โครงงานเลขที่ วศ.คพ. P002-2/2567

เรื่อง

แพลตฟอร์มสำหรับจัดเก็บข้อมูลโปรเจกต์จบของวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่

โดย

นายธนวินท์ สายทอง รหัส 640610304
นายณฐพงศ์ พงศาวลีศรี รหัส 640610630
นายพิชยุทธ หันชัยเนาว์ รหัส 640610653

โครงงานนี้
เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่
ปีการศึกษา 2567

PROJECT No. CPE P002-2/2567

Project Box: A Platform for Archiving Senior Projects at CMU Engineering

Thanawin Saithong **640610304**

Nathaphong Phongsawaleesri **640610630**

Pichayut Hunchainao **640610653**

A Project Submitted in Partial Fulfillment of Requirements
for the Degree of Bachelor of Engineering
Department of Computer Engineering
Faculty of Engineering
Chiang Mai University
2024

หัวข้อโครงงาน	: แพลตฟอร์มสำหรับจัดเก็บข้อมูลโปรเจกต์จบของวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่	
	: Project Box: A Platform for Archiving Senior Projects at CMU Engineering	
โดย	: นายธนวินท์ สายทอง	รหัส 640610304
	นายณฐพงศ์ พงศาวลีศรี	รหัส 640610630
	นายพิชยุทธ หันชัยเนาว์	รหัส 640610653
ภาควิชา	: วิศวกรรมคอมพิวเตอร์	
อาจารย์ที่ปรึกษา	: ผศ. โดม โพธิกานนท์	
ปริญญา	: วิศวกรรมศาสตรบัณฑิต	
สาขา	: วิศวกรรมคอมพิวเตอร์	
ปีการศึกษา	: 2567	

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่ ได้อนุมัติให้โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต (สาขาวิศวกรรมคอมพิวเตอร์)

..... หัวหน้าภาควิชาวิศวกรรมคอมพิวเตอร์
(รศ.ดร. สันติ พิทักษ์กิจนุกูร)

คณะกรรมการสอบโครงงาน

..... ประธานกรรมการ
(ผศ. โดม โพธิกานนท์)

..... กรรมการ
(อ.ดร. ชินวัตร อิศราดิสัยกุล)

..... กรรมการ
(นิรันดร์ พิสุทธอานนท์)

หัวข้อโครงงาน	: แพลตฟอร์มสำหรับจัดเก็บข้อมูลโปรเจกต์จบของวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่	
	: Project Box: A Platform for Archiving Senior Projects at CMU Engineering	
โดย	: นายธนวินท์ สายทอง	รหัส 640610304
	นายณฐพงศ์ พงศาวลีศรี	รหัส 640610630
	นายพิชยุทธ หันชัยเนาว์	รหัส 640610653
ภาควิชา	: วิศวกรรมคอมพิวเตอร์	
อาจารย์ที่ปรึกษา	: ผศ. โดม โพธิกานนท์	
ปริญญา	: วิศวกรรมศาสตรบัณฑิต	
สาขา	: วิศวกรรมคอมพิวเตอร์	
ปีการศึกษา	: 2567	

บทคัดย่อ

แพลตฟอร์มนี้ถูกพัฒนาขึ้นโดยมีจุดประสงค์หลักเพื่อเก็บรวบรวมโครงงานวิศวกรรม ของนักศึกษาคณะวิศวกรรมศาสตร์มหาวิทยาลัยเชียงใหม่ และใช้ในการสร้างแหล่งข้อมูลกลางที่เป็นระบบและเข้าถึงได้ง่าย นักศึกษาสามารถค้นหาและศึกษาโครงงานของรุ่นพี่ เพื่อนำไปใช้เป็นแรงบันดาลใจหรือตัวอย่างในการทำโครงงาน โดยระบบได้ออกแบบให้สามารถค้นหาได้จากหมวดหมู่ต่างๆ หรือสามารถค้นหาโครงงานที่เกี่ยวห้องได้โดยต้นหาจาก keyword ที่เกี่ยวข้องใน pdf ของโครงงานนั้นๆ

Project Title	:	Project Box: A Platform for Archiving Senior Projects at CMU Engineering	
Name	:	Thanawin Saithong	640610304
		Nathaphong Phongsawaleesri	640610630
		Pichayut Hunchainao	640610653
Department	:	Computer Engineering	
Project Advisor	:	Asst. Prof. Dome Potikanond	
Degree	:	Bachelor of Engineering	
Program	:	Computer Engineering	
Academic Year	:	2024	

ABSTRACT

This platform was developed with the main purpose of collecting engineering projects of students of the Faculty of Engineering, Chiang Mai University, and used to create a centralized and easily accessible source of information. Students can search for and study the projects of their seniors to use as inspiration or examples for doing projects. The system is designed to be searchable by various categories or to search for projects related to the room by searching for relevant keywords in the PDF of that project.

กิตติกรรมประกาศ

โครงงานนี้จะไม่สามารถสำเร็จได้ถ้าไม่ได้ความกรุณาจาก ผศ. โดม โพธิกานนท์ อาจารย์ที่ปรึกษาโครงงาน ที่ได้สละเวลาให้ความช่วยเหลือให้คำแนะนำและสนับสนุนในการทำโครงงานนี้รวมถึงอ.ดร.ชินวัตร อิศราดิสัยกุล และ ผศ.ดร.นิรันดร์ พิสุทธอานนท์ ที่ ให้คำปรึกษาจนทำให้โครงงานเล่มนี้เสร็จ สมบูรณ์ไปได้

นายธนวินท์ สายทอง
นายณฐพงศ์ พงศาวลีศรี
นายพิชยุทธ หันชัยเนาว์
2 กุมภาพันธ์ 2567

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ที่มาของโครงงาน

ที่มาของโครงงานนั้นเกิดจากที่การเก็บโครงงานในปัจจุบันขอภาควิศวกรรมศาสาตร์ยังมีการเก็บกระจัดกระจายผู้ที่สนใจต้องการเข้าถึงโครงงานต้องติดต่อผู้ที่เก็บโครงงานโดยตรง และไม่สามารถเข้าถึงโครงงานได้เอง ทำให้เกิดปัญหาในการเข้าถึงข้อมูลที่ต้องการ

จากสาเหตุที่กล่าวมาข้างต้นทำให้ผู้พัฒนาต้องการพัฒนาแพลตฟอร์มที่เป็นศูนย์กลางการเก็บโครงงานรวมถึงทำให้โครงงานสามารถเข้าถึงได้โดยง่ายโดยการทำให้อยู่ในรูปแบบของ Web Application ที่ช่วยให้ผู้ใช้สามารถอับโหลด จัดเก็บ และค้นหาโครงงานได้อย่างเป็นระบบ

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อสร้างระบบศูนย์กลางที่สามารถรวบรวมและจัดเก็บข้อมูลโปรเจกต์จบของนักศึกษาจากทุกภาควิชาในคณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่
2. ช่วยให้นักศึกษาสามารถค้นหาและศึกษาข้อมูลโปรเจกต์ รุ่นพี่เพื่อใช้อ้างอิงและเป็นแนวทาง
3. เพื่อง่ายต่อการค้นหาโปรเจ็คที่อยู่นอกเหนือจากภาควิชาของตนเอง

1.3 ขอบเขตของโครงงาน

1.3.1 ขอบเขตด้านฮาร์ดแวร์

1. คอมพิวเตอร์หรือโน๊ตบุ๊คที่ใช้สำหรับระบบต้องสามารถเชื่อมต่อสัญญาณอินเทอร์เน็ต

1.3.2 ขอบเขตด้านซอฟต์แวร์

1. เว็บไซต์ไม่รองรับการแสดงผลแบบ Responsive ในขนาดหน้าจอที่เล็ก(เช่นใน โทรศัพท์มือถือ)

1.4 ประโยชน์ที่ได้รับ

1. ช่วยให้นักศึกษาและนักพัฒนาสามารถจัดเก็บและเข้าถึงโปรเจ็กต์ได้ง่าย
2. พิ่มโอกาสในการเรียนรู้จากโปรเจ็กต์ที่มีอยู่แล้ว
3. ช่วยองค์กรหรือมหาวิทยาลัยในการบริหารจัดการโปรเจ็กต์ของนักศึกษาได้อย่างมีประสิทธิภาพ

1.5 เทคโนโลยีและเครื่องมือที่ใช้

1.5.1 Next.js

รูปที่ 1.1: รูปจาก https://medium.com/geekculture/why-should-you-learn-next-js-in-2021-what-are-the-benefits-8292d79bc50c

Next.js[?] คือ JavaScript webapps framework ถูกสร้างขึ้น on top จาก library อย่าง React, Webpack, และ Babel ขึ้นมาอีกที มีจุดเด่นคือ เป็น SSR (server-side rendering) ตั้งแต่ต้น

1.5.1.1 ข้อดีของ Next.js

- สามารถทำ SSR ได้ง่าย
- มีการจัดการ SEO ที่ดี
- Hot reload เวลาเราแก้ไขไฟล์ หน้าเว็บของเราจะถูก refresh โดยอัตโนมัติ
- Project Structure ที่ชัดเจนที่ถูกออกแบบมาให้เรียบร้อยแล้ว
- Routing ด้วยความที่มี project structure การทำ routingจึงสามารถ auto routing ได้

1.5.2 ElysiaJs

รูปที่ 1.2: รูปจาก https://sadewawicak25.medium.com/file-upload-and-security-validation-on-elysia-js-2-d6c57b023441

ElysiaJS[?] คือ Framework ในการพัฒนา API ด้วยภาษา Typescript โดยมีจุดเด่นคือ ความเร็วที่เร็วกว่า Express ถึง 21 เท่า (เนื่องจาก ElysiaJS มีการใช้ Bun เป็น Runtime) และอีกจุดเด่นคือ End-to-end Type Safety หรือชนิดของข้อมูลที่ชัดเจน ทำให้เวลาเราทำงานร่วมกับผู้อื่นสามารถทำได้สะดวกมากยิ่งขึ้น เพราะไม่ต้องมาทะเลาะกันเรื่องชนิดของข้อมูลที่ส่งให้กัน อีกทั้งยังมี Community ที่เติบโตเร็ว

1.5.3 Gin

รูปที่ 1.3: รูปจาก https://www.askme.co.th/article/what-is-docker/

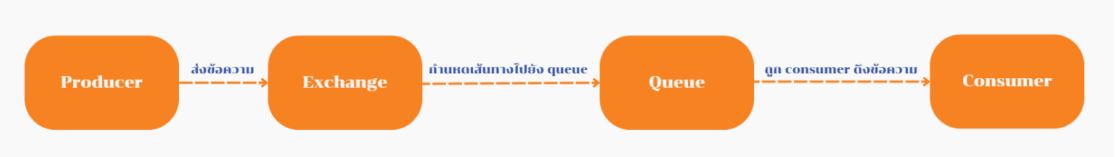
Gin[?] เป็น web framework ที่เขียนด้วยภาษา golang ที่ถูกพัฒนาต่อมาจาก Martini API ที่หยุดพัฒนาไปแล้ว โดย Gin จะใช้ customized httprouter ทำให้มีประสิทธิภาพด้านความเร็วที่สูงมากกว่า Martini ถึง 40 [?]เท่า ทำให้มีperformance กับ productivity ที่ดี

1.5.3.1 Feature สำคัญของ Gin

- **JSON validation** สามารถแปลงและตรวจสอบ JSON ของ HTTP request
- **Routes grouping** จัดกลุ่ม routes ของ request ว่า request ไหนต้องมีการ authorization หรือไม่จำเป็นต้องมี การแยก request ด้วย version ของ API โดยสามารถจัดกลุ่มได้อย่างไม่จำกัดและไม่กระทบกับประสิทธิภาพ
- **Middleware support** incoming HTTP request จะถูกจัดการด้วย chain ของ middleware และ action สุดท้าย
- **Rendering build-in** ง่ายสำหรับสร้าง API ที่ render เป็น JSON, XML และ HTML
- **Error management** สามารถจัดการ error ที่เกิดขึ้นในระดับ application และ HTTP ได้

1.5.4 RabbitMQ

RabbitMQ [?]ซอฟต์แวร์ที่เป็นตัวกลางรับส่งข้อความระหว่างแอปพลิเคชันต่างๆ ผู้ไปรับรับข้อความจากผู้ส่ง (แอปพลิเคชันหนึ่ง) เก็บไว้รอการคัดแยก และส่งต่อให้ผู้รับ (แอปพลิเคชันอีกแอปพลิเคชันหนึ่ง) เหมาะสำหรับการทำแอปพลิเคชันที่ต้องมีการจัดคิวในการส่งข้อความ ระบบที่เป็นไมโครเซอร์วิสเอาไว้สื่อสารกันได้อย่างมีประสิทธิภาพ ทำให้สามารถแบ่งงานขนาดใหญ่เป็นงานย่อยๆ และส่งไปยังระบบอื่นๆ เพื่อประมวลผลได้นั่นเอง



รูปที่ 1.4: รูปจาก https://www.borntodev.com/2024/06/09/rabbitmq-nodejs/

1.5.4.1 คำศัพท์ต่างๆที่เกี่ยวข้อง

- **Producer** คือ ผู้ส่งข้อความ
- **Consumer** คือ ผู้รับข้อความ
- **Queue** คือ คิวข้อความ
- **Exchange** คือ ตัวกลางในการส่งข้อความ
- **Binding** คือ การเชื่อมต่อระหว่าง Exchange กับ Queue
- **Channel** คือ ช่องสื่อสารระหว่าง Producer และ Consumer
- **Connection** คือ การเชื่อมต่อระหว่าง RabbitMQ กับ Producer และ Consumer

1.5.5 Docker

รูปที่ 1.5: รูปจาก https://www.docker.com/

Docker[?]เป็นแพลตฟอร์มโอเพนซอร์สที่ช่วยในการสร้าง ทดสอบ และปรับใช้แอปพลิเคชันในรูปแบบของคอนเทนเนอร์ คอนเทนเนอร์เป็นสภาพแวดล้อมที่แยกจากกันที่สามารถรันแอปพลิเคชันได้ โดยที่ไม่ต้องกังวลเกี่ยวกับการกำหนดค่าหรือการติดตั้งซอฟต์แวร์เพิ่มเติมในระบบปฏิบัติการหลักของเซิร์ฟเวอร์ที่ใช้

สิ่งที่ทำให้ Docker แตกต่างจากเทคโนโลยีอื่นๆ เช่น Virtual Machines (VMs) คือ Docker ใช้ Kernel ของระบบปฏิบัติการเดียวกันในการรันคอนเทนเนอร์แต่ละตัว ทำให้มีประสิทธิภาพในการใช้งานทรัพยากรสูงขึ้น และทำให้คอนเทนเนอร์ใช้เวลาในการเริ่มต้นที่รวดเร็ว

1.5.6 องค์ประกอบหลักของ Docker

- **Docker Engine** เป็นซอฟต์แวร์ที่รันอยู่เบื้องหลังซึ่งทำหน้าที่สร้างและจัดการคอนเทนเนอร์ในระบบ มันมีองค์ประกอบย่อย 2 ส่วนที่สำคัญ ได้แก่

    1. Docker Daemon (dockerd): เป็นโปรแกรมหลักที่รันอยู่เบื้องหลังและรับคำสั่งจาก Docker Client ผ่าน API โดย Daemon จะจัดการทุกอย่างที่เกี่ยวข้องกับคอนเทนเนอร์ เช่น การสร้าง, การเริ่มต้น, การหยุด, และการลบคอนเทนเนอร์
    2. Docker CLI (docker): คือส่วนที่ผู้ใช้ใช้ในการสั่งการ Docker ผ่านคำสั่งต่างๆ ผ่าน Command Line Interface เช่น การสร้างคอนเทนเนอร์ (docker run), การสร้างอิมเมจ (docker build), และการจัดการเครือข่าย (docker network)

- **Docker Image** คือไฟล์แบบคงที่ที่บรรจุโค้ดแอปพลิเคชันและทุกสิ่งที่แอปพลิเคชันนั้นต้องการในการรัน เช่น ไลบรารี, การตั้งค่า, และไฟล์ระบบ Image ถูกสร้างจากไฟล์ที่เรียกว่า Dockerfile ซึ่งเป็นไฟล์ที่กำหนดขั้นตอนในการติดตั้งและตั้งค่าแอปพลิเคชัน

  Dockerfile Syntax ที่ใช้ในการสร้าง Image มีคำสั่งที่เป็นลำดับขั้นตอน ยกตัวอย่างเช่น

    1. FROM: ระบุ Image เบื้องต้น เช่น Ubuntu, Alpine หรือ Node.js
    2. RUN: รันคำสั่งใน Image เช่น การติดตั้งแพ็คเกจ
    3. COPY/ADD: คัดลอกไฟล์จากโฮสต์เข้าสู่ Image
    4. CMD/ENTRYPOINT: กำหนดคำสั่งที่รันเมื่อคอนเทนเนอร์เริ่มทำงาน

- **Docker Container** คือ สิ่งที่ถูกสร้างจาก Docker Image และเป็นสภาพแวดล้อมที่แยกจากกันที่สามารถรันแอปพลิเคชันได้โดยมีคุณสมบัติดังนี้

-
    1. แยกการทำงานจากระบบปฏิบัติการโฮสต์ แต่ยังใช้เคอร์เนลร่วมกัน
    2. สามารถสร้าง, ลบ, หยุด, และรีสตาร์ทได้อย่างง่ายดาย
    3. สามารถแชร์ทรัพยากรเครือข่ายและไฟล์ระหว่างคอนเทนเนอร์ต่างๆ ได้

1.5.6.1 ความแตกต่างระหว่าง Docker กับ Virtual Machines

ตารางที่ 1.1: เปรียบเทียบคุณสมบัติของ Docker และ Virtual Machine (VM)

คุณสมบัติ	Docker	Virtual Machine (VM)
การใช้เคอร์เนล	ใช้เคอร์เนลของระบบปฏิบัติการโฮสต์	จำลองระบบปฏิบัติการเต็มรูปแบบ
ขนาดไฟล์	ขนาดเล็ก	ขนาดใหญ่
เวลาเริ่มต้น	เริ่มต้นได้อย่างรวดเร็ว	ใช้เวลามากขึ้น
การใช้ทรัพยากร	ใช้ทรัพยากรน้อย	ใช้ทรัพยากรมาก
ความยืดหยุ่น	เหมาะสำหรับแอปพลิเคชันแบบไมโครเซอร์วิส	เหมาะสำหรับการจำลองระบบขนาดใหญ่
การแยกทรัพยากร	แยกการทำงานในระดับแอปพลิเคชัน	แยกระบบปฏิบัติการทั้งหมด

โดยรวมแล้ว Docker เหมาะสำหรับการรันแอปพลิเคชันที่ต้องการความยืดหยุ่นในการปรับใช้และทรัพยากรที่มีข้อจำกัด ในขณะที่ VM เหมาะกับการรันระบบที่ต้องการการแยกสภาพแวดล้อมอย่างสมบูรณ์ เช่น การรันหลายระบบปฏิบัติการในเครื่องเดียวกัน

1.5.7 Nginx Proxy Manager (NPM)

รูปที่ 1.6: https://sparwan.com/en/blogs/news/tutoriel-installation-de-nginx-proxy-manager

เป็น open-source ที่ออกแบบมาเพื่อช่วยให้การจัดการพร็อกซีของ Nginx, SSL ,Access Lists และอื่นๆ โดยสร้างขึ้นมาเพื่อจุดประสงค์ให้ง่ายต่อการใช้งานโดยมี Dashboard ให้ เหมาะสำหรับผู้ใช้ที่ไม่เชี่ยวชาญการใช้ Nginx ผ่าน CLI นอกจากนี้ ยังรองรับ SSL ฟรีผ่าน Let's Encrypt รวมถึงสามารถใช้งานได้บน Docker และรองรับการใช้งานหลาย User อีกด้วย

1.5.8 PostgreSQL

รูปที่ 1.7: รูปจาก https://www.fullstackpython.com/postgresql.html

PostgreSQL คือ ระบบฐานข้อมูลแบบ Relational Database Management System (RDBMS) ที่เป็นโอเพนซอร์ส มีความสามารถในการจัดการข้อมูลที่มีความซับซ้อน รวมถึงการจัดการข้อมูลที่มีความสัมพันธ์กัน

1.5.9 Figma

รูปที่ 1.8: รูปจาก https://www.figma.com/

Figma เป็นเครื่องมือออกแบบ UI/UX ที่ใช้งานผ่านเว็บ ทำให้สามารถทำงานร่วมกันได้ง่าย สามารถสร้าง Wireframe, Mockup, Prototype และ Design System ได้

1.6 แผนการดำเนินงาน

ขั้นตอนการดำเนินงาน	มิ.ย. 2563	ก.ค. 2563	ส.ค. 2563	ก.ย. 2563	ต.ค. 2563	พ.ย. 2563	ธ.ค. 2563	ม.ค. 2564	ก.พ. 2564
ศึกษาค้นคว้า									
ชิล									
เผา									
ทดสอบ									

1.7 บทบาทและความรับผิดชอบ

อธิบายว่าในการทำงาน นศ. มีการกำหนดบทบาทและแบ่งหน้าที่งานอย่างไรในการทำงาน จำเป็นต้องใช้ความรู้ใดในการทำงานบ้าง

1.8 ผลกระทบด้านสังคม สุขภาพ ความปลอดภัย กฎหมาย และวัฒนธรรม

แนวทางและโยชน์ในการประยุกต์ใช้งานโครงงานกับงานในด้านอื่นๆ รวมถึงผลกระทบในด้านสังคมและสิ่งแวดล้อมจากการใช้ความรู้ทางวิศวกรรมที่ได้

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

การทำโครงงาน เริ่มต้นด้วยการศึกษาค้นคว้า ทฤษฎีที่เกี่ยวข้อง หรือ งานวิจัย/โครงงาน ที่เคยมีผู้นำเสนอไว้แล้ว ซึ่งเนื้อหาในบทนี้ก็จะเกี่ยวกับการอธิบายถึงสิ่งที่เกี่ยวข้องกับโครงงาน เพื่อให้ผู้อ่านเข้าใจเนื้อหาในบทถัดๆ ไปได้ง่ายขึ้น

2.1 ระบบฐานข้อมูล (Database System)

ระบบฐานข้อมูล (Database System) หมายถึงโครงสร้างสารสนเทศที่ประกอบด้วย รายละเอียดของข้อมูลที่เกี่ยวข้องกันที่จะนำมาใช้ในระบบต่าง ๆ ร่วมกัน ซึ่งผู้ใช้สามารถจัดการกับ ข้อมูลได้ในลักษณะต่าง ๆ ทั้งเพิ่ม ลบ หรือแก้ไขตลอดจนการเรียกดูข้อมูล ส่วนใหญ่จะเป็นการประยุกต์นำเอาระบบคอมพิวเตอร์เข้ามาช่วยในการจัดการฐานข้อมูล
ระบบฐานข้อมูล มีคำศัพท์ต่างๆที่เกี่ยวข้องดังนี้

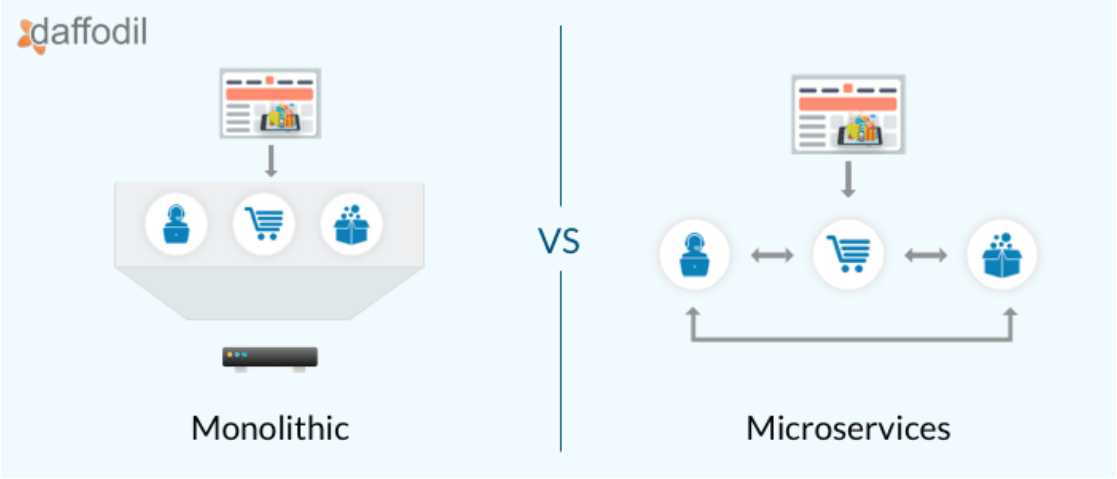
1. เอนทิตี้ (Entity) หมายถึงสิ่งที่เราสนใจจะเก็บข้อมูล เช่น นักศึกษา อาจารย์ วิชาการ หรือห้องเรียน
2. แอตทริบิวต์ (Attribute) หมายถึงคุณสมบัติของเอนทิตี้ เช่น ชื่อ นามสกุล หรือรหัสนักศึกษา
3. ความสัมพันธ์ (Relationship) หมายถึงความสัมพันธ์ระหว่างเอนทิตี้ โดยที่เอนทิตี้หนึ่งสามารถมีความสัมพันธ์กับเอนทิตี้อีกเอนทิตี้หนึ่งได้ เช่น นักศึกษาสามารถลงทะเบียนเรียนในหลายวิชา และวิชาใด ๆ ก็สามารถมีนักศึกษาหลายคนลงทะเบียนเรียนได้
4. คีย์หลัก (Primary Key) หมายถึงคีย์ที่ใช้เพื่อระบุเอนทิตี้นั้น ๆ อย่างชัดเจน และไม่สามารถซ้ำกันได้
5. คีย์นอก (Foreign Key) หมายถึงคีย์ที่เป็นคีย์หลักของเอนทิตี้หนึ่ง และเป็นคีย์ที่อยู่ในเอนทิตี้อีกเอนทิตี้หนึ่ง

2.2 ไมโครเซอร์วิส (Microservices)

Microservice[?] หรือ Microservice Architecture คือสถาปัตยกรรมการออกแบบ Service หรือก็คือออกแบบซอฟต์แวร์ โดยการที่ในชื่อมีคำว่า Micro นำหน้าอยู่ก็เพราะว่าเป็นการออกแบบที่ทำให้ Service มีขนาดเล็กเพื่อแก้ไขจุดด้อยของสถาปัตยกรรมการออกแบบอื่นๆ

2.2.1 Monolithic VS Microservice

หาก Microservice เป็นการออกแบบ Service ให้มีขนาดเล็ก การจะเทียบให้เห็นภาพชัดเจนที่สุดก็ต้องเทียบกับ Monolithic ที่เป็นระบบที่มีขนาดใหญ่ โดย Monolithic จะเป็นระบบที่มีการทำงานทั้งหมดอยู่ใน Service เดียว



รูปที่ 2.1: รูปจาก nsights.daffodilsw.com

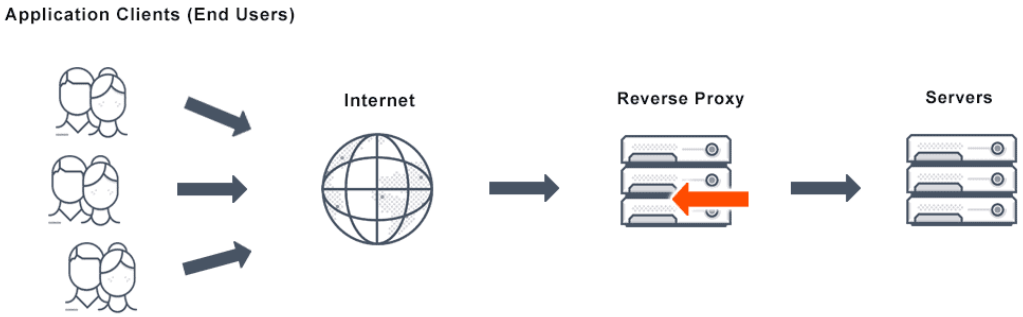
2.2.1.1 ความแตกต่างระหว่าง Monolithic และ Microservice

- **Monolithic** เป็นชื่อของสถาปัตยกรรมการออกแบบซอฟต์แวร์หรือ Service ที่มีคนใช้งานเป็นจำนวนมากและมีมาอย่างยาวนาน โดยเป็นลักษณะของระบบที่การทำงานทุกอย่างจะรวมอยู่ในกลุ่มก้อนเดียวกัน และใช้งาน Database เดียวกัน (อย่างในภาพจะเห็นว่าเป็นเว็บไซต์ขายสินค้าที่มีฟังก์ชันจัดการผู้ใช้, ตะกร้าสินค้า และการส่งสินค้า รวมอยู่ด้วยกัน และใช้ฐานข้อมูลเดียวกัน)
- **Microservice** จะออกแบบโดยแยกการทำงานที่รวมกันเป็นก้อนใหญ่ๆของแบบ Monolithic ออกมาให้เล็กลงโดยอาจจะแยกตามบริการหรือตามฟังก์ชันการทำงานเลยก็ได้ (จากในภาพฟังก์ชันทั้งสามอย่างจะแยกออกจากกัน และไม่ได้ใช้ฐานข้อมูลเดียวกันในการเก็บข้อมูลอีกต่อไป เพราะแต่ละฟังก์ชันหรือบริการที่แยกออกมามีฐานข้อมูลเป็นของตัวเอง และสามารถติดต่อกันได้ผ่าน API)

2.2.1.2 ข้อดีและข้อเสียของ Microservice

- ข้อดี
    1. การทำงานหลักแต่ละส่วนของระบบ ถ้าเป็นไปได้ควรแยกออกเป็น service แต่ละอัน เช่น จัดการสินค้า กับจัดการการซื้อสินค้าก็แยกกันไปเลย
    2. มีที่เก็บข้อมูลของตัวเอง
- ข้อเสีย
    1. การจัดการระบบที่มีหลาย service อาจจะทำให้การจัดการระบบทำได้ยากขึ้น
    2. การทำงานของระบบที่แยกออกมาอาจจะทำให้การทำงานของระบบช้าลง

2.3 Reverse Proxy



รูปที่ 2.2: รูปจาก https://www.vmware.com/topics/reverse-proxy-server

Reverse Proxy[?] เป็นเซิร์ฟเวอร์พร็อกซีที่ทำหน้าที่เป็นตัวกลางระหว่าง ไคลเอนต์ (Client) และ เซิร์ฟเวอร์ต้นทาง (Origin Server) โดยไคลเอนต์จะส่งคำขอ (Request) ไปยัง Reverse Proxy และจากนั้น Reverse Proxy จะส่งคำขอนั้นไปยังเซิร์ฟเวอร์ที่เหมาะสม แล้วรับคำตอบกลับมาเพื่อส่งต่อให้ไคลเอนต์

2.3.1 ประโยชน์ของการทำ Reverse Proxy

การทำ Reverse Proxy มีประโยชน์หลายอย่างโดยประโยชน์หลักๆ จะมีดังนี้

- ป้องกันการโจมตี DDoS
- ซ่อน IP จริงของเซิร์ฟเวอร์
- ใช้ SSL/TLS เพื่อเข้ารหัสข้อมูล
- กระจายโหลด (Load Balancing) ไปยังเซิร์ฟเวอร์หลายตัว
- แคชข้อมูล (Caching) ลดภาระของเซิร์ฟเวอร์ต้นทาง

2.4 Hypertext Transfer Protocol (HTTP)

HTTP (Hypertext Transfer Protocol) เป็นโปรโตคอลสื่อสารที่ใช้ในการส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์บนเครือข่ายอินเทอร์เน็ต โดย HTTP มีหน้าที่เป็นตัวกลา และเบราว์เซอร์ (web browsers) หรือแอปพลิเคชันอื่น ๆ ที่ใช้เครือข่ายอินเทอร์เน็ต งในการร้องขอและส่งข้อมูลระหว่างเว็บไซต์ (web servers)

2.5 Application Programing Interface (API)

คือ การเชื่อมต่อโปรแกรมประยุกต์ ในบริบทนี้ คำว่า ”Application” หมายถึงทุกซอฟต์แวร์ที่มีฟังก์ชันชัดเจน และ ”Interface” ก็คือตัวประสานหรือเป็นเหมือนสัญญาที่กำหนดวิธีการสื่อสารกันระหว่าง ”Application” API ทำงานใน 4 รูปแบบด้วยกัน โดยขึ้นอยู่กับเวลาและสาเหตุที่สร้าง API

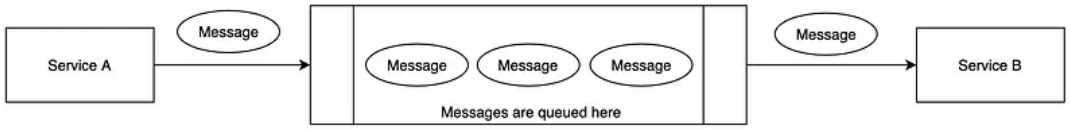
1. SOAP API - Simple Object Access Protocol (โปรโตคอลการเข้าถึงอ็อบเจกต์อย่างง่าย) ไคลเอ็นต์และเซิร์ฟเวอร์จะแลกเปลี่ยนข้อความโดยใช้ XML ซึ่งเป็น API ที่มีความยืดหยุ่นน้อยซึ่งเคยได้รับความนิยมมากกว่านี้ในอดีต
2. RPC API - Remote Procedure Call (การเรียกใช้กระบวนการระยะไกล) ไคลเอ็นต์ดำเนินการฟังก์ชัน (หรือกระบวนการ) หนึ่งๆ บนเซิร์ฟเวอร์ และเซิร์ฟเวอร์ส่งผลลัพธ์กลับไปยังไคลเอ็นต์
3. Websocket API - Web API สมัยใหม่ที่ใช้อ็อบเจกต์ JSON ในการส่งข้อมูล WebSocket API รองรับการสื่อสารสองทางระหว่างแอปไคลเอ็นต์และเซิร์ฟเวอร์ เซิร์ฟเวอร์สามารถส่งข้อความเรียกกลับไปยังไคลเอ็นต์ที่เชื่อมต่อ จึงทำให้มีประสิทธิภาพมากกว่า REST API
4. REST API - API ที่ได้รับความนิยมและยืดหยุ่นที่สุดที่พบในเว็บไซต์ปัจจุบัน ไคลเอ็นต์ส่งคำขอไปยังเซิร์ฟเวอร์เป็นข้อมูล เซิร์ฟเวอร์ใช้ข้อมูลอินพุตจากไคลเอ็นต์นี้เพื่อเริ่มต้นฟังก์ชันภายในและส่งคืนข้อมูลเอาต์พุตกลับไปยังไคลเอ็นต์

2.6 Json Web Token (JWT)

JWT[**?**]เป็นมาตรฐานแบบเปิด (RFC 7519) ที่กำหนดรูปแบบข้อมูลที่มีขนาดเล็กและสามารถตรวจสอบได้ในตัวเอง เพื่อใช้ในการส่งข้อมูลระหว่างฝ่ายต่างๆ อย่างปลอดภัยในรูปแบบของ JSON ข้อมูลนี้สามารถตรวจสอบและเชื่อถือได้ เนื่องจากมีการลงนามดิจิทัล (digitally signed)

2.7 Message Queue

Message Queue[**?**] (เรียกย่อๆว่า MQ) เป็นส่วนประกอบหนึ่งที่สำคัญในการออกแบบระบบขนาดใหญ่ โดย MQ ทำหน้าที่ในการรับ Message จากต้นทาง เก็บรักษาไว้ตามลำดับที่รับ Message เข้ามา และเปิดให้ปลายทาง มาหยิบ Message ออกไปทีละ 1 Message (หรือมากกว่า) ตามลำดับที่กำหนดไว้ ตามประเภทของ Queue นั้นๆโดยที่ MQ นั้นเอง ก็มีหลากหลายประเภท หลายยี่ห้อผู้ผลิต และหลากหลายลักษณะการใช้งาน แต่ในพื้นฐานแล้ว ก็จะมีลักษณะเหมือนกัน ลองมาดูภาพการทำงานแบบคร่าวๆกันครับ

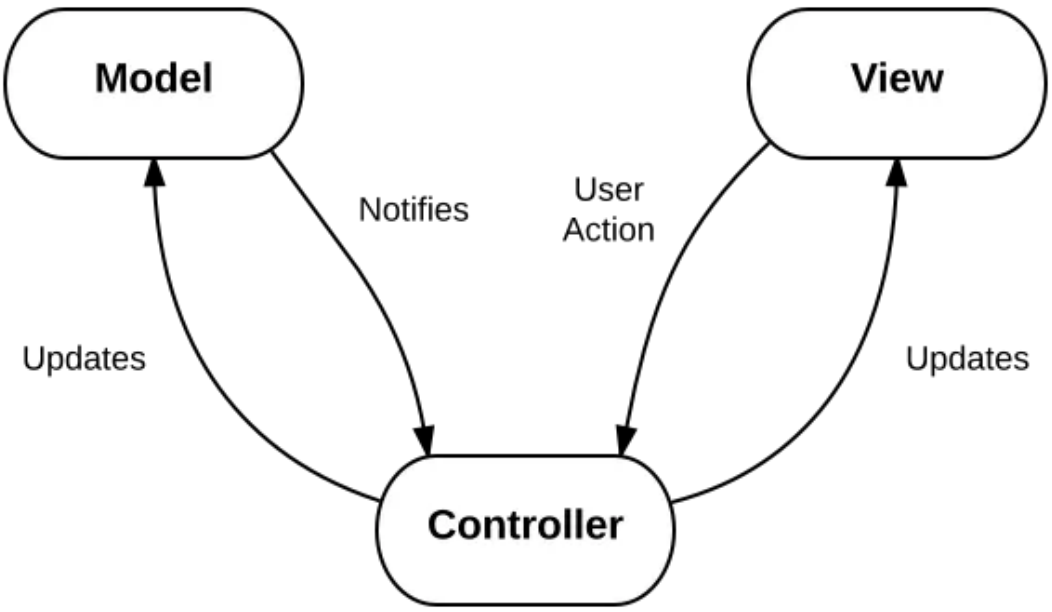


รูปที่ 2.3:
รูปจาก https://panitw.medium.com/การใช้งาน-message-queue-pattern-65b90a6c4364

จากภาพ เราเรียก Service A ว่า Producer (ผู้ผลิต) และ Service B ว่า Consumer (ผู้บริโภค) โดย Producer จะสร้าง Message และส่งเข้าไปรอไว้ในคิว เพื่อให้ Consumer มาหยิบข้อความไปใช้ ข้อความที่ส่งเข้าไปใน MQ ก็จะถูกเก็บรักษาเอาไว้รอให้ Consumer มาหยิบโดยความเร็วข้อความที่ส่งเข้ามา อาจจะไม่เท่ากับความเร็วของข้อความที่ถูกดึงออกไป เช่นถ้า Producer ส่งข้อความทีละ 10 ข้อความต่อนาที แต่ Consumer อ่านไปทำได้ทีละ 1 ข้อความต่อนาที ก็จะทำให้มีข้อความค้างใน MQ เพิ่มขึ้นเรื่อยๆ 9 ข้อความต่อนาที

2.8 Model-View-Controller (MVC)

MVC หรือ Model-View-Controller[**?**] เป็นรูปแบบการออกแบบโปรแกรมที่มีการแบ่งส่วนการทำงานออกเป็น 3 ส่วน ได้แก่ Model, View และ Controller เป็น Software Design Pattern หรือแนวทางการออกแบบซอฟต์แวร์รูปแบบหนึ่งของการเขียนซอฟต์แวร์



รูปที่ 2.4: รูปจาก https://commons.wikimedia.org/wiki/File:MVC-basic.svg

- **Model** คือส่วนที่เก็บข้อมูล และจัดการกับข้อมูล โดย Model จะไม่รู้จัก View จะส่งข้อมูลกลับไปยัง Controller เท่านั้น
- **View** คือส่วนที่แสดงผลข้อมูล และรับข้อมูลจากผู้ใช้ โดย View จะไม่รู้จัก Model แต่จะส่งข้อมูลกลับไปยัง Controller เท่านั้น
- **Controller** คือส่วนที่ควบคุมการทำงานของระบบ โดย Controller จะรับข้อมูลจาก View และส่งข้อมูลไปยัง Model และจาก Model ก็จะส่งข้อมูลกลับไปยัง View

จุดประสงค์ของ MVC

- เพื่อแยกโค้ดออกเป็นส่วน ทำให้เราเปลี่ยนแปลงบางส่วนของโค้ดได้ โดยไม่กระทบกับส่วนอื่น ทำให้ Maintenance โค้ดทีหลังได้ง่าย
- ทำให้โค้ดสามารถถูกเขียนพร้อมกันโดยโปรแกรมเมอร์หลายคนได้
- สามารถนำโค้ดมาใช้ซ้ำได้

2.9 Elasticsearch

เนื่องด้วยการ Search project ด้วย full text search บน SQL Database ไม่ตอบโจทย์หากข้อมูลมีจำนวนมากและต้องการค้นหาจาก Text ที่มีความยาวมาก Elasticsearch จึงเข้ามาช่วยแก้ปัญหาในส่วนนี้ โดย Elastic Search คือ Search Engine ที่ได้รับความนิยมอย่างแพร่หลาย ถูกออกแบบมาให้สามารถจัดเก็บ ค้นหา และวิเคราะห์ข้อมูลขนาดใหญ่ได้อย่างรวดเร็วและมีประสิทธิภาพ โดยใช้โครงสร้างแบบ Distributed (กระจายศูนย์) และ RESTful API เพื่อให้สามารถเข้าถึงและจัดการข้อมูลได้ง่าย คุณสมบัติของ Elastic search มีดังนี้ Full text search รองรับการค้นหาข้อมูลแบบเต็มรูปแบบและแม่นยำ, Distributed Architecture ทำงานแบบกระจายข้อมูลที่เรียกว่า sharding เพื่อรองรับโหลดขนาดใหญ่และเพิ่มประสิทธิภาพ, Scalability สามารถขยายขนาดได้ง่ายทั้งในรูปแบบ vertical scaling และ horizontal scaling , Near Real-Time Search สามารถค้นหาและดึงข้อมูลที่เพิ่งถูกเพิ่มเข้ามาได้อย่างรวดเร็ว, Integration รองรับการเชื่อมต่อกับ Logstash, Kibana และ Beats เพื่อสร้าง solution สำหรับการวิเคราะห์ข้อมูลและการ Monitor

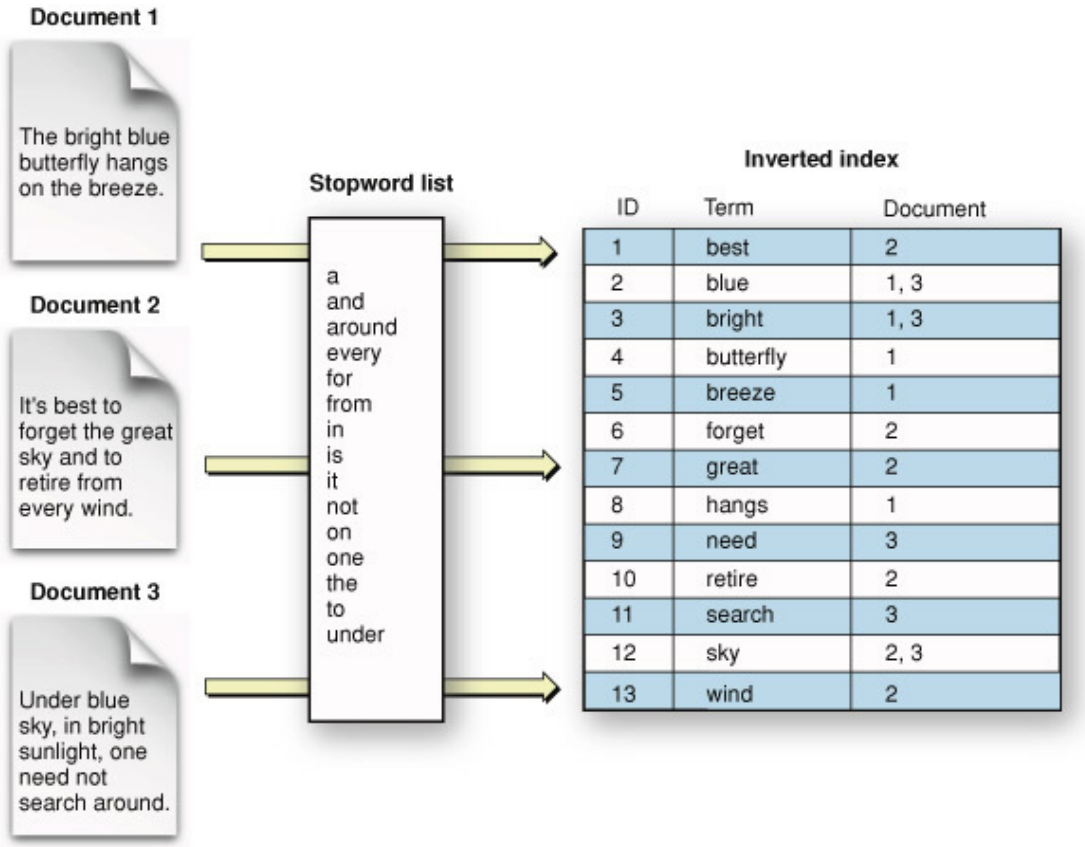
2.10 Inverted Index

Inverted Index เป็นโครงสร้างข้อมูลที่ถูกใช้ใน Elasticsearch เพื่อเพิ่มประสิทธิภาพในการค้นหาข้อมูลแบบ Full-Text Search โดยหลักการของ Inverted Index คือการสร้างดัชนีที่ทำให้สามารถค้นหาคำในเอกสารขนาดใหญ่ได้อย่างรวดเร็ว โครงสร้างของ Inverted Index มีดังนี้:

- extbfTokenization ข้อความจะถูกแยกออกเป็นคำย่อยๆ (Token) เช่น ”Elasticsearch is fast” จะแยกออกเป็น [”elasticsearch”, ”is”, ”fast”].
- extbfNormalization คำจะถูกปรับรูปแบบ เช่น เปลี่ยนเป็นตัวพิมพ์เล็กทั้งหมด หรือกำจัดเครื่องหมายพิเศษ.

- extbfPosting List สร้างดัชนีโดยเก็บตำแหน่งของคำที่ปรากฏในเอกสาร เช่น ”fast” อาจอยู่ในเอกสารที่ 1, 2 และ 5.

การใช้ Inverted Index ทำให้ Elasticsearch สามารถค้นหาข้อมูลได้เร็วขึ้นมากเมื่อเทียบกับการสแกนข้อมูลทั้งหมดแบบดั้งเดิมใน SQL Database.



รูปที่ 2.5: ตัวอย่างโครงสร้างของ Inverted Index

2.11 Kibana

Kibana เป็นเครื่องมือสำหรับแสดงผลและวิเคราะห์ข้อมูลที่ถูกจัดเก็บใน Elasticsearch โดยมีอินเทอร์เฟซแบบกราฟิกที่ช่วยให้ผู้ใช้สามารถสร้างแดชบอร์ด (Dashboard), ค้นหา, วิเคราะห์ และแสดงข้อมูลในรูปแบบต่างๆ ได้อย่างง่ายดาย Kibana มักถูกใช้ร่วมกับ Elasticsearch ในงานที่ต้องจัดเก็บ ค้นหา วิเคราะห์ และแสดงผลข้อมูล เช่น:

- การตรวจสอบเซิร์ฟเวอร์หรือแอปพลิเคชัน (Application Monitoring)
- ระบบค้นหาในเว็บไซต์หรือเอกสาร (Search Systems)
- การสร้างแดชบอร์ดสรุปข้อมูลธุรกิจ (Business Analytics)
- การตรวจจับภัยคุกคาม (Security Monitoring)

2.12 Spring Boot

Spring Boot เป็นเฟรมเวิร์กที่พัฒนาต่อยอดจาก Spring Framework โดยมีจุดเด่นคือทำให้การตั้งค่าซับซ้อนของ Spring ง่ายขึ้น และช่วยให้พัฒนาเว็บแอปพลิเคชัน หรือ REST API ได้เร็วขึ้น คุณสมบัติของ Spring Boot มีดังนี้:

- **Auto Configuration** – ตั้งค่าเริ่มต้นอัตโนมัติ เช่น Database, Security, Logging
- **Embedded Server** – มี Web Server ในตัว (Tomcat, Jetty, Undertow)
- **Spring Boot Starter** – ใช้ dependency สำเร็จรูป ทำให้ง่ายต่อการใช้งานเครื่องมือต่างๆ เช่น พัฒนาเว็บแอปและ REST API
- **Spring Boot Actuator** – ใช้ตรวจสอบสถานะระบบ เช่น health check
- **Microservices Ready** – รองรับการพัฒนา Microservices ได้ง่าย

2.13 Hibernate

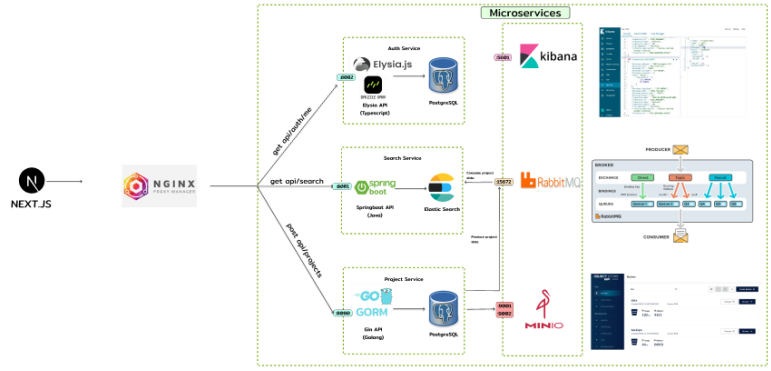
Hibernate เป็นเฟรมเวิร์กที่ใช้สำหรับ ORM (Object-Relational Mapping) ใน Java ซึ่งช่วยให้สามารถทำงานกับฐานข้อมูลได้สะดวกขึ้นโดยไม่ต้องเขียน SQL โดยตรง คุณสมบัติหลักของ Hibernate มีดังนี้:

- **Mapping Object กับ Database** – แปลง Java Object ให้สามารถบันทึกลงฐานข้อมูลได้อัตโนมัติ
- **HQL (Hibernate Query Language)** – ภาษา Query ที่สามารถใช้แทน SQL และเป็นอิสระจากฐานข้อมูล
- **Lazy Loading** – โหลดข้อมูลเฉพาะส่วนที่จำเป็นเพื่อลดการใช้ทรัพยากร
- **Transaction Management** – รองรับการทำงานกับ Transaction และ ACID properties
- **Database Independence** – ใช้งานได้กับหลายฐานข้อมูล เช่น MySQL, PostgreSQL, Oracle

บทที่ 3
โครงสร้างและขั้นตอนการทำงาน

3.1 สถาปัตยกรรมระบบ

โครงงานนี้ได้ออกแบบสถาปัตยกรรมของระบบเป็นแบบ Microservices โดยบทนี้จะกล่าวถึงโครงสร้างของระบบทั้งหมด และอธิบายถึงแต่ละส่วนของระบบ โดยระบบจะถูกแบ่งออกเป็นส่วนย่อยๆ แต่ละส่วนจะมีหน้าที่และความรับผิดชอบในการทำงานที่แตกต่างกัน



รูปที่ 3.1: รูปภาพแสดงสถาปัตยกรรมของระบบ

จากรูปที่ 3.1 แสดงถึงสถาปัตยกรรมของระบบที่ถูกแบ่งออกเป็นส่วนย่อยๆ ดังนี้

1. **Frontend** ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการแสดงผลของระบบ โดยส่วนนี้จะถูกพัฒนาด้วย Next.js
2. **Reverse Proxy** ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการจัดการการเชื่อมต่อระหว่าง Frontend และ Backend โดยส่วนนี้จะถูกพัฒนาด้วย Nginx Proxy Manager(NPM)
3. **Backend** ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการจัดการข้อมูลและการทำงานของระบบ โดยส่วนนี้จะถูกแบ่งออกเป็นส่วนย่อยๆ ดังนี้
    - **Auth Service** ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการจัดการข้อมูลของผู้ใช้งาน โดยส่วนนี้จะถูกพัฒนาด้วย Typescript Elysia.js
    - **Project Service** ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการจัดการข้อมูลของโครงงาน โดยส่วนนี้จะถูกพัฒนาด้วย Go Gin
    - **Search Service** ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการค้นหาข้อมูลของโครงงาน โดยส่วนนี้จะถูกพัฒนาด้วย Java Sprint Boot & Elasticsearch

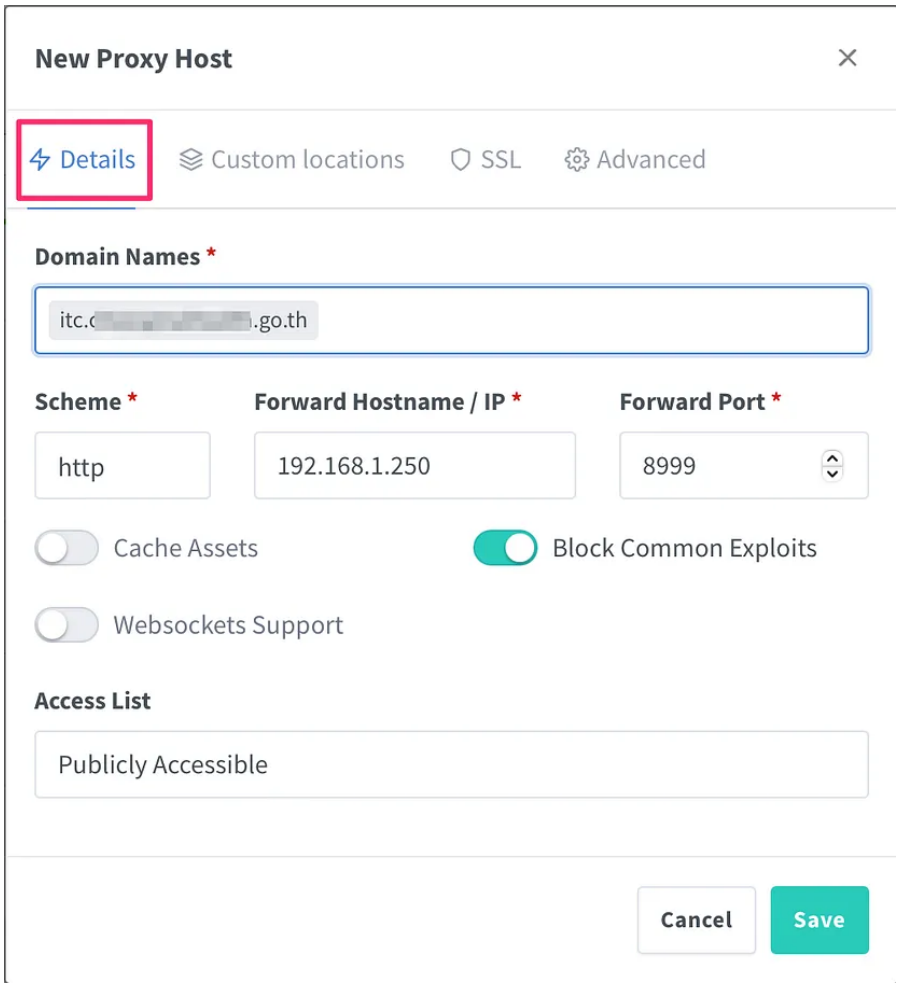
3.2 Frontend (Waiting for edit)

ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการแสดงผลของระบบ โดยส่วนนี้จะถูกพัฒนาด้วย Next.js โดยส่วนนี้จะถูกแบ่งออกเป็นส่วนย่อยๆ ดังนี้

- **Home Page** ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการแสดงผลหน้าแรกของเว็บไซต์
- **Project Page** ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการแสดงผลหน้าโครงงาน
- **Search Page** ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการแสดงผลหน้าค้นหา
- **Profile Page** ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการแสดงผลหน้าโปรไฟล์

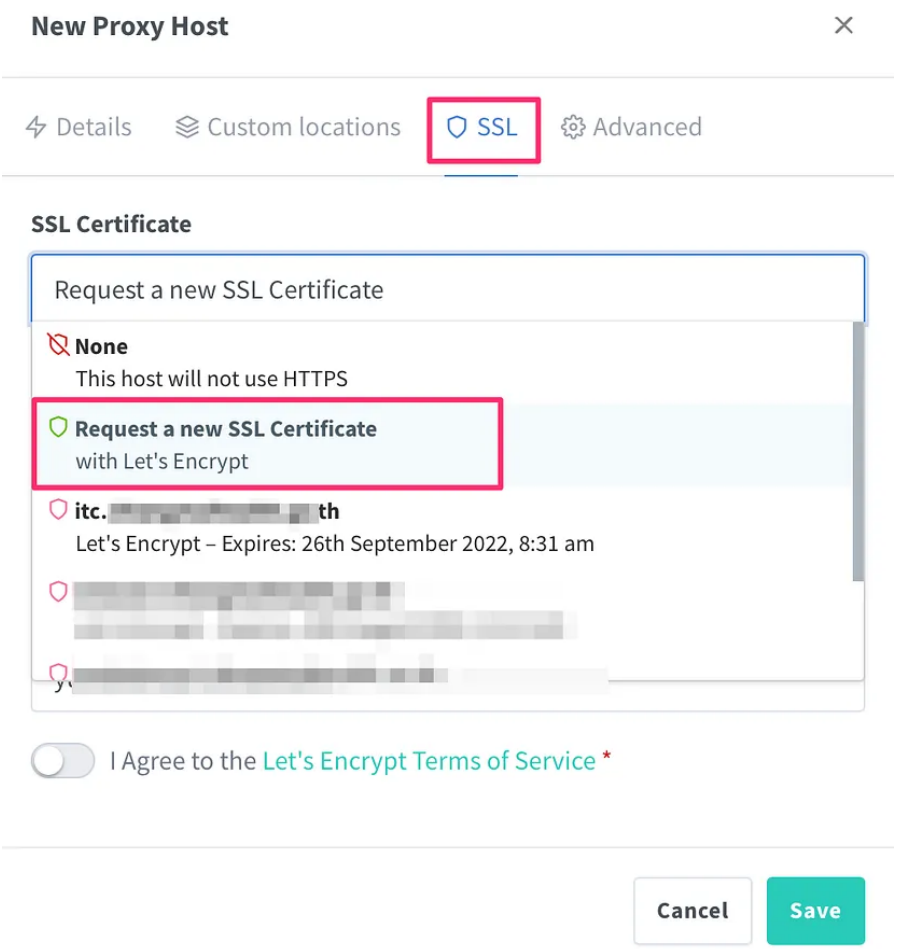
3.3 Reverse Proxy

ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการจัดการการเชื่อมต่อระหว่าง Frontend และ Backend โดยส่วนนี้จะถูกพัฒนาด้วย Nginx Proxy Manager(NPM)[?] มีหน้าที่คือ การแปลง ip addres ของ server ให้เป็น domain name ยกตัวอย่างเช่น แปลง IP Addres `139.59.117.147:8080` → `domainname.com` โดยส่วนนี้จะทำให้เราสามารถเข้าถึงเว็บไซต์ผ่าน domainname.com ได้ และยังสามารถจัดการการเชื่อมต่อระหว่าง Frontend และ Backend ได้อีกด้วย



รูปที่ 3.2: รูปภาพแสดงการทแปลง IP ของ Nginx Reverse Proxy

อีก feature นึงที่สำคัญของ Nginx Reverse Proxy คือการจัดการ SSL Certificate โดย Reverse Proxy จะทำหน้าที่ในการจัดการ SSL Certificate ให้กับเว็บไซต์ ซึ่งจะทำให้เว็บไซต์มีความ



รูปที่ 3.3: รูปภาพแสดงการจัดการ SSL Certificate ของ Nginx Reverse Proxy

ปลอดภัยมากขึ้น และยังสามารถใช้งานได้ทั้งบน http และ https อีกด้วย

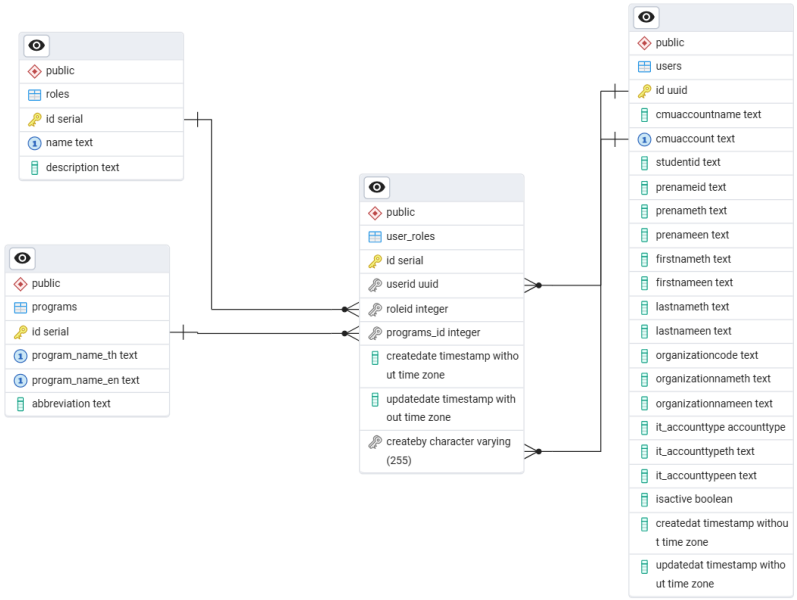
3.4 Backend

ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการจัดการข้อมูลและการทำงานของระบบ โดยส่วนนี้จะถูกแบ่งออกเป็นส่วนย่อยๆ ดังนี้

3.4.1 Auth Service

ส่วนนี้เป็นส่วนที่ทำหน้าที่ในการจัดการข้อมูลของผู้ใช้งาน โดยส่วนนี้จะถูกพัฒนาด้วย Typescript Elysia.js โดยส่วนนี้จะมีหน้าที่คือ จัดการข้อมูลของผู้ใช้งานทำหน้าที่ในการเข้าสู่ระบบ และจัดการสิทธิ์ของผู้ใช้งาน(Role) โดยมีการใช้งาน Oauth2 ในการจัดการการเข้าสู่ระบบ และมีการใช้งาน JWT เพื่อสร้าง Token ส่งกลับไปยัง Frontend

3.4.1.1 Database Schema



รูปที่ 3.4: รูปภาพแสดง Database Schema ของ Auth Service

โครงสร้างของฐานข้อมูล

- **users** ตารางนี้จะเป็นตารางที่เก็บข้อมูลของผู้ใช้งาน
- **roles** ตารางนี้จะเป็นตารางที่เก็บข้อมูลของตำแหน่งของผู้ใช้งาน
- **programs** ตารางนี้จะเป็นตารางที่เก็บข้อมูลของโปรแกรมหรือหลักศสูตรที่มีอยู่ในระบบ
- **user_roles** ตารางนี้จะแสดงถึงความสัมพันธ์ของผู้ใช้งานกับตำแหน่งของผู้ใช้งานในโปรแกรมหรือหลักสูตรนั้นๆ

3.4.2 Project Service

3.4.3 Search Service

บทที่ 4
การทดลองและผลลัพธ์

ในบทนี้จะทดสอบเกี่ยวกับการทำงานในฟังก์ชันหลักๆ

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผล

นศ. ควรสรุปถึงข้อจำกัดของระบบในด้านต่างๆ ที่ระบบมีในเนื้อหาส่วนนี้ด้วย

5.2 ปัญหาที่พบและแนวทางการแก้ไข

ในการทำโครงงานนี้ พบว่าเกิดปัญหาหลักๆ ดังนี้

5.3 ข้อเสนอแนะและแนวทางการพัฒนาต่อ

ข้อเสนอแนะเพื่อพัฒนาโครงงานนี้ต่อไป มีดังนี้

ภาคผนวก

ภาคผนวก ก
The first appendix

Text for the first appendix goes here.

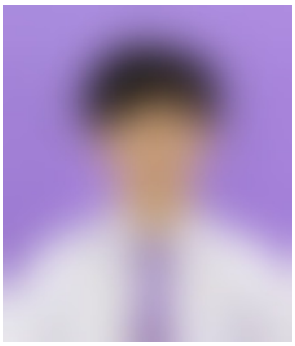
ก.1 Appendix section

Text for a section in the first appendix goes here.
test ทดสอบฟอนต์ serif ภาษาไทย
test ทดสอบฟอนต์ sans serif ภาษาไทย
test ทดสอบฟอนต์ teletype ภาษาไทย
test ทดสอบฟอนต์ teletype ภาษาไทย
ตัวหนา serif ภาษาไทย sans serif ภาษาไทย teletype ภาษาไทย
ตัวเอียง serif ภาษาไทย sans serif ภาษาไทย teletype ภาษาไทย
ตัวหนาเอียง serif ภาษาไทย sans serif ภาษาไทย teletype ภาษาไทย
https://www.example.com/test_ทดสอบ_url

ภาคผนวก ข
คู่มือการใช้งานระบบ

Manual goes here.

ประวัติผู้เขียน



Your biosketch goes here. Make sure it sits inside the `biosketch` environment.