



Backend with Node.js & Express

ปรียากรณ์ สมยา
10 มีนาคม 2568



CONTENTS



01 Overview

04 Connecting to MySQL

02 Project Setup and Install Dependencies

05 Creating API Routes and API Code

03 Setting up Express Server

06 Testing API with Postman

07 Summary & Q&A

01

Overview

Node.js and Express

Overview of Node.js and Express



Node.js

- JavaScript runtime สำหรับพัฒนา backend
- สร้างขึ้นบนเอ็นจิน V8 ของ Chrome ช่วยให้ นักพัฒนาสามารถเรียกใช้ JavaScript บนฝั่ง เซิร์ฟเวอร์ได้
- ช่วยให้สร้างแอปพลิเคชันที่ปรับขนาดได้และมี ประสิทธิภาพสูง
- ใช้โมเดล I/O ที่ไม่มีการบล็อกตามเหตุการณ์

Express.js

- Framework สำหรับสร้างเว็บเซิร์ฟเวอร์
- ที่มีความยืดหยุ่นและเรียบง่าย
- คุณสมบัติที่แข็งแกร่งสำหรับการสร้างแอปพลิเคชันเว็บและ API
- ช่วยลดความซับซ้อนของการกำหนดเส้นทาง
- การรวมมิดเดิลแวร์
- การจัดการคำขอและการตอบกลับ

02

Project Setup and Install Dependencies

- Install Node.js and create a project
- Install Express and required libraries

Project Setup



1. Install Node.js

download from [Node.js official website](https://nodejs.org/) or install

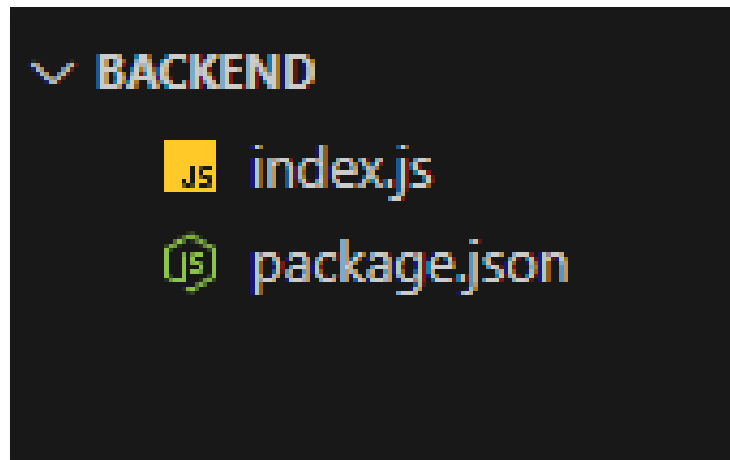
```
sh
```

```
node -v  
npm -v
```

2. Set Up a New Project

```
sh
```

```
mkdir backend && cd backend  
npm init -y
```



Project Setup



3. Install Dependency

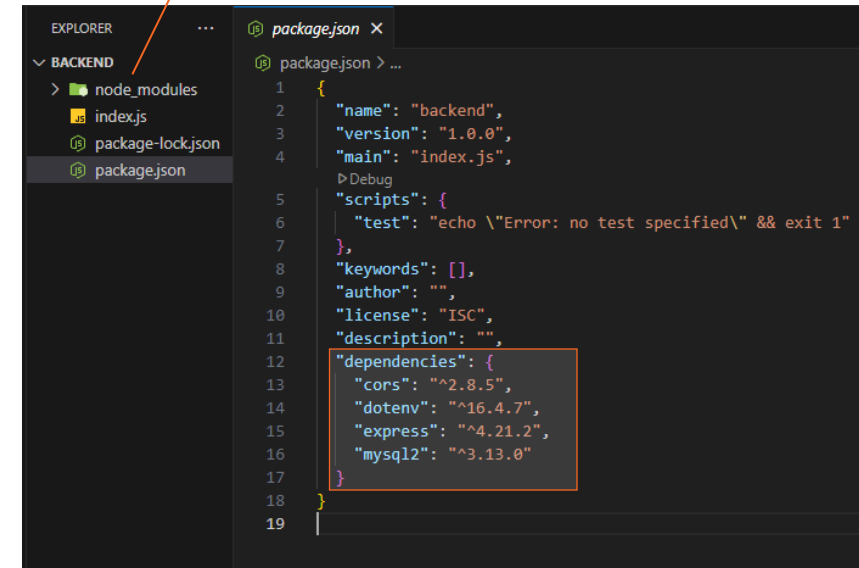
หมายถึง ซอฟต์แวร์, ไลบรารี, หรือแพ็คเกจอื่น ๆ ที่โปรเจกต์ของคุณต้องใช้เพื่อทำงานได้

```
sh
```

```
npm install express mysql2 cors dotenv
```

Npm	Details
express	เป็นเว็บเฟรมเวิร์กยอดนิยมสำหรับ Node.js ที่ช่วยให้สร้าง API
mysql2	เป็นไลบรารีที่ใช้เชื่อมต่อและจัดการ MySQL database ใน Node.js
cors (Cross-Origin Resource Sharing)	เป็น middleware สำหรับ Express ที่ช่วยจัดการการเข้าถึง API จากโดเมนอื่น
dotenv	ใช้สำหรับโหลดตัวแปรสภาพแวดล้อม (.env) มาใช้ใน Node.js

```
> node_modules
```



03

Setting up Express Server

Setting up Express Server



```
sh
```

```
node index.js
```

The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project structure with a 'BACKEND' folder containing 'node_modules', 'index.js', 'package-lock.json', and 'package.json'. The 'index.js' file is selected. The main editor area shows the code for 'index.js':

```
1  const express = require('express');
2  const app = express();
3
4  app.get('/', (req, res) => {
5    res.send('Hello World!');
6  });
7
8  app.listen(3000, () => console.log('Server running on port 3000'));
9
10
```

At the bottom, the TERMINAL panel shows the command 'node index.js' being executed, resulting in the output 'Server running on port 3000'.

04

Connecting to MySQL

Use .env file

Connecting to MySQL



.env

```
EXPLORER  ...  dbjs  .env
BACKEND
  > node_modules
  .env
  db.js
  index.js
  package-lock.json
  package.json

1  PORT=3000
2  DB_HOST=localhost
3  DB_USER=root
4  DB_PASSWORD=1234
5  DB_NAME=demotb
6
7
8
9
10
11
12
13
14
15
16
17
```

db.js

```
EXPLORER  ...  dbjs
BACKEND
  > node_modules
  .env
  db.js
  index.js
  package-lock.json
  package.json

1  require('dotenv').config();
2  const mysql = require('mysql2/promise');
3
4  const pool = mysql.createPool({
5    host: process.env.DB_HOST,
6    user: process.env.DB_USER,
7    password: process.env.DB_PASSWORD,
8    database: process.env.DB_NAME,
9    waitForConnections: true,
10  });
11
12  module.exports = pool;
13
14
15
16  async function checkConnection() {
17    try {
18      const connection = await pool.getConnection(); // * ลองขอ connection
19      console.log('✅ MySQL Connected!');
20      connection.release(); // * คืน connection กลับไปที่ pool
21    } catch (err) {
22      console.error('❌ MySQL Connection Error:', err);
23    }
24  }
25
26  checkConnection();
27
```

sh

node index.js

05

Creating API Routes and API Code

แบบที่ 1 กำหนด Route โดยตรง

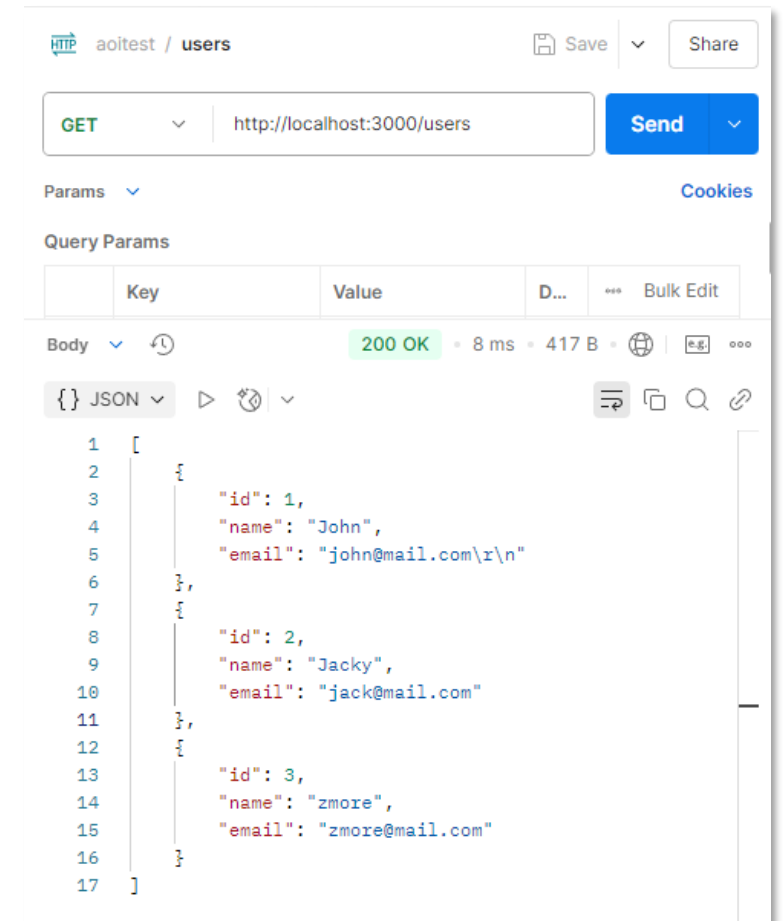
แบบที่ 2 แยก Routes ออกไปเป็นไฟล์

Creating API Routes and API Code



แบบที่ 1 กำหนด Route โดยตรง

```
index.js x
index.js > ...
1  const express = require('express');
2  const cors = require('cors');
3  const pool = require('./db'); // ไฟล์ database connection
4  const app = express();
5
6  app.use(cors());
7  app.use(express.json());
8
9  app.get('/', async (req, res) => {
10     res.send('Hello World');
11 });
12
13 app.listen(3000, () => console.log('Server running on port 3000'));
14
15
16 app.get('/users', async (req, res) => {
17     const [rows] = await pool.query('SELECT * FROM users');
18     res.json(rows);
19 });
20
21 app.post('/users', async (req, res) => {
22     const { name, email } = req.body;
23     const result = await pool.query('INSERT INTO users (name, email) VALUES (?, ?)', [name, email]);
24     res.json({ id: result[0].insertId, name, email });
25 });
26
27
28
```

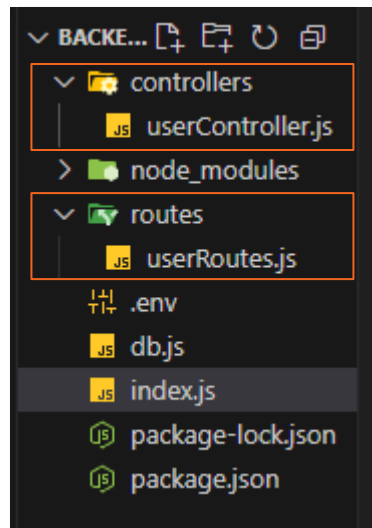


Creating API Routes and API Code

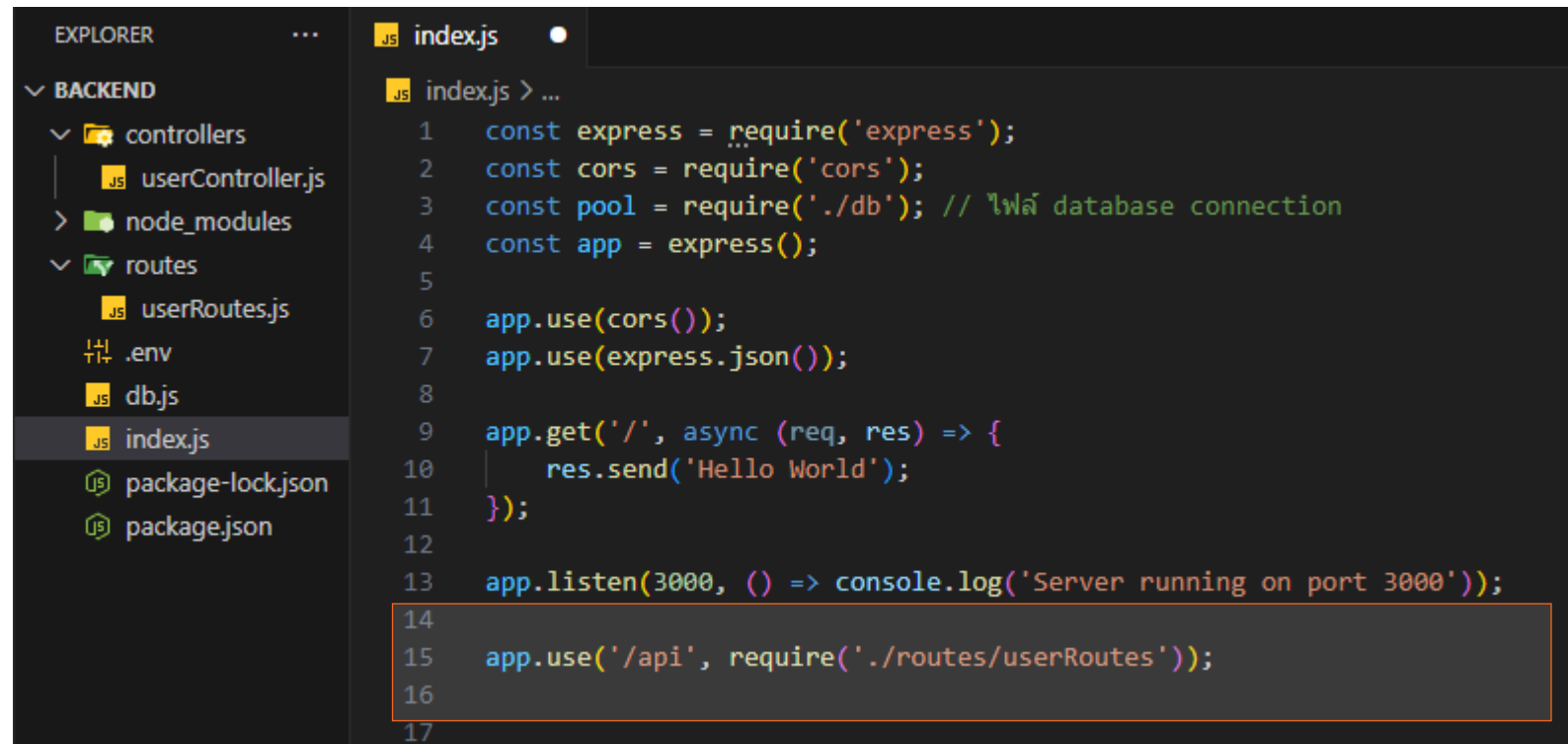


แบบที่ 2 แยก Routes ออกไปเป็นไฟล์

1. folder structure



2. Add Code in 'index.js'

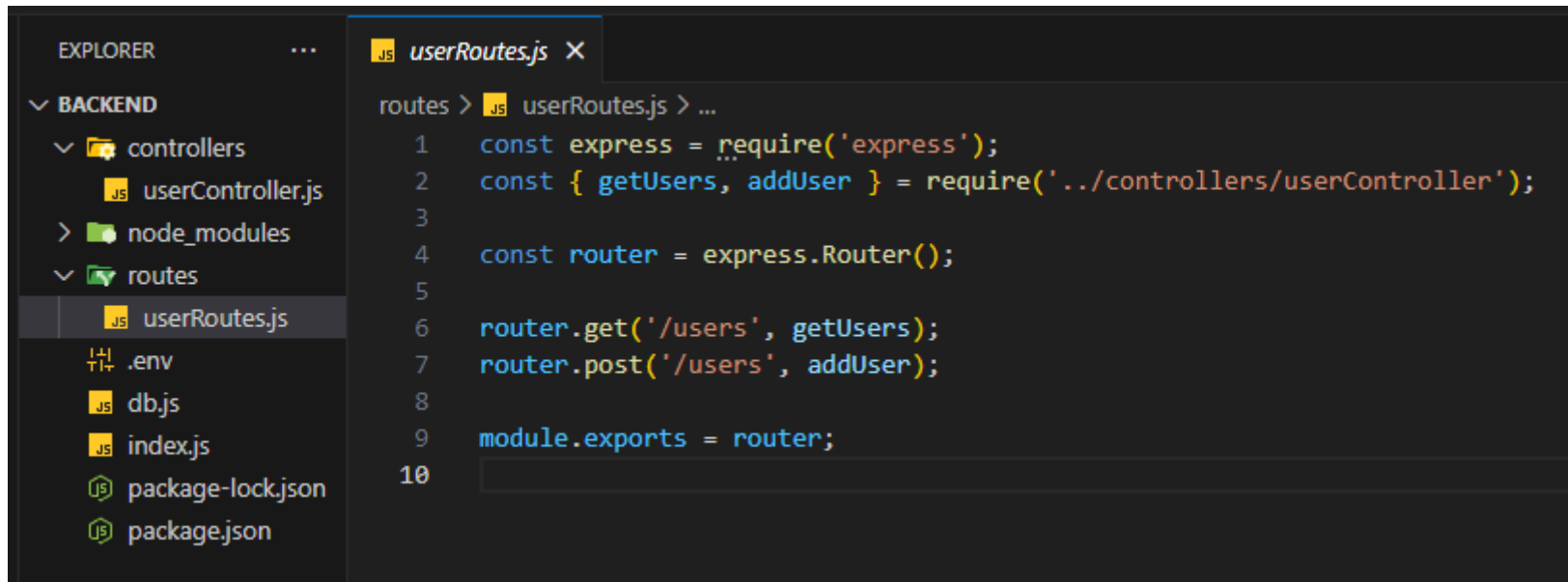


Creating API Routes and API Code



แบบที่ 2 แยก Routes ออกไปเป็นไฟล์

3. routes/userRoutes.js



The screenshot shows the VS Code interface. On the left, the Explorer sidebar is open, showing the project structure under the 'BACKEND' folder. The 'routes' folder is expanded, and 'userRoutes.js' is selected. The main editor area displays the code for 'userRoutes.js'.

```
routes > JS userRoutes.js > ...
1  const express = require('express');
2  const { getUsers, addUser } = require('../controllers/userController');
3
4  const router = express.Router();
5
6  router.get('/users', getUsers);
7  router.post('/users', addUser);
8
9  module.exports = router;
10
```

Creating API Routes and API Code



แบบที่ 2 แยก Routes ออกไปเป็นไฟล์

4. controllers/userController.js

```
EXPLORER  ...  JS userController.js X
└─ BACKEND
  └─ controllers
    └─ JS userController.js
  └─ node_modules
  └─ routes
    └─ JS userRoutes.js
    .env
    JS db.js
    JS index.js
    package-lock.json
    package.json

controllers > JS userController.js > ...
1  const db = require('../db');
2
3  exports.getUsers = async (req, res) => {
4    try {
5      const [results] = await db.query('SELECT * FROM users');
6      res.json(results);
7    } catch (err) {
8      res.status(500).json({ error: err.message });
9    }
10 };
11
12 exports.addUser = async (req, res) => {
13   const { name, email } = req.body;
14   try {
15     const [results] = await db.query('INSERT INTO users (name, email) VALUES (?, ?)', [name, email]);
16     res.json({ message: 'User added successfully', userId: results.insertId });
17   } catch (err) {
18     res.status(500).json({ error: err.message });
19   }
20 };
21
```

06

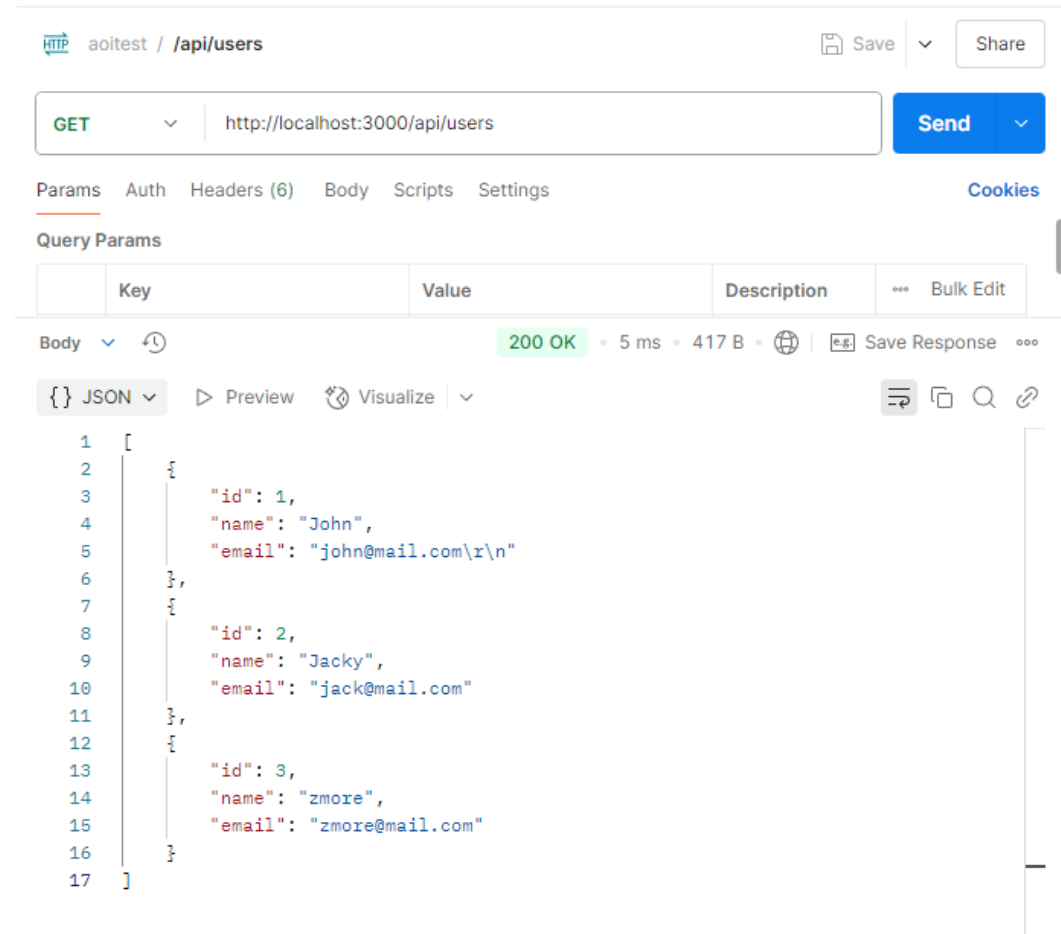
Testing API with Postman

Method: GET,POST

Testing API with Postman



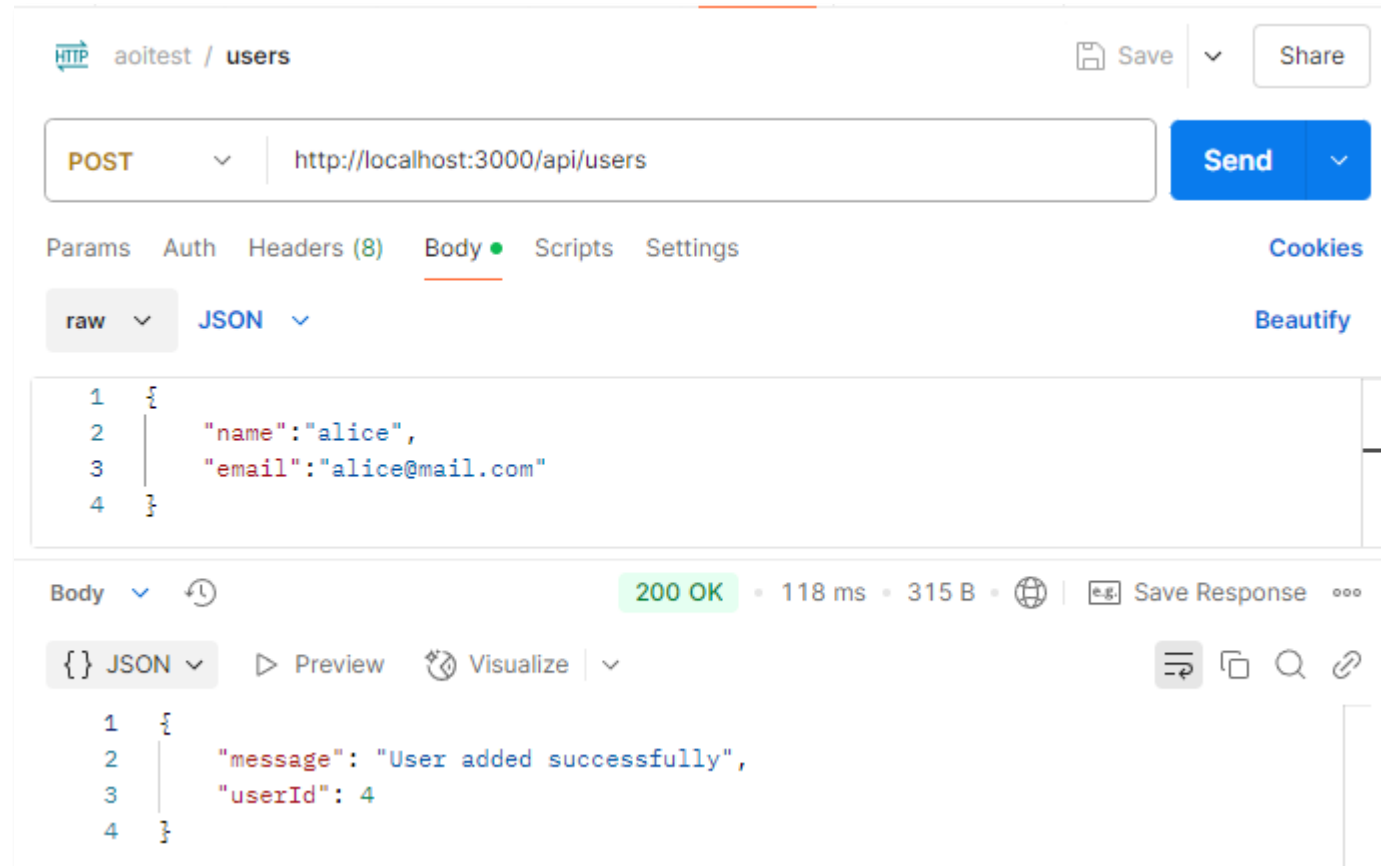
1. Method: GET (http://localhost:3000/api/users/)



Testing API with Postman



1. Method: POST (http://localhost:3000/api/users/)



07

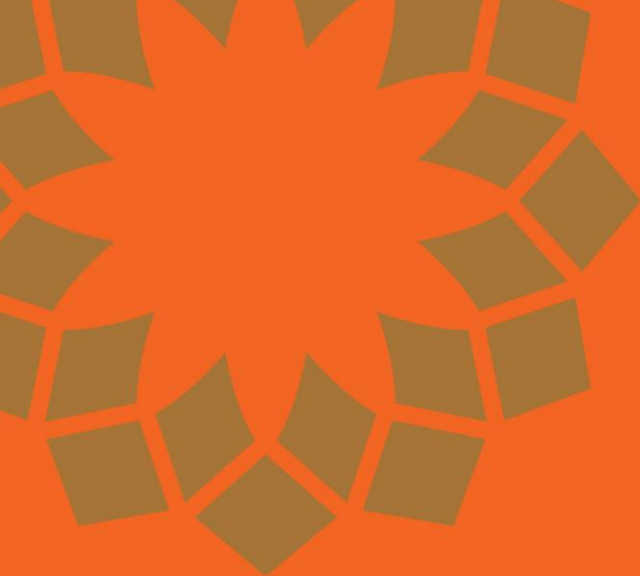
Summary & Q&A



Summary



Database	ไลบรารีที่ใช้	ตัวอย่าง Connection String
MySQL / MariaDB	mysql2 หรือ sequelize	{ host: 'localhost', user: 'root', password: 'pass', database: 'testdb' }
PostgreSQL	pg หรือ sequelize	postgres://user:pass@localhost:5432/testdb
SQL Server (MSSQL)	mssql	{ server: 'localhost', user: 'sa', password: 'pass', database: 'testdb' }
Oracle	oracledb	{ user: 'system', password: 'pass', connectString: 'localhost/XE' }
Ingres	odbc	DSN=IngresDB;UID=user;PWD=password
SQLite	sqlite3 หรือ sequelize	{ filename: './data.db' }



Innovate the Future

