

## Task Board - Software Architecture Learning Lab

Screenshots ที่ต้องส่ง:

- VM IP Address

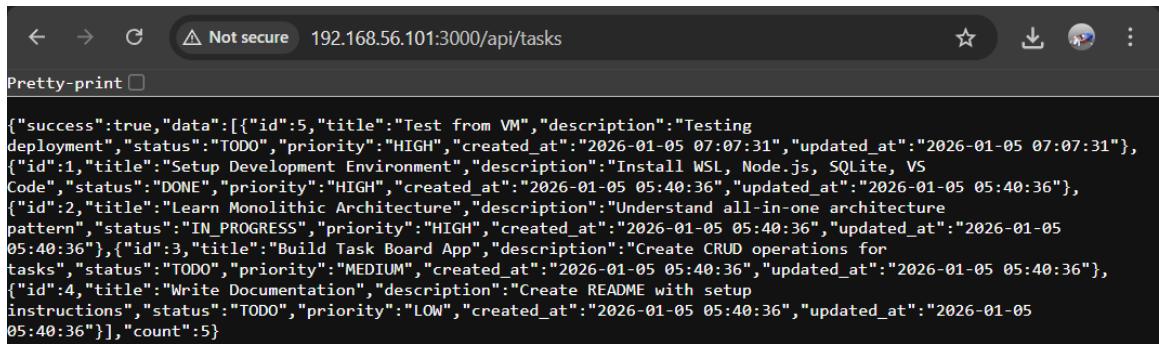
```
devlab@ubuntu:~$ hostname -I  
10.0.2.15 192.168.56.101 fd17:625c:f037:2:a00:27ff:fead:11bd
```

- PM2 Status

```
devlab@ubuntu:~$ pm2 status
```

<b>id</b>	<b>name</b>	<b>mode</b>	<b>�</b>	<b>status</b>	<b>cpu</b>	<b>memory</b>
0	task-board-api	fork	0	online	0%	58.0mb

- API Response (จาก local browser) [http://VM\_IP:3000/api/tasks]



A screenshot of a web browser window displaying the JSON response from the API endpoint `http://192.168.56.101:3000/api/tasks`. The browser's address bar shows the URL. Below the address bar, there are buttons for back, forward, and refresh, along with a "Not secure" warning icon. To the right of the address bar are icons for star, download, and more. A "Pretty-print" checkbox is checked. The main content area shows the JSON data:

```
{"success":true,"data":[{"id":5,"title":"Test from VM","description":"Testing deployment","status":"TODO","priority":"HIGH","created_at":"2026-01-05 07:07:31","updated_at":"2026-01-05 07:07:31"}, {"id":1,"title":"Setup Development Environment","description":"Install WSL, Node.js, SQLite, VS Code","status":"DONE","priority":"HIGH","created_at":"2026-01-05 05:40:36","updated_at":"2026-01-05 05:40:36"}, {"id":2,"title":"Learn Monolithic Architecture","description":"Understand all-in-one architecture pattern","status":"IN_PROGRESS","priority":"HIGH","created_at":"2026-01-05 05:40:36","updated_at":"2026-01-05 05:40:36"}, {"id":3,"title":"Build Task Board App","description":"Create CRUD operations for tasks","status":"TODO","priority":"MEDIUM","created_at":"2026-01-05 05:40:36","updated_at":"2026-01-05 05:40:36"}, {"id":4,"title":"Write Documentation","description":"Create README with setup instructions","status":"TODO","priority":"LOW","created_at":"2026-01-05 05:40:36","updated_at":"2026-01-05 05:40:36"}],"count":5}
```

#### 4. Frontend ทำงาน (แสดงการสร้าง task)

The screenshot shows a task management interface titled "All Tasks". At the top, there is a filter dropdown set to "All Tasks". Below the filter, there are three columns representing different stages of task status:

- To Do (3)**: Contains three tasks:
  - Test from VM** (HIGH priority): Testing deployment. Created: Jan 5, 07:07 AM. Buttons: NEXT STATUS, DELETE.
  - Build Task Board App** (MEDIUM priority): Create CRUD operations for tasks. Created: Jan 5, 05:40 AM. Buttons: NEXT STATUS, DELETE.
  - Write Documentation** (LOW priority): Create README with setup instructions. Created: Jan 5, 05:40 AM. Buttons: NEXT STATUS, DELETE.
- In Progress (1)**: Contains one task:
  - Learn Monolithic Architecture** (HIGH priority): Understand all-in-one architecture pattern. Created: Jan 5, 05:40 AM. Buttons: NEXT STATUS, DELETE.
- Done (1)**: Contains one task:
  - Setup Development Environment** (HIGH priority): Install WSL, Node.js, SQLite, VS Code. Created: Jan 5, 05:40 AM. Buttons: NEXT STATUS, DELETE.

#### 5. PM2 Logs

```
devlab@ubuntu:~$ pm2 logs --lines 20
[TAILING] Tailing last 20 lines for [all] processes (change the value with --lines option)
/home/devlab/.pm2/pm2.log last 20 lines:
PM2      | 2026-01-05T08:10:26: PM2 log: Stopping app:task-board-api id:0
PM2      | 2026-01-05T08:10:26: PM2 log: App [task-board-api:0] exited with code [0] via signal [SIGINT]
PM2      | 2026-01-05T08:10:26: PM2 log: pid=900 msg=process killed
PM2      | 2026-01-05T08:10:26: PM2 log: App [task-board-api:0] starting in -fork mode-
PM2      | 2026-01-05T08:10:26: PM2 log: App [task-board-api:0] online

/home/devlab/projects/task-board-api/logs/error-0.log last 20 lines:
/home/devlab/projects/task-board-api/logs/output-0.log last 20 lines:
0|task-boa | 2026-01-05T07:12:59: i  [INFO] GET /config.js
0|task-boa | 2026-01-05T07:48:31: i  [INFO] GET /api/tasks
0|task-boa | 2026-01-05T07:48:31: i  [INFO] GET /favicon.ico
0|task-boa | 2026-01-05T07:51:20: i  [INFO] GET /config.js
0|task-boa | 2026-01-05T07:51:25: i  [INFO] GET /config.js
0|task-boa | 2026-01-05T07:51:45: i  [INFO] GET /config.js
0|task-boa | 2026-01-05T07:51:50: i  [INFO] GET /config.js
0|task-boa | 2026-01-05T08:10:26: i  [INFO] 🔍 Shutting down...
0|task-boa | 2026-01-05T08:10:26: ✓ ปิดการเชื่อมต่อฐานข้อมูล
0|task-boa | 2026-01-05T08:10:26: [dotenv@17.2.3] injecting env (3) from .env -- tip: 📁 backup p and recover secrets: https://dotenvx.com/ops
0|task-boa | 2026-01-05T08:10:27: [dotenv@17.2.3] injecting env (0) from .env -- tip: ✅ audit secrets and track compliance: https://dotenvx.com/ops
0|task-boa | 2026-01-05T08:10:27: ✓ เชื่อมต่อฐานข้อมูลสำเร็จ: /home/devlab/projects/task-board-api/database/tasks.db
0|task-boa | 2026-01-05T08:10:27: i  [INFO] 🚀 Server running at http://0.0.0.0:3000
0|task-boa | 2026-01-05T08:10:27: i  [INFO] 🎨 Environment: production
0|task-boa | 2026-01-05T08:10:35: i  [INFO] GET /config.js
0|task-boa | 2026-01-05T08:10:50: i  [INFO] GET /config.js
```

## 6. Network Ping Test

```
ladadmin@LAPTOP-L1RC473V:~/class/engse207/week5-client-server$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=63 time=1.77 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=63 time=1.22 ms
64 bytes from 192.168.56.101: icmp_seq=3 ttl=63 time=0.763 ms
64 bytes from 192.168.56.101: icmp_seq=4 ttl=63 time=0.754 ms
64 bytes from 192.168.56.101: icmp_seq=5 ttl=63 time=0.860 ms
64 bytes from 192.168.56.101: icmp_seq=6 ttl=63 time=0.671 ms
64 bytes from 192.168.56.101: icmp_seq=7 ttl=63 time=1.04 ms
^C
--- 192.168.56.101 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6183ms
```

## ข้อมูล VM Access

### VM IP address

```
devlab@ubuntu:~$ hostname -I
10.0.2.15 192.168.56.101 fd17:625c:f037:2:a00:27ff:fead:11bd
```

### SSH credentials (username เท่านั้น ไม่ใส่ passwords)

VM IP Address: 192.168.56.101

SSH Port: 22

Username: devlab

### API endpoint URLs

Base API URL: http://192.168.56.101:3000

#### Tasks API:

- GET http://192.168.56.101:3000/api/tasks
- GET http://192.168.56.101:3000/api/tasks/:id
- POST http://192.168.56.101:3000/api/tasks
- PUT http://192.168.56.101:3000/api/tasks/:id
- DELETE http://192.168.56.101:3000/api/tasks/:id
- PATCH http://192.168.56.101:3000/api/tasks/:id/next-status

## Task Board - Software Architecture Learning Lab

### บทนำ

รายงานฉบับนี้จัดทำขึ้นเพื่อเปรียบเทียบสถาปัตยกรรมซอฟต์แวร์จำนวน 3 รูปแบบ ได้แก่ Monolithic Architecture (Week 3), Layered Architecture (Week 4) และ Client-Server Architecture (Week 5) โดยอ้างอิงจากประสบการณ์จริงในการพัฒนา ทดสอบ และทำการ Deploy ระบบบนเครื่อง Local และ Virtual Machine (VM)

### ส่วนที่ 1 การเปรียบเทียบสถาปัตยกรรมทั้ง 3 แบบ

ด้านการประเมิน	Monolithic	Layered	Client-Server	คำอธิบายเพิ่มเติม
จำนวนไฟล์หลัก	น้อย	ปานกลาง	มาก	แยกส่วนมากขึ้น ไฟล์เพิ่มขึ้น
ความซับซ้อน	ต่ำ	ปานกลาง	สูง	Client-Server ต้องจัดการ Network และ IP
สภาพแวดล้อม	เครื่องเดียว	เครื่องเดียว	2 เครื่อง	Local + VM
การ Deploy	ง่ายที่สุด	ง่าย	ซับซ้อนมาก	ต้องตั้งค่า SSH / PM2 / Firewall
ประสิทธิภาพ	สูง	สูง	ปานกลาง	มี Latency จาก Network
Maintainability	ต่ำ	สูง	สูง	Layered ช่วยจัดระเบียบโค้ด
Scalability	ต่ำ	ปานกลาง	สูง	รองรับผู้ใช้งานจำนวนมาก
Security	ต่ำ	ปานกลาง	สูง	ต้องออกแบบ API อย่างรอบคอบ

## ส่วนที่ 2 การประเมินคุณลักษณะเชิงคุณภาพ (Quality Attributes)

Attribute	Monolithic	Layered	Client-Server	เหตุผลในการประเมิน
Performance	9	8	7	มีค่า Latency จากเครือข่าย
Scalability	3	6	9	เพิ่มเครื่องฟัง Server ได้
Reliability	5	7	9	ใช้ PM2 ดูแล Process ได้
Security	4	6	9	ต้องออกแบบ Network และ API
Maintainability	5	9	8	Layered ทำให้โค้ดอ่านง่าย

## ส่วนที่ 3 การเลือกสถาปัตยกรรมให้เหมาะสมกับสถานการณ์

- ระบบ Startup ขนาดเล็ก → เหมาะกับ Monolithic
- ระบบองค์กรขนาดกลาง → เหมาะกับ Layered
- ระบบที่ต้องรองรับผู้ใช้งานจำนวนมาก → เหมาะกับ Client-Server

## ส่วนที่ 4 ประสบการณ์จากการ Deploy จริง

### ปัญหาที่พบ:

- พอร์ตชนกัน (EADDRINUSE)
- CORS บล็อกการเชื่อมต่อ
- โปรแกรม VSCode Remote หลุดการเชื่อมต่อ
- ต้องตั้งค่า HOST = 0.0.0.0
- ต้องใช้ PM2 จัดการ Process

### แนวทางแก้ไข:

- ตรวจสอบพอร์ตด้วย netstat และ lsof
- ตั้งค่า CORS ให้รองรับ Client
- ใช้ PM2 start / restart
- บันทึก process ด้วย pm2 save

## ส่วนที่ 5 บทเรียนที่ได้รับ

1. สถาปัตยกรรมมีผลโดยตรงต่อความซับซ้อนของการ Deploy ระบบ
2. Layered Architecture ช่วยให้โค้ดอ่านง่ายและแก้ไขได้ง่ายขึ้น
3. Client-Server ต้องเข้าใจเรื่อง Network, Firewall และ Process Manager
4. การวางแผนโครงสร้างตั้งแต่ต้นช่วยลดปัญหาในระยะยาว

## สรุปผลการเรียนรู้

จากการทำงานในทั้งสามสัปดาห์ ผู้จัดทำมีความเข้าใจความแตกต่างของสถาปัตยกรรมทั้งในด้านโครงสร้าง การออกแบบ การ Deploy และผลกระทบต่อการพัฒนาและบำรุงรักษาระบบซึ่งเป็นประสบการณ์ที่สามารถนำไปประยุกต์ใช้กับงานจริงในอนาคตได้.