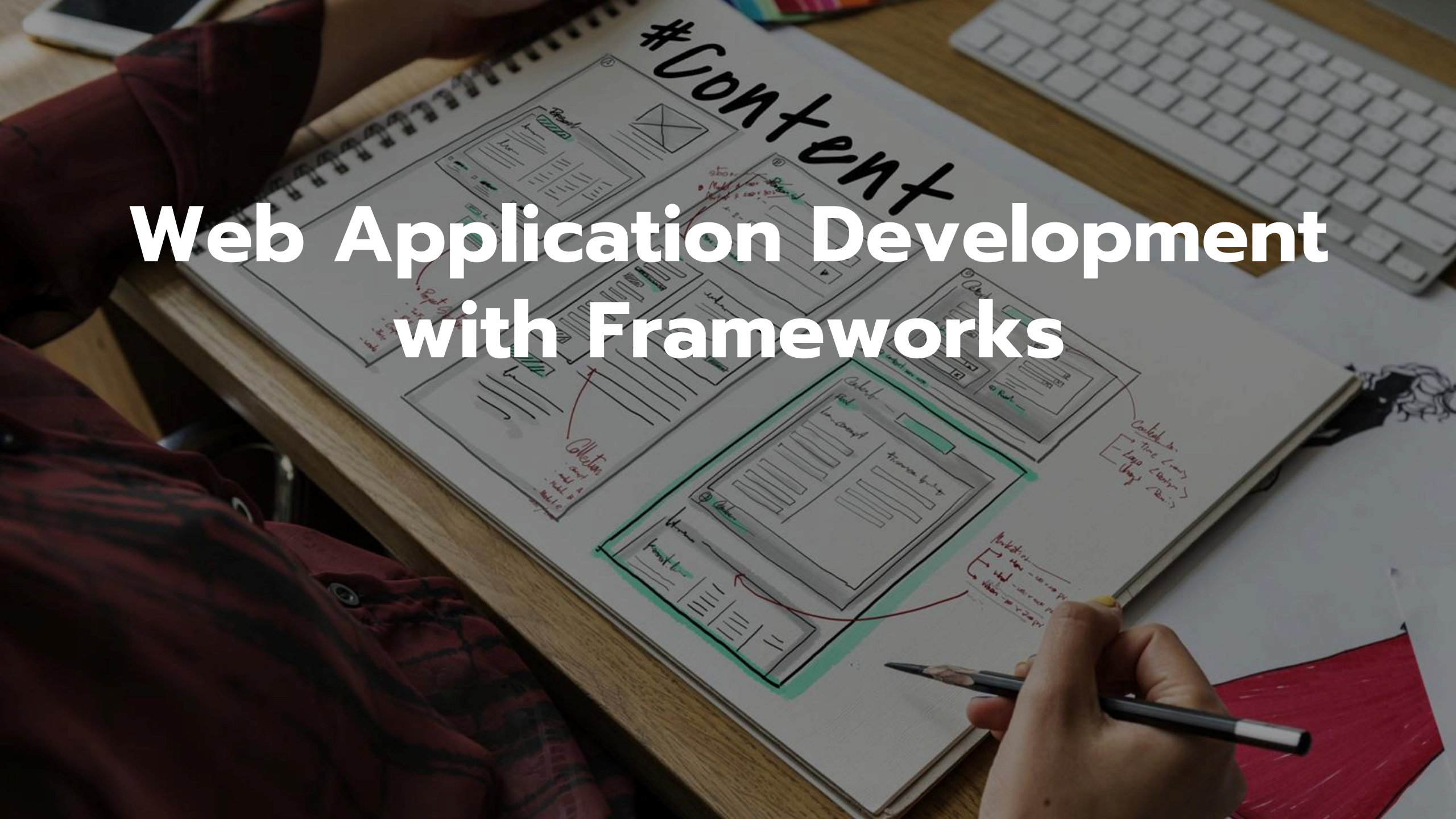
01-406-063-303 การพัฒนาเว็บแอปพลิเคชันด้วยเฟรมเวิร์ค

Web Application Development with Frameworks

3(2-2-5)

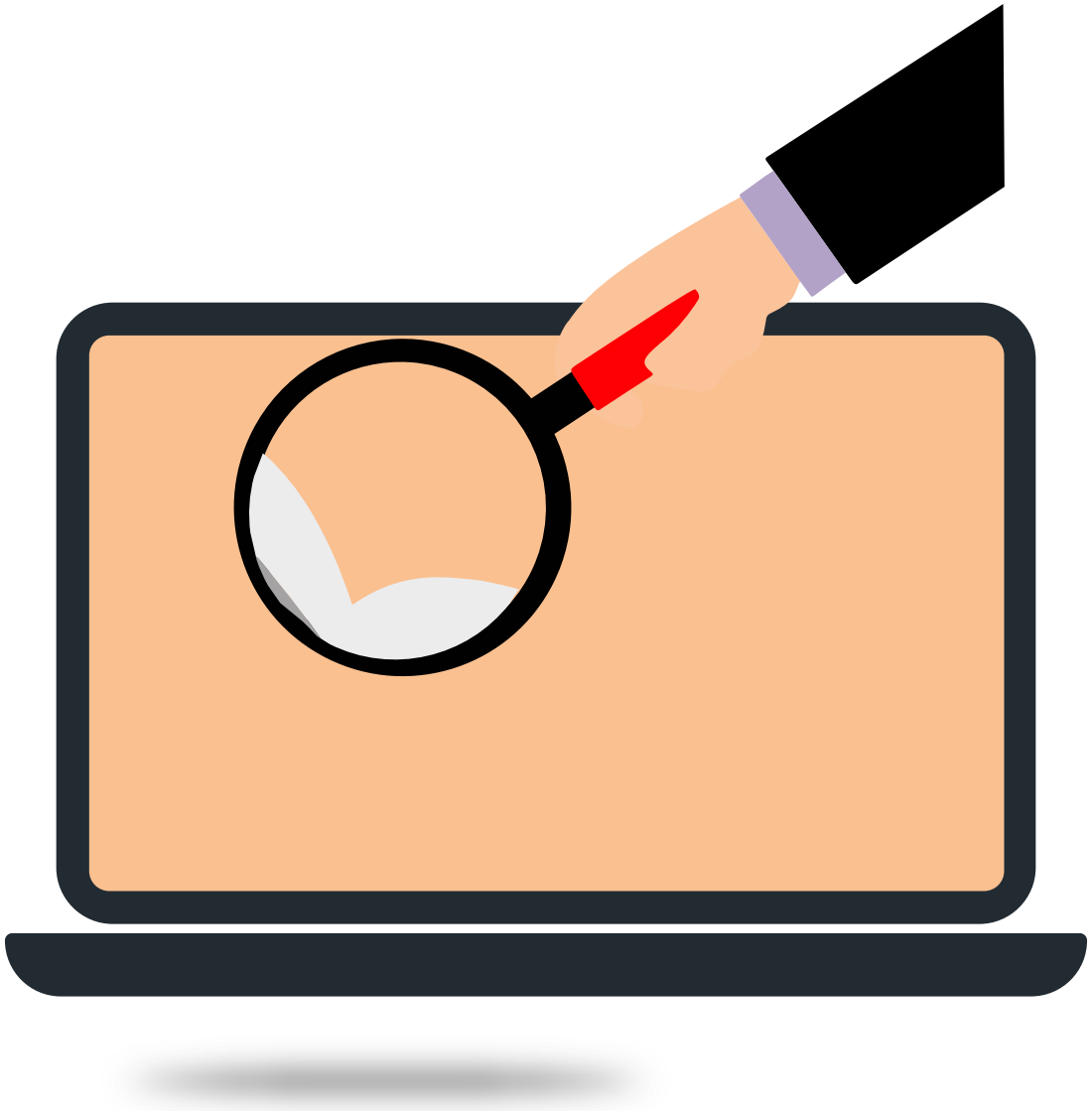
Suda Tipprasert ,Ph.D.

Information System, Business Administration,
Rajamangala University of Technology Isan



Web Application Development with Frameworks

ผู้สอน



อาจารย์ ดร.สุดา ทิพย์ประเสริฐ

สาขาระบบสารสนเทศ คณะบริหารธุรกิจ

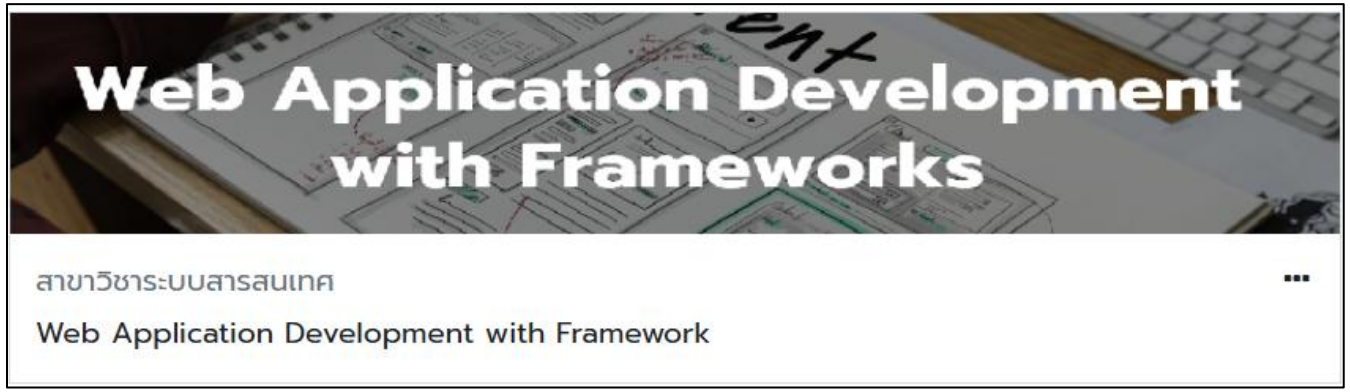
มหาวิทยาลัยเทคโนโลยีราชมงคลอีสาน

E-mail : suda.ti@rmuti.ac.th

0868769035

LMS

https://lms.rmuti.ac.th/course/view.php?id=4165



เข้าตามกลุ่มเรียน
BC4/2A รหัสเข้ากลุ่ม BC-444222a
BC4/2B รหัสเข้ากลุ่ม BC-444222b

ผลลัพธ์การเรียนรู้ของหลักสูตร (PLOs)

PLO 1 เลือกใช้เครื่องมือด้านเทคโนโลยีดิจิทัลที่ผสานกันได้อย่างเหมาะสมต่อบริบททางธุรกิจ

PLO 2 วิเคราะห์ปัญหาในระบบธุรกิจและเลือกวิธีการแก้ปัญหาด้านเทคโนโลยีดิจิทัลที่เหมาะสมได้

PLO 3 ออกแบบนวัตกรรมดิจิทัลด้านธุรกิจได้อย่างสร้างสรรค์

PLO 4 พัฒนานวัตกรรมดิจิทัลด้านธุรกิจโดยประยุกต์ใช้แนวคิดของความเป็นผู้ประกอบการ

PLO 5 มีภาวะผู้นำและผู้ตาม ทำงานเป็นทีมร่วมกับผู้อื่นได้

PLO 6 สื่อสารระหว่างบุคคล และนำเสนองานได้

PLO 7 ตระหนักและเคารพสิทธิของผู้อื่น มีความรับผิดชอบต่อหน้าที่ องค์กร สังคม และสิ่งแวดล้อม

PLO 8 มีคุณธรรม จริยธรรม มีความซื่อสัตย์สุจริต ปฏิบัติตามหลักจรรยาบรรณในวิชาชีพ

คำอธิบายรายวิชา

การพัฒนาเว็บแอปพลิเคชัน เฟรมเวิร์คสำหรับการพัฒนาเว็บแอปพลิเคชัน การเลือกเฟรมเวิร์ค การพัฒนาและการเลือกเอพีไอสำหรับพัฒนาเว็บแอปพลิเคชัน

Web application development; frameworks for web application; frameworks choosing; web application development; choosing an API for web application development

ผลลัพธ์การเรียนรู้ของรายวิชา (CLOs)

CLO1: ใช้หลักการการพัฒนาเว็บแอปพลิเคชันในการเลือกใช้เฟรมเวิร์คสาหรับการพัฒนาเว็บแอปพลิเคชันได้อย่างเหมาะสม

CLO2: พัฒนาเว็บแอปพลิเคชันจากเฟรมเวิร์คและเอพีไอ

CLO3: มีคุณธรรม จริยธรรม มีความซื่อสัตย์สุจริต ปฏิบัติตามหลักจรรยาบรรณในวิชาชีพ

เนื้อหา

Chapter 1: พื้นฐานการพัฒนาเว็บแอปพลิเคชัน
- What is Web Application?
- สถาปัตยกรรมเว็บแอปพลิเคชัน
- สถาปัตยกรรมไคลเอนต์เซิร์ฟเวอร์
- เครื่องมือที่ใช้ในการพัฒนาเว็บแอปพลิเคชัน

Chapter 2: แนวคิดของ Framework สำหรับเว็บ
- ความหมายและประโยชน์ของ Framework
- ประเภทของ Framework: Frontend vs Backend vs Full-Stack
- การเลือกใช้ Framework ที่เหมาะสมกับงาน

Chapter 3: การเริ่มต้นกับ Laravel Framework
- การติดตั้ง Laravel และเครื่องมือที่เกี่ยวข้อง (Composer, XAMPP, etc.)
- โครงสร้างของ Laravel
- Routing และ Controller เบื้องต้น

Chapter 4: Laravel - การจัดการข้อมูล
- การใช้ Blade Template
- การเชื่อมต่อฐานข้อมูล
- การสร้าง Model และ Migration

Chapter 5: Authentication และ Middleware
- การสร้างระบบ Login/Logout ด้วย Laravel Auth
- การป้องกัน route ด้วย Middleware
- การจัดการสิทธิ์ผู้ใช้

Chapter 6: การทำ CRUD ด้วย Laravel
- การสร้างหน้า Create, Read, Update, Delete
- การจัดการฟอร์มและการ Validate ข้อมูล
- ตัวอย่างการใช้งานจริงแบบ Mini Project

Chapter 7: การใช้งาน API และ Web Services
- ความเข้าใจพื้นฐานของ REST/RESTful API
- การเชื่อมต่อ API ภายนอกด้วย Laravel
- เอพีไอสำหรับพัฒนาเว็บแอปพลิเคชัน

Chapter 8: การปรับใช้ (Deployment)
- การ deploy เว็บไปยัง Hosting หรือ Cloud (เช่น Render, Vercel, Hostinger)
- การสรุปองค์ความรู้

โครงสร้างรายสัปดาห์

สัปดาห์ 1: แนะนำรายวิชา – รูปแบบการเรียน, การวัดผล, ความรู้พื้นฐาน

สัปดาห์ 2: พื้นฐานการพัฒนาเว็บ สถาปัตยกรรมเว็บแอป Client/Server

สัปดาห์ 3: แนวคิดของ Framework – ความหมาย, ประเภท, ตัวอย่าง

สัปดาห์ 4: Laravel Framework ติดตั้ง, โครงสร้าง, Routing

สัปดาห์ 5: Laravel (ต่อ) – Blade, Controller, Migration

สัปดาห์ 6-7: Authentication – การจัดการผู้ใช้, JWT, Middleware

สัปดาห์ 8: สอบกลางภาค

สัปดาห์ 9-10: การทำ CRUD – การสร้าง อ่าน แก้ไข ลบ ข้อมูล

สัปดาห์ 11-12: การทำ CRUD – การสร้าง อ่าน แก้ไข ลบ ข้อมูล (ต่อ)

สัปดาห์ 13: API และ Web Services – REST API, การเชื่อมต่อ

สัปดาห์ 14–15: APIDeployment – การเผยแพร่เว็บ, การตั้งค่า Hosting

สัปดาห์ 16: สอบปลายภาค

เกณฑ์การให้คะแนน

วิธีการประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
สอบกลางภาค	8	25%
สอบปลายภาค	17	25%
สอบย่อย	14	15%
จิตพิสัย	ตลอดภาคการศึกษา	10%
งานที่มอบหมายและแบบฝึกหัดประจำหน่วย - .ใบงาน 25%	ตลอดภาคการศึกษา	25%

*** การตัดเกรดใช้วิธีการ อิงเกณฑ์ ***
** จิตพิสัย มาสาย (ไม่ทันเช็คชื่อ 2 ครั้ง = ขาด 1 ครั้ง)
เช็คชื่อประมาณ 15 นาทีแรก หลังจากนั้นถือมาว่าสาย

เกณฑ์ผ่านรายวิชาผู้ที่จะผานรายวิชานี้จะต้อง

-มีเวลาเข้าชั้นเรียนไม่ต่ำกว่าร้อยละ80ของเวลาเรียน

-ได้คะแนนรวมทั้งรายวิชาไม่ต่ำกว่าร้อยละ50ของคะแนนรวม

เกณฑ์การให้คะแนน

คะแนน	เกรด
คะแนนร้อยละ 80 ขึ้นไป	A
คะแนนร้อยละ 75-79	B+
คะแนนร้อยละ 70-74	B
คะแนนร้อยละ 65 -69	C+
คะแนนร้อยละ 60-64	C
คะแนนร้อยละ 55-59	D+
คะแนนร้อยละ 50-54	D
คะแนนร้อยละ 49 ลงไป	F

เวลาเข้าเรียนไม่ต่ำกว่าร้อยละ 80 ขาดได้ไม่เกิน 3 ครั้ง

ส่งงานที่ได้รับมอบหมายทั้งหมด**ครบ**และ**ตรงตามระยะเวลาที่กำหนด**

หากนักศึกษาผู้ใดไม่สามารถเข้าสอบได้ ให้มาติดต่อผู้สอนทันที

กรณี ป่วย,ได้รับอุบัติเหตุ มีกิจธุระสำคัญทางราชการ

จะต้องทำจดหมายลาหรือมีหนังสือชี้แจงล่วงหน้า ทั้งนี้ขึ้นอยู่กับดุลย

พินิจของผู้สอน

**กรณี พบการทุจริตในการสอบ ให้ F ในรายวิชา

แบบทดสอบก่อนเรียน

Chapter 1

พื้นฐานการพัฒนาเว็บแอปพลิเคชัน

Chapter 1 พื้นฐานการพัฒนาเว็บแอปพลิเคชัน

What is Web Application?

Web Application Architecture : สถาปัตยกรรมเว็บแอปพลิเคชัน

Client/Server Architecture : สถาปัตยกรรมไคลเอนต์เซิร์ฟเวอร์

เครื่องมือที่ใช้พัฒนาเว็บแอปพลิเคชัน

What is Web Application?

เว็บแอปพลิเคชัน (Web Application) เป็นซอฟต์แวร์ที่ทำงานบนเว็บเบราว์เซอร์ โดยผู้ใช้สามารถเข้าถึงและใช้งานได้ผ่านอินเทอร์เน็ต ไม่ต้องติดตั้งโปรแกรมลงบนเครื่องคอมพิวเตอร์โดยตรง ทำให้มีความยืดหยุ่นสูงและเข้าถึงได้ง่ายจากทุกที่ทุกเวลา

What is Web Application?

องค์ประกอบหลักของเว็บแอปพลิเคชัน

1 ไคลเอนต์ (Client): ส่วนนี้คือ **เว็บเบราว์เซอร์** ที่ผู้ใช้ใช้งาน เช่น Google Chrome, Mozilla Firefox, Microsoft Edge หรือ Safari ทำหน้าที่ร้องขอข้อมูลและแสดงผลลัพธ์ที่ได้รับจากเซิร์ฟเวอร์

2 เซิร์ฟเวอร์ (Server): คอมพิวเตอร์ที่เก็บไฟล์และรันโปรแกรมของเว็บแอปพลิเคชัน ทำหน้าที่ประมวลผลคำขอจากไคลเอนต์และส่งข้อมูลกลับไป

3 ฐานข้อมูล (Database): ระบบที่ใช้เก็บข้อมูลของเว็บแอปพลิเคชัน เช่น ข้อมูลผู้ใช้, สินค้า, หรือบทความต่างๆ

1 ส่วนติดต่อผู้ใช้ (Frontend)

คือส่วนที่ผู้ใช้เห็นและโต้ตอบ เช่น หน้าเว็บ ฟอร์มต่างๆ เทคโนโลยีที่ใช้ HTML, CSS, JavaScript, Frameworks (เช่น React, Vue)

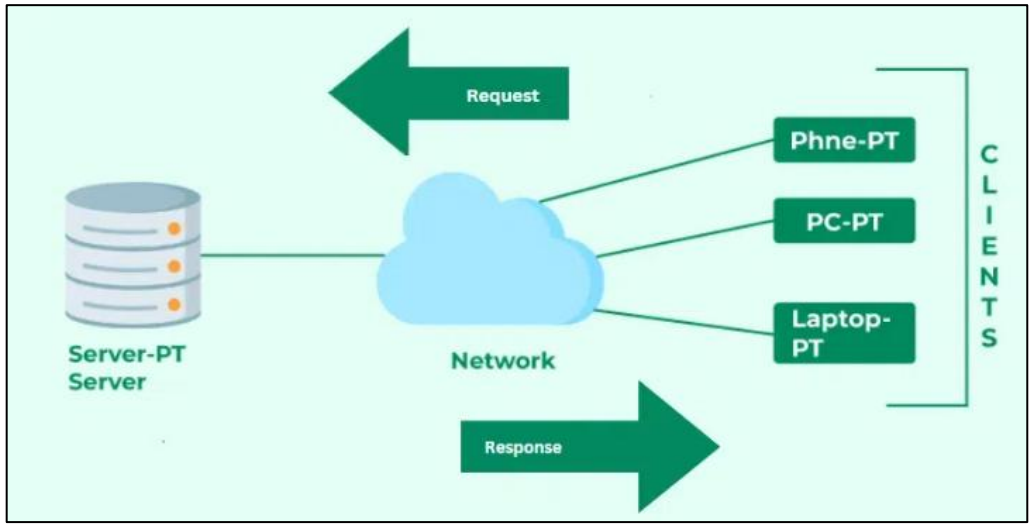
2 ส่วนประมวลผล (Backend)

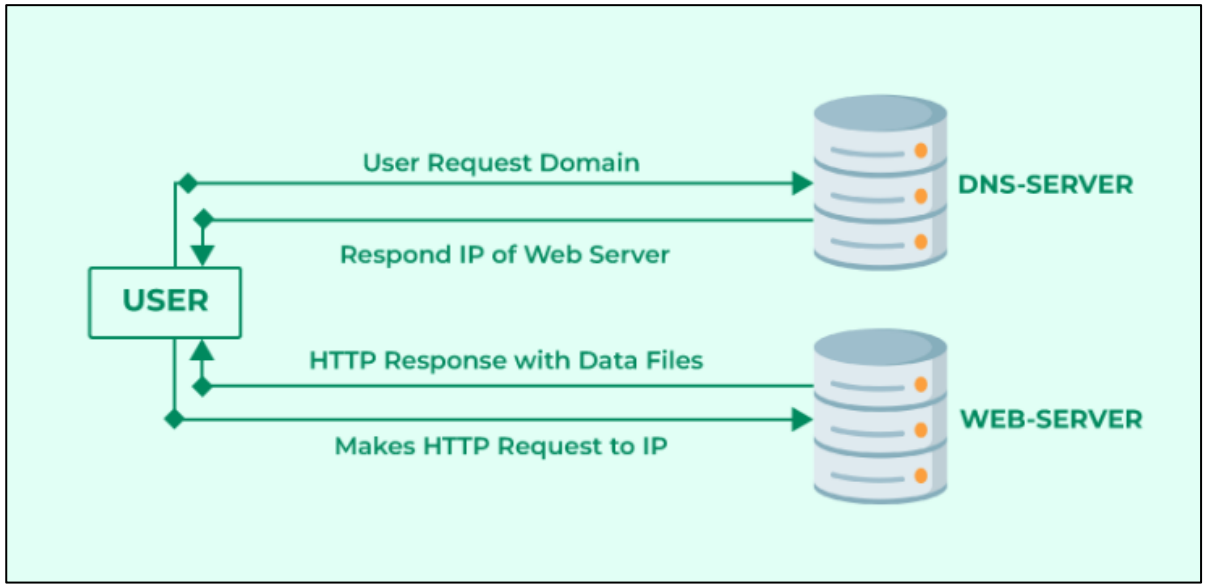
ทำหน้าที่ประมวลผลคำสั่ง ติดต่อฐานข้อมูล และควบคุมการทำงานของระบบ เทคโนโลยีที่ใช้ PHP, Python, Node.js, Laravel, Django เป็นต้น

3 ฐานข้อมูล (Database)

เก็บข้อมูลของระบบ เช่น ข้อมูลผู้ใช้ รายการสินค้า ระบบฐานข้อมูลยอดนิยม MySQL, PostgreSQL, MongoDB

แนวคิดของ Web Application





การทำงานของเว็บแอปพลิเคชัน

การทำงานของเว็บแอปพลิเคชันสามารถสรุปขั้นตอนง่ายๆ ได้ดังนี้

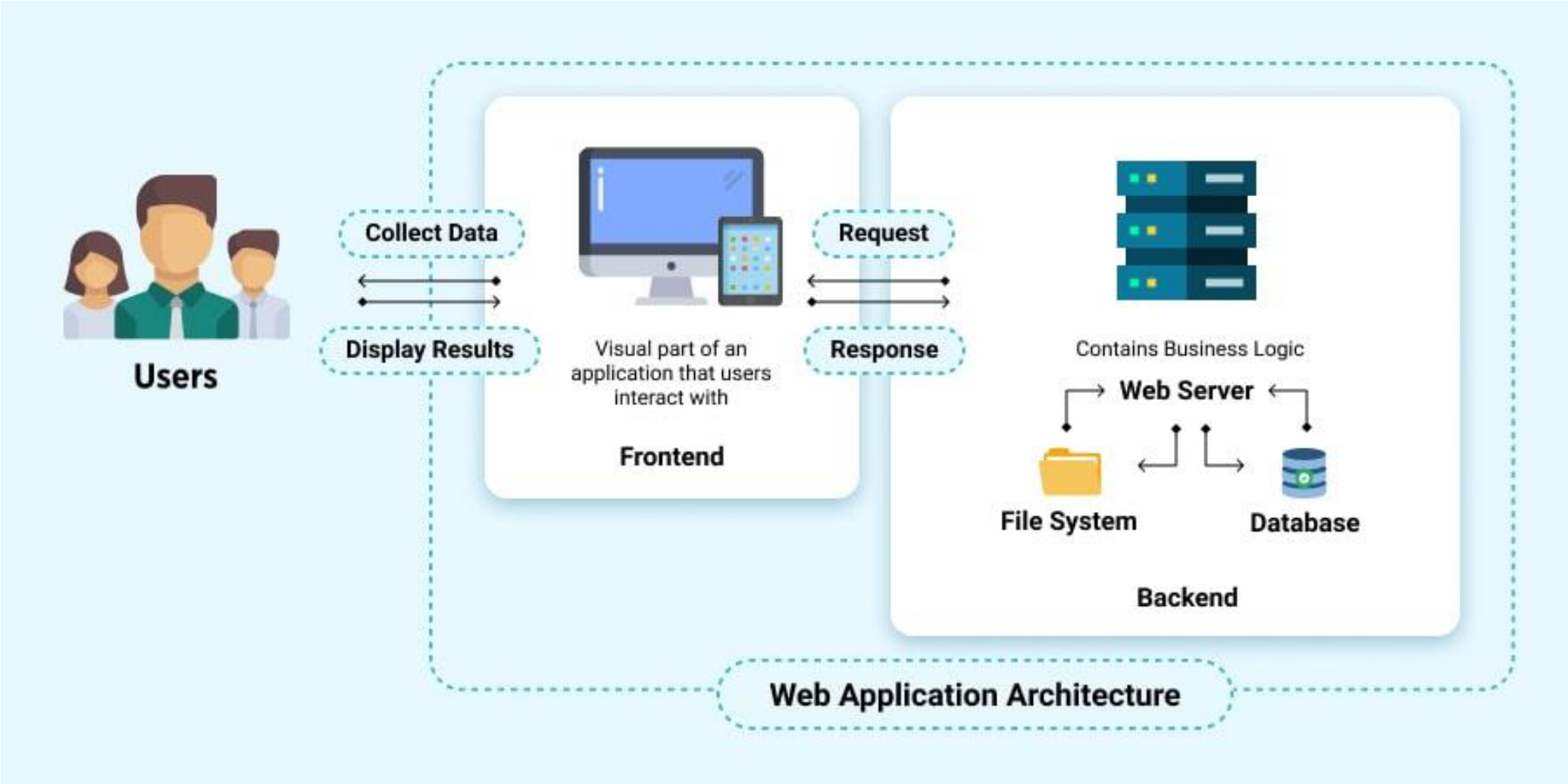
1. ผู้ใช้ร้องขอ (Request) ผู้ใช้เปิดเว็บเบราว์เซอร์และพิมพ์ URL ของเว็บแอปพลิเคชัน หรือคลิกที่ลิงก์ ซึ่งเป็นการส่งคำขอไปยังเซิร์ฟเวอร์

2. เซิร์ฟเวอร์ประมวลผล (Process) เซิร์ฟเวอร์ได้รับคำขอ จะทำการประมวลผลข้อมูลตามที่ร้องขอ อาจมีการเรียกใช้ข้อมูลจากฐานข้อมูล หรือรันโค้ดโปรแกรมเพื่อสร้างผลลัพธ์

3. เซิร์ฟเวอร์ส่งการตอบกลับ (Response) หลังจากประมวลผลเสร็จสิ้น เซิร์ฟเวอร์จะส่งข้อมูลกลับไปยังเว็บเบราว์เซอร์ของไคลเอนต์ โดยมักอยู่ในรูปแบบของไฟล์ HTML, CSS, JavaScript, รูปภาพ หรือข้อมูลอื่นๆ

4. เว็บเบราว์เซอร์แสดงผล (Render) เว็บเบราว์เซอร์ได้รับข้อมูลและทำการแปลความหมาย (render) เพื่อแสดงผลลัพธ์ให้ผู้ใช้เห็นบนหน้าจอ

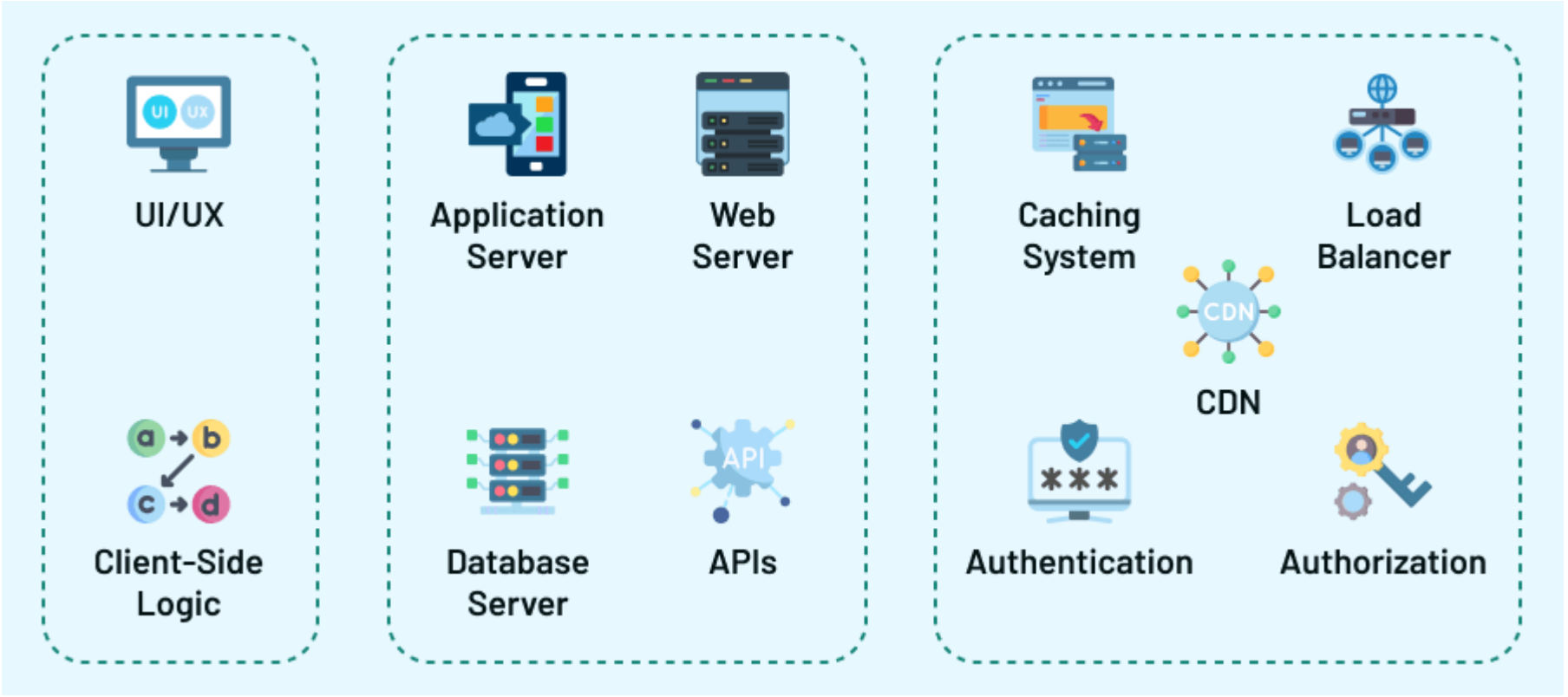
Web Application Architecture สถาปัตยกรรมเว็บแอปพลิเคชัน



https://www.velvetech.com/blog/web-application-architecture/

Web Application Architecture สถาปัตยกรรมเว็บแอปพลิเคชัน

องค์ประกอบของสถาปัตยกรรมเว็บแอปพลิเคชัน Main Components of Web Application Architecture



Web Application Architecture สถาปัตยกรรมเว็บแอปพลิเคชัน

Frontend Components

สามารถแบ่งออกได้เป็น 2 องค์ประกอบ คือ

1.การออกแบบหน้าจอ (UX/UI Design) – เป็นส่วนที่ผู้ใช้มองเห็นและโต้ตอบกับแอปพลิเคชัน ซึ่งสามารถสร้างขึ้นโดยใช้ **HTML, CSS และ JavaScript**

2.ตรรกะฝั่งไคลเอนต์ (Client-side logic) – หรือก็คือโค้ดที่ทำงานภายในเบราว์เซอร์ เพื่อจัดการพฤติกรรมของ UI และการโต้ตอบต่าง ๆ โดยใช้ **Frontend Frameworks** เช่น **React, Angular หรือ Vue**

Web Application Architecture สถาปัตยกรรมเว็บแอปพลิเคชัน

Backend Components

- **เว็บเซิร์ฟเวอร์ (Web Server)** ทำหน้าที่จัดการและประมวลผลคำร้องขอ (Request) จากฝั่ง Frontend และส่งผลลัพธ์กลับไป (Response) ตัวอย่างของเว็บเซิร์ฟเวอร์ ได้แก่ **Apache, Nginx หรือ Node.js**
- **แอปพลิเคชันเซิร์ฟเวอร์ (Application Server)** เป็นส่วนที่เก็บ **ตรรกะทางธุรกิจ (Business Logic)** ซึ่งจะประมวลผลคำร้องขอ ใช้กฎต่าง ๆ และจัดการกับกระบวนการที่ซับซ้อน โดยอาจพัฒนาแบ็กเอนด์ขึ้นเองด้วยภาษา **Python, Java หรือ .NET** ร่วมกับเฟรมเวิร์ก เช่น **Django, Express, Laravel, Spring**
- **เซิร์ฟเวอร์ฐานข้อมูล (Database Server)** ใช้ในการเก็บและจัดการข้อมูลของแอปพลิเคชัน โดยอาจเป็นฐานข้อมูลแบบ **SQL** เช่น **MySQL, PostgreSQL** หรือแบบ **NoSQL** เช่น **MongoDB และ Cassandra**
- **API (Application Programming Interface)** ช่วยให้ระบบต่าง ๆ สามารถสื่อสารกันได้ ไม่ว่าจะเป็นการเชื่อมต่อระหว่างบริการต่าง ๆ หรือระหว่าง Frontend กับ Backend โดย API อาจเป็นแบบภายในหรือภายนอก และมีหลายประเภท เช่น **RESTful, GraphQL หรือ SOAP** ขึ้นอยู่กับความต้องการของแอปพลิเคชัน

Web Application Architecture สถาปัตยกรรมเว็บแอปพลิเคชัน

Optional Components

Optional Components คือส่วนประกอบเสริม ที่ไม่จำเป็นต้องมีทุกระบบ

- ❑ ระบบแคช (Caching System) ช่วยให้การดึงข้อมูลที่ใช้งานบ่อยทำได้เร็วขึ้น ส่งผลให้ประสิทธิภาพโดยรวมดีขึ้น
- ❑ ตัวกระจายโหลด (Load Balancer) ช่วยกระจายทราฟฟิกที่เข้ามายังเซิร์ฟเวอร์หลายเครื่อง เพื่อใช้ทรัพยากรอย่างมีประสิทธิภาพ เพิ่มความเสถียร และป้องกันการทำงานหนักเกินไปของเซิร์ฟเวอร์
- ❑ CDN (Content Delivery Network) ช่วยกระจายเนื้อหาไปใกล้กับตำแหน่งของผู้ใช้มากขึ้น เพื่อลดเวลาในการโหลด เหมาะสำหรับไฟล์แบบสแตติก เช่น รูปภาพหรือวิดีโอ
- ❑ ระบบยืนยันตัวตนและการกำหนดสิทธิ์ (Authentication & Authorization) ช่วยให้การเข้าถึงแอปปลอดภัย โดยมักจัดการผ่านผู้ให้บริการระบุตัวตน เช่น OAuth, JWT (JSON Web Token) ซึ่งเป็นวิธีการยืนยันตัวตนและการกำหนดสิทธิ์แบบ stateless ที่นิยมใช้ใน Web Application สมัยใหม่ โดยเฉพาะในระบบที่มี API

สถาปัตยกรรม Client/Server

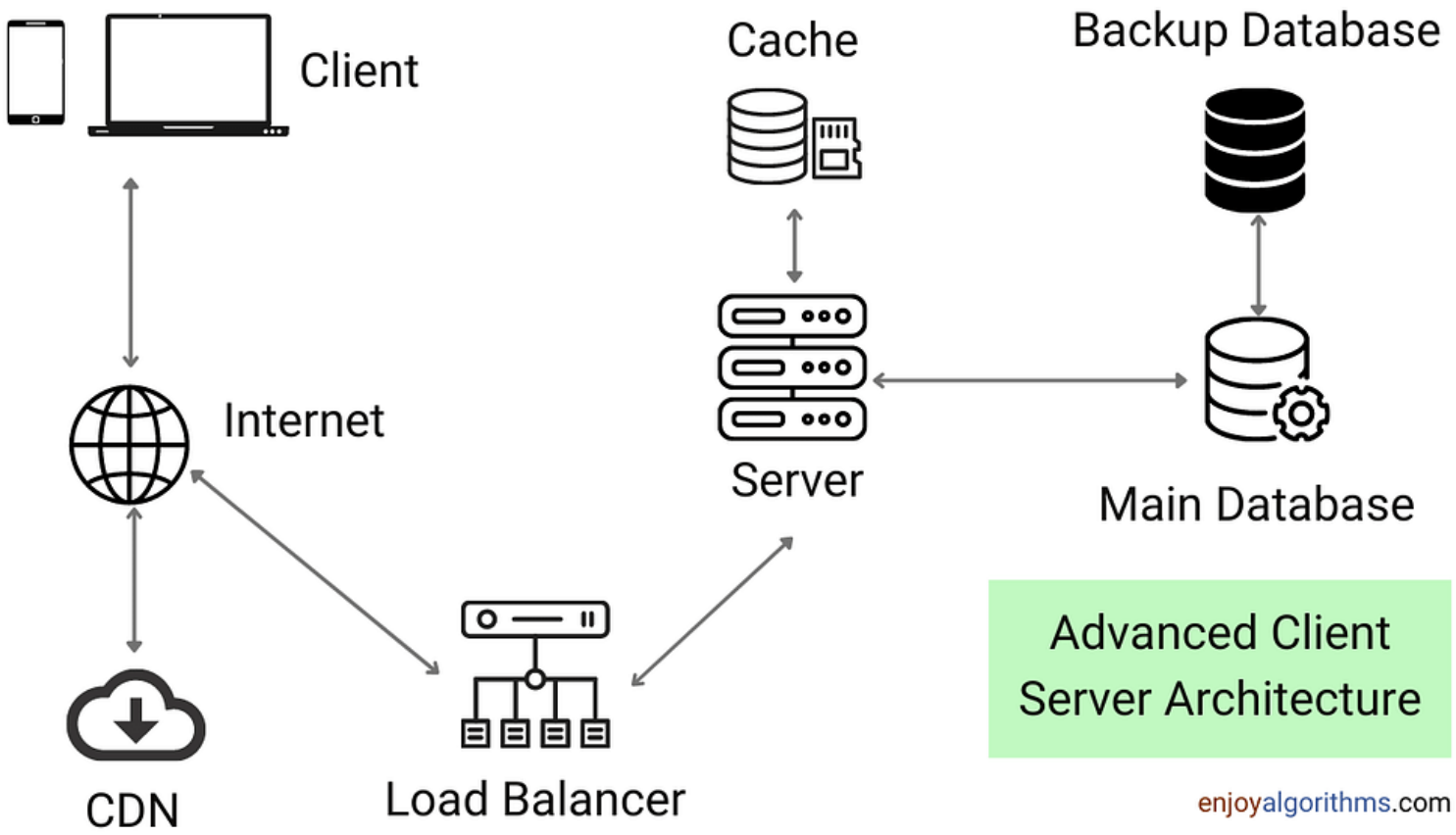
What is client-server architecture?

สถาปัตยกรรมแบบไคลเอนต์-เซิร์ฟเวอร์ เป็นระบบแบบกระจาย (distributed system) ประเภทหนึ่งที่ประกอบด้วยไคลเอนต์และเซิร์ฟเวอร์ โดยเซิร์ฟเวอร์มีหน้าที่เป็นโฮสต์ (ให้บริการ), จัดการ และส่งมอบบริการต่าง ๆ ให้กับไคลเอนต์ ในระบบนี้ ไคลเอนต์จะเชื่อมต่อกับเซิร์ฟเวอร์และสื่อสารกันผ่านเครือข่ายคอมพิวเตอร์โดยใช้อินเทอร์เน็ต ดังนั้น เมื่อไคลเอนต์ต้องการใช้บริการใด ๆ ก็จะส่งคำร้องขอไปยังเซิร์ฟเวอร์ ซึ่งเซิร์ฟเวอร์จะทำการประมวลผลคำร้องขอนั้น และส่งผลลัพธ์กลับมายังไคลเอนต์

https://www.enjoyalgorithms.com/blog/client-server-architecture

สถาปัตยกรรม Client/Server

Components of client-server architecture



https://www.enjoyalgorithms.com/blog/client-server-architecture

สถาปัตยกรรม Client/Server

Components of client-server architecture

สถาปัตยกรรมแบบไคลเอนต์-เซิร์ฟเวอร์ที่ซับซ้อนอาจมีหลายองค์ประกอบ โดยในระดับพื้นฐานจะมีองค์ประกอบสำคัญอยู่ 4 ส่วน ได้แก่ ไคลเอนต์ (Client), ตัวกระจายโหลด (Load Balancer), เซิร์ฟเวอร์ (Servers), และโพรโตคอลเครือข่าย (Network Protocols)

•**ไคลเอนต์ (Client)** คือซอฟต์แวร์ที่ร้องขอทรัพยากรและบริการที่เซิร์ฟเวอร์จัดเตรียมไว้ให้ โดยจะทำงานบนคอมพิวเตอร์ของผู้ใช้หรือเครื่องระยะไกล และเชื่อมต่อกับเซิร์ฟเวอร์

•**เซิร์ฟเวอร์ (Server)** คือโปรแกรมซอฟต์แวร์ที่รับและประมวลผลคำร้องขอจากไคลเอนต์ โดยจะทำงานอยู่บนเครื่องระยะไกลและสามารถเข้าถึงได้จากคอมพิวเตอร์หรือเวิร์กสเตชันของผู้ใช้ ตัวอย่างเช่น เว็บเซิร์ฟเวอร์ คือเซิร์ฟเวอร์ชนิดหนึ่งที่รับและประมวลผลคำร้องขอจากเบราวเซอร์

https://www.enjoyalgorithms.com/blog/client-server-architecture

สถาปัตยกรรม Client/Server

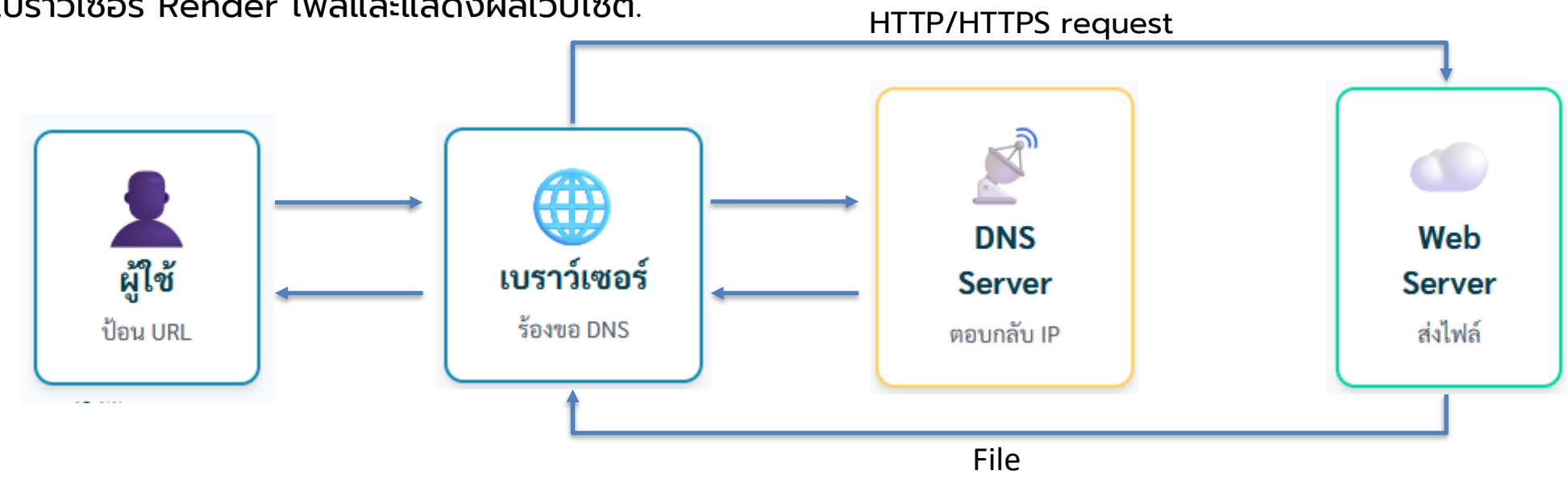
Components of client-server architecture

- **ตัวกระจายโหลด (Load Balancer)** เป็นระบบหรืออุปกรณ์ที่ช่วยกระจายคำร้องขอจากไคลเอนต์ไปยังเซิร์ฟเวอร์หลายเครื่อง เพื่อให้การทำงานมีประสิทธิภาพมากขึ้น ป้องกันไม่ให้เซิร์ฟเวอร์เครื่องใดเครื่องหนึ่งรับภาระมากเกินไป และช่วยเพิ่มความเสถียรของระบบ
- **โพรโตคอลเครือข่าย (Network Protocols)** เป็นชุดของกฎหรือมาตรฐานที่ใช้สำหรับการสื่อสารระหว่างไคลเอนต์และเซิร์ฟเวอร์ ตัวอย่างเช่น HTTP (สำหรับเว็บ), TCP/IP (สำหรับการรับส่งข้อมูลผ่านอินเทอร์เน็ต) ซึ่งช่วยให้ข้อมูลถูกส่งและรับอย่างถูกต้องและปลอดภัย

สถาปัตยกรรม Client/Server

client-server model ทำงานอย่างไร

1. ผู้ใช้ป้อน URL ของเว็บไซต์ลงในเบราว์เซอร์
2. เบราว์เซอร์ส่งคำขอไปยัง DNS server เพื่อค้นหา IP Address ของ Web Server
3. DNS server ส่ง IP Address กลับไปที่เบราว์เซอร์ เบราว์เซอร์ส่ง HTTP/HTTPS request ไปยัง IP Address ของ Web Server
4. Web Server ส่งไฟล์ที่จำเป็นสำหรับเว็บไซต์กลับไปยังเบราว์เซอร์
5. เบราว์เซอร์ Render ไฟล์และแสดงผลเว็บไซต์.



Web application development tools : เครื่องมือที่ใช้พัฒนาเว็บแอปพลิเคชัน

1. Frontend Frameworks & Libraries These are essential for building the user interface and user experience (UI/UX) of a web application. **React.js, Next.js, Tailwind CSS, Bootstrap**

NEXT.JS

Web application development tools : เครื่องมือที่ใช้พัฒนาเว็บแอปพลิเคชัน

2. Backend Frameworks & Runtimes: These handle the server-side logic, database interactions, and API development. **Node.js, Express.js, Django (Python), Flask (Python), Laravel (PHP)**, **ASP.NET Core (C#), Spring Framework (Java)**

Web application development tools : เครื่องมือที่ใช้พัฒนาเว็บแอปพลิเคชัน

3. Development Environments & Code Editors: These are the primary interfaces for writing and managing code. **Visual Studio Code (VS Code), Sublime Text, WebStorm, Chrome DevTools**

Web application development tools : เครื่องมือที่ใช้พัฒนาเว็บแอปพลิเคชัน

4. Version Control & Collaboration: Essential for managing code changes and facilitating team collaboration. **GitHub/GitLab/Bitbucket**

Web application development tools : เครื่องมือที่ใช้พัฒนาเว็บแอปพลิเคชัน

5. Containerization & Orchestration: For deploying and managing applications consistently across different environments.

•**Docker:** An open-source platform that allows developers to package applications and their dependencies into portable containers, ensuring consistent deployment.

•**Kubernetes (K8s):** An open-source system for automating the deployment, scaling, and management of containerized applications.

Web application development tools : เครื่องมือที่ใช้พัฒนาเว็บแอปพลิเคชัน

6. Package Managers & Task Runners: Tools for managing project dependencies and automating repetitive tasks.

•**npm (Node Package Manager):** The default package manager for Node.js, used for installing and managing JavaScript packages.

•**Grunt/Gulp:** JavaScript task runners used to automate repetitive development tasks like minification, compilation, and testing.

Web application development tools : เครื่องมือที่ใช้พัฒนาเว็บแอปพลิเคชัน

7. Web Servers / Web Servers Package

Web Servers คำว่า "เว็บเซิร์ฟเวอร์" สามารถหมายถึงได้ทั้ง ฮาร์ดแวร์ และ ซอฟต์แวร์ ที่ทำงานร่วมกัน หรือโปรแกรมที่เข้าใจและตอบสนองต่อคำขอจากเว็บเบราว์เซอร์ ซอฟต์แวร์ที่สำคัญที่สุดคือ **เซิร์ฟเวอร์ HTTP** (HTTP server) โดย HTTP (Hypertext Transfer Protocol) เป็นภาษาที่เว็บเบราว์เซอร์และเว็บเซิร์ฟเวอร์ใช้ในการสื่อสารกัน Web Server ทำหน้าที่ จัดเก็บไฟล์เว็บไซต์ทั้งหมดและส่งมอบไปยังเบราว์เซอร์ของไคลเอ็นต์เมื่อได้รับการร้องขอผ่าน HTTP,HTTPS Protocol

- **Nginx, Apache HTTP Server, Caddy, Microsoft IIS**

Web Servers Package

Web Servers Package คือ คือชุดรวมซอฟต์แวร์ (software bundle) ที่ประกอบด้วยเว็บเซิร์ฟเวอร์และส่วนประกอบสำคัญอื่นๆ ที่จำเป็นต่อการรันเว็บแอปพลิเคชัน

XAMPP, WAMP/MAMP/LAMP (specific distributions), Laragon (for Windows), Docker Desktop (and Docker Compose)

Web application development tools : เครื่องมือที่ใช้พัฒนาเว็บแอปพลิเคชัน

8. Databases

To store and manage application data.

•**Relational Databases (SQL):**

- **MySQL:** Popular open-source choice, widely used.
- **PostgreSQL:** Advanced open-source relational database, known for robustness and features.
- **SQL Server:** Microsoft's relational database management system, often used with ASP.NET Core.
- **SQLite:** Embedded database, good for smaller applications or local development.

•**NoSQL Databases:**

- **MongoDB:** Document-oriented database (JSON-like documents), flexible schema, good for rapidly changing data.
- **Redis:** In-memory data store, often used for caching, session management, and real-time data. ใช้สำหรับจัดเก็บข้อมูลในหน่วยความจำ เช่น session หรือ cache
- **Cassandra:** Distributed NoSQL database for handling large amounts of data across many servers.
- **Firebase/Firestore:** Google's NoSQL document database, often used for real-time applications and mobile backends.

Web application development tools : เครื่องมือที่ใช้พัฒนาเว็บแอปพลิเคชัน

9. API Development & Testing Tools

•**Postman:** Widely used tool for designing, testing, and documenting APIs.

•**Swagger/OpenAPI:** Standard for defining RESTful APIs, enabling automated documentation and client code generation. ใช้นิยาม/ทดสอบ RESTful

•**GraphQL:** A query language for APIs and a runtime for fulfilling those queries with your existing data. Provides more efficient data fetching than traditional REST.

Web application development tools : เครื่องมือที่ใช้พัฒนาเว็บแอปพลิเคชัน

10. Cloud Platforms & Serverless

For hosting and deploying web applications, often reducing server management overhead.

•**AWS (Amazon Web Services):** Offers a vast array of services including EC2 (virtual servers), Lambda (serverless functions), RDS (managed databases), S3 (object storage).

•**Azure (Microsoft Azure):** Microsoft's cloud computing platform with similar offerings (Virtual Machines, Azure Functions, Azure SQL Database).

•**Google Cloud Platform (GCP):** Google's suite of cloud services (Compute Engine, Cloud Functions, Cloud SQL, Firestore).

•**Heroku:** Platform-as-a-Service (PaaS) that simplifies deployment for various languages/frameworks.

•**Vercel / Netlify:** Primarily for frontend deployment with serverless functions for backend logic (often used with Next.js or other static site generators).

ใบงานที่ 1