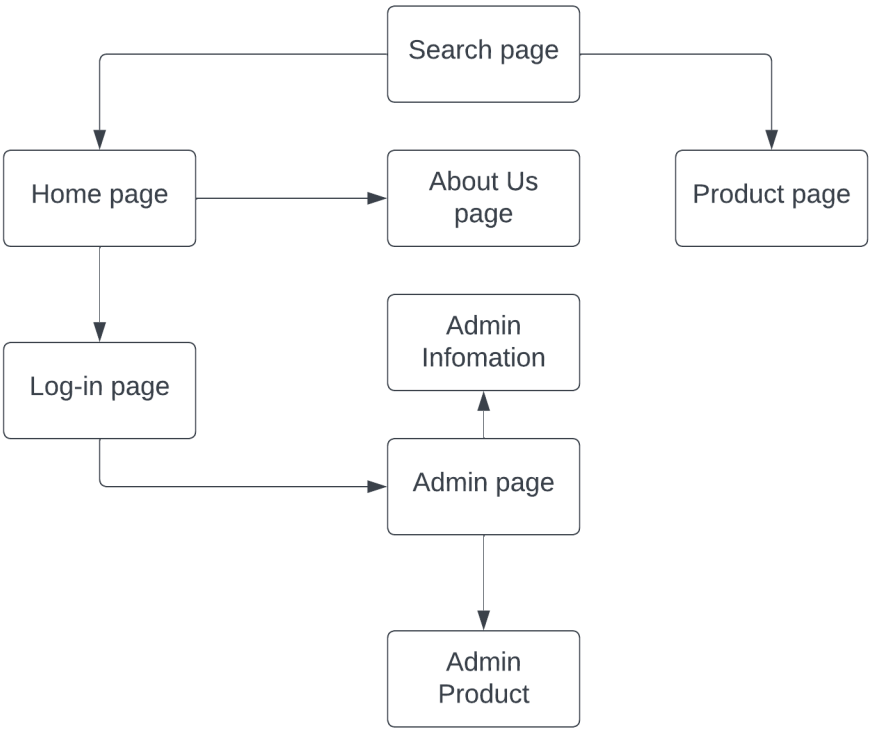
การแก้ไขรายการสินค้าของผู้ดูแลระบบ (Admin) นั้น สามารถทำได้อย่างละเอียดในทุก ๆ ขั้นตอน ตั้งแต่การสร้างรายการสินค้าใหม่ ไปจนถึงการแก้ไขข้อมูลของรายการสินค้าที่มีอยู่ เช่น ชื่อสินค้า รายละเอียดสินค้า รูปภาพสินค้า ราคาสินค้า สต็อกสินค้า เป็นต้น ข้อมูลของรายการสินค้าเหล่านี้จะถูกเก็บไว้ในระบบฐานข้อมูล ซึ่งผู้ดูแลระบบ (Admin) สามารถเข้าถึงและแก้ไขได้ผ่านระบบหลังบ้านของ Web Application

การค้นหาข้อมูลของสินค้าของผู้ใช้งาน (User) นั้น สามารถทำได้ตามเงื่อนไขที่ต้องการ เช่น ชื่อสินค้า หมวดหมู่สินค้า ราคาสินค้า เป็นต้น ข้อมูลของสินค้าที่ถูกค้นหาจะถูกแสดงผลให้ผู้ใช้งาน (User) ทราบ ข้อมูลเหล่านี้จะถูกดึงข้อมูลจากฐานข้อมูลของ Web Application

ตัวอย่างการใช้งาน Web Application นี้ เช่น

- ร้านค้าอุปกรณ์ช่างที่ต้องการขยายช่องทางการขายสินค้าไปยังออนไลน์
- โรงงานอุตสาหกรรมที่ต้องการจัดจำหน่ายอุปกรณ์ช่างให้กับลูกค้า
- ช่างที่ต้องการสั่งซื้ออุปกรณ์ช่างออนไลน์

2. แผนผังหน้าต่าง ๆ ของเว็บแอปพลิเคชัน



รูปภาพที่ 2.1 แผนผังการเชื่อมต่อของเว็บแอปพลิเคชัน

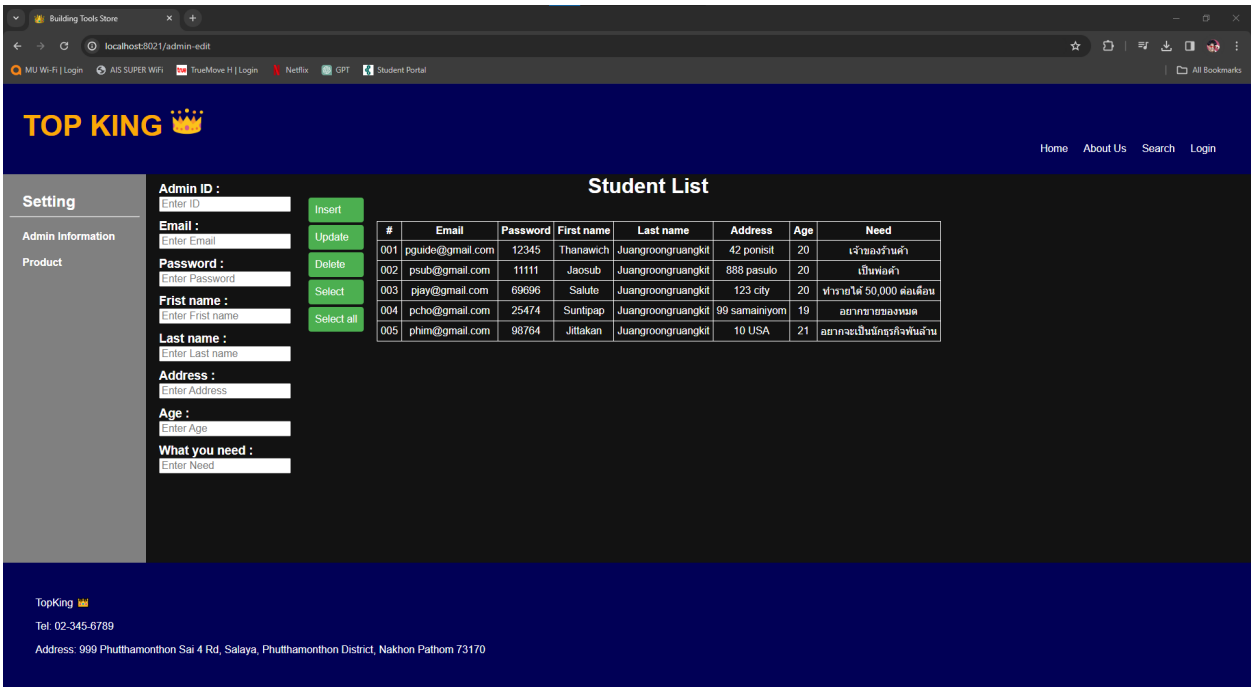
3. รายละเอียดหน้าเว็บต่าง ๆ ของผู้ดูแลระบบ (Admin)

ผู้ดูแลระบบสามารถแก้ไขข้อมูลในระบบฐานข้อมูลได้ 2 ส่วน ได้แก่ ข้อมูลผู้ดูแลระบบ และ ข้อมูลสินค้า

ข้อมูลผู้ดูแลระบบ

- Admin ID: รหัสประจำตัวผู้ดูแลระบบ
- Email: ที่อยู่อีเมลของผู้ดูแลระบบ
- Password: รหัสผ่านของผู้ดูแลระบบ
- First name: ชื่อจริงของผู้ดูแลระบบ
- Last name: นามสกุลของผู้ดูแลระบบ
- Address: ที่อยู่ของผู้ดูแลระบบ
- Age: อายุของผู้ดูแลระบบ
- What you need: สิ่งที่ต้องการจากผู้ดูแลระบบ

ข้อมูลเหล่านี้จะถูกเก็บไว้ในฐานข้อมูลของ Web Application ซึ่งผู้ดูแลระบบสามารถเข้าถึงและแก้ไขได้ผ่านหน้าข้อมูลผู้ดูแลระบบ

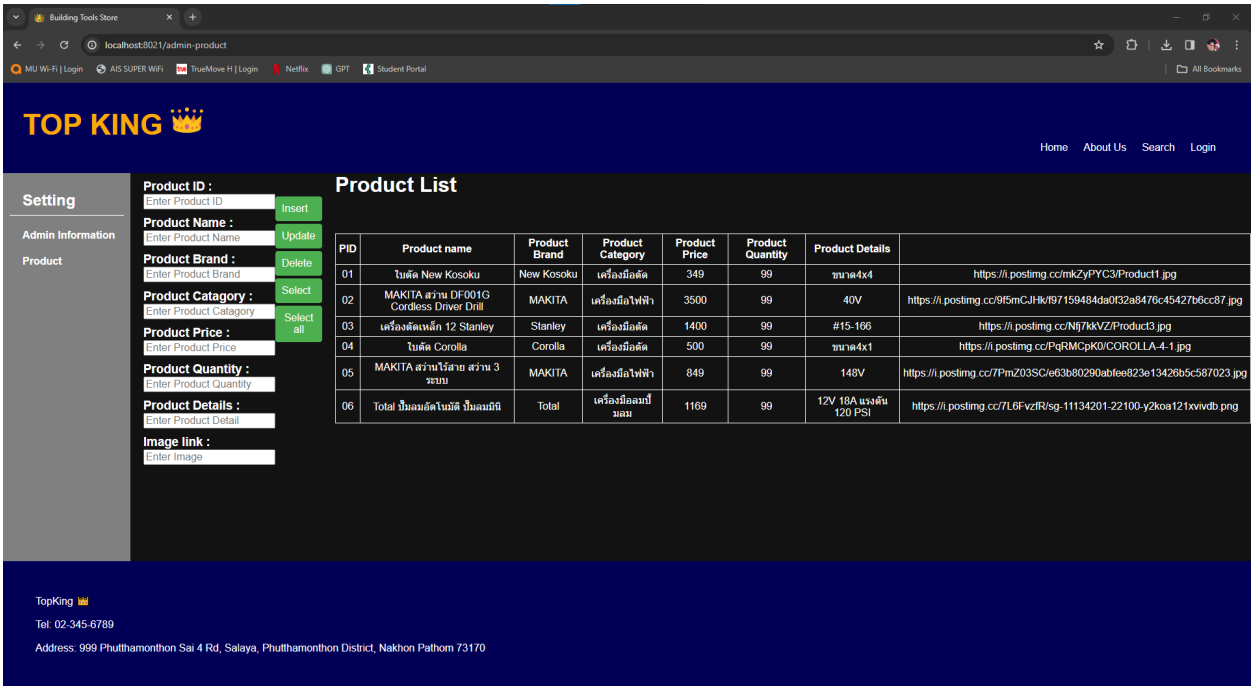


รูปภาพที่ 3.1 เว็บไซต์ของข้อมูลผู้ดูแลระบบ

ข้อมูลสินค้า

- Product ID: รหัสประจำสินค้า
- Product Name: ชื่อสินค้า
- Product Brand: ยี่ห้อสินค้า
- Product Category: หมวดหมู่สินค้า
- Product Price: ราคาสินค้า
- Product Quality: คุณภาพสินค้า
- Product Details: รายละเอียดสินค้า

ข้อมูลเหล่านี้จะถูกเก็บไว้ในฐานข้อมูลของ Web Application ซึ่งผู้ดูแลระบบสามารถเข้าถึงและแก้ไขได้ผ่านหน้าข้อมูลสินค้า



รูปภาพที่ 3.2 เว็บไซต์ของข้อมูลสินค้า

โดยมีจะปุ่มสำหรับคำสั่ง

- ปุ่ม Insert ใช้ในการเพิ่มข้อมูลใหม่ลงในฐานข้อมูล
- ปุ่ม Update ใช้ในการแก้ไขข้อมูลในฐานข้อมูล
- ปุ่ม Delete ใช้ในการลบข้อมูลในฐานข้อมูล
- ปุ่ม Select ใช้ในการเลือกข้อมูลในฐานข้อมูล โดยสามารถเลือกตามเงื่อนไขที่ต้องการ เช่น รหัสประจำตัวผู้ดูแลระบบ ที่อยู่อีเมล ชื่อจริง เป็นต้น
- ปุ่ม Select all ใช้ในการเลือกข้อมูลทั้งหมดในฐานข้อมูล

ตัวอย่างการใช้งานหน้าของ ผู้ดูแลระบบ (Admin)

- ผู้ดูแลระบบต้องการเพิ่มข้อมูลผู้ดูแลระบบใหม่ ผู้ดูแลระบบจะต้องป้อนข้อมูลลงในช่องข้อความ Admin ID, Email, Password, First name, Last name, Address, Age, และ What you need จากนั้นกดปุ่ม Insert
- ผู้ดูแลระบบต้องการแก้ไขข้อมูลผู้ดูแลระบบที่มีอยู่ ผู้ดูแลระบบจะต้องเลือกข้อมูลผู้ดูแลระบบที่ต้องการแก้ไข จากนั้นป้อนข้อมูลที่ต้องการแก้ไขลงในช่องข้อความ Email, Password, First name, Last name, Address, Age, และ What you need จากนั้นกดปุ่ม Update
- ผู้ดูแลระบบต้องการลบข้อมูลผู้ดูแลระบบที่มีอยู่ ผู้ดูแลระบบจะต้องเลือกข้อมูลผู้ดูแลระบบที่ต้องการลบ จากนั้นกดปุ่ม Delete
- ผู้ดูแลระบบต้องการค้นหาข้อมูลผู้ดูแลระบบ ผู้ดูแลระบบจะต้องป้อนข้อมูลที่ต้องการค้นหาลงในช่องข้อความ รหัสประจำตัวผู้ดูแลระบบ, ที่อยู่อีเมล, หรือ ชื่อจริง จากนั้นกดปุ่ม Select
- ผู้ดูแลระบบต้องการเลือกข้อมูลผู้ดูแลระบบทั้งหมด ผู้ดูแลระบบจะต้องกดปุ่ม Select all

4. รายละเอียดของเว็บเซอร์วิสและโค้ด

4.1 รายการไฟล์ที่ใช้ในโครงงาน

```
sec1_gr1_pj2_ID017_035_051_065_105 (Root)
        > sec1_gr1_src
                > CallWs
                        <> CallAdminWs.js
                        <> CallLoginWS.js
                        <> CallProductWs.js
                        <> CallSearchWs.js
                > HTML
                        > CSS
                                <> Add-Product.css
                                <> AdminInform.css
                                <> admin-product.css
                                <> all.css
                                <> default.css
                                <> delete-page.css
                                <> edit-page.css
                                <> edit-product.css
                                <> him-styles.css
                                <> Home.css
                                <> Login.css
                                <> Search_page.css
                                <> styles.css
                                <> Useredit.css
                        <> About_us.html
                        <> admin-product.html
                        <> Home.html
                        <> Index.html
                        <> Login.html
                        <> Product1.html
                        <> Product2.html
                        <> Product3.html
                        <> Product4.html
                        <> Product5.html
                        <> Product6.html
                        <> Search_page.html
                        <> User(edit).html
                        <> user-solid.svg
```

```
                > picture
                <> ClientServer.js
                <> ecosys.config.js
                <> index.js
                <> package.js
                <> package-lock.js
        > sec1_gr1_ws__src
                > backend_WS
                        <> adminWS.js
                        <> callingadminWS.js
                        <> callingloginWS.js
                        <> callingproductWS.js
                        <> callingsearchWS.js
                        <> loginWS.js
                        <> productWS.js
                        <> searchWS.js
                <> .env
                <> app1
                <> package
                <> package-lock
                <> sec1_gr1_database
        <> README.TXT
```

4.2 รายละเอียดของเว็บเซอร์วิส

ในโครงงานนี้จะมีการสร้างและใช้งานเว็บเซอร์วิส ทั้งหมด 4 ตัว ดั้งต่อไปนี้

4.2.1 adminWS.js

โค้ดนี้เป็นการเขียนแอปพลิเคชัน Node.js โดยใช้ Express framework เพื่อทำการจัดการข้อมูล

ผู้ดูแลระบบ (admin) โดยแบ่งการทำงานออกเป็น 6 ฟังก์ชันหลักๆ ดังนี้

1. ฟังก์ชัน get("/") แสดงข้อความว่าอยู่ในหน้าการทำงาน admin

```
router.get("/", function (req, res) {
  return res.send({ message: "you are in admin page" });
});
```

2. ฟังก์ชัน get("/admins") แสดงผลข้อมูลผู้ดูแลระบบทั้งหมด

```
router.get("/admins", function (req, res) {
  connection.query("SELECT * FROM ADMINS", function (error, results) {
    if (error) throw error;
    return res.send({ error: false, data: results, message: "Admin lists" });
  });
});
```

3. ฟังก์ชัน get("/admin") แสดงผลข้อมูลผู้ดูแลระบบตามเงื่อนไขที่กำหนด

```
router.get("/admin", function (req, res) {
  let aid = req.query.AID;
  let aemail = req.query.EMAIL;
  let afname = req.query.FNAME;

  let sql = `SELECT * FROM ADMINS
            WHERE AID LIKE "%${aid}%" AND
            EMAIL LIKE "%${aemail}%" AND
            FNAME LIKE "%${afname}%";`
```

```
  connection.query(sql, function (error, results) {
    if (error || results.length === 0)
      return res.send({
        error: true,
        message: "Admin is not found.",
      });
    return res.send({
      error: false,
      data: results,
      message: "Admin retrieved",
    });
  });
});
```

4. ฟังก์ชัน post("/admin") เพิ่มข้อมูลผู้ดูแลระบบใหม่

```
router.post("/admin", function (req, res) {
  let admin = req.body;
  connection.query(
    "INSERT INTO ADMINS SET ? ", admin, function (error, results) {
      if (error) throw error;
      return res.send({
        error: false,
        data: results.affectedRows,
        message: "New admin has been added.",
      });
    }
  );
});
```

5. ฟังก์ชัน put("/admin") แก้ไขข้อมูลผู้ดูแลระบบ

```
router.put("/admin", function (req, res) {
  let admin_id = req.body.AID;
  let admin = req.body;
  connection.query("UPDATE ADMINS SET ? WHERE AID = ?", [admin,
  admin_id], function (error, results) {
    if (error) throw error;
    return res.send({
      error: false,
      data: results.affectedRows,
      message: "Admin has been updated.",
    });
  });
});
```

6. ฟังก์ชัน delete("/admin") ลบข้อมูลผู้ดูแลระบบ

```
router.delete("/admin", function (req, res) {
  let admin_id = req.body.AID;
  connection.query("DELETE FROM ADMINS WHERE AID = ?", [admin_id],
  function (error, results) {
    if (error) throw error;
    return res.send({
      error: false,
      data: results.affectedRows,
      message: "Admin has been deleted.",
    });
  });
});
```

การทำงานโดยรวมของโค้ดนี้ มีดังนี้

1. ทำการนำเข้าไลบรารีที่จำเป็น ได้แก่ express, mysql, dotenv, cors

```
const express = require('express');
const mysql = require('mysql2');
const app = express();
const router = express.Router();
const dotenv = require('dotenv');
dotenv.config();
const cors = require('cors');
```

2. สร้างตัวเชื่อมต่อฐานข้อมูล

```
let connection = mysql.createConnection({
  host: process.env.MYSQL_HOST,
  user: process.env.MYSQL_USERNAME,
  password: process.env.MYSQL_PASSWORD,
  database: process.env.MYSQL_DATABASE
});
```

3. กำหนดค่าให้แอปพลิเคชันรองรับ CORS

```
let whiteList = ["http://localhost:8021", "http://localhost:8022"];
let corsOptions = {
  origin: whiteList,
  methods: "GET,POST,PUT,DELETE",
};
app.use(cors(corsOptions));
router.use(cors(corsOptions));
```

4. กำหนดให้แอปพลิเคชันรองรับการส่งและรับข้อมูล JSON และ URLencoded

```
app.use(router);
router.use(express.json());
router.use(express.urlencoded({ extended: true }));
```

5. Export router เพื่อนำไปใช้ในส่วนอื่นของแอปพลิเคชัน

```
module.exports = router;
```

4.2.2 loginWS.js

โค้ดนี้เป็นการเขียน Node.js ที่ให้บริการ RESTful API สำหรับการตรวจสอบการเข้าสู่ระบบ (login) โดยมีการตั้งค่าต่าง ๆ

การทำงานโดยรวมของโค้ดนี้ มีดังนี้

1. ทำการนำเข้าไลบรารีที่จำเป็น ได้แก่ express, mysql, dotenv, cors

```
const express = require('express');
const mysql = require('mysql2');
const app = express();
const dotenv = require('dotenv');
dotenv.config();
const router = express.Router();
app.use(router);
const cors = require('cors');
```

2. กำหนดค่าให้แอปพลิเคชันรองรับ CORS

```
let whiteList = ["http://localhost:8021", "http://localhost:8022"];
let corsOptions = {
  origin: whiteList,
  methods: "GET,POST,PUT,DELETE",
};
app.use(cors(corsOptions));
router.use(cors(corsOptions));
```

3. กำหนดให้แอปพลิเคชันรองรับการส่งและรับข้อมูล JSON และ URLencoded

```
router.use(express.json());
router.use(express.urlencoded({ extended: true }));
```

4. สร้างตัวเชื่อมต่อฐานข้อมูล

```
let connection = mysql.createConnection({
    host : process.env.MYSQL_HOST,
    user : process.env.MYSQL_USERNAME,
    password : process.env.MYSQL_PASSWORD,
    database : process.env.MYSQL_DATABASE
});
```

6. ฟังก์ชัน Login

```
router.get("/login", function (req, res) {
  let email = req.query.USERNAME;
  let pwd = req.query.PWD;

  let sql = `SELECT * FROM ADMINS
            WHERE EMAIL LIKE "${email}" AND
            PWD LIKE "${pwd}";`

  connection.query(sql, function (error, results) {
    if (error || results.length === 0)
      return res.send({
        error: true,
        message: "Failed to login.",
      });
    return res.send({
      error: false,
      data: results,
      message: "You are login.",
    });
  });
```

```
});
```

7. Export router เพื่อนำไปใช้ในส่วนอื่นของแอปพลิเคชัน

```
module.exports = router;
```

4.2.3 productWS.js

โค้ดนี้เป็นการเขียน Node.js ที่ให้บริการ RESTful API สำหรับการจัดการข้อมูลสินค้า (product) ด้วยการให้รองรับการดำเนินการ CRUD (Create, Read, Update, Delete) สินค้า (product) โดยแบ่งการทำงานออกเป็น 6 ฟังก์ชันหลักๆ ดังนี้

1. ฟังก์ชัน get("/") แสดงข้อความว่า "you are in product page"

```
router.get("/", function (req, res) {
    return res.send({ message: "you are in product page" });
});
```

2. ฟังก์ชัน get("/products") แสดงผลข้อมูลสินค้าทั้งหมด

```
router.get("/products", function (req, res) {
  connection.query("SELECT * FROM product", function (error, results) {
    if (error) throw error;
    return res.send({ error: false, data: results, message: "Product lists" });
  });
});
```

3. ฟังก์ชัน get("/product") แสดงผลข้อมูลสินค้าตามเงื่อนไขที่กำหนด

```
router.get("/product", function (req, res) {
  let id = req.query.PID;
  let brand = req.query.PBRAND;
  let cat = req.query.PCAT;

  let sql = `SELECT * FROM product
            WHERE PID LIKE "%${id}%" AND
            PBRAND LIKE "%${brand}%" AND
```

```
            PCAT LIKE "%${cat}%";`


  connection.query(sql, function (error, results) {
    if (error || results.length === 0)
      return res.send({
        error: true,
        message: "Product is not found.",
      });
    return res.send({
      error: false,
      data: results,
      message: "Product retrieved",
    });
  });
});
```

4. ฟังก์ชัน post("/product") เพื่อเพิ่มข้อมูลสินค้าใหม่

```
router.post("/product", function (req, res) {
    let product = req.body;
    connection.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 added.",
        });
      }
    );
});
```

5. ฟังก์ชัน put("/product") เพื่ออัปเดตข้อมูลสินค้า

```
router.put("/product",function (req, res) {
    let product_id = req.body.PID;
    let product = req.body;

    connection.query("UPDATE product SET ? WHERE PID = ?",[product,
    product_id],function (error, results) {
        if (error)
          throw(error)
        return res.send({
          error: false,
          data: results.affectedRows,
          message: "Product has been updated.",
        });
      }
    );
});
```

6. ฟังก์ชัน delete("/product") เพื่อลบข้อมูลสินค้า

```
router.delete("/product",function (req, res) {
  let product_id = req.body.PID;

  connection.query("DELETE FROM product WHERE PID =
  ?",[product_id],function (error, results) {
      if (error)
        throw(error)
      return res.send({
        error: false,
        data: results.affectedRows,
        message: "Product has been deleted.",
```

```
      });
    }
  );
});
```

การทำงานโดยรวมของโค้ดนี้ มีดังนี้

1. ทำการนำเข้าไลบรารีที่จำเป็น ได้แก่ express, mysql, dotenv, cors

```
const express = require('express');
const mysql = require('mysql2');
const app = express();
const dotenv = require('dotenv');
dotenv.config();
const router = express.Router();
app.use(router);
const cors = require('cors');
```

2. กำหนดค่าให้แอปพลิเคชันรองรับ CORS

```
let whiteList = ["http://localhost:8021", "http://localhost:8022"];
let corsOptions = {
  origin: whiteList,
  methods: "GET,POST,PUT,DELETE",
};
app.use(cors(corsOptions));
router.use(cors(corsOptions));
```

3. กำหนดให้แอปพลิเคชันรองรับการส่งและรับข้อมูล JSON และ URLencoded

```
router.use(express.json());
router.use(express.urlencoded({ extended: true }));
```

4. สร้างตัวเชื่อมต่อฐานข้อมูล

```
let connection = mysql.createConnection({
```

```
    host : process.env.MYSQL_HOST,
    user : process.env.MYSQL_USERNAME,
    password : process.env.MYSQL_PASSWORD,
    database : process.env.MYSQL_DATABASE
});
```

4.2.4 searchWS.js

โค้ดนี้เป็นการเขียน Node.js ที่ให้บริการ RESTful API สำหรับการค้นหาข้อมูลสินค้า (products) โดยมีการตั้งค่าต่าง ๆ ดังนี้

สินค้า (product) โดยแบ่งการทำงานออกเป็น 2 ฟังก์ชันหลักๆ ดังนี้

1. ฟังก์ชัน get("/") แสดงข้อความว่า "you are in product page"

```
router.get("/", function (req, res) {
    return res.send({ message: "you are in product page" });
});
```

2. ฟังก์ชัน get("/products") แสดงผลข้อมูลสินค้าทั้งหมด

```
router.get("/search", function (req, res) {
  let brand = req.query.PBRAND;
  let cat = req.query.PCAT;
  let pname = req.query.PNAME;

  let sql = `SELECT * FROM product
            WHERE PNAME LIKE "%${pname}%" AND
            PCAT LIKE "%${cat}%" AND
            PBRAND LIKE "%${brand}%";`

  connection.query(sql, function (error, results) {
      if (error || results.length === 0)
        return res.send({
          error: true,
```

```
          message: "Product is not found.",
        });
      return res.send({
        error: false,
        data: results,
        message: "Product retrieved",
      });
  });
});
```

การทำงานโดยรวมของโค้ดนี้ มีดังนี้

1. ทำการนำเข้าไลบรารีที่จำเป็น ได้แก่ express, mysql, dotenv, cors

```
const express = require('express');
const mysql = require('mysql2');
const app = express();
const dotenv = require('dotenv');
dotenv.config();
const router = express.Router();
app.use(router);
const cors = require('cors');
```

2. กำหนดค่าให้แอปพลิเคชันรองรับ CORS

```
let whiteList = ["http://localhost:8021", "http://localhost:8022"];
let corsOptions = {
  origin: whiteList,
  methods: "GET",
};
app.use(cors(corsOptions));
router.use(cors(corsOptions));
```

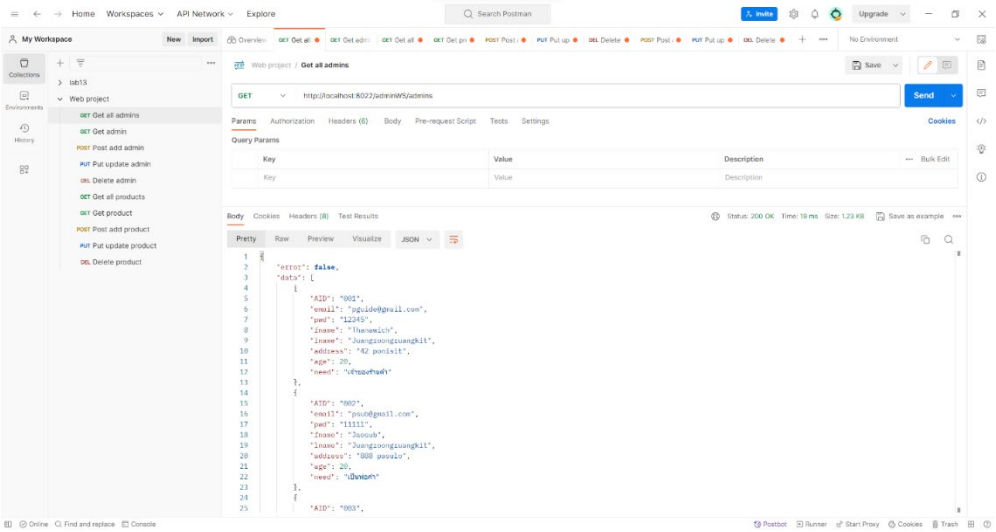
3. กำหนดให้แอปพลิเคชันรองรับการส่งและรับข้อมูล JSON และ URLencoded

```
router.use(express.json());
router.use(express.urlencoded({ extended: true }));
```

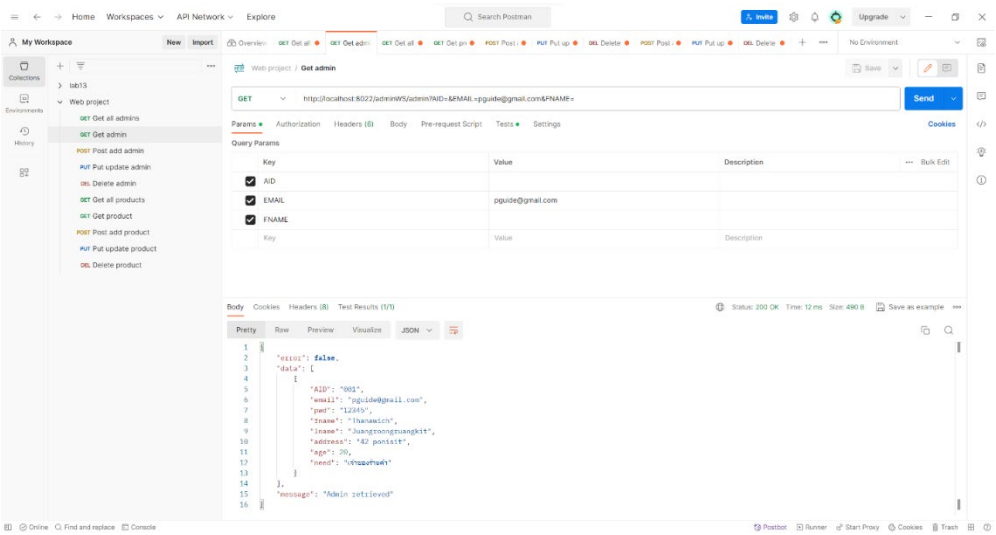
4. สร้างตัวเชื่อมต่อฐานข้อมูล

```
let connection = mysql.createConnection({
    host : process.env.MYSQL_HOST,
    user : process.env.MYSQL_USERNAME,
    password : process.env.MYSQL_PASSWORD,
    database : process.env.MYSQL_DATABASE
});
```

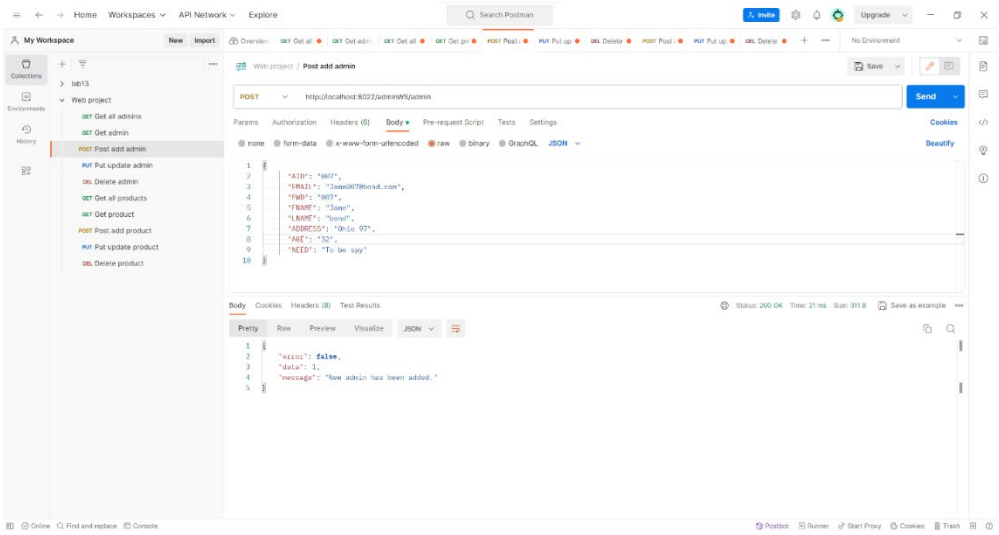
5. ผลการทดสอบของเว็บเซอร์วิสทั้งหมด โดยใชโปรแกรม Postman



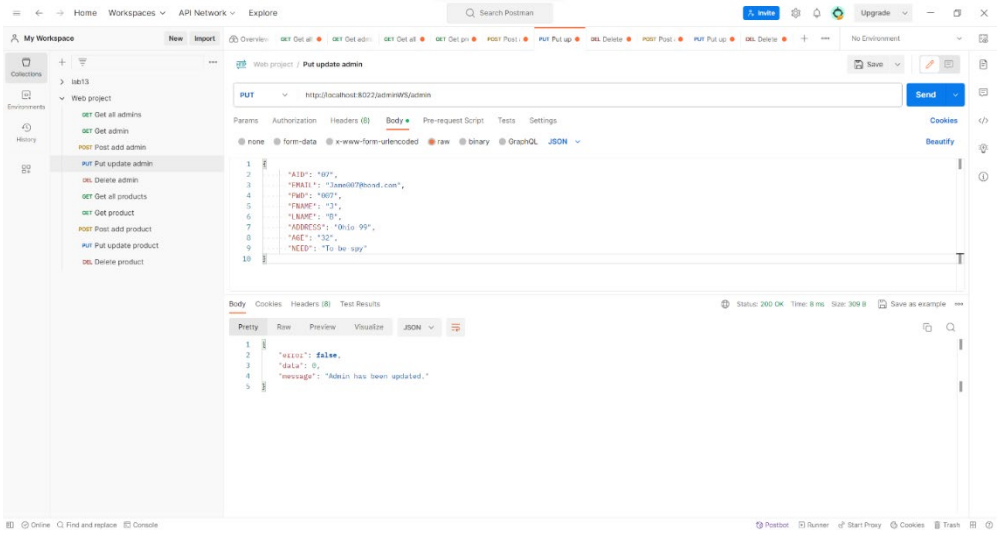
รูปภาพที่ 5.1 method GET ของ url adminWS/admins



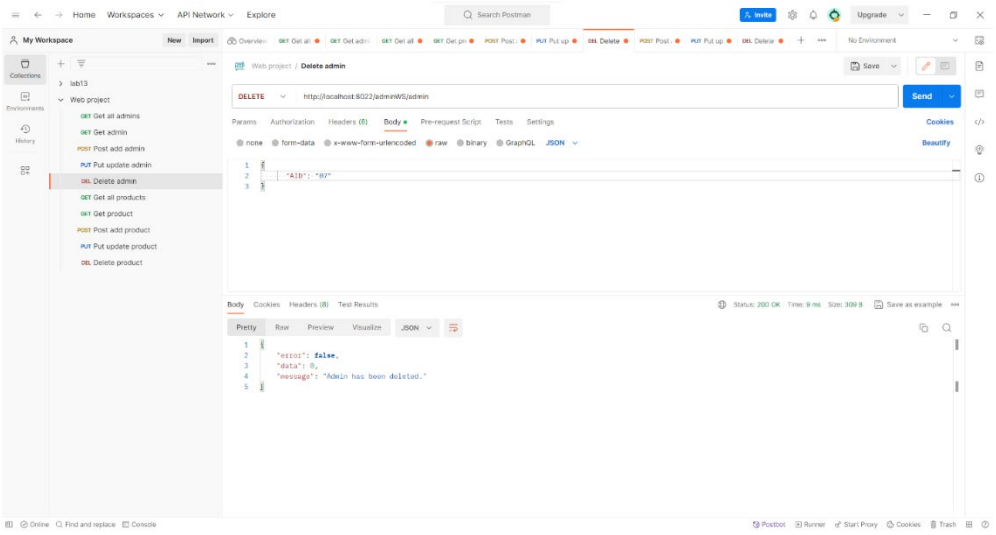
รูปภาพที่ 5.2 method GET ของ url adminWS/admin



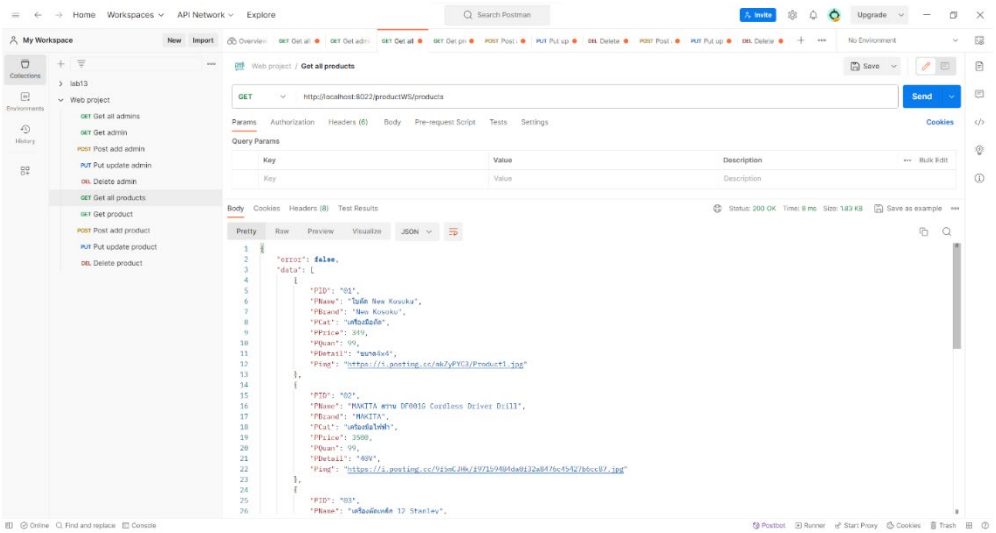
รูปภาพที่ 5.3 method POST ของ url adminWS/admin



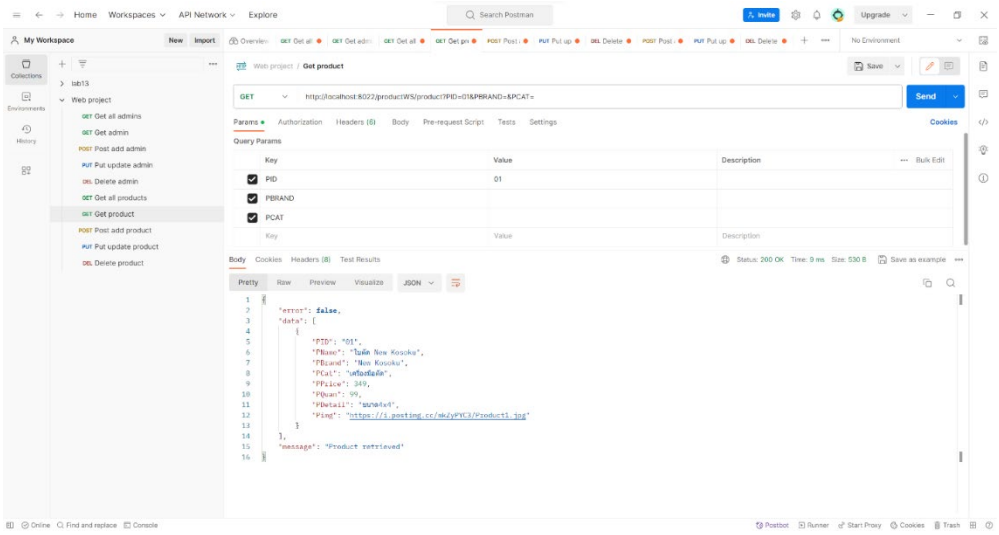
รูปภาพที่ 5.4 method PUT ของ url adminWS/admin



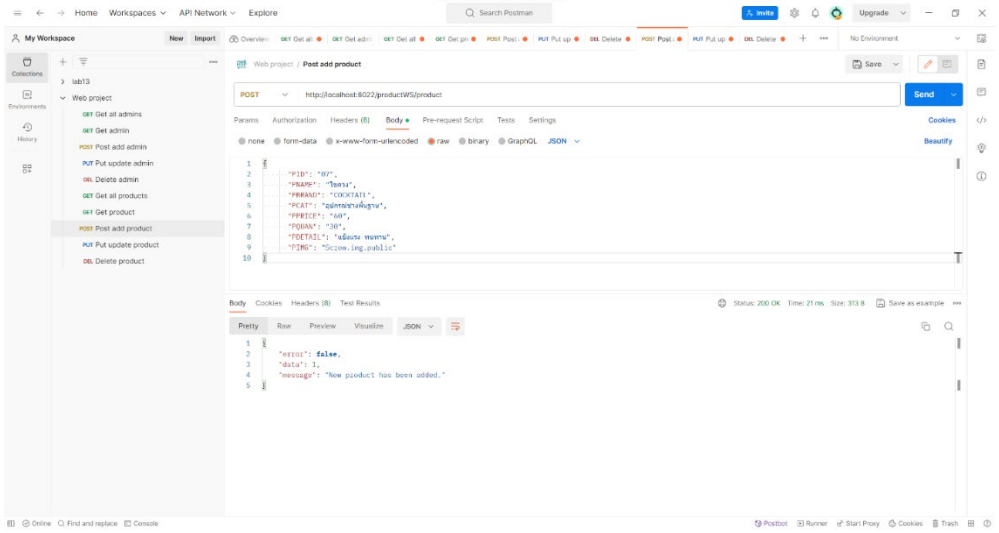
รูปภาพที่ 5.5 method DELETE ของ url adminWS/admin



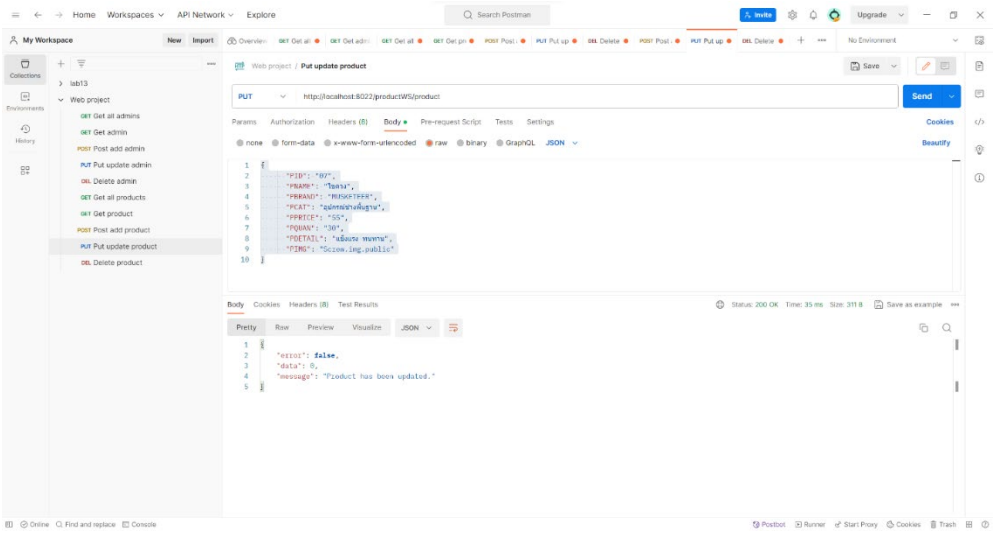
รูปภาพที่ 5.6 method GET ของ url productWS/products



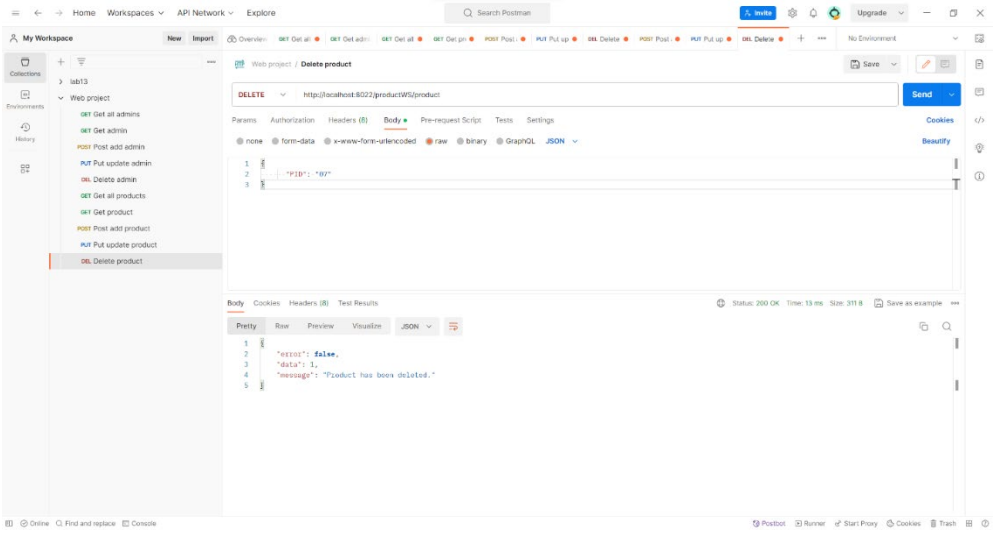
รูปภาพที่ 5.7 method GET ของ url productWS/product



รูปภาพที่ 5.8 method POST ของ url productWS/product



รูปภาพที่ 5.9 method PUT ของ url productWS/product



รูปภาพที่ 5.10 method DELETE ของ url productWS/product