เอกสารประกอบการเรียน วิชา 01204271
วิศวกรรมคอมพิวเตอร์เบื้องต้น
การทดลองที่ 1 การติดตั้งโปรแกรมพื้นฐานและสร้าง Restful API

วัตถุประสงค์

1. สามารถติดตั้งโปรแกรมที่ต้องใช้งานและตรวจสอบเวอร์ชันได้
2. สามารถสร้าง Restful API ได้
3. สามารถใช้โปรแกรม Postman ได้

การทดลองที่ 1.1 ติดตั้ง Node.js และตรวจสอบเวอร์ชัน

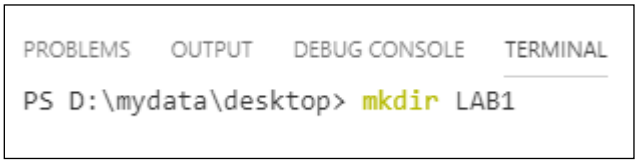
1. ให้ดาวน์โหลด Node.js ผ่านเว็บไซต์ https://nodejs.org/en/
2. หลังจากติดตั้งแล้วให้ตรวจสอบเวอร์ชัน โดยเปิด Node.js command prompt หรือ Command Prompt
3. พิมพ์คำสั่ง node –v และกด Enter จากนั้นตรวจสอบผล
4. พิมพ์คำสั่ง npm –v และกด Enter จากนั้นตรวจสอบผล

การทดลองที่ 1.2 ติดตั้ง Visual Studio Code และลง Extensions

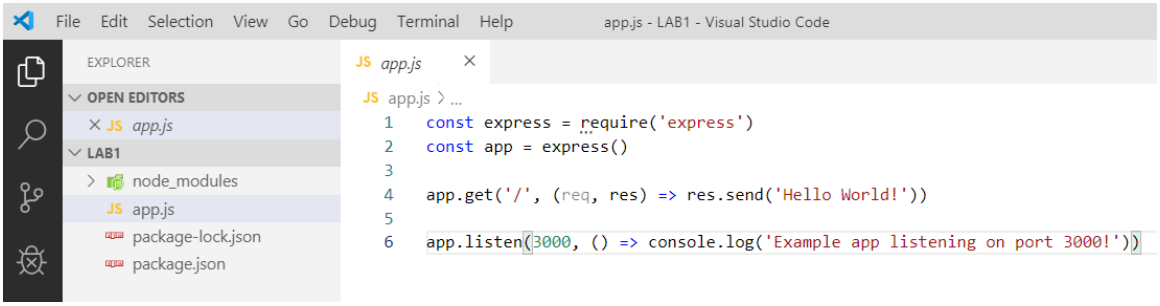
1. ให้ดาวน์โหลด Visual Studio Code ผ่านเว็บไซต์ https://code.visualstudio.com/
2. หลังจากติดตั้งแล้วให้เปิดโปรแกรม Visual Studio Code
3. ทดลองติดตั้ง Extensions

การทดลองที่ 1.3 การใช้คำสั่ง npm และติดตั้ง Express

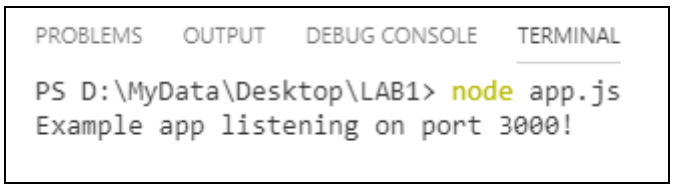
1. สร้างโปรเจคใหม่ โดยเปิด Terminal จากเมนู Terminal > New Terminal พิมพ์คำสั่ง mkdir ตามด้วยชื่อโปรเจค จากนั้นคลิก Enter และเปลี่ยน path ด้วยคำสั่ง cd ตามด้วยชื่อโปรเจค จากนั้นคลิก Enter ดังรูป



2. พิมพ์คำสั่ง npm init และกด Enter โปรแกรมจะให้เรากำหนดค่าต่างๆของโปรเจค
3. จะได้ไฟล์ package.json อยู่ใน Folder ของโปรเจค
4. ติดตั้ง Express โดยพิมพ์คำสั่ง npm install express และกด Enter
5. คลิกขวาที่โปรเจคเลือก New File > ตั้งชื่อไฟล์ app.js และกด Enter
6. พิมพ์คำสั่งดังรูป



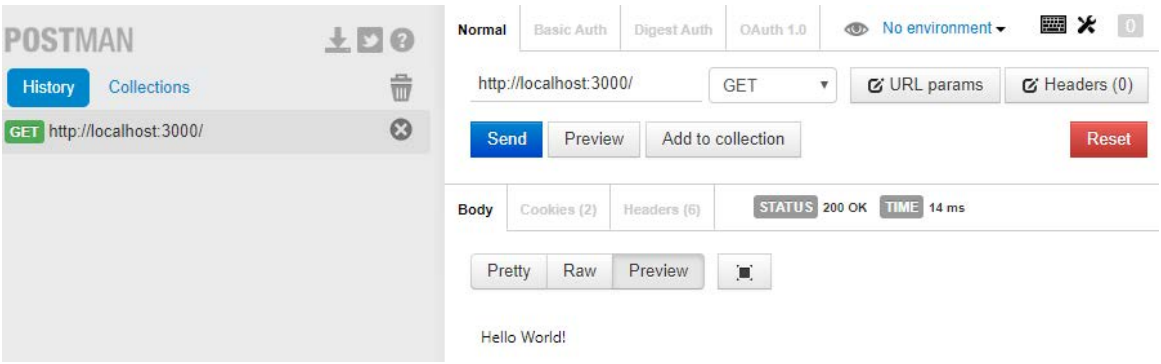
7. พิมพ์คำสั่ง node app.js ที่ Terminal และกด Enter (หยุดการทำงานให้กด Ctrl + C)



8. จากนั้นเปิด Browser และพิมพ์ URL > http://localhost:3000

การทดลองที่ 1.4 ติดตั้ง Tabbed Postman - REST Client ที่ Chrome (Extension)

1. เปิด Browser Chrome เข้า Chrome web store หา Tabbed Postman - REST Client และติดตั้ง https://chrome.google.com/webstore/search/Tabbed%20Postman%20-%20REST%20Client%20?hl=th
2. เปิดโปรแกรมทดลองใส่ url เลือก GET และกด Send ดูผลลัพธ์



การทดลองที่ 1.5 สร้าง Restful API อย่างง่าย

1. ติดตั้ง nodemon ด้วยคำสั่ง npm install -g nodemon
2. ติดตั้ง body-parser ด้วยคำสั่ง npm install body-parser
3. แก้ไขไฟล์ app.js ดังนี้

```
const express = require('express')
const app = express()
var bodyParser = require('body-parser');
app.use(bodyParser.json());
```

3.1 เพิ่ม users ประกอบด้วย id, name และ email ดังตัวอย่าง

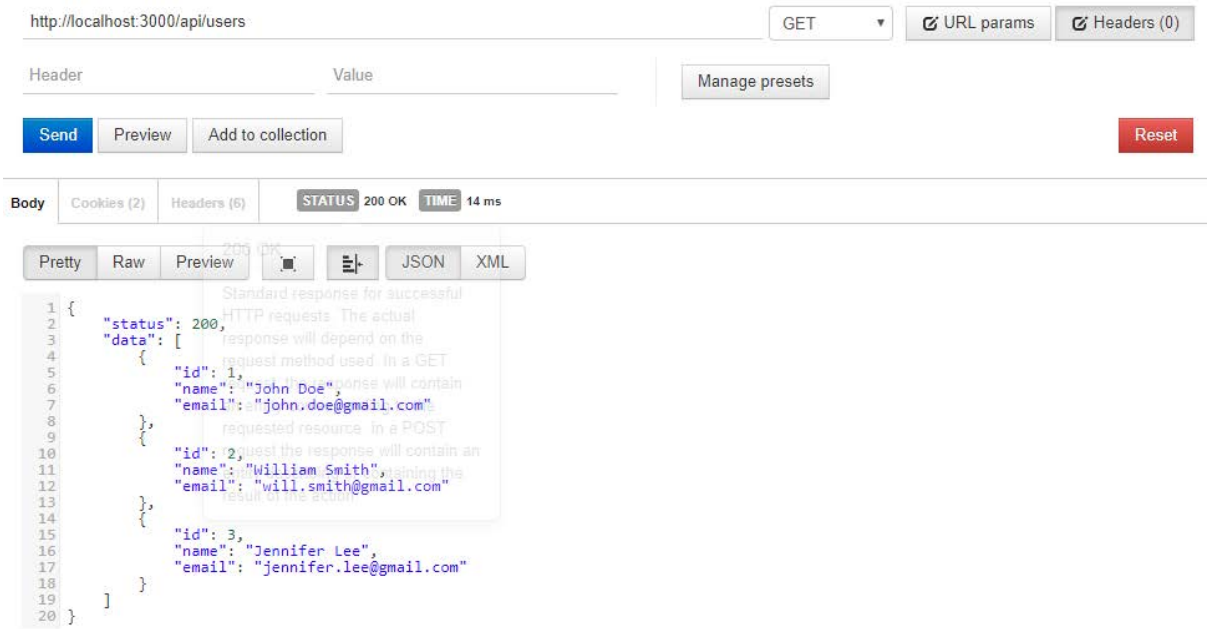
```
const users = [
    {
        "id":1,
        "name":"John Doe",
        "email":"john.doe@gmail.com"
    },
    {
        "id":2,
        "name":"William Smith",
        "email":"will.smith@gmail.com"
    },
    {
        "id":3,
        "name":"Jennifer Lee",
        "email":"jennifer.lee@gmail.com"
    }
]
```

3.2 เพิ่มคำสั่ง GET ข้อมูลทั้งหมด

```
app.get('/api/users', (req, res) => {
    const result = {
        "status":200,
        "data":users
    }
    return res.json(result)
})
```

3.3 พิมพ์คำสั่ง nodemon app.js เพื่อรัน API

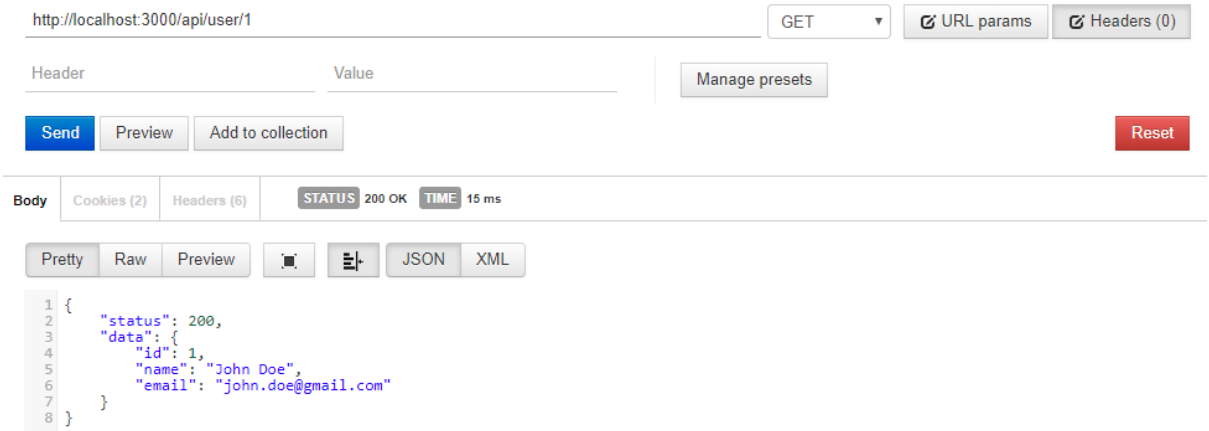
3.4 เปิด postman พิมพ์ url : http://localhost:3000/api/users เลือก GET และกด Send



3.5 เพิ่มคำสั่ง GET ข้อมูลตาม id

```
app.get("/api/user/:id", (req, res) => {
  // หาว่า ใน users มี user.id ที่ตรงกับ params.id หรือไม่ ถ้ามี ก็เก็บ user นั้นๆ
  let user = users.find(user => user.id === parseInt(req.params.id));
  // ถ้าไม่มี ก็ส่งสถานะ 400 และข้อความกลับออกไป
  if (!user)
    return res
      .status(400)
      .json({ status: 400, message: "Not found user with the given ID" });
  // มี user
  res.user = user;
  const result = {
    status: 200,
    data: res.user
  };
  return res.json(result);
});
```

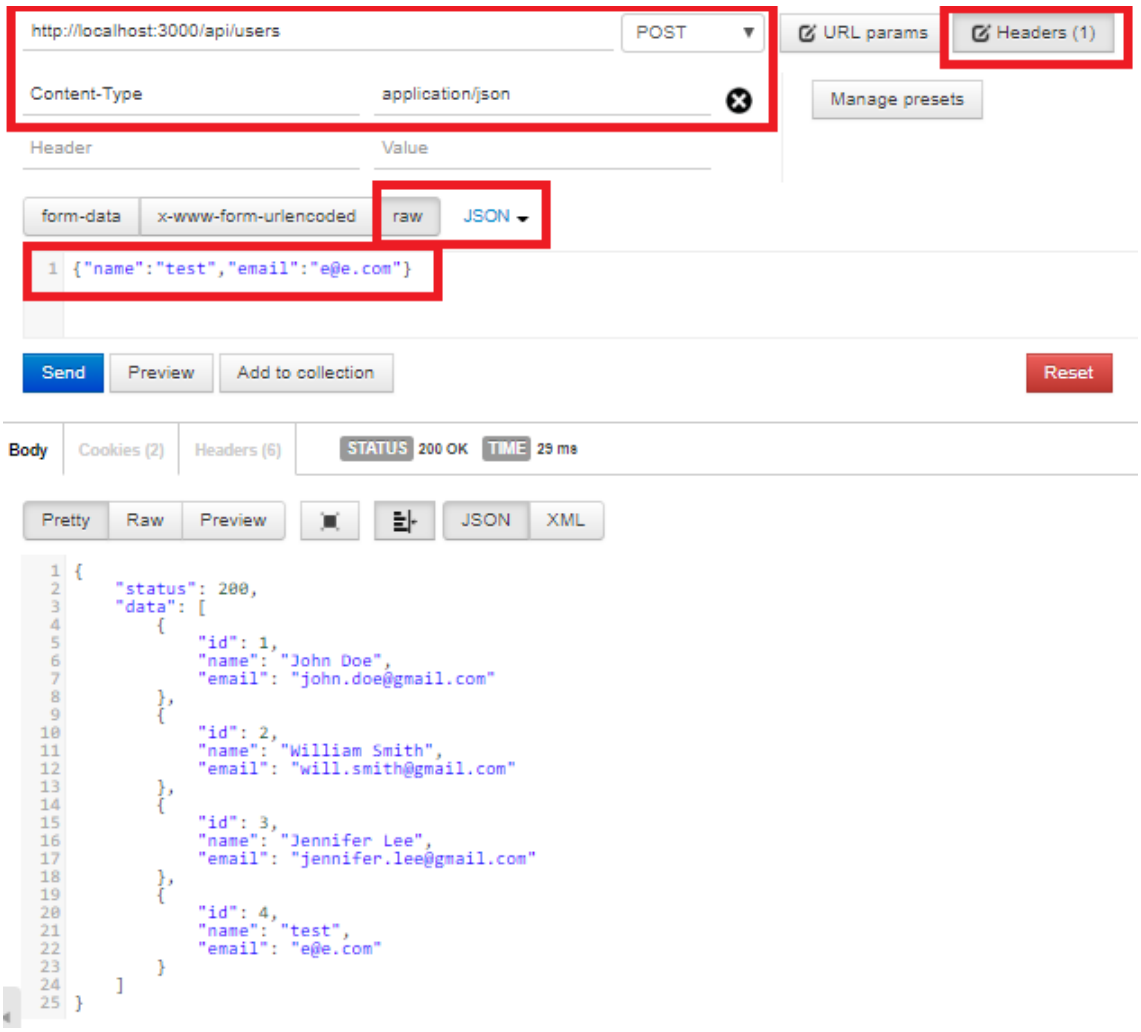
3.6 เปิด postman พิมพ์ url : http://localhost:3000/api/user/1 เลือก GET และกด Send



3.7 เพิ่มคำสั่ง POST เพื่อเพิ่มข้อมูล

```
app.post('/api/users', (req, res) => {
    let user = {
        "id":users.length+1, // เพิ่ม id จากค่าเดิมไปอีก 1
        "name":req.body.name, // ใช้ค่า name จากที่ส่งเข้ามา
        "email":req.body.email // ใช้ค่า email จากที่ส่งเข้ามา
    }
    users.push(user) // เพิ่มข้อมูลใหม่เข้าไปใน array users
    const result = { // รูปแบบผลลัพธ์ข้อมูลที่จะส่งกลับ
        "status":200,
        "data":users
    }
    return res.json(result)
})
```

3.8 เปิด postman พิมพ์ url : http://localhost:3000/api/users เลือก POST คลิก Headers กำหนดช่อง Header คือ Content-Type และช่อง Value คือ application/json จากนั้นเลือก raw และเลือก JSON และใส่ข้อมูลที่ต้องการเพิ่มในกล่องข้อความ เสร็จแล้วกด Send



3.9 เพิ่มคำสั่ง PUT เพื่อแก้ไขรายการ

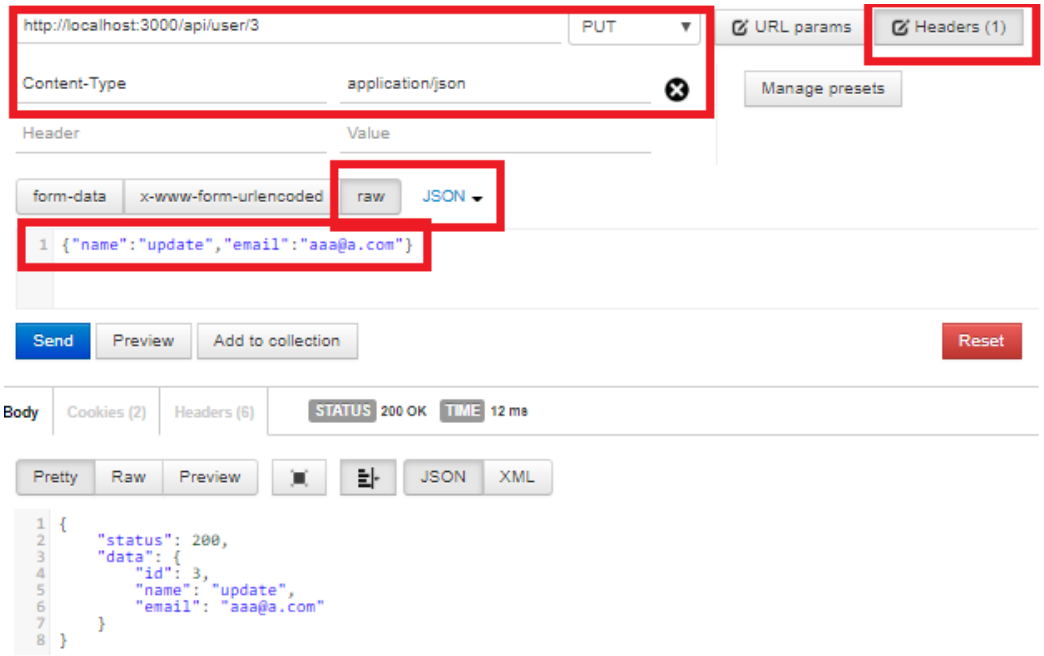
```
app.put('/api/user/:id', (req, res) => {
    let user = users.find( (user) => user.id === parseInt(req.params.id))
    if (!user)
      return res
        .status(400)
        .json({ status: 400, message: "Not found user with the given ID" });

    let user_index = users.findIndex( (user) => user.id === parseInt(req.params.id))

    user = { // กำหนดข้อมูลที่จะอัพเดท
        "id":user.id, // อัพเดท id ที่ตรงกับค่า params.id
        "name":req.body.name, // ใช้ข้อมูลใหม่ที่ส่งเข้ามาสำหรับ name
        "email":req.body.email // ใช้ข้อมูลใหม่ที่ส่งเข้ามาสำหรับ email
    }
    //แทนที่ข้อมูล
    users[user_index] = user

    // ส่งข้อมูลที่ได้อัพเดทเรียบร้อยแล้ว กลับออกไป
    const result = {
        "status":200,
        "data":user
    }
    return res.json(result)
})
```

3.10 เปิด postman พิมพ์ url : http://localhost:3000/api/user/3 เลือก PUT คลิก Headers กำหนดช่อง Header คือ Content-Type และช่อง Value คือ application/json จากนั้นเลือก raw และเลือก JSON และใส่ข้อมูลที่ต้องการแก้ไขในกล่องข้อความ เสร็จแล้วกด Send



3.11 เพิ่มคำสั่ง Delete เพื่อลบรายการ

```
app.delete("/api/user/:id", (req, res) => {
  let user = users.find(user => user.id === parseInt(req.params.id));
  if (!user)
    return res
      .status(400)
      .json({ status: 400, message: "Not found user with the given ID" });

  let user_index = users.findIndex(user => user.id === parseInt(req.params.id));
  users.pop(user_index);

  const result = {
    status: 200,
    data: users
  };
  return res.json(result);
});
```

3.12 เปิด postman พิมพ์ url : http://localhost:3000/api/user/3 เลือก DELETE แล้วกด Send