บทที่ 2
แนวคิด ทฤษฎี เอกสาร และงานวิจัยที่เกี่ยวข้อง

ในการพัฒนาระบบจัดการเว็บไซต์สำหรับคณะเทคโนโลยีสารสนเทศครั้งนี้ ผู้พัฒนาได้ทำการศึกษาและประยุกต์ใช้แนวคิด ทฤษฎี เทคโนโลยี และเครื่องมือที่เกี่ยวข้องในด้านต่างๆ เพื่อให้ระบบที่พัฒนาขึ้นมีประสิทธิภาพสูงสุด ดังต่อไปนี้

1. **วิศวกรรมซอฟต์แวร์ (Software Engineering)**
   ประยุกต์ใช้หลักการและกระบวนการที่เป็นระบบในการออกแบบ พัฒนา และทดสอบซอฟต์แวร์
2. **การวิเคราะห์และออกแบบเชิงวัตถุ (Object-Oriented Analysis and Design - OOAD)**
   ใช้เป็นแนวทางหลักในการวิเคราะห์ความต้องการและออกแบบสถาปัตยกรรมของระบบให้มีความยืดหยุ่นและง่ายต่อการบำรุงรักษา
3. **การออกแบบประสบการณ์และส่วนต่อประสานผู้ใช้ (UX/UI Design)**
   มุ่งเน้นการออกแบบที่คำนึงถึงผู้ใช้งานเป็นศูนย์กลาง (**User-Centered Design**) เพื่อให้ระบบสามารถใช้งานได้ง่ายและตอบสนองต่อความต้องการของผู้ใช้กลุ่มต่างๆ
4. **การออกแบบระบบฐานข้อมูล (Database System Design)**
   วางแผนและออกแบบโครงสร้างฐานข้อมูลเชิงสัมพันธ์ (**Relational Database**) เพื่อการจัดเก็บและจัดการข้อมูลอย่างเป็นระบบและมีประสิทธิภาพ
5. **ภาษาโปรแกรมและเฟรมเวิร์ก (Programming Languages and Frameworks)**
   เลือกใช้ภาษาและเทคโนโลยีที่ทันสมัยและเหมาะสมกับลักษณะของงานทั้งในส่วนหน้าบ้าน (**Frontend**) และส่วนหลังบ้าน (**Backend**)
6. **เทคโนโลยีเว็บเอพีไอ (Web API Technology)**
   ออกแบบและพัฒนา **RESTful API** เพื่อเป็นช่องทางในการสื่อสารและแลกเปลี่ยนข้อมูลระหว่างส่วนหน้าบ้านและส่วนหลังบ้านอย่างเป็นมาตรฐาน
7. **การยืนยันตัวตนด้วยโทเคน (Token-based Authentication)**
   ใช้เทคนิค **JSON Web Token (JWT)** เพื่อสร้างระบบการเข้าสู่ระบบที่ปลอดภัยและสามารถขยายขนาดได้ (**Scalable**)

8. **การปรับแต่งเว็บไซต์สำหรับเครื่องมือค้นหา (Search Engine Optimization - SEO)**
   ประยุกต์ใช้เทคนิคการแสดงผลฝั่งเซิร์ฟเวอร์ (**Server-Side Rendering**) และการออกแบบโครงสร้างเนื้อหาที่เป็นมิตรต่อการจัดทำดัชนีของเครื่องมือค้นหา

วิศวกรรมซอฟต์แวร์ (Software Engineering)

ศาสตร์ที่ประยุกต์ใช้หลักการและกระบวนการที่เป็นระบบ มีระเบียบแบบแผน และสามารถวัดผลได้ เพื่อนำไปสู่การพัฒนา การดำเนินงาน และการบำรุงรักษาซอฟต์แวร์ที่มีคุณภาพ เป้าหมายหลักคือการสร้างผลิตภัณฑ์ซอฟต์แวร์ที่ตอบสนองต่อความต้องการของผู้ใช้ได้อย่างถูกต้องสมบูรณ์ อยู่ในงบประมาณและกรอบเวลาที่กำหนด และง่ายต่อการบำรุงรักษาในระยะยาว

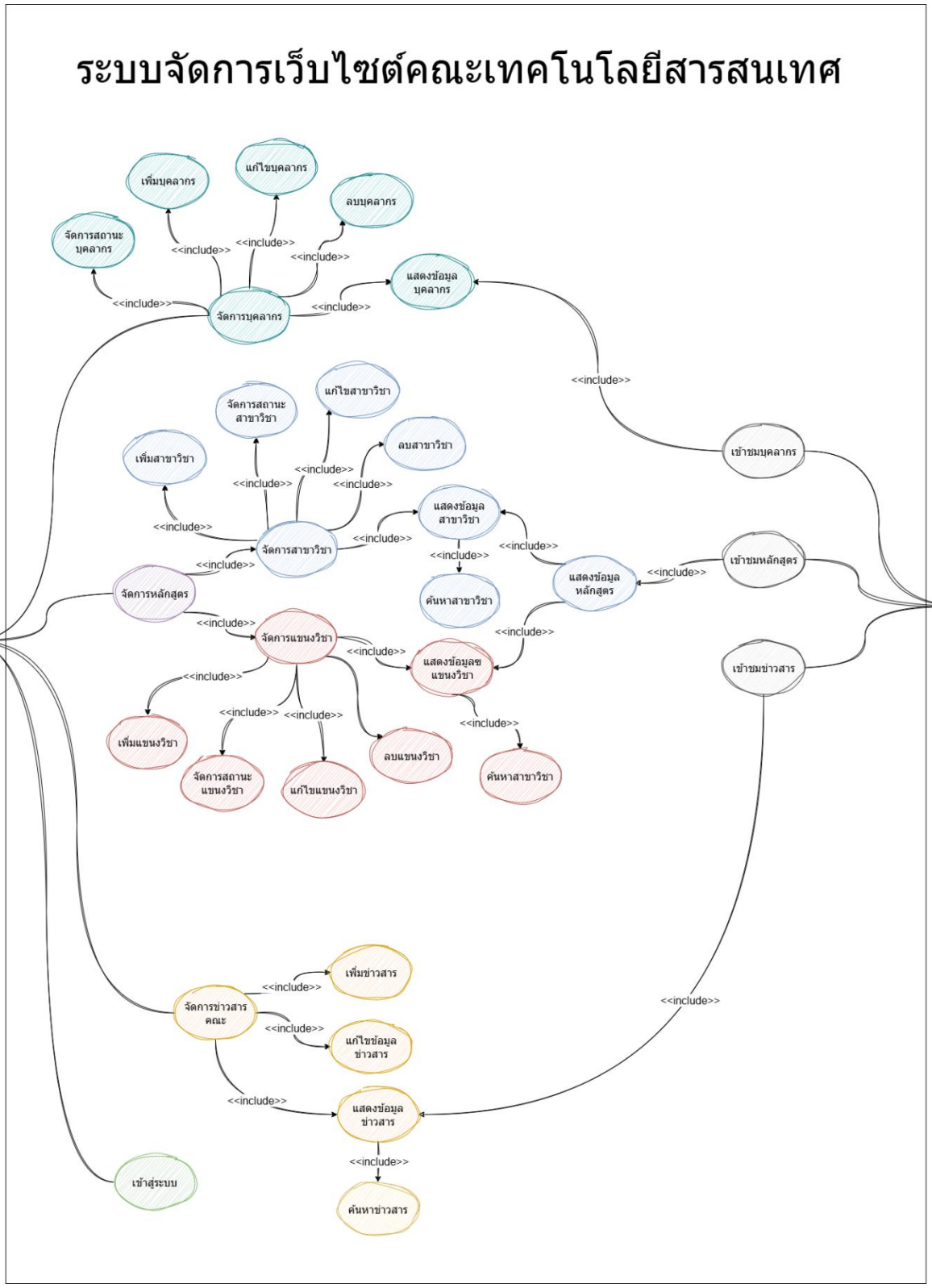
แนวทางนี้ครอบคลุมทุกขั้นตอนของ วงจรการพัฒนาซอฟต์แวร์ (Software Development Life Cycle - SDLC) ตั้งแต่การรวบรวมและวิเคราะห์ความต้องการ (Requirements Analysis), การออกแบบสถาปัตยกรรมและส่วนประกอบของระบบ (System Design), การลงมือเขียนโปรแกรม (Implementation), การทดสอบเพื่อรับประกันคุณภาพ (Quality Assurance & Testing), ไปจนถึงการติดตั้งใช้งาน (Deployment) และการบำรุงรักษา (Maintenance)

การประยุกต์ใช้ในโครงการ

ในโครงการพัฒนาระบบจัดการเว็บไซต์สำหรับคณะเทคโนโลยีสารสนเทศนี้ ผู้พัฒนาได้นำหลักการทางวิศวกรรมซอฟต์แวร์มาเป็นแนวทางหลักในการดำเนินงาน โดยมีการวางแผนโครงการอย่างเป็นลำดับขั้นตอน เริ่มจากการวิเคราะห์ความต้องการของกลุ่มผู้ใช้งานต่างๆ (เช่น ผู้ดูแลระบบ, อาจารย์, และผู้เยี่ยมชมทั่วไป) เพื่อกำหนดขอบเขตและคุณสมบัติของระบบ จากนั้นจึงนำไปสู่การออกแบบโครงสร้างเชิงเทคนิค การเลือกใช้เทคโนโลยีที่เหมาะสม และการพัฒนาซอฟต์แวร์อย่างมีแบบแผน นอกจากนี้ยังมีการวางแผนการทดสอบในแต่ละส่วนย่อย (Unit Testing) และการทดสอบทั้งระบบ (Integration Testing) เพื่อให้มั่นใจว่าซอฟต์แวร์ที่ส่งมอบมีข้อผิดพลาดน้อยที่สุดและสามารถทำงานได้อย่างมีเสถียรภาพ

การวิเคราะห์และออกแบบเชิงวัตถุ (Object-Oriented Analysis and Design - OOAD)

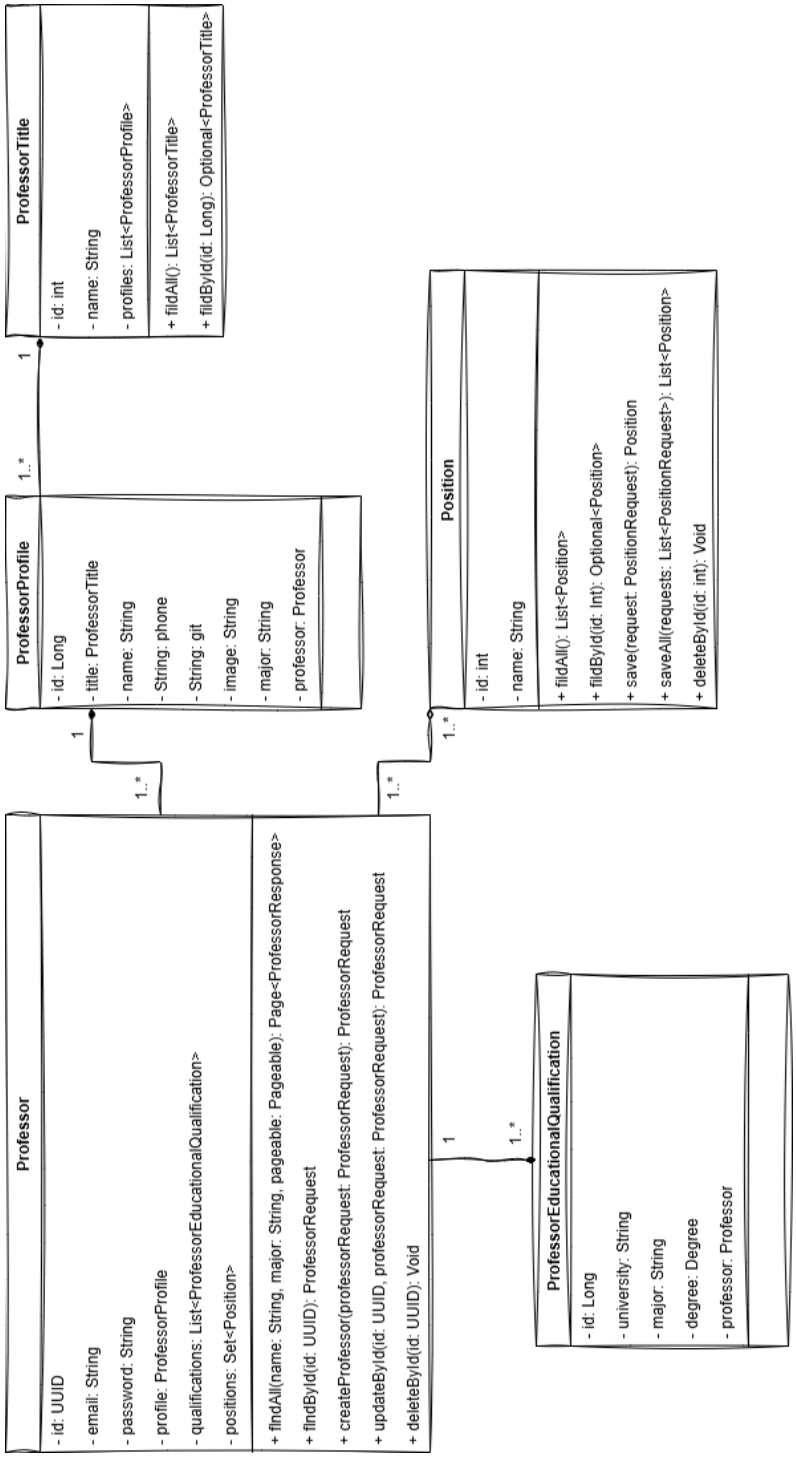
Use Case Diagram ระบบจัดการเว็บไซต์คณะเทคโนโลยีสารสนเทศ



Class Diagram ระบบจัดการเว็บไซต์คณะเทคโนโลยีสารสนเทศ

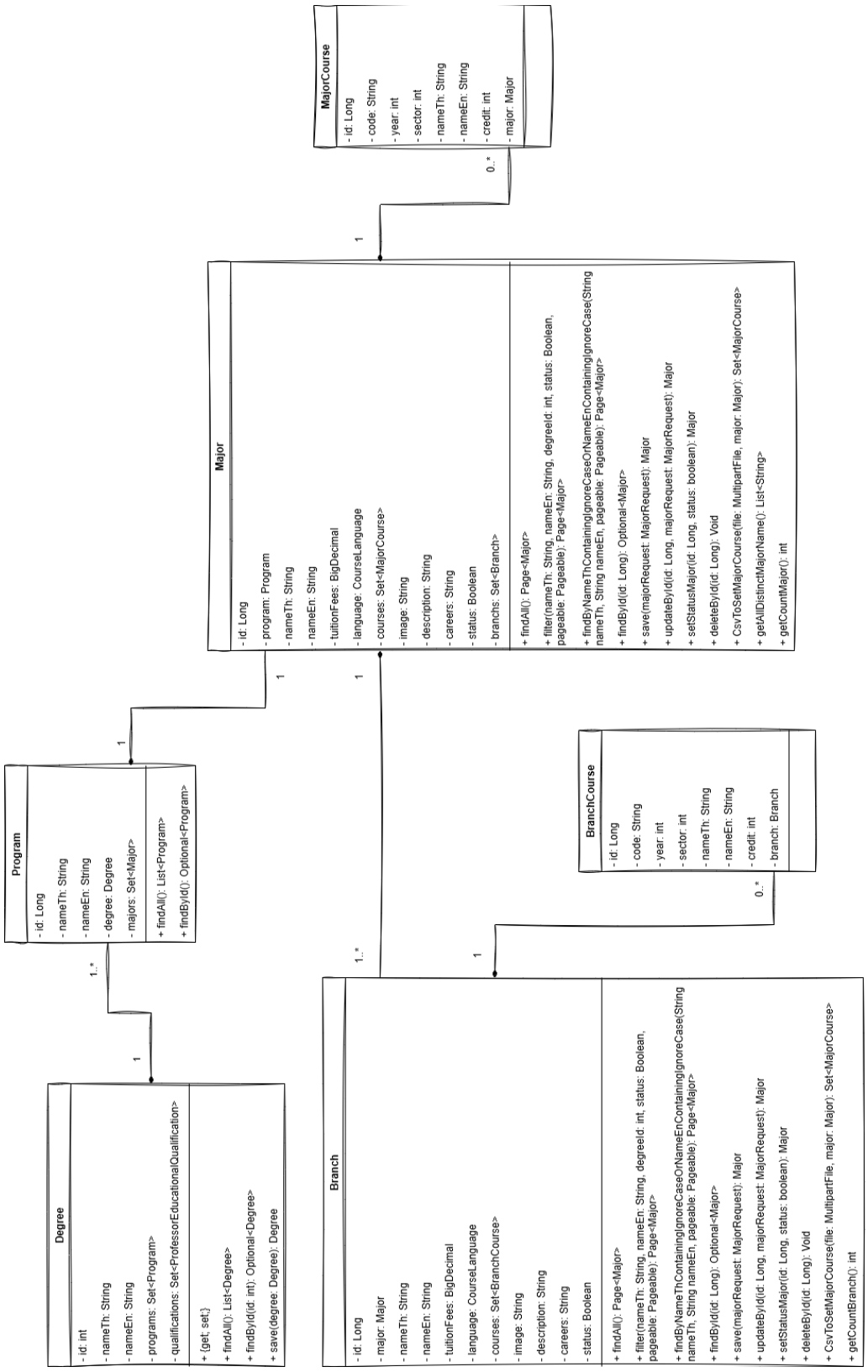
แบ่งเป็น 3 ระบบย่อย ได้แก่ ระบบจัดการบุคลากร , ระบบจัดการหลักสูตร และระบบจัดการข่าวสาร

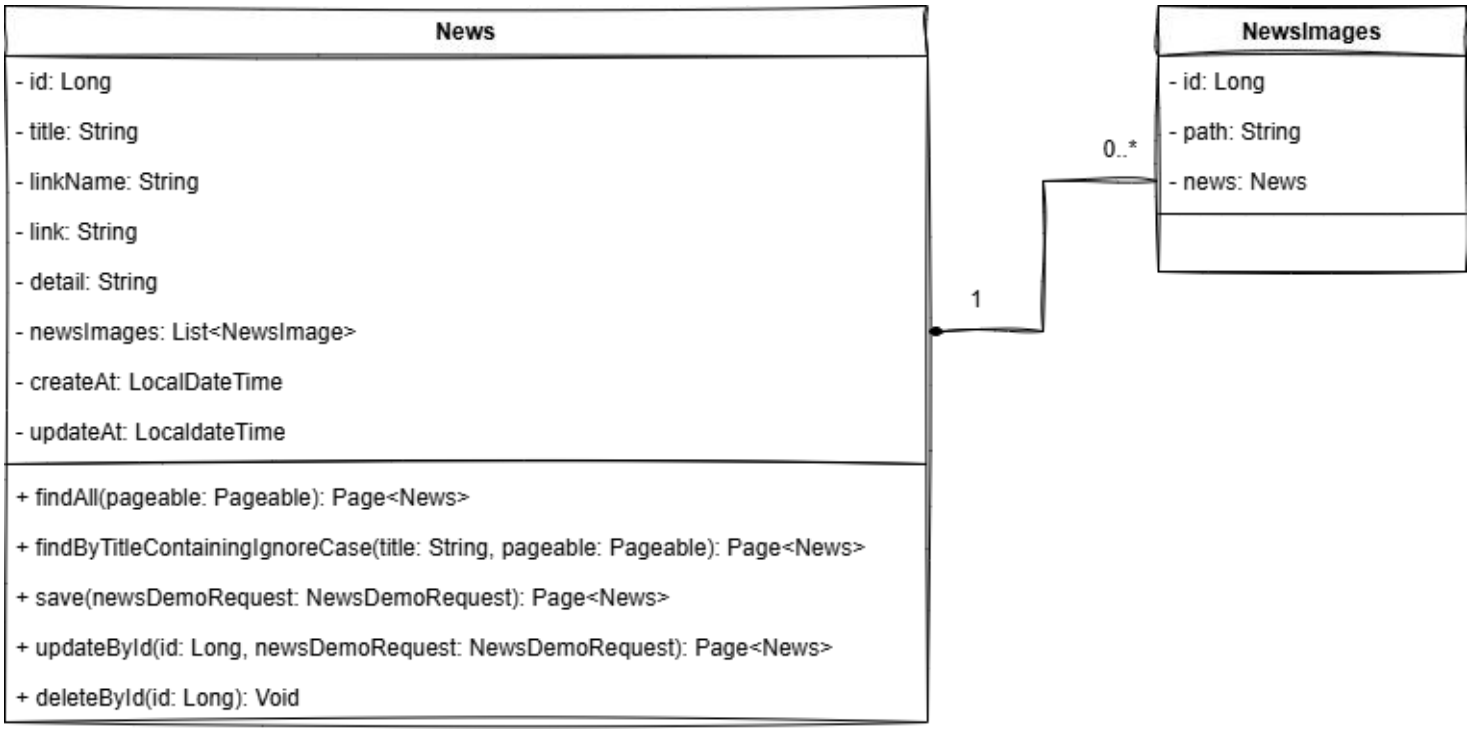
ระบบจัดการบุคลากร



ระบบจัดการหลักสูตร

ระบบจัดการข่าวสาร




News
- id: Long
- title: String
- linkName: String
- link: String
- detail: String
- newsImages: List<NewsImage>
- createAt: LocalDateTime
- updateAt: LocaldateTime
+ findAll(pageable: Pageable): Page<News>
+ findByTitleContainingIgnoreCase(title: String, pageable: Pageable): Page<News>
+ save(newsDemoRequest: NewsDemoRequest): Page<News>
+ updateById(id: Long, newsDemoRequest: NewsDemoRequest): Page<News>
+ deleteById(id: Long): Void
1
0..*
NewsImages
- id: Long
- path: String
- news: News

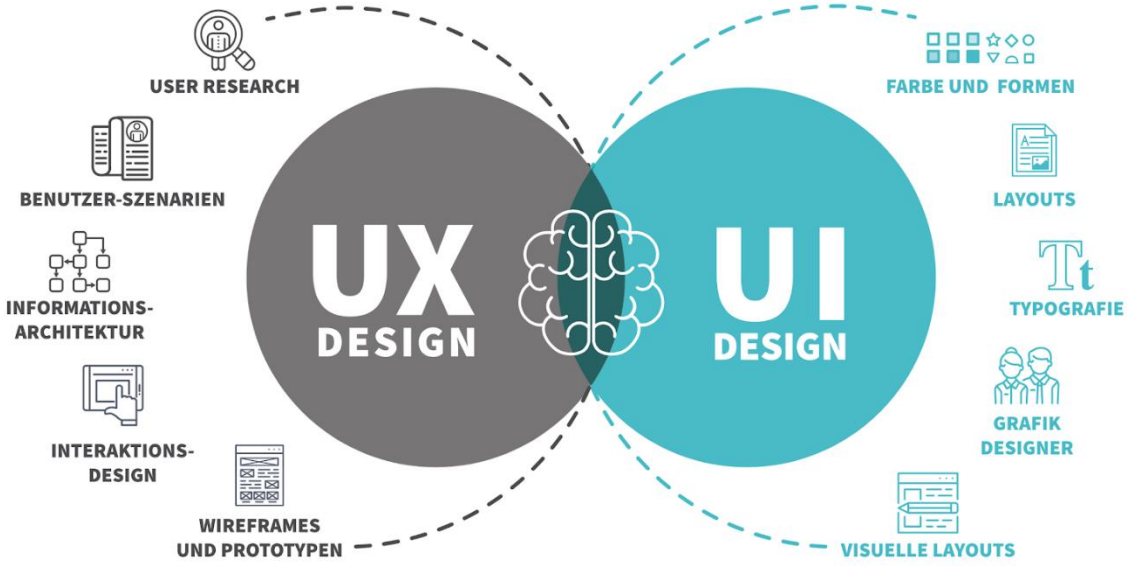
การออกแบบประสบการณ์และส่วนต่อประสานผู้ใช้ (UX/UI Design)

UX (User Experience) คือ ประสบการณ์และความรู้สึกโดยรวมของผู้ใช้ที่มีต่อการใช้งานระบบ ในขณะที่ UI (User Interface) คือ ลักษณะหน้าตาและส่วนประกอบที่ผู้ใช้มองเห็นและโต้ตอบด้วย เช่น ปุ่ม เมนู รูปภาพ การออกแบบที่ดีต้องคำนึงถึงทั้งสองส่วนควบคู่กันไป โดยมีเป้าหมายเพื่อให้ผู้ใช้สามารถบรรลุเป้าหมายได้อย่างราบรื่นและพึงพอใจ

แนวทางสำคัญคือ การออกแบบที่ยึดผู้ใช้เป็นศูนย์กลาง (User-Centered Design) ซึ่งให้ความสำคัญกับการทำความเข้าใจความต้องการ พฤติกรรม และข้อจำกัดของผู้ใช้กลุ่มเป้าหมาย เพื่อนำมาเป็นข้อมูลในการออกแบบ

การประยุกต์ใช้ในโครงการ: เนื่องจากเว็บไซต์ของคณะฯ มีผู้ใช้หลายกลุ่ม (นักศึกษาปัจจุบัน, ผู้ที่สนใจศึกษาต่อ, คณาจารย์, เจ้าหน้าที่) การออกแบบ UX/UI จึงมุ่งเน้น

- ความง่ายในการใช้งาน (Usability) จัดวางโครงสร้างเมนูและข้อมูลให้หาเจอง่าย ไม่ซับซ้อน
- การเข้าถึง (Accessibility) ออกแบบให้ทุกคนสามารถเข้าถึงข้อมูลได้
- การตอบสนอง (Responsive Design) หน้าเว็บสามารถแสดงผลได้อย่างเหมาะสมบนทุกขนาดหน้าจอ ตั้งแต่คอมพิวเตอร์เดสก์ท็อปไปจนถึงสมาร์ทโฟน



ภาษาโปรแกรมและเฟรมเวิร์ก (Programming Languages and Frameworks)

ในส่วนนี้จะอธิบายถึงเทคโนโลยีและเครื่องมือที่ถูกเลือกใช้ในการพัฒนาส่วนหน้าบ้าน (Frontend) และส่วนหลังบ้าน (Backend) ของระบบ

ภาษา TypeScript

TypeScript คือภาษาโปรแกรมแบบเปิดเผยซอร์สโค้ด (Open-source) ที่ได้รับการพัฒนาต่อยอดมาจากภาษา JavaScript (Superset of JavaScript) โดยการเพิ่มความสามารถของการกำหนดชนิดข้อมูลแบบคงที่ (Static Typing) เข้ามา จุดประสงค์หลักคือเพื่อช่วยให้นักพัฒนาสามารถตรวจจับข้อผิดพลาดเชิงโครงสร้างของโค้ดได้ตั้งแต่ในขั้นตอนการคอมไพล์ (Compile-time) ซึ่งเป็นการส่งเสริมความน่าเชื่อถือและความสะดวกในการบำรุงรักษาโค้ด (Code Reliability and Maintainability) โดยเฉพาะในโครงการขนาดใหญ่ที่มีความซับซ้อนสูง

การใช้งานในโปรเจค

ใช้เป็นภาษาหลักในการพัฒนาแอปพลิเคชันส่วนหน้าบ้าน (Frontend) ทั้งหมด เพื่อสร้างโค้ดที่มีความเสถียร ลดข้อผิดพลาดที่อาจเกิดขึ้น และสนับสนุนการทำงานร่วมกันของทีมพัฒนา

Node.js

JavaScript Runtime Environment ที่ทำให้สามารถรันโค้ด JavaScript นอกเว็บเบราว์เซอร์ได้ ในบริบทของ Frontend นั้น Node.js ไม่ได้ถูกใช้เพื่อรันเซิร์ฟเวอร์โดยตรง แต่มีความสำคัญอย่างยิ่งในฐานะเครื่องมือสำหรับการพัฒนา (Development Tooling)

การใช้งานในโปรเจค

เป็นพื้นฐานที่จำเป็นในการติดตั้งและจัดการไลบรารีฝั่ง Frontend ทั้งหมด รวมถึงใช้ในการรันคำสั่งต่างๆ เพื่อเริ่มต้นเซิร์ฟเวอร์สำหรับการพัฒนา (Development Server) และการ Build โปรเจคเพื่อนำไปใช้งานจริง

Vue.js Nuxt.js

Framework สำหรับสร้างเว็บแอปพลิเคชันบนพื้นฐานของ Vue.js โดย Nuxt.js ได้รับการออกแบบมาเพื่อทำให้การพัฒนาเว็บแอปพลิเคชันสมัยใหม่เป็นเรื่องง่ายและมีประสิทธิภาพสูง มาพร้อมกับฟีเจอร์ที่จำเป็นมากมายติดตั้งมาให้แล้ว

การใช้งานในโปรเจค

เป็น Framework หลักในการสร้าง User Interface (UI) ทั้งหมดของโปรเจค จัดการโครงสร้างของเว็บแอปพลิเคชัน การเชื่อมต่อระหว่างหน้าต่างๆ และการแสดงผลข้อมูลที่ได้รับมาจาก Backend

Tailwind CSS

Utility-First CSS Framework ที่แตกต่างจาก Framework อื่นๆ (เช่น Bootstrap) ตรงที่จะไม่มีคอมโพเนนต์สำเร็จรูปมาให้ (เช่น Button, Card) แต่จะให้คลาส (Class) ที่มีหน้าที่เฉพาะเจาะจงในการจัดสไตล์เพียงอย่างเดียว (เช่น flex, pt-4, text-center) นักพัฒนาจะนำคลาสเหล่านี้มาประกอบกันในไฟล์ HTML หรือ .vue เพื่อสร้างดีไซน์ที่ต้องการได้อย่างอิสระและรวดเร็ว

การใช้งานในโปรเจค

ใช้เป็นเครื่องมือหลักในการออกแบบและจัดสไตล์หน้าตาของเว็บแอปพลิเคชันทั้งหมด ทำให้สามารถสร้าง UI ที่มีความสวยงามและตอบสนองต่อทุกขนาดหน้าจอได้อย่างมีประสิทธิภาพ

Shadcn/vue

Shadcn/vue ไม่ใช่ไลบรารีคอมโพเนนต์สำเร็จรูปแบบดั้งเดิม แต่เป็น ชุดของคอมโพเนนต์ที่สามารถนำกลับมาใช้ใหม่ได้ (Reusable components) ที่สร้างขึ้นโดยใช้ Tailwind CSS และ Radix Vue (สำหรับจัดการการเข้าถึงและการทำงานเบื้องหลัง) จุดเด่นคือ แทนที่จะติดตั้งเป็นแพ็กเกจ นักพัฒนาจะใช้ Command Line Interface (CLI) เพื่อคัดลอกโค้ดของคอมโพเนนต์ที่ต้องการ (เช่น Button, Dialog, Table) เข้ามาไว้ในโปรเจคของตัวเองโดยตรง

การใช้งานในโปรเจค

ใช้เป็นชุดเครื่องมือสำหรับสร้าง UI คอมโพเนนต์พื้นฐานที่มีความซับซ้อน เช่น Dropdown Menus, Data Tables, Forms, Dialogs ซึ่งช่วยประหยัดเวลาในการพัฒนาและยังคงความสามารถในการปรับแต่งได้อย่างเต็มที่

ภาษา Java

ภาษาโปรแกรมเชิงวัตถุ (Object-Oriented Programming) ที่ได้รับความนิยมอย่างสูงและใช้งานมาอย่างยาวนาน มีจุดเด่นในเรื่องของความเสถียร (Robustness), ความปลอดภัย (Security) และความสามารถในการทำงานข้ามแพลตฟอร์ม (Platform Independent) ผ่านสิ่งที่เรียกว่า Java Virtual Machine (JVM)

การใช้งานในโปรเจค

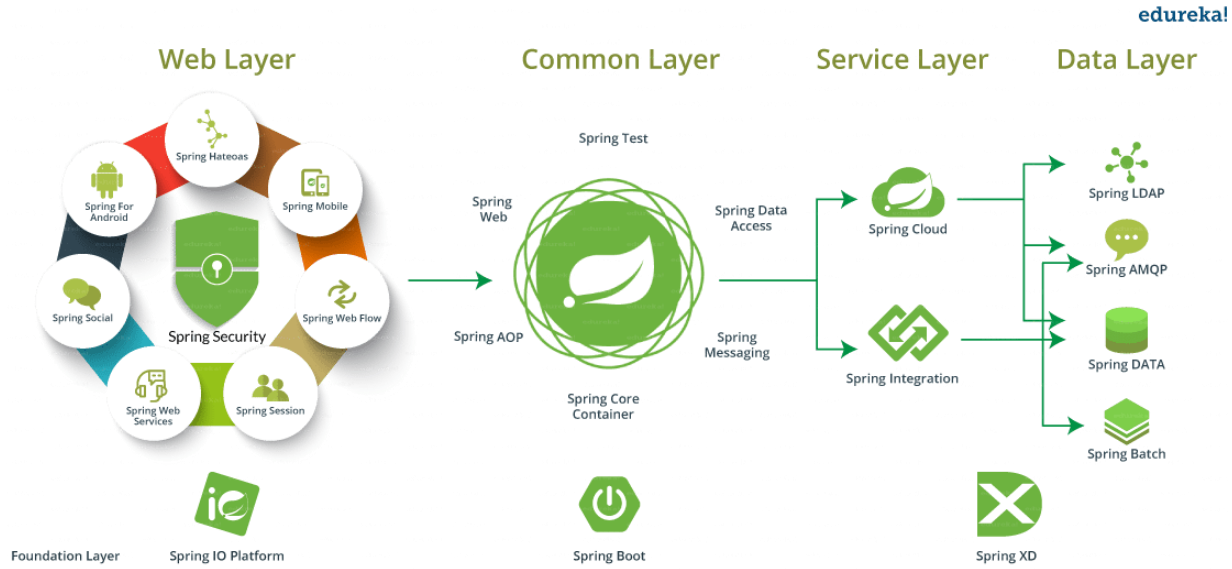
เป็นภาษาหลักที่ใช้ในการพัฒนาส่วนของ Backend ทั้งหมด เพื่อสร้าง API (Application Programming Interface) สำหรับรับส่งข้อมูลกับฝั่ง Frontend และจัดการตรรกะทางธุรกิจ (Business Logic) ทั้งหมดของระบบ

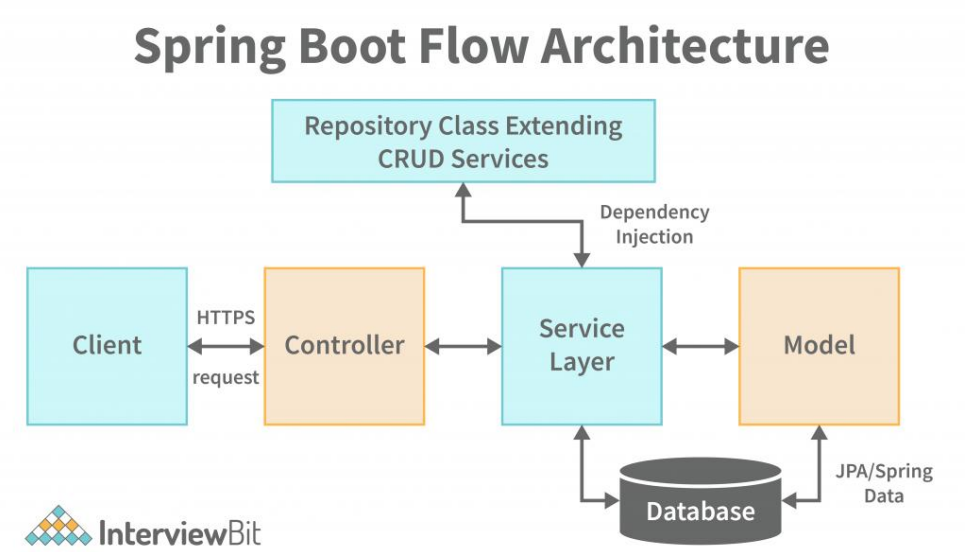
Spring Boot

Spring Boot คือ Framework ที่สร้างขึ้นบน Spring Framework อีกทีหนึ่ง โดยมีเป้าหมายเพื่อทำให้การสร้างแอปพลิเคชันด้วย Java และ Spring เป็นเรื่องที่ง่ายและรวดเร็วขึ้นอย่างมาก โดยลดขั้นตอนการตั้งค่าที่ยุ่งยากและซับซ้อนออกไป

การใช้งานในโปรเจค

เป็น Framework หลักในการสร้าง Backend API ทั้งหมดของโปรเจค จัดการเรื่องการเชื่อมต่อกับฐานข้อมูล (PostgreSQL), การสร้าง Endpoints สำหรับให้ Frontend เรียกใช้งาน, การจัดการความปลอดภัย (Security) และการจัดการส่วนประกอบต่างๆ ของแอปพลิเคชัน





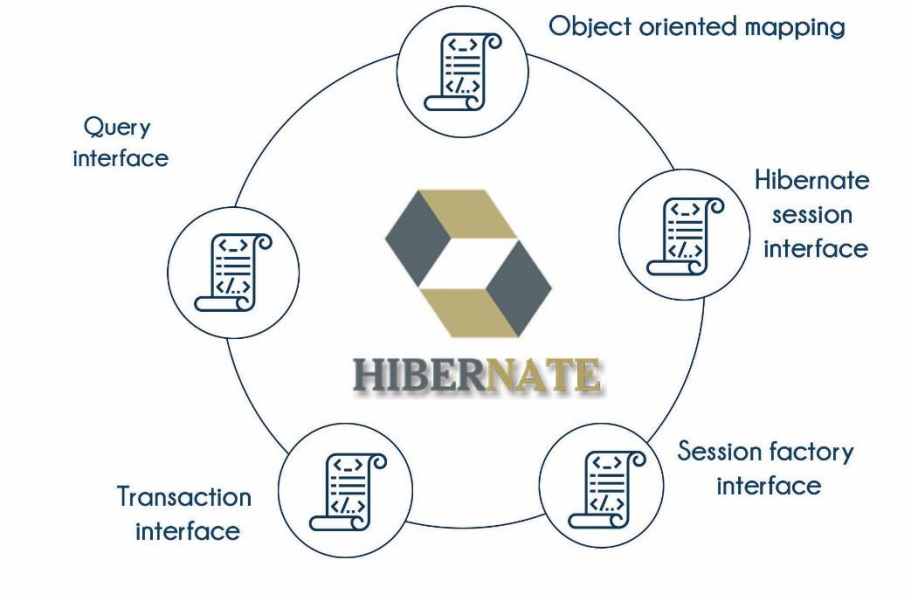
Hibernate

Object-Relational Mapping (ORM) เฟรมเวิร์กสำหรับภาษา Java ที่ทำหน้าที่เป็น "ตัวกลาง" เชื่อมระหว่างโค้ดที่เขียนในรูปแบบ เชิงวัตถุ (Object-Oriented) กับ ฐานข้อมูลเชิงสัมพันธ์ (Relational Database) ที่จัดเก็บข้อมูลในรูปแบบตาราง

หน้าที่หลักของ Hibernate คือการแปลงข้อมูลระหว่างสองรูปแบบนี้โดยอัตโนมัติ (Mapping) ทำให้นักพัฒนาสามารถจัดการข้อมูลในฐานข้อมูล (เช่น การบันทึก, การค้นหา, การแก้ไข, การลบ) ได้โดยการเรียกใช้เมธอดของอ็อบเจกต์ในภาษา Java โดยตรง โดยไม่จำเป็นต้องเขียนคำสั่ง SQL ที่ซับซ้อนด้วยตัวเอง ซึ่งช่วยลดความผิดพลาดและเพิ่มความเร็วในการพัฒนา

การประยุกต์ใช้ในโครงการ

ในส่วนหลังบ้าน (Backend) ของระบบ หากพัฒนาด้วยภาษา Java โครงการนี้ได้เลือกใช้ Hibernate เพื่อจัดการกับการสื่อสารกับฐานข้อมูล การทำเช่นนี้ช่วยให้นักพัฒนาสามารถมุ่งเน้นไปที่การพัฒนาตรรกะทางธุรกิจ (Business Logic) ของระบบจัดการเว็บไซต์ได้เต็มที่ เช่น การสร้างฟังก์ชันสำหรับเพิ่มข่าวประชาสัมพันธ์ หรือการดึงข้อมูลอาจารย์มาแสดงผล โดย Hibernate จะรับหน้าที่แปลงคำสั่งเหล่านี้เป็น SQL ที่เหมาะสมให้เอง ทำให้โค้ดมีความสะอาด อ่านง่าย และง่ายต่อการบำรุงรักษาในระยะยาว

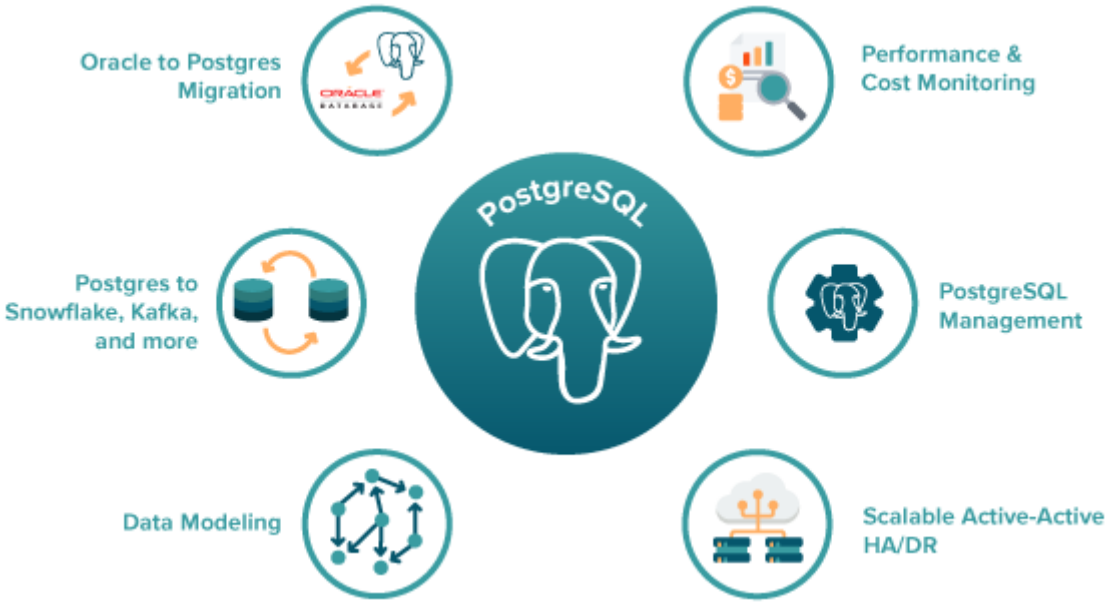


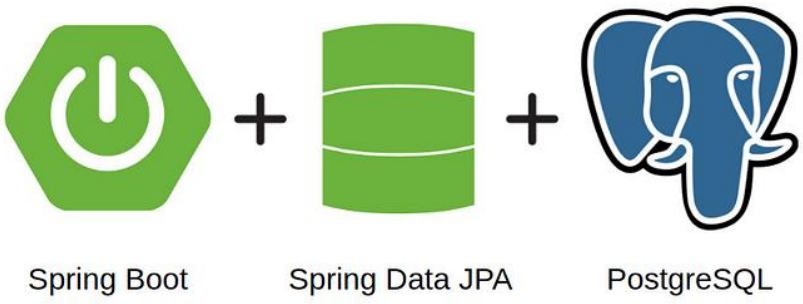
PostgreSQL

เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์แบบโอเพนซอร์ส (Open-Source Object-Relational Database Management System) ที่มีประสิทธิภาพสูงและได้รับความนิยมอย่างแพร่หลาย มีชื่อเสียงในด้านความน่าเชื่อถือ, ความยืดหยุ่น และการรองรับมาตรฐาน SQL อย่างเคร่งครัด

การใช้งานในโปรเจค

ใช้เป็นฐานข้อมูลหลักสำหรับจัดเก็บข้อมูลทั้งหมดของแอปพลิเคชัน เช่น ข้อมูลบุคลากร, ข้อมูลหลักสูตร และข้อมูลข่าวสาร โดยส่วนของ Backend (Spring Boot) จะทำหน้าที่เชื่อมต่อและจัดการข้อมูลใน PostgreSQL





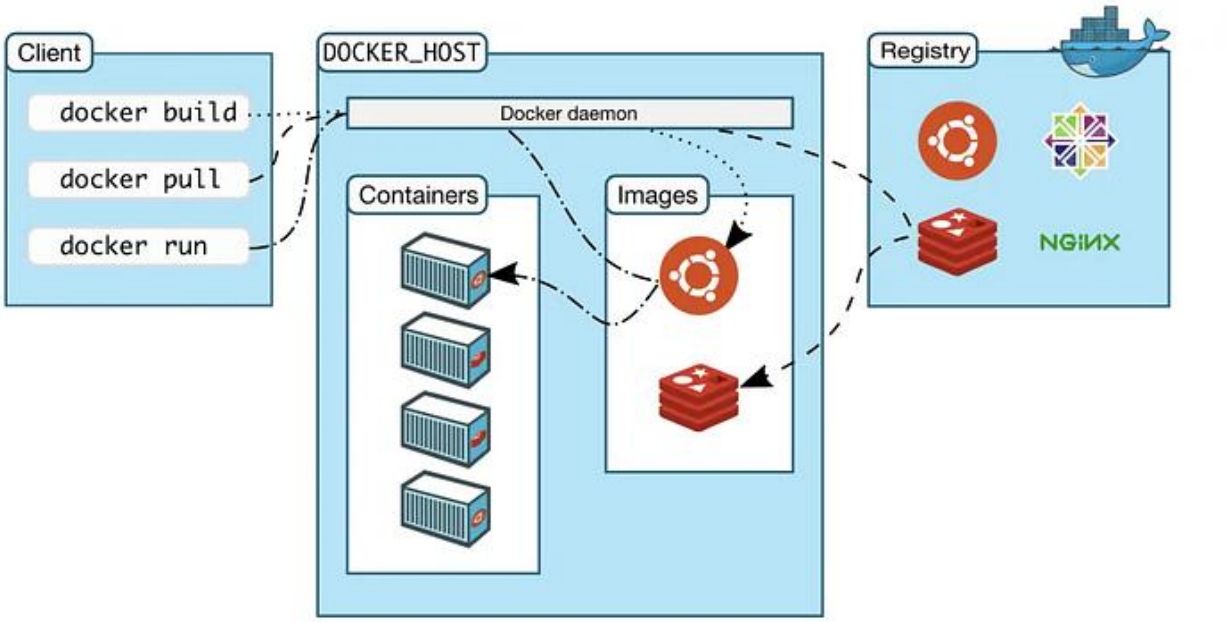
เทคโนโลยีคอนเทนเนอร์ (Docker)

วิธีการจำลองสภาพแวดล้อมการทำงานของซอฟต์แวร์ในระดับระบบปฏิบัติการ โดยการ "แพ็ก" แอปพลิเคชันพร้อมกับส่วนประกอบที่จำเป็นทั้งหมด (เช่น ไลบรารี, โค้ด, และการตั้งค่า) ไว้ในหน่วยที่เรียกว่า คอนเทนเนอร์ (Container) ซึ่งสามารถทำงานได้อย่างอิสระและสม่ำเสมอบนทุกสภาพแวดล้อม ไม่ว่าจะเป็นเครื่องคอมพิวเตอร์ของนักพัฒนา, เซิร์ฟเวอร์ทดสอบ, หรือเซิร์ฟเวอร์จริง (Production)

เทคโนโลยีนี้แตกต่างจากการจำลองเสมือน (Virtualization) แบบดั้งเดิมที่ต้องจำลองฮาร์ดแวร์ทั้งระบบ (Hypervisor) แต่คอนเทนเนอร์จะใช้ทรัพยากรจากระบบปฏิบัติการหลัก (Host OS) ร่วมกัน ทำให้มีขนาดเล็กกว่า, เริ่มทำงานได้เร็วกว่า, และใช้ทรัพยากรน้อยกว่ามาก เทคโนโลยีที่ได้รับความนิยมสูงสุดในปัจจุบันคือ Docker

การประยุกต์ใช้ในโครงการ

โครงการนี้ได้นำเทคโนโลยีคอนเทนเนอร์ (Docker) มาใช้เพื่อสร้างสภาพแวดล้อมในการพัฒนาและทดสอบระบบให้เหมือนกันสำหรับทีมพัฒนาทุกคน (Development Environment Parity) ซึ่งช่วยลดปัญหาคลาสสิกอย่าง "เครื่องฉันรันได้ แต่เครื่องเธอรันไม่ได้" (It works on my machine) นอกจากนี้ ยังช่วยให้การนำระบบขึ้นใช้งานจริง (Deployment) เป็นไปอย่างราบรื่นและรวดเร็ว โดยเพียงแค่ย้ายคอนเทนเนอร์ที่ผ่านการทดสอบแล้วไปยังเซิร์ฟเวอร์เป้าหมาย ก็สามารถมั่นใจได้ว่าระบบจะทำงานได้ถูกต้องเหมือนเดิม



เทคโนโลยีเว็บเอพีไอ (Web API Technology)

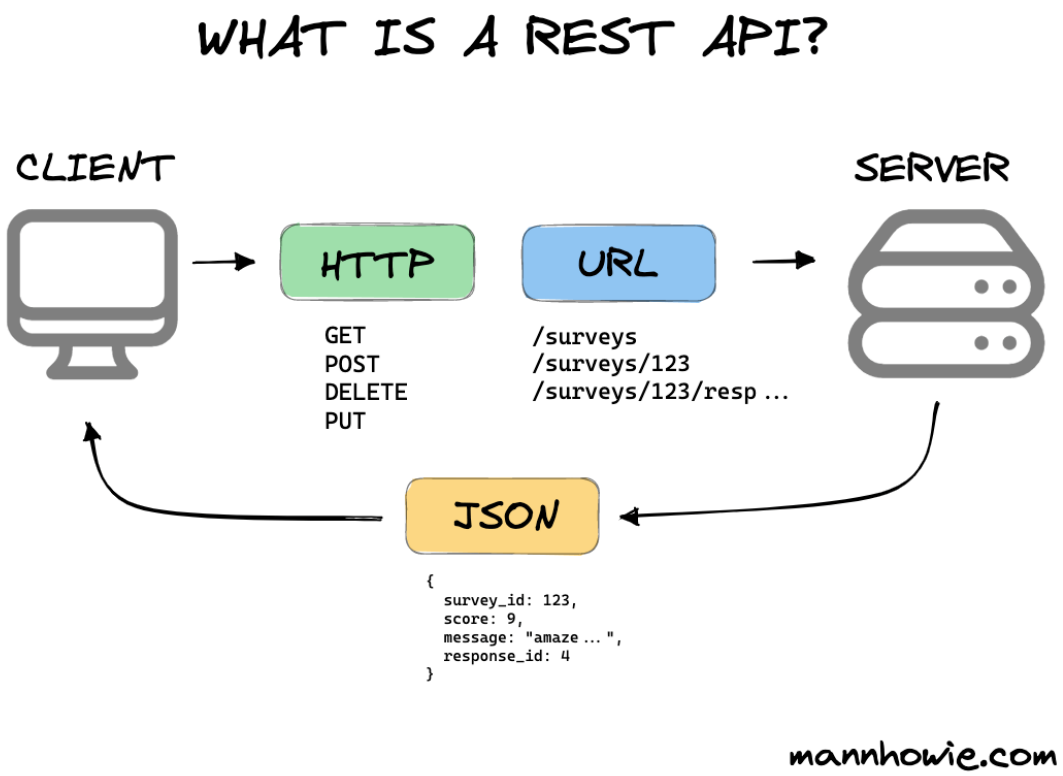
API (Application Programming Interface)

ช่องทางการสื่อสารที่กำหนดไว้เพื่อให้ซอฟต์แวร์สองส่วนสามารถคุยกันได้ สำหรับเว็บแอปพลิเคชันสมัยใหม่ นิยมใช้ **RESTful API** เป็นสถาปัตยกรรมในการออกแบบ API ซึ่งใช้โปรโตคอล HTTP มาตรฐาน (เช่น GET, POST, PUT, DELETE) ในการร้องขอและจัดการข้อมูล ทำให้การสื่อสารระหว่าง Frontend และ Backend เป็นไปอย่างเป็นระบบและเป็นอิสระต่อกัน

การประยุกต์ใช้ในโครงการ

ระบบนี้ได้พัฒนา RESTful API ขึ้นมาเพื่อทำหน้าที่เป็น "สะพาน" เชื่อมระหว่างส่วนหน้าบ้านและส่วนหลังบ้าน เมื่อผู้ใช้ต้องการดูข้อมูลข่าวสาร (Frontend) จะส่งคำร้องขอ (Request) ไปยัง API Endpoint (เช่น GET /api/news) และส่วนหลังบ้าน (Backend) จะส่งข้อมูลข่าวสารกลับไปในรูปแบบมาตรฐาน (เช่น JSON) เพื่อให้ Frontend นำไปแสดงผล วิธีนี้ทำให้สามารถพัฒนาและปรับปรุงแต่ละส่วนได้โดยไม่กระทบกัน

การยืนยันตัวตนด้วยโทเคน (Token-based Authentication)



เป็นการยืนยันตัวตนสำหรับผู้ใช้ในระบบดิจิทัล โดยหลังจากผู้ใช้เข้าสู่ระบบสำเร็จด้วยชื่อผู้ใช้และรหัสผ่าน เซิร์ฟเวอร์จะสร้าง "โทเคน" (Token) ซึ่งเป็นสายอักขระที่เข้ารหัสและมีข้อมูลของผู้ใช้และวันหมดอายุส่งกลับไปให้ผู้ใช้ จากนั้นในการร้องขอข้อมูลครั้งต่อๆ ไป ผู้ใช้จะต้องแนบโทเคนนี้มาด้วยเพื่อยืนยันว่าตนเองมีสิทธิ์เข้าถึงข้อมูลหรือกระทำการนั้นๆ

JSON Web Token (JWT)

เป็นมาตรฐานที่นิยมใช้ในการสร้างโทเคน เนื่องจากมีความปลอดภัยและไม่จำเป็นต้องเก็บสถานะการล็อกอินไว้ที่ฝั่งเซิร์ฟเวอร์ (Stateless) ทำให้ระบบสามารถรองรับผู้ใช้จำนวนมากได้ดี (Scalable)

การประยุกต์ใช้ในโครงการ

สำหรับส่วนจัดการเว็บไซต์ (Admin Panel) จะใช้ระบบ JWT ในการยืนยันตัวตน เมื่อผู้ดูแลระบบล็อกอินเข้ามา จะได้รับ JWT และต้องใช้โทเคนนี้ในการเรียก API ที่ต้องการสิทธิ์ เช่น การสร้างข่าวใหม่ หรือการแก้ไขข้อมูลหลักสูตร เพื่อให้มั่นใจว่าเฉพาะผู้มีสิทธิ์เท่านั้นที่สามารถแก้ไขข้อมูลสำคัญของเว็บไซต์ได้

การปรับแต่งเว็บไซต์สำหรับเครื่องมือค้นหา (Search Engine Optimization - SEO)

SEO คือกระบวนการปรับปรุงโครงสร้างและเนื้อหาของเว็บไซต์เพื่อให้เป็นมิตรต่อการทำงานของเครื่องมือค้นหา (Search Engines) เช่น Google โดยมีเป้าหมายเพื่อให้เว็บไซต์ติดอันดับที่ดีในการค้นหาแบบทั่วไป (Organic Search) ซึ่งจะช่วยเพิ่มการมองเห็นและจำนวนผู้เข้าชมเว็บไซต์

เทคนิคหนึ่งที่สำคัญสำหรับเว็บแอปพลิเคชันสมัยใหม่คือ การแสดงผลฝั่งเซิร์ฟเวอร์ (Server-Side Rendering - SSR) ซึ่งเซิร์ฟเวอร์จะสร้างหน้าเว็บ HTML ที่มีเนื้อหาครบถ้วนสมบูรณ์ก่อนที่จะส่งไปยังเบราว์เซอร์ของผู้ใช้ วิธีนี้แตกต่างจากการแสดงผลฝั่งไคลเอนต์ (Client-Side Rendering - CSR) ที่เบราว์เซอร์จะได้รับไฟล์ HTML ที่ว่างเปล่าแล้วใช้ JavaScript เพื่อดึงเนื้อหามาแสดงผลทีหลัง

การประยุกต์ใช้ในโครงการ

เพื่อให้บุคคลภายนอก เช่น ผู้ที่สนใจศึกษาต่อ สามารถค้นหาข้อมูลหลักสูตรหรือข่าวสารของคณะฯ ผ่าน Google ได้ง่าย โครงการนี้ได้ประยุกต์ใช้เทคนิค SSR เมื่อ Search Engine Crawler เข้ามาเก็บข้อมูล มันจะเห็นหน้าเว็บที่มีเนื้อหาครบถ้วนทันที ทำให้สามารถจัดทำดัชนี (Indexing) ได้อย่างถูกต้องและมีประสิทธิภาพ ซึ่งส่งผลดีต่ออันดับในการค้นหาโดยตรง