ระบบเช็คชื่อด้วยคิวอาร์โค้ด
QR Code check name

นายเจตพล ชินพันธ์

โครงงานนี้เป็นส่วนหนึ่งของการศึกษา
หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ คณะวิทยาศาสตร์
มหาวิทยาลัยอุบลราชธานี
ปีการศึกษา 2562

โครงงาน	:	ระบบเช็คชื่อด้วยคิวอาร์โค้ด QR Code check name
โดย	:	นายเจตพล ชินพันธ์
	:	
อาจารย์ที่ปรึกษา	:	ดร. เกรียงศักดิ์ ตรีประพิณ
ระดับการศึกษา	:	วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ปีการศึกษา	:	2562

ได้รับการพิจารณาให้เป็นส่วนหนึ่งของการศึกษา

ตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์

คณะกรรมการสอบประเมินความรู้โครงงานคอมพิวเตอร์

. . . . . อาจารย์ที่ปรึกษา
(ดร. เกรียงศักดิ์ ตรีประพิณ)

. . . . . กรรมการ
(ดร.ทศพร จูฉิม)

. . . . . กรรมการ
(ดร.ไพชยนต์ คงไชย)

. . . . . หัวหน้าภาควิชา
(ผศ.ดร. สุพจน์ สีบุตร)

วันที่ . . . / . . . / . . .

กิตติกรรมประกาศ

การพัฒนาโครงงานระบบเช็คชื่อด้วยคิวอาร์โค้ด สำเร็จลุล่วงได้ด้วยความกรุณาและความช่วยเหลือจากหลายๆ ท่าน ข้าพเจ้าขอขอพระคุณทุกท่าน ที่มีส่วนร่วมในการพัฒนาโครงงานนี้

ขอขอบพระคุณอาจารย์ที่ปรึกษา ดร.เกรียงศักดิ์ ตรีประพิณ อาจารย์ที่ปรึกษาโครงงานที่ได้แนะนำทฤษฎีและแนวทางในแก้ปัญหาต่าง ๆ ที่เกิดขึ้นระหว่างการพัฒนาระบบ อีกครั้งยังคอยตรวจสอบความก้าวหน้าของการทำงานเป็นระยะ ๆ รวมทั้งสร้างกำลังใจให้ผู้พัฒนาอยู่เสมอ

ขอขอบพระคุณเจ้าหน้าที่ภาควิชาคณิตศาสตร์ สถิติและคอมพิวเตอร์ ที่คอยเอื้อยอำนวยความสะดวกทั้งเรื่องอุปกรณ์และสถานที่ต่อการปฏิบัตงานของผู้พัฒนา

ขอกราบขอบพระคุณบิดา มารดา ที่คอยให้กำลังใจ คอยให้ความรักและความห่วงใยเสมอมา ตลอดจนคอยช่วยเหลือทุนทรัพย์ทางด้านการศึกษาและอุปกรณ์ในการพัฒนาโครงงาน

นายเจตพล ชินพันธ์

เมษายน 63

โครงงาน	:	ระบบเช็คชื่อด้วยคิวอาร์โค้ด
โดย	:	นายเจตพล ชินพันธ์
	:	
อาจารย์ที่ปรึกษา	:	ดร. เกรียงศักดิ์ ตรีประพิณ
ระดับการศึกษา	:	วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ปีการศึกษา	:	2562

บทคัดย่อ

โครงการนี้มีวัตถุประสงค์เพื่อพัฒนาการเรียนการสอนโดยสารเว็บแอปพลิเคชันด้วย Angular เพื่อเช็คชื่อในการเข้าเรียนซึ้งเป็นส่วนสำคัญในการให้คะแนนเสริมช่วยนักศึกษาและพัฒนานักศึกษาเกี่ยวกับวินัยในการมาเรียนให้ตรงเวลา โดยในเว็บแอปพลิเคชันนี้จะเป็นการจัดการข้อมูลคะแนนการมาเรียนของนักศึกษาซึ้งระบบจะมุ่งเน้นความรวดเร็วในการใช้งานและการเก็บข้อมูล นอกจากนี้ยังมีการนำเทคโนโลยี QRcode มาใช้ในระบบ โดยการให้อาจารย์เป็นผู้สร้าง QRcode และ ให้นักศึกษาเป็นผู้สแกนซึ้งนักศึกษาเพียงแค่สแกน QRcode ของอาจารย์และกรอกรหัสนักศึกษาเพื่อยืนยันตัวตนก็เป็นอันเสร็จสิ้นการเช็คชื่อ

คำสำคัญ: เว็บแอปพลิเคชัน เช็คชื่อเข้าเรียนด้วยคิวอาร์โค้ด

Topic	:	QR Code check name
Author	:	MR.JADTAPHON CHINNAPAN
	:	
Advisor	:	Kriengsak Treeprapin, Ph.D.
Degree	:	Bachelor of Science (Computer Science)
Academic Year	:	2019

Abstract

QRCode check name

The program aims to improve the teaching and learning of web applications with Angular to check attendance names, which are an important part of providing extra points to help students and students to develop discipline. time ,In this web application, it will manage student enrollment score, which will focus on speed of usage and data collection. In addition, QRcode technology is introduced in the system by having teachers create QRcode and have the students scan the students. Simply scan the QRcode of the professor and enter the student code to verify identity, then the name check is complete.

Keywords: Web Application QRCode check name

สารบัญ

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

บทที่ 1

บทนำ

1.1 ที่มาและเหตุผล

การเช็คชื่อเข้าเรียน เป็นสิ่งสำคัญในการตรวจสอบความกระตือรือร้นในการเข้าเรียนของนักศึกษา และเป็นตัวชี้วัดการให้คะแนนนักศึกษาของผู้สอนหรือเช็คยอดนักศึกษาที่เข้าเรียนในคลาสนั่นๆ ซึ่งการเช็คชื่อเป็นสิ่งสำคัญที่ช่วยคณะกรรมการตรวจสอบว่าและเป็นหลักฐานในการให้คะแนนของผู้สอนที่จะแสดงต่อคณะกรรมการเมื่อมีการตรวจสอบแหล่งที่มาของคะแนน

การเช็คชื่อในคลาสเรียนส่วนมากเป็นการเรียกชื่อนักศึกษาแล้วให้นักศึกษาขานตอบ ซึ่งจะทำให้การเช็คชื่อนั่นมีความล่าช้าและทำให้เสียเวลาในการเรียนการสอน ซึ่งเวลาที่เสียไปในการเช็คชื่อนักศึกษาสามารถนำไปใช้ทำสิ่งต่างๆ ได้ เช่น พักครึ่ง ทำแบบฝึกหัด ติว ตอบคำถามและเตรียมตัวสำหรับคาบต่อไป เป็นต้น นอกจากนี้ผู้สอนบ้างท่านอาจจะใช้วิธีสุ่มเช็คชื่อเพื่อลดเวลาในที่ใช้ในการเช็คชื่อในบ้างคาบเรียน แต่นั้นทำให้นักศึกษาส่วนมากคิดว่าผู้สอนอาจจะไม่เช็คชื่อในคาบเรียนนั่นจึงไม่เข้าเรียน และมีผู้สอนบ้างท่านยังใช้กระดาษในการเช็คชื่อ ซึ่งเป็นการสิ้นเปลืองกระดาษและอาจทำให้กระดาษชำรุด หรือสูญหาย ส่งผลทำให้ไม่มีหลักฐานในการให้คะแนนนักศึกษาที่เข้าเรียน และยังไม่สามารถระบุแหล่งที่มาของคะแนนที่ให้นักศึกษาต่อคณะกรรมการอีกด้วย

จากปัจจัยที่กล่าวมานั่น ผู้พัฒนาจึงมีความคิดที่จะพัฒนาระบบเช็คชื่อเข้าเรียน โดยจะเน้นให้ระบบใช้งานง่ายไม่สับซ้อน เข้าถึงง่าย และมีความรวดเร็ว โดยใช้เทคโนโลยี Quick Response Code (QR-Code) เพื่อให้ง่ายต่อการใช้งาน ซึ่งจะให้นักศึกษาสแกน QR-Code ของผู้สอน โดยผู้สอนจะเปิด QR-Code ให้นักศึกษาทุกคนในคลาสเรียนสแกนเพื่อเป็นการเช็คชื่อเข้าเรียน นอกจากนี้นักศึกษาไม่จำเป็นต้องโหลดแอพเพิ่มก็สามารถเช็คชื่อเข้าเรียนได้

1.2 วัตถุประสงค์

1. เพื่อพัฒนาระบบแอพการเช็คชื่อในคาลสเรียนที่มีขนาดใหญ่ มีนักศึกษาเยอะและลดระยะเวลาที่ใช้ในการเช็ดชื่อนักศึกษา
2. เพื่อเป็นแหล่งเก็บข้อมูลการเข้าเรียนของนักศึกษาและเป็นหลักฐานแสดงการมาเรียนของนักศึกษา ซึ่งจะเป็นแหล่งที่มาของคะแนนการเข้าเรียน
3. เพื่อลดการใช้กระดาษที่ใช้ในการเช็คชื่อ และป้องกันการสูดหายหรือชำรุดของข้อมูล

1.3 ขอบเขตของโครงงาน

1.3.1 ผู้สอน

- สามารถเพิ่ม และลบ คลาสเรียน
- สามารถตรวจสอบรายชื่อนักศึกในคลาสเรียน
- สามารถจัดการคะแนนของนักศึกษา
- สามารถสร้าง QR Code สำหรับให้นักศึกษาสแกนเพื่อเช็คชื่อได้
- สามารถเพิ่ม ลบและแก้ไข ชื่อนักศึกษาในคลาสเรียนได้
- สามารถรวมไฟล์คะแนน ของคลาสเรียนเดียวกัน

1.3.2 นักศึกษา

- สแกน QR-Code ของผู้สอนเพื่อเป็นการเช็คชื่อเข้าเรียน

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ช่วยอำนวยความสะดวกผู้สอนและนักศึกษามีเวลาในการเรียนการสอนมากขึ้น
2. ช่วยให้นักศึกษามีความกระตือรือร้นในการมาเรียนและตรงต่อเวลา
3. ผู้สอนมีแหล่งเก็บข้อมูลการเข้าเรียนของนักศึกษาและป้องกันการชำรุดหรือเสียหายของข้อมูล
4. ผู้สอนตรวจสอบพฤติกรรมการเข้าเรียนของนักศึกษาได้ง่าย

1.5 เครื่องมือที่ใช้ในการพัฒนา (Development tools)

1.5.1 ฮาร์ดแวร์

1. สมาร์ทโฟน (Smart phone)
    - ทำงานบนระบบปฏิบัติการแอนดรอย์เวอร์ชัน 5.0 หรือ API Level 21
    - หน่วยประมาลผลกลาง Mediatek MT6753 Octa-core ความเร็ว 1.3 กิกะเฮิร์ตซ์ (Gigahertz, GHz)
    - หน่วยประมวลผลกราฟฟิกอย่างน้อย Mali-T720MP3
    - หน่วยความจำหลักอย่างน้อย 2 กิกะไบต์ (Gigabyte, GB)
    - หน่วยความจำสำรองอย่างน้อย 16 กิกะไบต์ (Gigabyte, GB)
    - หน้าจอแสดงผลความละเอียดอย่างน้อย 1080 x 1920 พิกเซล (Pixel)
    - หน้าจอแสดงผลขนาดอย่างน้อย 5 นิ้ว
    - กล้องถ่ายรูปความละเอียดอย่างน้อย 13 เมกกะพิกเซล (Magapixel)
2. เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal computer)
    - ทำงานบนระบบปฏิบัติการวินดโวส์ 10 (Windows operating system 10)
    - หน่วยประมวลผลกลาง Intel(R) Core(TM) i7-7500U CPU ความเร็ว 2.70 กิกะเฮิร์ต (Gigahertz, GHz)
    - หน่วยประมวลผลกราฟฟิก NVIDIA GeForce 960M ความจำ 4 กิกะไบต์ (Gigabyte, GB)
    - หน่วยความจำหลัก 4 กิกะไบต์ (Gigabyte, GB)
    - หน่วยความจำสำรอง 1 เทระไบต์ (Terabyte: TB)

1.5.2 ซอฟต์แวร์ (Software)

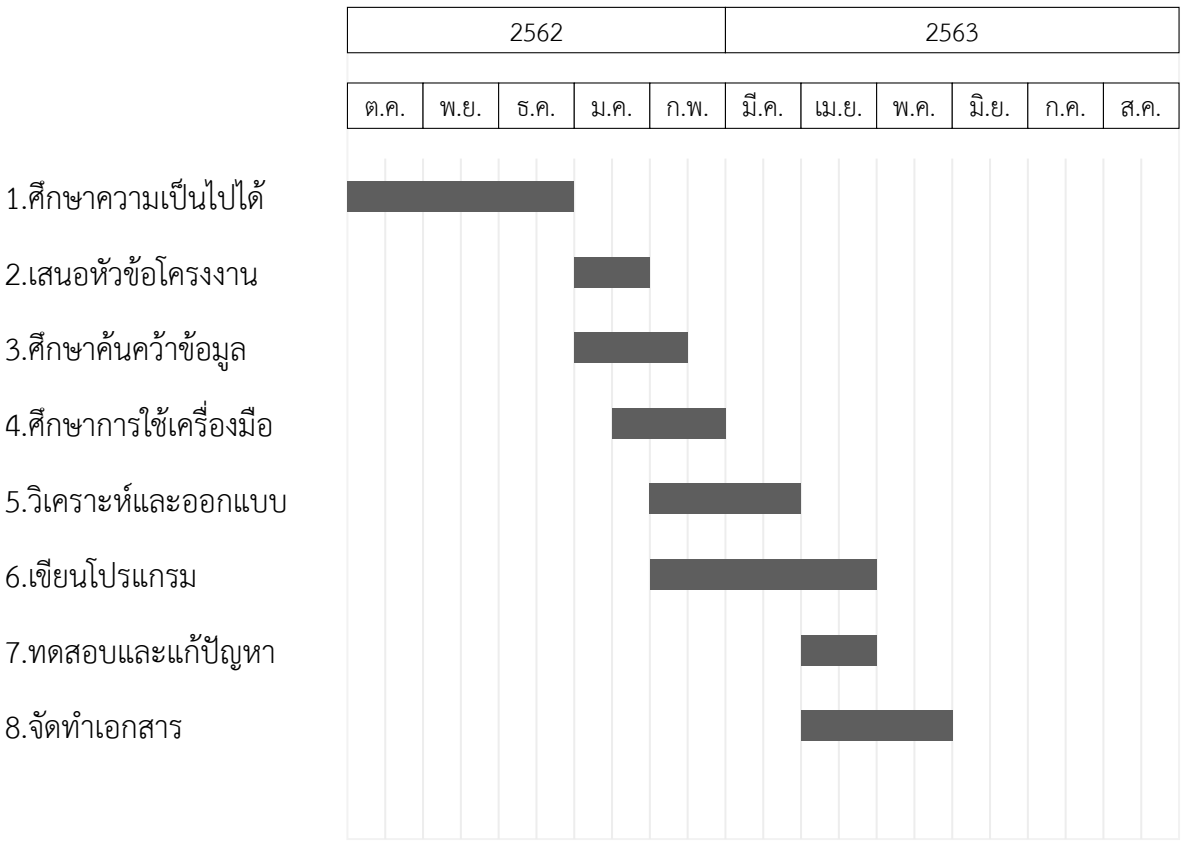
1. Angular เวอร์ชั่น 8 เป็นเฟรมเวิร์ค (Framework) สำหรับทำหน้าเว็บแสดงผล โดยใช้ JavaScript ในการพัฒนา
2. MongoDB คือฐานข้อมูลแบบไม่มีความสัมพันธ์ของตารางซึ่งมีโครงสร้างแบบ NoSQL(Not Only Structured Query Language) มีรูปแบบการจัดเก็บข้อมูลแบบ JSON(JavaScript Object Notation)

3. Echo Golang เวอร์ชั่น 1.13 ใช้สำหรับทำเว็บเซิฟเวอร์ (Web Server) พัฒนาด้วยภาษา Golang
4. Postman คือเครื่องมือสำหรับช่วยในพัฒนา API(Applocation Programming Interface) ใช้ในการทดสอบการทำงานของ Service รวมถึงจำรอง Service
5. Visual Studio Code คือเครื่องมือแก้ไขและปรับแต่งโค้ดสำหรับพัฒนาเว็บแอปพลิเคชัน
6. Google Chrome คือโปรแกรมเว็บเบราเซอร์ (Web Browser) ใช้สำหรับเปิดเว็บไซต์และเป็นเครื่องมือที่ช้สำหรับตรวจสอบ หรือหาจุดบกพร่องของเว็บแอปพลิเคชัน

1.5.3 แผนการดำเนินการ

ในการสร้างระบบเช็คชื่อด้วยคิวอาร์โค้ด ผู้พัฒนาได้แบ่งขั้นตอนการดำเนินงานไว้ด้วยกัน 7 ขั้นตอน ดังต่อไปนี้

ตารางที่ 1.1: ขั้นตอนการดำเนินงาน



บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

บทนี้จะเป็นรายละเอียดเกี่ยวกับทฤษฎีและงานวิจัยที่เกี่ยวของกับการพัฒนาโปรแกรมในครั้งนี้ โดยที่แต่ละหัวข้อจะมีความสัมพันธ์กันเป็นลำดับ โดยหัวข้อที่หนึ่งจะแนะนำความรู้เรื่อง การพัฒนาเว็บแอปพลิเคชัน เพื่อให้เข้าใจพื้นฐานเบื้องต้นเกี่ยวกับที่มาของโครงงาน หัวข้อที่สองที่สามจะช่วยเตรียมให้ผู้อ่านเข้าใจเทคโนโลยที่ใช้ในการออกแบบและพัฒนา ส่วนของระบบเช็คชื่อด้วยคิวอาร์โค้ด

2.1 ความรู้พื้นฐานของเว็บแอปพลิเคชัน

เว็บแอปพลิเคชัน [1] คือ การพัฒนาระบบงานบนเว็บ ซึ่งมีระบบมีการ ไหลเวียนในแบบ Online ทั้งแบบ Local ภายในวง LAN และ Global ออกไปยังเครือข่าย อินเตอร์เน็ต ทำให้เหมาะสำหรับงานที่ต้องการข้อมูลแบบ Real Time การทำงานของ Web Application นั้นโปรแกรมส่วนหนึ่งจะวางตัวอยู่บน Rendering Engine ซึ่งตัว Rendering Engine จะทำหน้าที่หลัก ๆ คือ นำเอาชุดคำสั่งหรือรูปแบบโครงสร้างข้อมูลที่ใช้ในการแสดงผล นำมาแสดงผล บนพื้นที่ส่วนหนึ่งในจอภาพ โปรแกรมส่วนที่วางตัวอยู่บน Rendering Engine จะทำหน้าที่หลัก ๆ คือ การเปลี่ยนแปลงแก้ไขสิ่งที่แสดงผล จัดการตรวจสอบข้อมูลที่รับเข้ามาเบื้องต้นและการประมวล บางส่วนแต่ส่วนการทำงานหลัก ๆ จะวางตัวอยู่บนเซอร์เวอร์ในลักษณะ Web Application แบบ เบื้องต้น ฝั่งเซิร์ฟเวอร์จะประกอบไปด้วยเว็บเซิร์ฟเวอร์ซึ่งทำหน้าที่เชื่อมต่อกับไคลเอนต์ตาม โปรโตคอล HTTP/HTTPS โดยนอกจากเว็บเซิร์ฟเวอร์จะทำหน้าที่ส่งไฟล์ที่เกี่ยวเนื่องกับการแสดงผล ตามมาตรฐาน HTTP ตามปกติทั่วไปแล้ว เว็บเซิร์ฟเวอร์จะมีส่วนประมวลผลซึ่งอาจจะเป็นตัว แปลภาษา เช่น Script Engine ของภาษา PHP หรืออาจจะมีการติดตั้ง .NET Framework ซึ่งมีส่วนแปลภาษา CLR ที่ใช้แปลภาษา Intermediate จากโค้ดที่เขียนด้วย VB.NET หรือ C.NET หรืออาจจะเป็น J2EE ที่มีส่วนแปลไบต์โค้ดของคลาสที่ได้จากโปรแกรมภาษาจาวา เป็นต้น

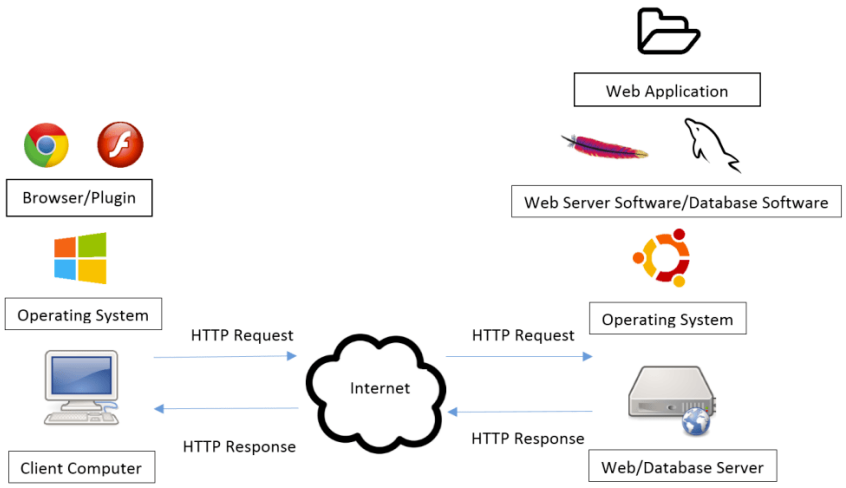
2.1.1 ความสำคัญของเว็บแอปพลิเคชัน

จากอดีตที่ยังไม่มีระบบเว็บแอปพลิเคชันการทำงานของคอมพิวเตอร์ส่วนบุคคล จะเป็น ลักษณะที่เรียกว่า Standalone ยกตัวอย่างเช่น เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรมเช่น Microsoft

office ประกอบไปด้วยโปรแกรม Ms-Word, Ms-Excel เป็นต้น โปรแกรมเหล่านี้จะเรียกว่าเป็น Desktop application กล่าวคือติดตั้งที่เครื่องคอมพิวเตอร์แต่ละบุคคล หรือโปรแกรมสำหรับงานบัญชีบางหน่วยงานติดตั้งที่เครื่องคอมพิวเตอร์พิวเตอร์เป็นลักษณะไคลแอนเซิร์ฟเวอร์(Client-Server application) ฐานข้อมูลไว้ที่เซิร์ฟเวอร์ (Server) และติดตั้งตัวโปรแกรมบัญชีที่เครื่องใช้งาน (Client) ซึ่งตอบสนองความต้องการเพิ่มขึ้นในด้าน Multi-User หรือใช้งานพร้อมๆกันได้หลาย ๆ คน โดยใช้ ฐานข้อมูลเดียวกัน เก็บฐานข้อมูลไว้ที่ส่วนกลาง เทคโนโลยี Desktop Application ไม่สามารถ ตอบสนองความต้องการการบริหารจัดการได้ โดยเฉพาะการทำธุรกิจที่ต้องปรับเปลี่ยนไปตลอดเวลา ข้อมูลมีการเคลื่อนไหวตลอดเวลา เพื่อตอบสนองภาวะตลาดที่แปรเปลี่ยน ระบบ Client-Server Application ตัวโปรแกรมมีความซับซ้อน การแก้ไข การ Upgrade ทำได้ยุ่งยาก อย่างกรณี หาก ต้องการ Upgrade หรือเพิ่มคุณสมบัติเพิ่มเติมให้กับ Application ที่ตัวเซิร์ฟเวอร์ต้องหยุดระบบ ทั้งหมด และเมื่อ Upgrade ที่เซิร์ฟเวอร์แล้ว ก็จำเป็นต้อง Upgrade ที่ Client ด้วย หากระบบมี ผู้ใช้งานจำนวนมาก จะยิ่งเพิ่มความยุ่งยากมากขึ้น นอกจากนี้ยังไม่รวมปัญหาว่า ที่เครื่อง Client มี ความหลากหลายและแตกต่างกัน เช่น OS (Operating System) ที่ต่างกัน สเปคเครื่องที่แตกต่างกัน ซึ่งหากการ Upgrade แล้วมีความจ าเป็นต้องใช้สเปคเครื่องที่สูงขึ้นที่ฝั่ง Client จำเป็นต้อง Upgrade ตัวเครื่องคอมพิวเตอร์ตามไปด้วย ปัญหาเหล่านี้ ถูกจัดการด้วยเทคโนโลยี Web Application เพราะ Web Application สามารถตอบสนองปัญหาข้างต้นได้เป็นอย่างดี และสามารถแทนที่ Desktop Application ที่เป็น Client-Server Application ได้เป็นอย่างดีตัวโปรแกรมของ Web Application จะถูกติดตั้งไว้ที่ Server คอยให้บริการกับ Client และที่ Client ก็ไม่ต้องติดตั้งโปรแกรมเพิ่มเติม สามารถใช้โปรแกรมประเภท Brower ที่ติดมากับ OS ใช้งานได้ทันทีอย่าง Internet Explorer หรือ โปรแกรมฟรี ได้แก่ FireFox, Google Chrome ซึ่งกำลังเป็นที่นิยมเป็นอย่างมาก ด้วยความสามารถ ของ Brower ที่หลากหลาย ทำให้ไม่จ ากัดว่าเครื่องที่ใช้เป็น OS อะไร หรืออุปกรณ์อะไร อย่าง อุปกรณ์ TouchPad หรือ SmartPhone ก็สามารถเรียกใช้งานได้ ลดข้อจำกัดเรื่องสถานที่ใช้งานอีก ด้วย จุดเด่นอีกอย่างหนึ่ง คือ ข้อมูลที่ส่งหากัน ระหว่าง Client กับ Server มีปริมาณน้อยมาก ทำให้ เราสามารถย้ายเซิร์ฟเวอร์ไปอยู่บนเครือข่าย Internet ได้และสามารถใช้งานผ่าน Internet Connection ที่มีความเร็วต่างๆได้ จุดเด่นนี้ทำให้ สามารถใช้ Application เหล่านี้จากทุก ๆ แห่งใน โลกได้

2.1.2 ส่วนประกอบของเว็บแอปพลิเคชัน

การทำความเข้าใจส่วนประกอบของเว็บแอปพลิเคชัน ถือว่าเป็นสิ่งสำคัญเพราะถ้านักพัฒนาโปรแกรม สามารถมองภาพโดยรวมของระบบได้ทั้งหมด จะสามารถเข้าใจถึงกระบวนการทำงานได้ดียิ่งขึ้น และสามารถนำไปช่วยในการออกแบบโปรแกรมที่ต้องการพัฒนาเพื่อให้เกิดประสิทธิภาพในการทำงาน



รูปที่ 2.1: โครงสร้างของระบบปฏิบัติการแอนดรอยด์

ที่มา : https://www.blog.tamacorp.com/wp-content

จากโครงสร้างของระบบปฏิบัติการแอนดรอยด์ในรูปที่ 2.1 จะสังเกตได้ว่า มีการแบ่งออกเป็นส่วน ๆ โดยจะแบ่งเป็น 2 ส่วยใหญ่ๆ คือ ส่วนประกอบฝั่งผู้ใช้งาน (Client-side Techology) และส่วนประกอบฝั่งเซิร์ฟเวอร์ (Server-side Tchnology)

1. ส่วนประกอบฝั่งผู้ใช้งาน

จากรูป 2.1 ด้านซ้ายของรูปจะเป็นเทคโนโลยีฝั่งผู้ใช้งาน ซึ่งประกอบด้วย 3 ส่วนหลักๆ

1.1 เว็บเบราว์เซอร์ (Web Browser) เป็นซอฟต์แวร์ที่ผู้ใช้งานใช้เข้าถึงเว็บแอปพลิเคชั่น โดยเริ่มต้นให้ผู้ใช้งานใส่ URL หรือว่าชื่อของเว็บไซต์ที่ต้องการเข้าใช้งาน เช่น www.google.com เมื่อเบราว์เซอร์ได้รับชื่อเว็บไซตีก็จะทำการแปลงชื่อเว็บไซต์เป็น IP address ผ่าน DNS หลังจากนั่นเบราว์เซอร์จะทำการสร้าง HTTP requst เพื่อส่งคำร้องไปยังเว็บเซิร์ฟเวอร์ผ่านทางเครือข่ายอินเตอร์เน็ต เมื่อได้รับ HTTP response

จากเว็บเซิร์ฟเวอร์แล้ว เบราว์เซอร์จะทำหน้าที่ในการอ่าน และแปลง HTTP response ให้เป็นข้อมูลที่ใช้ในการแสดงผลให้กับผู้ใช้งาน

1.2 ส่วนต่อความสามารถเว็บและเบราว์เซอร์(Web Plugin และ Browser Add-on/Extension)

- Web Plugin คือโปรแกรมที่ถูกเขียนให้ทำงานร่วมกับเว็บเบราว์เซอร์ Web Plugin ที่เป็นที่รู้จัก เช่น Adobe Flash Plugin, Pdf reader และ Java Applet เป็นต้น ซึ่งเหล่านี้จะถูกเบราว์เซอร์เรียกใช้ก็ต่อเมื่อเว็บไซต์ที่เข้าใช้งานมีเนื้อหาที่ต้องแสดงผลโดย Plugin
- Browser Add-on/Extension เป็นโปรแกรมที่ใช้ในการเพิ่มความสามารถให้กับเบราวเซอร์ เช่น ส่วนเพิ่มความสามารถที่ช่วยในการจัดการไฟล์ดาวโหลด ส่วนเพิ่มความสามารถที่ช่วยในการดาวโหลดไฟล์วีดิโอ เป็นต้น ซึ่งส่วนเพิ่มความสามารถเบราว์เซอร์เหล่านี้จะเน้นเพิ่มความสามารถให้กับเบราว์เซอร์ มากกว่าการประมวลผลเนื้อหาเว็บไซต์

1.3 ระบบปฏิบัติการ (Operating System) คือ ระบบปฏิบัติการทำหน้าที่ในการจัดการกับทรัพยากรของเครื่องคอมพิวเตอร์ ทำหน้าที่ในการรับ HTTP request จากเบราว์เซอร์และส่งต่อไปให้กับอินเทอร์เน็ต DNS ในระบบปฏิบัติการทำหน้าที่ในการแปลง URL ให้เป็น IP Address เพื่อค้นหาเครื่องเว็บเซิร์ฟเวอร์ สร้างการเชื่อมต่อ (TCP connection) ระหว่างเครื่องผู้ใช้งานและเครื่องเซิร์ฟเวอร์ ดังนั้นการทำงานของระบบปฏิบัติการจะเป็นสิ่งที่ผู้ใช้งานมองไม่เห็นแต่ก็มีความสำคัญมาก

2. ส่วนประกอบฝั่งเซิร์ฟเวอร์

เว็บเซิร์ฟเวอร์ที่ทำหน้าที่เป็นผู้ให้บริการแก่ผู้ใช้งานเว็บไซต์ประกอบไปด้วยเทคโนโลยีและซอฟต์แวร์หลายส่วนทำงานร่วมกัน โดยซอฟต์แวร์หลักที่ใช้ในการให้บริการของเว็บเซิร์ฟเวอร์ประกอบไปด้วย 4 ส่วนประกอบหลัก

1.1 เว็บแอปพลิเคชั่น (Web Application) คือส่วนสำคัญของเว็บไซต์เนื่องจากทำหน้าที่ติดต่อกับผู้ใช้งาน รับและแสดงข้อมูล ประมวลผลข้อมูล จัดการข้อมูลในฐานข้อมูลและอื่น ๆ เรียกได้ว่าเว็บแอปพลิเคชั่นเป็นซอฟต์แวร์ที่ให้บริการผู้ใช้งานทั่วโลกผ่านอินเทอร์เน็ต หากนักพัฒนาได้เขียนเว็บแอปพลิเคชั่นตาม Model-View-Controller (MVC) แล้วก็จะสามารถแบ่งเว็บแอปพลิเคชันออกได้เป็นสามส่วนหลัก ๆ คือ

- ส่วนที่ติดต่อกับผู้ใช้งานเพื่อรับข้อมูลและแสดงผล (View)

- ส่วนที่ประมวลผลการทำงาน (Controller)
- ส่วนที่ใช้ในการติดต่อและจัดการกับข้อมูลและฐานข้อมูล (Model)

นักพัฒนาสามารถพัฒนาเว็บแอปพลิเคชันได้ด้วยภาษาคอมพิวเตอร์ที่หลากหลาย เราสามารถแบ่งภาษาที่ใช้ในการพัฒนาเว็บแอปพลิเคชั่นออกเป็นสองส่วนคือ Front-End Technology ใช้สำหรับพัฒนา View (ส่วนติดต่อกับผู้ใช้งาน) และ Back-End Technology ใช้สำหรับพัฒนา Model และ Controller (ส่วนประมวลผลและจัดการข้อมูล)

Front-End Web Technology หมายถึงส่วนของเทคโนโลยีที่ใช้ในการสร้างส่วนติดต่อกับผู้ใช้งาน ในการสร้างเว็บแอปพลิเคชั่น Front-End Technology ที่เป็นที่แพร่หลายได้แก่ HTML, CSS, และ JavaScript ซึ่งภาษาคอมพิวเตอร์เหล่านี้ถูกใช้อย่างแพร่หลายในการสร้างส่วนติดต่อกับผู้ใช้งานของเว็บแอปพลิเคชัน ความหลากหลายของ Front-End Web Technology ถูกจำกัดด้วยมาตรฐานกลางที่ออกโดยองค์กรที่ไม่แสดงหาผลกำไรอย่าง World Wide Web Consortium (W3C) ซึ่งเป็นผู้กำหนดมาตรฐาน HTML, CSS, และ JavaScript เพื่อให้ผู้พัฒนาเบราว์เซอร์ให้แสดงผลข้อมูลในรูปแบบเดียวกัน เพื่อความสะดวกแก่ผู้ใช้งานและนักพัฒนา ซึ่งเบราว์เซอร์ในปัจจุบันต่างรองรับการประมวลผลของ HTML, CSS และ JavaScript โดยสมบูรณ์แม้ว่าจะมีความแตกต่างในการแสดงผลไปบ้าง(เล็กน้อย)ในบางเบราว์เซอร์

Back-End Web Technology หมายถึงส่วนของเทคโนโลยีที่เป็นส่วนประมวลผลตรรกะและการทำงานของเว็บแอปพลิเคชัน ไม่ว่าจะเป็นการตรวจสอบสิทธิ์การเข้าใช้ การเรียกดูและจัดเก็บข้อมูล การทำงานของเว็บแอปพลิเคชันในส่วนของ Back-End จะเริ่มหลังจากเว็บแอปพลิเคชั่นได้รับ HTTP request มาจากผู้ใช้งาน ทำการประมวลผล และส่งข้อมูลกลับไปให้กับผู้ใช้งาน เทคโนโลยีที่ใช้ในการพัฒนา Back-End ของเว็บแอปพลิเคชันจะมีความหลากหลายกว่า Front-End เนื่องจากไม่มีข้อจำกัดด้านมาตรฐานกลางดัง Front-End technology ที่ต้องรองรับมาตรฐานที่กำหนดโดย W3C เพื่อให้ทำงานกับเว็บเบราว์เซอร์ได้อย่างไม่มีปัญหา

1.2 เว็บเซิร์ฟเวอร์ซอฟต์แวร์ (Web Server Software) เป็นโปรแกรมที่ทำงานอยู่บน web server ซึ่งหน้าที่หลักของ web server software คือการประมวลผล HTTP request ที่ได้รับมาและตอบกลับด้วย HTTP response ให้กับผู้ใช้งาน ปัจจุบันมี

web server software หลายตัวที่ถูกใช้งานอย่างแพร่หลายเช่น Apache HTTP server, Internet Information Service (IIS) และ Nginx ยังมี web server software ตัวอื่นอีกมากในท้องตลาดที่ไม่ได้กล่าวถึงในที่นี้ อย่างไรก็ตาม web server software ที่ได้รับความนิยมอย่างแพร่หลายมากที่สุดในปัจจุบันคือ Apache HTTP server และผู้ใช้งานมักจะใช้คู่กับ PHP (ตัวแปลภาษา PHP) และ MySQL (ฐานข้อมูล)

1.3 ระบบปฏิบัติการ (Operating System) ระบบปฏิบัติการบนฝั่งของเซิร์ฟเวอร์มีหน้าที่ในการจัดการกับทรัพยากรของเครื่องเซิร์ฟเวอร์ เช่น CPU memory และ bandwidth เป็นต้น เนื่องจาก web application เป็นบริการที่เปิดให้ผู้ใช้งานเข้าถึงได้ตลอดเวลา ดังนั้น ระบบปฏิบัติการบนเซิร์ฟเวอร์จึงต้องมีความเสถียรและสามารถจัดการกับทรัพยากรของเครื่องได้อย่างมีประสิทธิภาพ

2.2 ความรู้พื้นฐานภาษาโก (Go)

ภาษาโก [2] คือ ภาษาโปรแกรมที่เป็นโอเพนซอร์ส (Open-Source) ถูกสร้างขึ้นจากบริษัทกูเกิล ในปี ค.ศ. 2007 โดย Robert Griesemer, Rob Pike, และ Ken Thompson เผยแพร่อย่างเป็นทางการ ในปี ค.ศ. 2009 ภาษาโกมีไวยากรณ์คล้ายกับภาษาซี (C) แต่ก็ยอมให้ผู้พัฒนาซอฟต์แวร์สามารถเขียนโปรแกรม ในรูปแบบของภาษาสคริปต์ได้ (script language)

ภาษาโกถูกออกแบบมาสำหรับการเขียนซอฟต์แวร์ระบบเป็นหลัก มีลักษณะเป็นแบบ statically typed language หมายถึง ชนิดข้อมูลจะถูกกำหนดในช่วงเวลาของการคอมไพล์ โปรแกรม (compile time) มี garbage collection เพื่อช่วยเพิ่มประสิทธิภาพการทำงานของ โปรแกรม มีการสนับสนุนการคอมไพล์โปรแกรมเพื่อนำไปใช้บนแพลตฟอร์มอื่นๆ (cross-compiler) และมีการสนับสนุนการทำงานแบบภาวะพร้อมกัน ที่ถูกออกแบบขึ้นมาตามแนวคิดของ Hoare’s Communicating Sequential Processes นอกจากนี้การเขียนโปรแกรมด้วยภาษาโกนั้นง่าย เพราะ บางส่วนของภาษาโกมีลักษณะคล้ายคลึงกับภาษาสคริปต์

2.2.1 ชุดคำสั่งในภาษาโก

คำสั่งในภาษาโก (The Go Authors, 2011) มีคำสั่งจำนวนมากให้เลือกใช้งานตามความเหมาะสม แต่ในระบบนี้ขอนำเสนอ 6 คำสั่งได้แก่ คำสั่ง if-else คำสั่ง func คำสั่ง go คำสั่ง var คำสั่ง for และคำสั่ง channel ซึ่งคำสั่งเหล่านี้เพียงพอต่อการพัฒนาโปรแกรมด้วยภาษาโก

2.2.2.1 คำสั่ง if-else เป็นคำสั่งสำหรับเลือกเส้นทางการทำงานของโปรแกรมตามเงื่อนไขที่ได้ กำหนดไว้ รูปแบบของคำสั่ง if-else แสดงดังภาพที่ 2.2 จากภาพถ้าเงื่อนไข exp เป็นจริง กลุ่มคำสั่ง ที่อยู่ภายใต้ if จะทำงาน หากเงื่อนไขเป็นเท็จกลุ่มคำสั่งที่อยู่ภายใต้ else จะถูกทำงานแทน โครงสร้างของคำสั่ง if-else แสดงดังภาพที่ 2.2

```
1
2 if exp {
3       //body if
4       } else{
5       // body else
6        }
```

รูปที่ 2.2: โครงสร้างคำสั่ง if-else

2.2.2.2 คำสั่ง func เป็นคำสั่งสำหรับสร้างฟังก์ชันเพื่ออำนวยความสะดวกในการประมวลผล กลุ่มคำสั่งต่างๆ รูปแบบของคำสั่ง func แสดงดังภาพที่ 2.3 จากภาพจะเห็นได้ว่าฟังก์ชันชื่อว่า A จะ สามารถรับค่าอยู่ในรูปแบบของ ชื่อว่าตัวแปร param1 ตามด้วย ptype1 คือชนิดของตัวแปร param1 และสามารถรับได้หลายๆค่าโดยคั่นด้วยเครื่องหมายจุลภาค (,) และสามารถคืนค่าได้โดยตัว แปร val1 ตามด้วยชนิดของตัวแปร vtype1 และสามารถคืนค่าได้หลายๆค่าโดยคั่นด้วยเครื่องหมาย จุลภาค โครงสร้างคำสั่ง func แสดงดังภาพที่ 2.3

```
1
2 func A(param1 ptype1 | […param …ptype ,] ) (val1
     vtype1 | [val.. …vtype ,]) { //body func } body
     else
3                          }
```

รูปที่ 2.3: โครงสร้างคำสั่ง func

2.2.2.3 คำสั่ง go เป็นคำสั่งที่กำหนดให้โปรแกรมทำงานแบบภาวะพร้อมกันอย่างอัตโนมัติ ตัวอย่างรูปแบบของคำสั่งแสดงได้ในภาพที่ 2.4 การทำงานของคำสั่งโก โปรแกรมจะถูกแบ่งออกเป็น โพรเซสย่อยๆหลายอันทำงานแบบภาวะพร้อมกัน ซึ่งภาษาโกเรียกโพรเซสย่อยแต่ละอันว่า goroutines โปรแกรมเมอร์ไม่จำเป็นต้องรู้เรื่องของการแบ่งแยกโปรแกรมออกเป็นโพรเซสย่อยๆ ตัว ภาษาโกจะเป็นตัวจัดการให้โครงสร้างการใช้ของคำสั่ง go แสดงดังภาพที่ 2.4

```
1
2 func A() { //body func} go A()
```

รูปที่ 2.4: โครงสร้างคำสั่ง go

2.2.2.4 คำสั่งที่ใช้สำหรับการประกาศตัวแปร แสดงดังภาพที่ 2.5 จากภาพเราสามารถประกาศตัวแปรได้ 2 รูปแบบ 1 การประกาศแบบใช้คำสั่ง var และ 2 การประกาศแบบย่อโดยใช้ เครื่องหมาย := ในการประกาศตัวแปร ทั้งสองแบบมีความแตกต่างกันที่การประกาศในรูปแบบที่ 1 สามารถประกาศโดยระบุชนิดของข้อมูลหรือหากไม่ระบุชนิดข้อมูลภาษาโกจะทำการกำหนดชนิด ข้อมูลให้ตามค่าที่อยู่ทางขาวมือของเครื่องหมายเท่ากับโดยอัตโนมัติ และสามารถประกาศได้ทั้งภายใน และภายนอกฟังก์ชัน ส่วนการประกาศในรูปแบบที่ 2 นั้นสามารถประกาศได้เฉพาะภายในฟังก์ชัน เท่านั้นและไม่ต้องระบุชนิดของข้อมูลโดยภาษาโกจะกำหนดชนิดข้อมูลให้อัตโนมัติเช่นกัน โครงสร้าง การประกาศตัวแปร แสดงดังภาพที่ 2.5

```
1
2 //  การประกาศโดยใช้var แบบไม่ระบุชนิด
3 var num = 10
4 // การประกาศโดยใช้var แบบระบุชนิด
5 var num int = 10
6 // การประกาศแบบย่อnum := 10
```

รูปที่ 2.5: โครงสร้างการประกาศตัวแปร

2.2.2.5 คำสั่ง for เป็นคำสั่งที่ทำงานแบบวนซ้ำ โดยจะทำการวนซ้ำาถ้าหากเงื่อนไขเป็นจริง รูปแบบของคำสั่งแสดงดังภาพที่ 2.6 จากภาพถ้า condition เป็นจริงกลุ่มคำสั่งที่อยู่ภายใต้ for จะ ทำงานวนไปเรื่อยๆจนกว่าเงื่อนไขจะเป็นเท็จ โดยในภาษาโกนั้นจะไม่มีคำสั่ง while หรือ do-while ดังเช่นภาษาอื่น มีเพียงคำสั่ง for ที่สามารถเขียนแทนคำสั่งเหล่านั้นได้ โครงสร้างของคำสั่ง for แสดง ดังภาพที่ 2.6

```
1
2 // คล้ายfor ในภาษาC for init; condition; post { }
3 // คล้ายwhile ในภาษาC for condition { }
4 // คล้ายfor(;;) ในภาษาC for { }
```

รูปที่ 2.6: โครงสร้างคำสั่ง for

2.2.2.6 คำสั่ง channel เป็นคำสั่งที่ใช้สำหรับเป็นช่องทางในการติดต่อสื่อสารรับหรือส่งข้อมูล ระหว่าง goroutines ที่ทำงานแบบภาวะพร้อมกันโดยจะใช้เครื่องหมาย “<-” ในการระบุว่าต้องการ รับหรือส่งข้อมูล ดังแสดงในภาพที่ 2.7

```
1
2 // รับค่าจากchannel <— channel
3 // ส่งค่าเข้าchannel channel <—
```

รูปที่ 2.7: โครงสร้างคำสั่ง channel

2.3 ความรู้พื้นฐาน Java Script

JavaScript [3] คือ ภาษาคอมพิวเตอร์สำหรับการเขียนโปรแกรมบนระบบอินเทอร์เน็ต ที่กำลังได้รับความนิยมอย่างสูง Java JavaScript เป็นภาษาสคริปต์เชิงวัตถุที่เรียกว่า ”สคริปต์” (Script) ซึ่งในการสร้างและพัฒนาเว็บไซต์ (ใช่ร่วมกับ HTML) เพื่อให้เว็บไซต์ของดูมีการเคลื่อนไหว สามารถตอบสนองผู้ใช้งานได้มากขึ้น ซึ่งมีวิธีการทำงานในลักษณะ ”แปลความและดำเนินงานไปทีละคำสั่ง” (Interpret) หรือเรียกว่า อ็อบเจ็กโอเรียนเต็ด (Object Oriented Programming) ที่มีเป้าหมายในการออกแบบและพัฒนาโปรแกรมในระบบอินเทอร์เน็ตสำหรับผู้เขียนด้วยภาษา HTML สามารถทำงานข้ามแพลตฟอร์มได้ โดยทำงานร่วมกับภาษา HTML และภาษา Java Script ทำงานได้ทั้งทางฝั่งไคลเอนต์ (Client) และทางฝั่งเซิร์ฟเวอร์ (Server)

JavaScript ถูกพัฒนาขึ้นโดย เน็ตสเคปคอมมิวนิเคชันส์ (Netscape Communications Corporation) โดยใช้ชื่อว่า Live Script ออกมาพร้อมกับ Netscape Navigator2.0 เพื่อใช้สร้างเว็บเพจโดยติดต่อกับเซิร์ฟเวอร์แบบ Live Wire ต่อมาเน็ตสเคปจึงได้ร่วมมือกับ บริษัทซันไมโครซิสเต็มส์ปรับปรุงระบบของบราวเซอร์เพื่อให้สามารถติดต่อใช้งานกับภาษาจาวาได้ และได้ปรับ-ปรุง LiveScript ใหม่เมื่อ ปี 2538 แล้วตั้งชื่อใหม่ว่า JavaScript

JavaScript สามารถทำให้ การสร้างเว็บเพจ มีลูกเล่น ต่าง ๆ มากมายและสามารถโต้ตอบกับผู้ใช้ได้อย่างทันที เช่น การใช้เมาส์คลิกหรือการกรอกข้อความในฟอร์ม เป็นต้น

เนื่องจาก JavaScript ช่วยให้ผู้พัฒนาสามารถสร้างเว็บเพจได้ตรงกับความต้องการและมีความน่าสนใจมากขึ้นประกอบกับเป็นภาษาเปิด ที่ใครก็สามารถนำไปใช้ได้ ดังนั้นจึงได้รับความนิยมเป็นอย่างสูง มีการใช้งานอย่างกว้างขวาง รวมทั้งได้ถูกกำหนดให้เป็นมาตรฐานโดย ECMA การทำงานของ JavaScript จะต้องมีการแปลความคำสั่ง ซึ่งขั้นตอนนี้จะถูกจัดการโดยบราวเซอร์ (เรียกว่าเป็น client-side script) ดังนั้น JavaScript จึงสามารถทำงานได้ เฉพาะบนบราวเซอร์ที่สนับสนุน ซึ่งปัจจุบันบราวเซอร์เกือบทั้งหมดก็สนับสนุน JavaScript แล้ว อย่างไรก็ดีสิ่งที่ต้องระวังคือ JavaScript มีการพัฒนาเป็นเวอร์ชั่นใหม่ ๆ ออกมาด้วยดังนั้น ถ้านำโค้ดของเวอร์ชั่นใหม่ไปรันบนบราวเซอร์รุ่นเก่าที่ยังไม่สนับสนุนอาจจะทำให้เกิดข้อผิดพลาด (Error) ได้

2.3.1 ลักษณะการทำงานของ JavaScript

JavaScript เป็นภาษาสคริปต์เชิงวัตถุหรือเรียกว่าอ็อบเจ็กโอเรียนเต็ด (Object Oriented Programming) ที่มีเป้าหมายในการออกแบบและพัฒนาโปรแกรมในระบบอินเทอร์เน็ตสำหรับผู้เขียนเอาสารด้วยภาษา HTML สามารถทำงานข้ามแพลตฟอร์มได้ทำงานร่วมกับภาษา HTML และภาษาจาวาได้ทั้งทางฝั่งไคลเอนต์ (Client) และ ทางฝั่งเซิร์ฟเวอร์ (Server) โดยมีลักษณะการทำงานดังนี้

```
1  // import ฟังก์ชั่นสำคัญของangular/core
2  import {Component, OnInit} from ‘@angular/core‘;
3
4  @Component({ประกาศชื่อ
5
6  // selector เพื่อให้component อื่นสามารถนำไปใช้ได้
7
8  selector: ’student’,ระบุชื่อไฟล์
9
10 // html ที่ใช้ฟังก์ชันในcomponent โดยตรง
11
12 templateUrl: ’./student.component.html’,ระบุ
13
14 // style ที่นำไปใช้กับhtml
15
16 styleUrls: [’./student.component.css’]
17 })
18
19 export class StudentComponent implements OnInit {
20
21 constructor() { }
22
23 ngOnInit() { }
24
25 }
```

รูปที่ 2.8: ตัวอย่างคำสั่งภาษา TypeScript ด้วย minted

- Navigator JavaScript เป็น Client-Side JavaScript ซึ่งหมายถึง JavaScript ที่ถูกแปลทางฝั่งไคลเอนต์ หมายถึงฝั่งเครื่องคอมพิวเตอร์ของผู้ใช้ไม่ว่าจะเป็นเครื่องพีซี (Personal

computer, PC) เครื่องแมคอินทอช (Macintosh) หรืออื่น ๆ จึงมีความเหมาะสมต่อการใช้งานของผู้ใช้ทั่วไปเป็นส่วนใหญ่

- LiveWire JavaScript เป็น Server-Side JavaScript ซึ่งหมายถึง JavaScript ที่ถูกแปลทางฝั่งเซิร์ฟเวอร์ (หมายถึงฝั่งเครื่องคอมพิวเตอร์ของผู้ให้บริการเว็บโดยอาจจะเป็นเครื่องของซันซิลิคอมกราฟิกส์หรืออื่น ๆ) สามารถใช้ได้เฉพาะกับ LIveWire ของเน็ตสเคป โดยตรง

2.3.2 JavaScript กับ HTML

การเขียน JavaScript อาจเขียนรวมอยู่ในไฟล์เดียวกันกับ HTML ได้ ซึ่งแตกต่างจากการเขียนโปรแกรมภาษา Java ที่ต้อง เขียนแยกออกเป็นไฟล์ต่างหากไม่สามารถเขียนรวมอยู่ในไฟล์เดียวกับ HTML ได้ วิธีการเขียน JavaScript เพื่อสั่งให้เว็ปเพจทำงาน มีอยู่ด้วยกัน 2 วิธี ดังนี้

- เขียนด้วยชุดคำสั่งและฟังก์ชันของ JavaScript เอง
- เขียนตามเหตการณ์ที่เกิดขึ้นตามการใช้งานจากชุดคำสั่งของ HTML

เมื่อเริ่มใช้งานโปรแกรมบราวเซอร์จะอ่านข้อมูลจากส่วนบนของเพจ HTML และทำงานไปตามลำดับจาก บนลงล่าง (top-down) โดยเริ่มที่ส่วน < HEAD >...< /HEAD > ก่อนจากนั้นจึงทำงานในส่วน < BODY >...< /BODY > เป็นลำดับต่อมา การทำงานของ JavaScript ดูไม่แตกต่างไปจาก HTML เท่าใดนัก แต่ HTML จะวางเลย์เอาต์โครงสร้างของอ็อบเจ็กต์ภายใน และส่วนเชื่อมโยงกับเว็บเพจเท่านั้น ในขณะที่ JavaScript สามารถเพิ่มเติมส่วนของการเขียนโปรแกรมและลอจิกเข้าไป

2.3.3 โครงสร้างภาษา

1. ตัวแปร (Variable) หมายถึง ชื่อหรือสัญลักษณ์ที่ตั้งขึ้นสำหรับการเก็บค่าใด ๆ ที่ไม่คงที่ โดยการจองเนื้อที่ในหน่วยความจำของระบบเครื่องที่เก็บข้อมูลซึ่งสามารถอ้างอิงได้ มีขนาดขึ้นอยู่กับชนิดของข้อมูลและค่าของข้อมูล ซึ่งค่าในตัวแปรนี้สามารถเปลี่ยนแปลงได้ตามคำสั่งในการประมวลผล
2. การตั้งชื่อ (Identifier or Name) เป็นชื่อที่ตั้งขึ้นมาเพื่อกำหนดให้เป็นชื่อของโปรแกรมหลัก, ฟังก์ชัน, ตัวแปร, ค่าคงที่, คำสั่ง และคำสงวน โดยมีหลักการตั้งชื่อว่า
    - ขึ้นต้นด้วยตัวอักษรในภาษาอังกฤษ ตามด้วยตัวอักษรหรือตัวเลขใด ๆ ก็ได้
    - ห้ามเว้นช่องว่าง

- ห้ามใช้สัญลักษณ์พิเศษ ยกเว้นขีดล่าง (_) และดอลล่าร์ ($)
- สำหรับความยาวของชื่อใน JavaScript จะมีความยาวเท่าใดก็ได้ แต่ที่นิยมใช้ ไม่เกิน 20 ตัวอักษร
- การตั้งชื่อมีข้อพึงระวังว่า จะต้องไม่ซ้ำกับคำสงวน (Reserve word) และตัวอักษรของชื่อจะจำแนกแตกต่างกันระหว่างอักษรตัวพิมพ์เล็กกับอักษรตัวพิมพ์ใหญ่
- ควรจะตั้งชื่อโดยให้ชื่อนั้นมีสื่อความหมายให้เข้ากับข้อมูล สามารถอ่านและเข้าใจได้ ตัวอย่างชื่อที่ถูกต้อง Hahaha, I_Love_you, Doll$ เป็นต้น

3. คำสงวน(Reserve word)เป็นคำที่มีความหมายเฉพาะตัวในภาษา JavaScript สงวนไม่ให้มีการตั้งชื่อซ้ำกับชื่อโปรแกรม, ฟังก์ชัน, ตัวแปร, ค่าคงที่ และคำสั่ง คำสงวน สามารถเรียกใช้ได้ทันทีโดยไม่ต้องมากำหนดความหมายใหม่แต่อย่างใด
4. ชนิดของข้อมูลของตัวแปร (Data Type) เป็นการกำหนดประเภทค่าของข้อมูลให้กับตัวแปร เพื่อให้เหมาะสมกับการอ้างอิงข้อมูลจากตัวแปรในการใช้งาน ชนิดข้อมูลของตัวแปรนั้นมีอยู่ด้วยกัน 4 ชนิด ได้แก่
    - number หมายถึง ข้อมูลชนิดตัวเลข ประกอบด้วย เลขจำนวนเต็ม (Integer) และเลขจำนวนจริง (Floating)
    - logical หมายถึง ข้อมูลทางตรรกะ มี 2 สถานะ คือ จริง (True) และเท็จ (False)
    - string หมายถึง ข้อมูลที่เป็นข้อความ ซึ่งจะต้องกำหนดไว้ในเครื่องหมายคำพูด (”...”)
    - null หมายถึง ไม่มีค่าข้อมูลใดๆ ซึ่งค่า null ใช้สำหรับการยกเลิกพื้นที่เก็บค่าของตัวแปรออกจากหน่วยความจำ
5. การประกาศตัวแปร (Declarations) เป็นการกำหนดชื่อและชนิดข้อมูลให้กับตัวแปรเพื่อนำไปใช้ในโปรแกรม โดยการตั้งชื่อจะต้องคำนึงถึงค่าของข้อมูลและ ชนิดของข้อมูลที่อ้างอิง นอกจากนี้การตั้งชื่อควรให้สื่อความหมายของข้อมูล และอักษรของชื่อจะจำแนกแตกต่างกันระหว่างอักษรตัวพิมพ์เล็กกับอักษรตัวพิมพ์ใหญ่

   รูปแบบ Var ชื่อตัวแปร; เป็นรูปแบบการประกาศตัวแปรปกติหรือ Var ชื่อตัวแปร = ข้อมูล; เป็นรูปแบบการกำหนดค่าเริ่มต้น ในกรณีที่ต้องการกำหนดตัวแปรหลายตัวในบรรทัดเดียวกันให้ใช้เครื่องหมาย คอมม่า (,) คั่นระหว่างชื่อตัวแปรและปิดท้ายด้วยเครื่องหมายเซมิโคล่อน (;) การกำหนดค่าให้กับตัวแปร

   รูปแบบ ชื่อตัวแปร = ค่าของข้อมูล โดยที่ค่าของข้อมูล ได้แก่

- ข้อมูลที่เป็นตัวเลข โดยกำหนดตัวเลขไปได้เลย เช่น num = 500
- ข้อมูลในทางตรรกะ ได้แก่ จริง (True) หรือ เท็จ (False) เช่น test = True;
- ข้อมูลสตริง ให้กำหนดอยู่ในเครื่องหมายคำพูด (”...”) เช่น name = ”Adisak”;

ตัวแปรมี 2 จำพวก หากกำหนดชื่อตัวแปรไว้ที่โปรแกรมหลักโดยไม่ได้อยู่ภายในขอบเขตฟังก์ชันใด ๆ เรียกว่าเป็นตัวแปรแบบโกลบัล (Global) ตัวแปรจำพวกนี้จะมีค่าคงอยู่ในหน่วยความจำตลอดการทำงานของโปรแกรม ทำให้สามารถเรียกใช้ได้จากทุก ๆ ส่วนของโปรแกรม รวมถึงภายในฟังก์ชันต่าง ๆ ด้วย แต่ถ้ากำหนดตัวแปรไว้ภายในขอบเขตฟังก์ชันใด ๆ จะเรียกว่าเป็นตัวแปรแบบ โลคัล (Local) เพราะจะเป็นตัวแปรที่มีค่าคงอยู่ และสามารถเรียกใช้ได้เฉพาะ ภายในขอบเขตของฟังก์ชันนั้น ๆ เท่านั้น

6. ตัวแปรแบบอาร์เรย์ (Array) หมายถึงตัวแปรซึ่งมีค่าได้หลายค่าโดยใช้ชื่ออ้างอิงเพียงชื่อเดียว ด้วยการใช้หมายเลขลำดับเป็นตัวจำแนกความแตกต่างของค่าตังแปรแต่ละตัว ถ้าจะเรียกตัวแปรชนิดนี้ว่า ”ตัวแปรชุด” ก็เห็นจะไม่ผิดนัก ตัวแปรชนิดนี้มีประโยชน์มาก ลองคิดถึงค่าข้อมูลจำนวน 100 ค่า ที่ต้องการเก็บไว้ในตัวแปรจำนวน 100 ตัว อาจทำให้ต้องกำหนดตัวแปรที่แตกต่างกันมากถึง 100 ชื่อ กรณีอย่างนี้ควรจะทำอย่างไรดี แต่ด้วยการใช้สมบัติอาร์เรย์ สามารถนำตัวแปรหลาย ๆ ตัวมาอยู่รวมเป็ฯชุดเดียวกันได้และสามารถเรียกใช้ตัวแปรทั้งหมดโดยระบุผ่านชื่อเพียงชื่อเดียวเท่านั้น ด้วยการระบุหมายเลขลำดับ หรือดัชนี (index) กำกับตามหลังชื่อตัวแปร ตัวแหรเพียงชื่อเดียวจึงมีความสามารถเทียบได้กับตัวแปรนับร้อยตัว พันตัว (ตัวที่ 1) ในตัวแปรแบบอาร์เรย์มีดัชนีเป็น 0 ส่วนตัวแปรต่อ ๆ ไปก็จะมีดัชนีเป็น 1,2,3,... ไปตามลำดับ เมื่อต้องการระบุชื่อตัวแปรแบบอาร์เรย์แต่ละตัว ก็จะใช้รูปแบบ name[0], name[1],... เรียงต่อกันไปเรื่อย ๆ สามารถสร้างตัวแปรอาร์เรย์ใหม่ด้วย myArray = new Array() ดังนี้

```
1 myArray[0] = 17;
2 myArray[1] = ”Nun”;
3 myArray[2] = ”Stop”;
```

รูปที่ 2.9: ตัวแปรอาร์เรย์

7. ค่าคงที่ (Literal หรือ Constant) หมายถึง ค่าของข้อมูลที่เมื่อกำหนดแล้วจะทำการเปลี่ยน-แปลงค่าเป็นอย่างอื่นไม่ได้ ชนิดข้อมูลของค่าคงที่ได้แก่

- เลขจำนวนเต็ม (Integer) เป็นตัวเลขที่ไม่มีเศษทศนิยม สามารถเขียนให้อยู่ในแบบเลขฐานสิบ (0-9), เลขฐานสิบหก (0-9, A-F) หรือ เลขฐานแปด (0-7) โดยการเขียนเลขฐานแปดให้ นำหน้าด้วย O (Octenary) ส่วนการเขียนเลขฐานสิบหกให้นำหน้าด้วย Ox หรือ OX (Hexadenary)
- เลขจำนวนจริง (Floating) ใช้รูปแบบการเขียนโดยประกอบไปด้วยตัวเลข จุดทศนิยม และตัวเลขยกกำลัง E (Exponential) เช่น □ 5.00E2 จะหมายถึงค่า 5.00 คูณด้วย 10 ยกกำลัง 2 จะมีค่าเป็น 500 □ 3.141E5 จะหมายถึงค่า 3.141 คูณด้วย 10 ยกกำลัง 5 จะมีค่าเป็น 314,1000
- ค่าบูลีน (Boolean) เป็นค่าคงที่เชิงตรรกะ คือมีค่าเป็น จริง(True) และ เท็จ(False) เท่านั้น
- ข้อความสตริง (String) เป็นค่าคงที่แบบข้อความที่อยู่ภายในเครื่องหมายคำพูด (”...” หรือ ’...’) เช่น ”บริษัท เอ็กซ์ทรีม จำกัด”, ’นางนฤมล เวชตระกูล’

8. รหัสคำสั่งพิเศษ (Character escape code) เป็นการกำหนดรหัสเพื่อควบคุมงานพิมพ์สตริงโดยใช้เครื่องหมาย Backslash นำหน้าตัวอักษรที่ต้องการกำหนดเป็นรหัส เพื่อให้กลายเป็นรหัสคำสั่งพิเศษ รหัส Character escape code
9. นิพจน์ (Expression) เป็นข้อคำสั่งที่ใช้กำหนดค่าของข้อมูล เช่น การบวกตัวเลข การเปรียบเทียบข้อมูล โดยการกำหนดชื่อของตัวแปร ตามด้วยเครื่องหมายที่ต้องการกระทำ (Operations) ต่อข้อมูลเป็นผลให้เกิดค่าข้อมูลใหม่ค่าหนึ่งให้กับตัวแปรเพื่อนำไปใช้งาน
   นิพจน์ JavaScript มีด้วยกัน 3 ชนิดดังนี้
   - นิพจน์คณิตศาสตร์ (Arithmetic) เป็นนิพจน์ที่ใช้เครื่องหมายทางคณิตศาสตร์เป็นตัวกระทำ ผลลัพธ์ที่ได้จะมีค่าเป็นตัวเลขให้กับตัวแปร เช่น ให้ตัวแปร num เก็บตัวเลข 5000 จะเขียนได้ดังนี้ num = 5000;
   - นิพจน์ตรรกะ (Logical) เป็นนิพจน์ในการเปรียบเทียบข้อมูลโดยใช้เครื่องหมายในการเปรียบเทียบเพื่อตรวจสอบข้อมูลในการเปรียบเทียบว่าจริงหรือเท็จ เช่น กำหนดให้

```
1 a = 50;
2 b = 70;
3 c = b>a;
```

รูปที่ 2.10: การเปรียบเทียบว่าจริงหรือเท็จ

ผลลัพธ์ที่ได้คือ c จะมีค่าเป็นจริง (True)

- นิพจน์ข้อความ (String) เป็นนิพจน์เกี่ยวกับการกำหนดข้อความ การเชื่อประโยคข้อ-ความ ใช้ประมวลผลข้อความในลักษณะต่าง ผลลัพธ์ที่ได้จึงมีค่าเป็นตัวอักษรหรือข้อความเสมอ เช่น ให้ตัวแปร name เก็บชื่อ Adisak จะเขียนได้ดังนี้ name = ”Adisak”;

10. ตัวดำเนินการ (Operator) หมายถึง เครื่องหมายกำหนดกรรมวิธีทางคณิตศาสตร์, พีชคณิต, บูลีน, การเปรียบเทียบ ระหว่างข้อมูล 2 ตัว ซึ่งเรียกว่า โอประแรนด์ (Operand) โดยอาจมีค่าเป็นตัวเลข ข้อความ ค่าคงที่ หรือตัวแปรต่าง ๆ
11. ชนิดของตัวดำเนินการ
    - ตัวดำเนินการคณิตศาสตร์ (Arithmetic operator) หมายถึง ใช้สำหรับคำนวณโอประแรนด์ที่เป็นค่าคงที่หรือตัวแปรก็ได้ โดยให้ค่าผลลัพธ์เป็นตัวเลขค่าเดียว
    - ตัวดำเนินการเชิงเปรียบเทียบ (Comparison operator) หมายถึง เครื่องหมายในการเปรียบเทียบข้อมูล ผลลัพธ์ที่ได้จะมีค่าตรรกบูลลีนเป็น จริง (True) และ เท็จ (False)
    - ตัวดำเนินการเชิงตรรกะ (Logical operator) เป็นเครื่องหมายที่ให้ค่าจริง (True) และ เท็จ (False) ในการเปรียบเทียบ
    - ตัวดำเนินการเชิงข้อความ (String operator) เป็นการเชื่อมประโยคข้อความเข้าด้วยกัน (concatenation) โดยใช้เครื่องหมายบวก (+) เป็นตัวกระทำ เช่น

```
1 Name = ”Bodin”;
2 Say = ”Hey ”+Name;
```

รูปที่ 2.11: ตัวดำเนินการเชิงข้อความ

ผลลัพธ์ที่ได้ Say จะมีข้อความเป็น Hey Bodin

- ตัวดำเนินการระดับบิต (Bitwise operator) เป็นการดำเนินการเชิงตรรกะในระดับบิต โดยจะใช้มุมมองในแบบเลขฐาน 2 มาจัดการกับข้อมูล นั่นคือ ข้อมูลตัวเลขนั้นจะถูกแปลงเป็นเลขฐานสองในหน่วยความจำในขณะที่มีการดำเนินการเชิงตรรกะในระดับบิต ซึ่งโดยปกติแล้วการกระทำใน JavaScript จะอยู่ในระดับตัวอักษร ที่เรียกว่า ระดับไบต์ (byte)

2.4 ความรู้พื้นฐาน Angular Fronted Framework

Angular [4] เป็นเฟรมเวิร์คที่เน้นเรื่องการทำ User Interface คือเป็นจาวาสคริปต์เฟรมเวิร์คที่เข้ามาช่วยเรื่องการแสดงผล เป็นเฟรมเวิร์คที่เปิดโอเพนซอร์ส โดยวัตถุประสงค์หลังของการงานสำหรับสร้างแอพลิเคชั่นในฝั่งไคลเอ็นในรูปแบบของ HTML, CSS และ JavaScript/ TypeScript ซึ่ง TpyeScript จะถูก compile ไปเป็น JavaScript

แองกูลาร์เป็นเฟรมเวิร์ค ช่วยจัดการและทำให้การพัฒนาเว็บง่ายขึ้น เน้นพัฒนาส่วนติดต่อผู้ใช้ของเว็บ เช่น components, declarative UI, hot-reloading, time-travel debugging และอื่น ๆ ให้สะดวกต่อการใช้งาน พยายยามลดสิ่งที่ไม่จำเป็นต่อการพัฒนาเว็บออกทำให้มีขนาดเล็กและง่ายต่อการนำไปใช้ของนักพัฒนา

2.4.1 การติดตั้ง

การติดตั้งเพื่อใช้งานแองกูลาร์จำเป็นต้องติดตั้ง Nodejs เวอร์ชั่น 6.9 ขึ้นไป และติดตั้ง npm เวอร์ชั่น 3 ขึ้นไป เมื่อตรวจสอบการติดตั้งว่าพร้อมแล้ว ก็สามารถติดตังและรันแองกูลาร์ได้ดังวิธีต่อไปนี้

- Angular Cli เป็นชุดคำสั่งที่ใช้ในการสร้าง Project ด้วย Angular ซึ่งได้มีการรวบรวมชุดเครื่องมือรวมถึงไลบรารีต่าง ๆ ที่จำเป็นต่อการพัฒนาไว้ โดยเราสามารถเลือก Templates เพื่อใช้งานตามความเหมาะสมของงานได้
    1. ทำการใช้คำสั่ง npm install -g @angular/cli เพื่อติดตั้ง angular CLI
    2. ทำการใช้คำสั่ง ng new my-project เพื่อการสร้าง Project
    3. ทำการใช้คำสั่ง cd my-project เพื่อเข้าไปที่ path ของ Project
    4. ทำการใช้คำสั่ง npm install เพื่อติดตั้ง node module ซึ่งเป็นชุดเครื่องมือที่ใช้สำหรับพัฒนาเว็บแอปพลิเคชัน

5. ทำการใช้คำสั่ง ng serve เริ่มการทำงานของโปรแกรม

2.4.2 คุณลักษณะของแองกูลาร์

ในหัวข้อนี้จะขอกล่าวถึงคุณลักษณะของแองกูลาร์เวอร์ชั่น 8 เพราะเป็นเวอร์ชั่นที่ใช้พัฒนาระบบนี้

- Angular CLI 8 เปลี่ยนมาใช้ dart-sass แทน node-sass ทำให้การ build ไฟล์ Sass เร็วขึ้นอีกมาก
- ใน Angular 8 รองรับการใช้งาน compiler (และ runtime) ตัวใหม่ที่ชื่อว่า Ivy ทำให้ประสิทธิภาพของโปรเจค Angular เราดีขึ้นมาก สามารถทำได้ตอนสร้างโปรเจคใหม่ ด้วยคำสั่ง ดังนี้

```
1 ng new project –enable-ivy
```

รูปที่ 2.12: สร้างโปรเจคใหม่แบบ Ivy

2.4.3 ข้อดีของ Angular

- ช่วยแยก Logic การตัดสินใจออกจากโค๊ด (Code) การแสดงผล
- ช่วยแยกหน้าเว็บออกเป็น Component ทำให้การจัดการง่ายขึ้นและนำกลับมาใช้ได้
- ช่วยจัดการเรื่อง Dynamic data
- มีการเก็บสถานะต่าง ๆ ไว้ที่จุดเดียวเพื่อให้ง่ายต่อการจัดการและการเรียกใช้งานของ Component โดยการใช้งาน Angular

2.5 ความรู้พื้นฐาน MongoDB

MongoDB [5] คือ ฐานข้อมูลแบบไม่มีความสัมพันธ์ของตารางแบบมีโครงสร้าง (Structured Query Language: SQL) หรือเรียกอีกชื่อหนึ่งว่า NoSQL มีรูปแบบการจัดการเก็บข้อมูลแบบ JSON (JavaScript Object Notation)

2.5.1 ทำไมถึงเลือกใช้ MongoDB

ในการพัฒนาซอฟต์แวร์อาจพบว่าการเปลี่ยนแปลงของข้อมูลจากระบบไปยังฐานข้อมูลเชิงโครงสร้างและต้องนำข้อมูลนั้นมาแสดงผลนั้นมีความยุ่งยาก ซึ่งในบางครั้งอาจจะมีการแก้ไขจัดการปัญหาด้วยการทำ Normalization แต่ผลลัพธ์ที่ได้ส่วนใหญ่ระบบกลับไม่สามารถรองรับการใช้งานจำนวนมากได้ MongoDB จึงมีบทบาทเพื่อมาแก้ไขปัญหาดังกล่าว ทำให้ไม่จำเป็นต้องแปลงข้อมูลซ้ำซ้อน และโดยเฉพาะในปัจจุบันระบบงานมีรูปแบบข้อมูลหลากหลานมาก มีปริมาณสูง ทำให้ระบบจัดเก็บข้อมูลต้องพร้อมรองรับได้ โดยสาเหตุหลักที่สมควรเลือกใช้งานฐานข้อมูล MongoDB คือ เพื่อปรับปรุงการพัฒนาระบบงานที่ใช้งานฐานข้อมูล เพื่อปรับปรุงประสิทธิภาพในการเข้าถึงข้อมูล เมื่อข้อมูลมีปริมาณมาก เพื่อลกเวลาในการรับส่งข้อมูล และเพิ่มจำนวน transaction/request ที่ถูกสร้างขึ้น

2.5.2 Schema-less คืออะไร

Shema-less คือ การไม่กำหนดโครงสร้างของฐานข้อมูล NoSQL ซึ่งมีความแตกต่างจาก SQL ปกติ เช่น Collection มีแค่ name ต่อมาเราสามารถเพิ่มการเก็บ position ก็สามารถเพิ่มได้เลย

```
1 { "name" : "Phon"}
2
3 { "name" : "Phon", "position" : "Developer"}
```

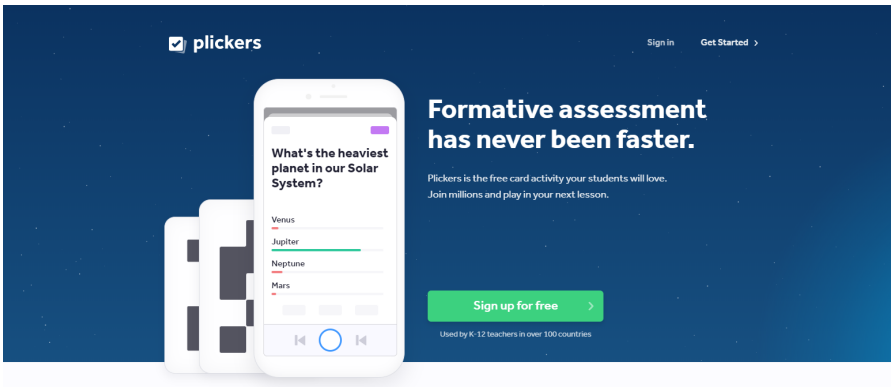
รูปที่ 2.13: ตัวอย่างข้อมูลใน Collection

2.6 เอกสารและงานวิจัยที่เกี่ยวข้อง

กล่าวถึงเอกสาร งานวิจัย หรือระบบงานที่คล้ายกันโดยแบ่งเป็น subsection โดยแต่ละหัวข้อให้อธิบายความสำคัญ ฟังก์ชันการทำงาน ข้อจำกัดหรือข้อแตกต่างจะระบบที่จะทำ เช่น

2.6.1 เว็บไซต์ Plickers

Plickers [6] เป็นเว็บไซต์ที่ให้บริการคุณครูและอาจารย์เก็บข้อมูลของนักเรียน นักศึกษาโดยจะมีกระดาษเป็นโค้ดให้นักเรียน นักศึกษาถือซึ่งกระดาษจะมีด้านที่แตกต่างกัน 4 ด้าน โดยแต่ละด้านจะมีตัวเลขเฉพาะของนักเรียน นักศึกษาแต่ละคนอยู่ที่มุมกระดาษ ซึ่งโค้ดที่นักเรียน นักศึกษาที่ได้นั่นจะนำมาใช้ในการตอบคำถามหรือเช็คชื่อมาเรียนโดยคุณครูหรืออาจารย์เป็นผู้สแกนแล้วให้นักเรียน นักศึกษาพลิกกระดาษโค้ดของตนเองเพื่อเลือกคำตอบ เว็บไซต์ plickers สามารถเข้าไปใช้งานได้ที่ https://get.plickers.com ซึ่งเป็นระบบเปิดให้ใช้งานได้ฟรี และมีฟังชั่นต่างๆมากมายให้ลอกใช้งาน



รูปที่ 2.14: หน้าแรกของเว็บไซต์ plickers

ที่มา : https://get.plickers.com

- ข้อดี

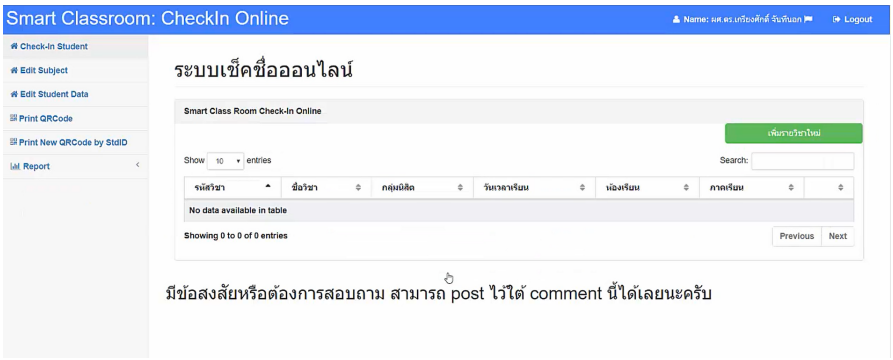
  - ฟังก์ชันการใช้งานหลากหลาย เช่น สามารถทำแบบฝึกหัดออนไลน์ได้ สามารถใช้ในการเช็คชื่อเข้าเรียนได้ และอัพโหลดสื่อการสอนได้ เป็นต้น
  - สามารถใช้ได้ทั้งคอมพิวเตอร์ และโทรศัพท์มือถือ โดยเข้าได้ทั้งในเว็บเบราว์เซอร์และแอปพลิเคชันบนโทรศัพท์มือถือ
  - ใช้งานได้ฟรี

- ข้อเสีย

  - ในการเช็คชื่อหรือการทำแบบฝึกหัดยังจำเป็นจะต้องปริ้นคิวอาร์โค้ดให้นักเรียน นักศึกษาถือซึ่งถ้าหากคิวอาร์โค้ดชำรุดหรือขาด ก็จะไม่สามารถใช้งานคิวอาร์โค้ดได้

2.6.2 เว็บไซต์ smart classroom QR-Code Check-in

smart classroom QR-Code Check-in [?] เป็นเว็บไซต์ที่ให้บริการคุณครูและอาจารย์ฟรีในการเช็คชื่อแบบออนไลน์บนเว็บเบราว์เซอร์ เว็บไซต์นี้พัฒนาโดย ผู้ช่วยศาสตราจารย์ ดร.เกรียงศักดิ์ จันทีนอก และอาจารย์ ดร.รัตนาวดี สนธิประสาท คณะการบัญชีและการจัดการ มหาวิทยาลัยมหาสารคามซึ่งเปิดให้คุณครูและอาจารย์ใช้งานได้ฟรี



รูปที่ 2.15: หน้าแรกของเว็บไซต์ smart classroom QR-Code Check-in

ที่มา : http://www.mse-exam.net/qr

- ข้อดี

  - ใช้งานง่าย และฟรี

  - สามารถใช้ได้ทั้งคอมพิวเตอร์ และโทรศัพท์มือถือ โดยเข้าได้ทั้งในเว็บเบราว์เซอร์และแอปพลิเคชันบนโทรศัพท์มือถือ

- ข้อเสีย

  - ในการเช็คชื่อหรือการทำแบบฝึกหัดยังจำเป็นจะต้องปริ้นคิวอาร์โค้ดให้นักเรียน นักศึกษาถือซึ่งถ้าหากคิวอาร์โค้ดชำรุดหรือขาด ก็จะไม่สามารถใช้งานคิวอาร์โค้ดได้

2.6.3 โปรแกรม Student Care

Student Care [7] เป็นโปรแกรสำหรับเช็คชื่อนักเรียนโดยใช้การสแกนในหน้าผ่านกล้อง AI โดยระบบจะตรวจจับใบหน้านักเรียนแล้วตรวจสอบข้อมูลในฐานข้อมูล ซึ่งอำนวยความสะดวกให้กับคุณครูและนักเรียนในความรวดเร็วของการเช็คชื่อ นอกจากนี้ยังมีระบบแจ้งเตือนไปให้ผู้ปกครองว่านักเรียนมาเรียนหรือไม่ โดยระบบจะแจ้งเตือนผ่านทาง Line Application

รูปที่ 2.16: หน้าแรกของเว็บไซต์ Student Care

ที่มา : http://www.student.co.th/

- ข้อดี
  - มีเทคโนโลยีที่ทันสมัยและใช้งานง่าย
  - สามารถใช้ได้ทั้งคอมพิวเตอร์ และโทรศัพท์มือถือ โดยเข้าได้ทั้งในเว็บเบราว์เซอร์และแอป-พลิเคชันบนโทรศัพท์มือถือ
  - มีการแจ้งเตือนถึงผู้ปกครองของนักเรียน - ไม่จำเป็นต้องใช้คีย์การ์ดหรือคิวอาร์โค้ด
- ข้อเสีย
  - การจะใช้งานระบบ Student Care จำเป็นต้องเสียค่าบริการรายเดือน
  - ระบบสแกนใบหน้ายังไม่สามารถแยกภาพถ่ายกับคนจริงๆ ได้

บทที่ 3

การวิเคราะห์และออกแบบระบบ

การวิเคราะห์และออกแบบระบบก่อนดำเนินการจริงเป็นอีกหนึ่งขั้นตอนที่มีความสำคัญมาก เพราะการวิเคราะห์และออกแบบระบบนั้นเป็นการกระทำที่ทำให้ผู้พัฒนาเห็นรายละเอียดส่วนย่อยของงานทั้งหมด เพิ่มประสิทธิภาพในการวางแผน การทำงาน และยังช่วยลดปัญหาที่อาจจะเกิดขึ้นในระหว่างพัฒนา เพื่อให้ระบบมีความสมบูรณ์มากยิ่งขึ้น เนื่องจากการวิเคราะห์และออกแบบระบบนั้นจะช่วยให้ให้บริการ จัดการทรัพยากรได้อย่างคุ้มค่าและตรงตามความต้องการของระบบ

การวิเคราะห์และออกแบบระบบเช็คชื่อด้วยคิวอาร์โค้ด ในบทนี้จะมีขั้นตอนเพื่อให้เห็นการดำเนินงานอย่างมีระบบ ในหัวข้อแรกจะนำเสนอภาพรวมของระบบ ก่อนจะนำเสนอเอกสารแสดงความต้องการของระบบซึ่งจะทำให้เห็นที่มาของเพจต่าง ๆ ในขั้นตอนของการออกแบบในหัวข้อที่สาม ส่วนหัวข้อที่เหลือจะแสดงแผนภาพการการทำงานของระบบโดยใช้ UML diagram ซึ่งประกอบไปด้วย Use Case, Class และ Sequence Diagram เพื่อแสดงรายละเอียดของระบบก่อนนำไปเขียนคำสั่งด้วยภาษาโปรแกรมในบทต่อไป

3.1 โครงสร้างภาพรวมของระบบ (System Architecture) เป็นการออกแบบภาพรวมและเทคโนโลยีของระบบ

3.2 System Requirements คือ ความต้องการหรือสิ่งที่ระบบควรจะทำ หรือหน้าที่หลักของระบบที่จะต้องทำ

3.3 User Interface Design เป็นการออกแบบส่วนต่อประสานกับผู้ใช้

3.4 Use Case Diagram เป็นแผนภาพที่ใช้แสดงให้ทราบว่าระบบทำงานหรือมีหน้าที่ใดบ้าง

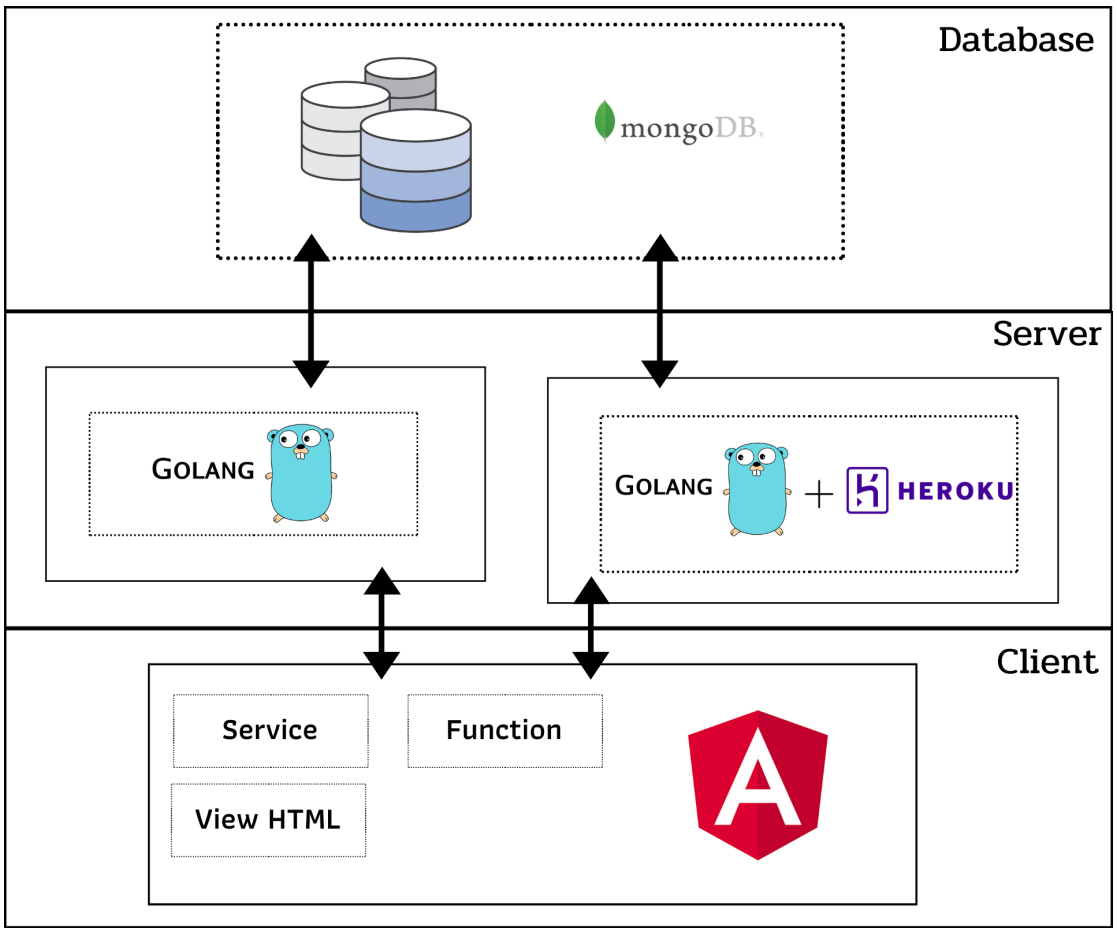
3.5 Class Diagram เป็นแผนภาพที่ใช้แสดง Class และความสัมพันธ์ระหว่าง Class

3.6 Sequence Diagram เป็นแผนภาพที่ใช้แสดงให้เห็นถึงการตอบโต้ข้อมูลระหว่างคลาส เรียงตามลำดับของเวลาที่เกิดเหตุการณ์จากน้อยไปมาก

3.1 โครงสร้างภาพรวมของระบบ

ความหมายของ System Architecture [8] หมายถึง กรอบโครงสร้างของระบบที่อธิบายความสัมพันธ์ขององค์ประกอบต่าง ๆ ไปจนถึงขั้นการเชื่อมต่อกันของระบบย่อยต่าง ๆ โดยจัดกลุ่มองค์ประกอบไว้ในหลาย ๆ ลักษณะเพื่อให้ผู้เกี่ยวข้อง (Stakeholder) จากพื้นฐานสาขาอาชีพที่แตกต่าง กันสามารถทำความเข้าใจได้ง่าย เช่น การจัดแบ่งองค์ประกอบตามลักษณะการทำงานของระบบ (functional components) เป็นต้น

การออกแบบ System architecture แสดงภาพรวมและเทคโนโลยีของระบบเช็คชื่อด้วยคิวอาร์โค้ด มีรายละเอียดดังรูปที่ 3.1



รูปที่ 3.1: System architecture ระบบเช็คชื่อด้วยคิวอาร์โค้ด

จากรูปที่ 3.1 สามารถอธิบายโครงสร้างและเทคโนโลยีของระบบโดยแบ่งเป็น 3 ส่วนหลัก ดังนี้

1. Database ระบบใช้บริการฐานข้อมูลแบบ NoSQL ของ MongoDB
2. Server กระบวนการทำงานในส่วนของเซิฟเวอร์ (server) แบ่งเป็น 2 ส่วนได้แก่
    - Heroku เป็นบริการสำหรับเป็น Api กลางเพื่อเปลี่ยนสถานะของคิวอาร์โค้ด ซึ่งในที่นี้ใช้ Go ในการพัฒนา
    - ชุดบริการ Api ใช้สำหรับการทำงานกับบริการต่าง ๆ ของระบบตัวอย่างเช่น ใช้สำหรับเก็บข้อมูลชั้นเรียน และเก็บคะแนนของนักศึกษา เป็นต้น โดยใช้ Go ในการพัฒนา
3. Client เว็บแอปพลิเคชันวัตถุประสงค์ในใช้งานเพื่อรองรับการทำงานของผู้ใช้งานบนบราวเซอร์โดยพัฒนาด้วย Angular

3.2 System Requirements

3.2.1 Functional Requirements

ระบบเช็คชื่อด้วยคิวอาร์โค้ด แบ่งความสามารถของระบบตามประเภทของผู้ใช้งานดังนี้

1. อาจารย์
    - สามารถสร้างและเพิ่มชั้นเรียนได้
    - สามารถตรวจสอบรายชื่อของนักศึกษาได้
    - สามารถจัดการคะแนนของนักศึกษาได้
    - สามารถคิวอาร์โค้ดให้นักศึกษาสแกนได้
    - สามารถกำหนดเวลาของคิดอาร์โค้ดได้
    - สามารถเพิ่ม ลบและแก้ไข ชื่อกับรหัสนักศึกษาได้
    - สามารถดาวน์โหลดไฟล์รายชื่อของนักศึกษาได้
    - สามารถรวมไฟล์คะแนน สำหรับวิชาเดียวกันได้
2. นักศึกษา
    - สามารถสแกนคิวอาร์โค้ดของอาจารย์เพื่อยืนยันการมาเรียนได้

3.2.2 Non-functional Requirements

1. เว็บแอปพลิเคชัน
    - ใช้โปรโตคอล (Protocol) แบบ HTTPS (Hypertext Transfer Protocol Secure) ในการสื่อสารที่ช่วยรักษาความสมบูรณ์ถูกต้องของข้อมูลผู้ใช้และเก็บข้อมูลไว้เป็น

ความลับระหว่างคอมพิวเตอร์ของผู้ใช้กับเว็บไซต์

- รองรับการใช้งานของผู้ใช้พร้อมกันอย่างน้อย 100 คน

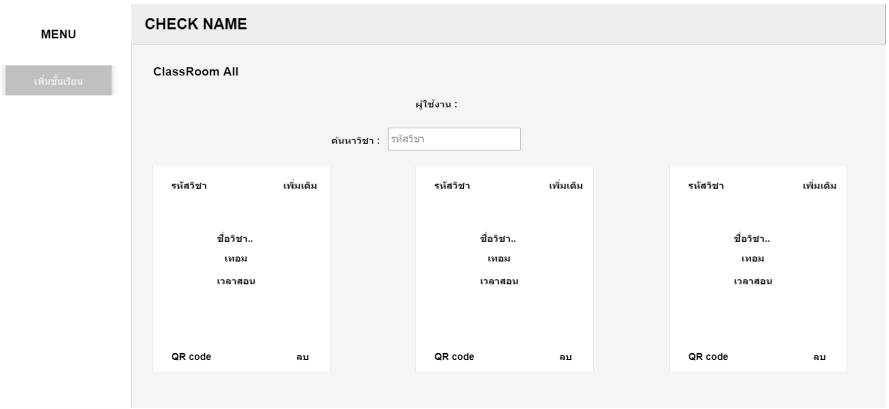
3.3 User Interface Design

ในการออกแบบ User Interface Design ของระบบเช็คชื่อด้วยคิวอาร์โค้ด ใช้การออกแบบใน 2 ส่วนคือ ส่วนของอาจารย์ และส่วนของนักศึกษา

3.3.1 ส่วนของอาจารย์

โดยการออกแบบหน้าจอของอาจารย์ประกอบด้วยส่วนต่างๆดังนี้

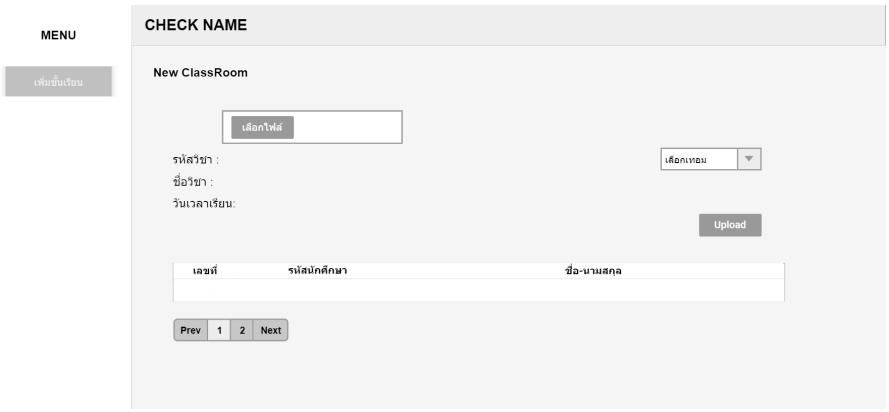
3.3.1.1 การออกแบบหน้าหลัก



รูปที่ 3.2: หน้าหลัก

จากภาพที่ 3.2 แสดงหน้าจอสำหรับให้อาจารย์ใช้ทำการตรวจสอบชั้นเรียน

3.3.1.2 การออกแบบหน้าเพิ่มชั้นเรียน



รูปที่ 3.3: หน้าจอสร้างชั้นเรียน

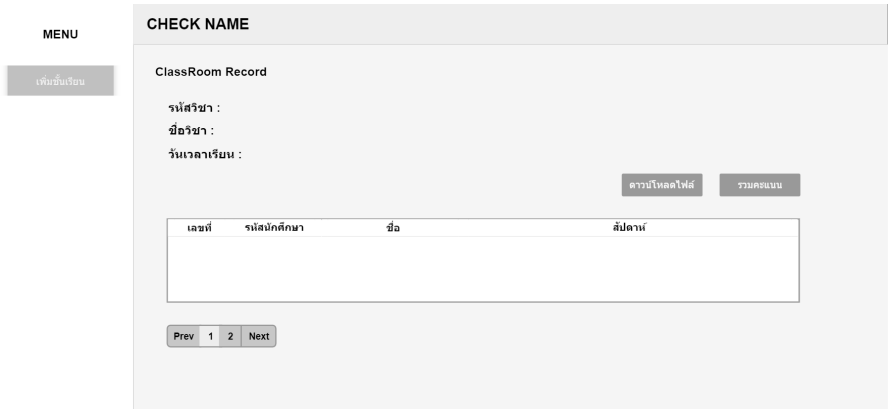
จากภาพที่ 3.3แสดงหน้าจอสร้างชั้นเรียนสำหรับอาจารย์ใช้ทำการ เพิ่มชั้นเรียนใหม่ โดยการเพิ่มไฟล์รายชื่อนักศึกษาและใส่ข้อมูลให้ครบ

3.3.1.3 การออกแบบหน้าจัดการคะแนนนักศึกษา

รูปที่ 3.4: หน้าจอจัดการคะแนนนักศึกษา

จากภาพที่ 3.4 แสดงหน้าจอจัดการคะแนนนักศึกษาเพื่อตรวจสอบคะแนนของนักศึกษาและจัดการคะแนนของนักศึกษา

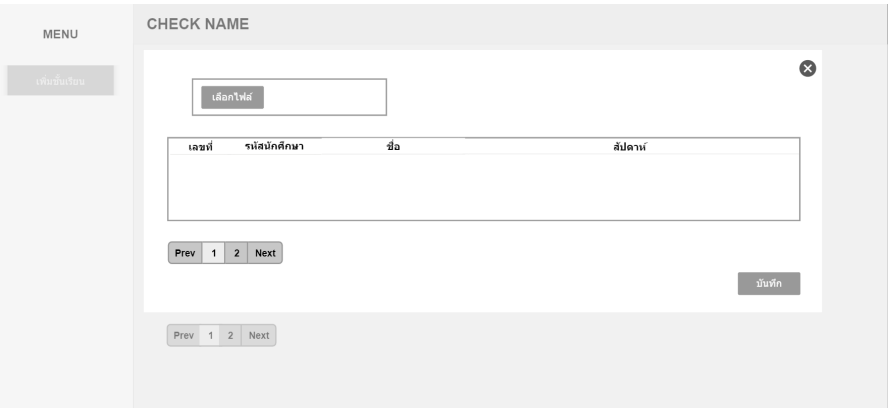
3.3.1.4 การออกแบบหน้าประวัติการเข้าเรียน



รูปที่ 3.5: หน้าจอประวัติการเข้าเรียน

จากภาพที่ 3.5แสดงหน้าจอประวัติการเข้าเรียนเพื่อดูข้อมูลการเข้าเรียนของนักศึกษา และในหน้านี้ยังใช้ในการเข้าถึงการดาวน์โหลดไฟล์ หรือเข้าถึงหน้ารวมไฟล์ได้

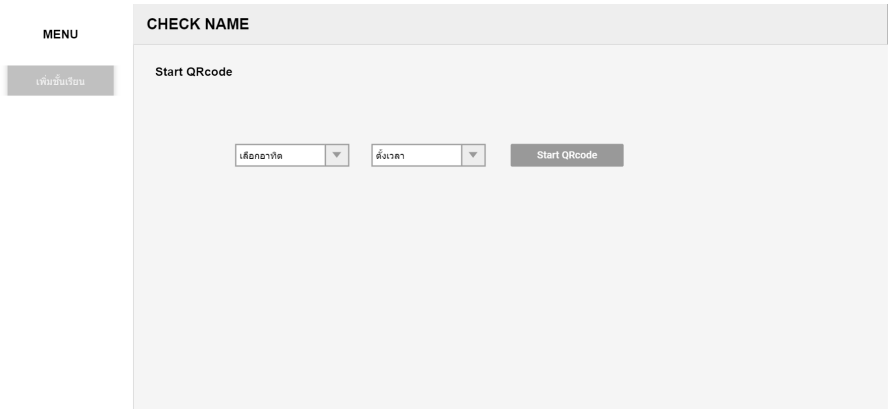
3.3.1.5 การออกแบบหน้ารวมไฟล์



รูปที่ 3.6: หน้ารวมไฟล์

จากภาพที่ 3.6 แสดงหน้าจอหน้ารวมไฟล์เป็น Pop-up เพื่อจัดการรวมไฟล์โดยการเลือกไฟล์ที่จะทำการรวม ซึ่งไฟล์ที่จะสามารถทำการรวมไฟล์ได้นั่นต้องเป็นวิชาเดียวกันเท่านั่น

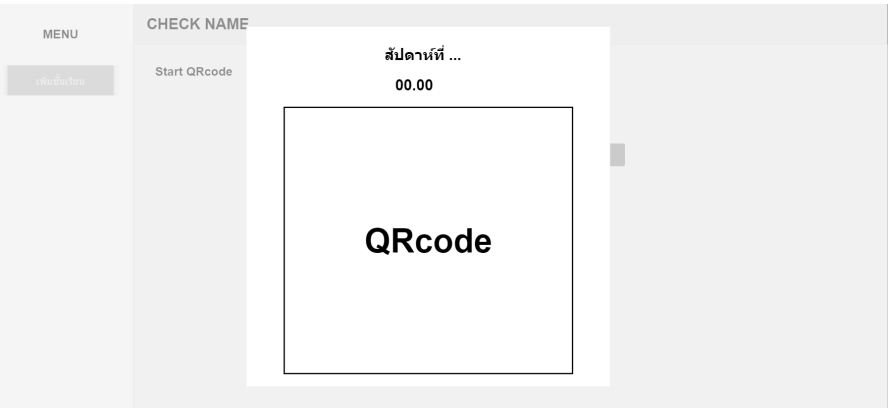
3.3.1.6 การออกแบบหน้าสร้างคิวอาร์โค้ด



รูปที่ 3.7: หน้าสร้างคิวอาร์โค้ด

จากภาพที่ 3.7 แสดงหน้าจอสร้างคิวอาร์โค้ด เพื่อให้อาจารย์ตั้งค่าต่างๆของคิวอาร์โค้ด ได้แก่ เวลา และอาทิตย์ที่สร้างคิวอาร์โค้ด

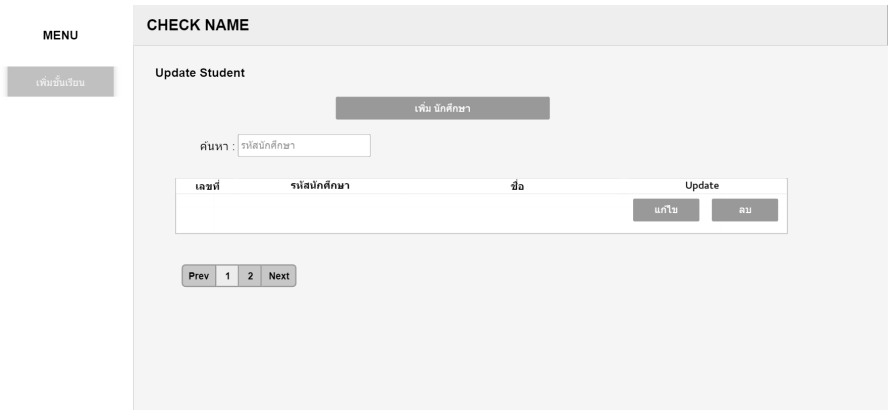
3.3.1.7 การออกแบบหน้าแสดงคิวอาร์โค้ด



รูปที่ 3.8: หน้าแสดงคิวอาร์โค้ด

จากภาพที่ 3.8 แสดงหน้าจอแสดงคิวอาร์โค้ด หน้านี้จะแสดงก็ต่อเมื่ออาจารย์ตั้งค่าและกดสร้างคิวอาร์โค้ดเท่านั่น โดยจะแสดงคิวอาร์โค้ดที่สร้างรวมถึงเวลาที่เหลือในการสแกน

3.3.1.8 การออกแบบหน้าจัดการนักศึกษา



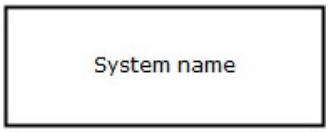
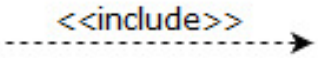
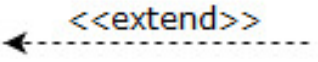
รูปที่ 3.9: หน้าจัดการนักศึกษา

จากภาพที่ 3.9 แสดงหน้าจอจัดการนักศึกษาเพื่อจัดการข้อมูลของนักศึกษา เช่น แก้ไข ลบ และเพิ่มนักศึกษา เป็นต้น

3.4 Use Case Diagram

Use Case Diagram เป็นแผนผังเพื่อแสดงฟังก์ชันแสดงการทำงานของระบบโดยรวม แสดงส่วนประกอบในระบบและกิจกรรมที่เกิดขึ้นในระบบซึ่งในระบบเช็คชื่อด้วยคิวอาร์โค้ด ผู้ใช้จำเป็นต้องเข้าสู่ระบบเพื่อใช้งานระบบ สัญลักษณ์ที่ใช้ในการเขียน Use Case Diagram แสดงในตารางที่ 3.1

ตารางที่ 3.1: สัญลักษณ์ของ Use case Diagram

สัญลักษณ์	การใช้งาน
Use case	Use case คือส่วนย่อยของระบบงาน แทนด้วยวงรีและชื่อของ Use case ภายในวงรี
	Actor คือบุคคลหรือระบบงานอื่นที่ใช้งานระบบหรือได้รับประโยชน์จากระบบซึ่งอยู่ภายนอกระบบ แทนด้วยรูปคนและมีชื่อบทบาทการใช้งานระบบ
	เส้นตรงที่แสดงถึงการใช้งาน Use case ของผู้กระทำ
	กรอบ สี่เหลี่ยม แสดง ถึง ขอบเขต ของ ระบบ โดย แสดง ชื่อ ระบบ ภายในหรือด้านบนกรอกสี่เหลี่ยม Use case อยู่ภายในกรอบสี่เหลี่ยม และ actor อยู่ภายนอกกรอบสี่เหลี่ยม
	ความสัมพันธ์แบบ «includes» แสดงว่า Use case หนึ่งดำเนินการตามขั้นตอนของ Use case อื่น โดยแทนด้วยสัณลักษณ์ลูกศรเส้นประ ซึ่ง Use case ที่หางลูกศรเรียกใช้งาน Use case ที่หัวลูกศรทุกครั้งที่มีการทำงาน
	ความสัมพันธ์แบบ «extend» แสดงว่า Use case หนึ่งดำเนินการตามขั้นตอนของ Use case อื่น โดยแทนด้วยสัญลักษณ์ลูกศรเส้นประ ซึ่ง Use case ที่หัวลูกศรเรียกใช้งาน Use case ที่หางลูกศร แต่การใช้งานไม่จำเป็นต้องเกิดขึ้นทุกครั้งขึ้นอยู่กับเงื่อนไขระหว่างการทำงาน

รูปที่ 3.10: Use Case Diagram ของระบบเช็คชื่อด้วยคิวอาร์โค้ด

ตารางที่ 3.2: อธิบาย Use Case หน้าที่ของระบบ ในภาพที่ 3.10

Use Case	คำอธิบาย
สร้างวิชา	อาจารย์สามารถสร้างชั้นเรียนได้ โดยการที่จะสร้างชั้นเรียนใหม่จำเป็นจะต้องอัพโหลดไฟล์ที่ดาวน์โหลดมาจากระบบของมหาวิทยาลัยเท่านั่น ซึ่งไฟล์จะต้องเป็นไฟล์ Excel และไม่สามารถสร้างชั้นเรียนซ้ำได้
อัพเดทข้อมูล นักศึกษา	อาจารย์สามารถจัดการรายชื่อนักศึกษาได้โดยการ เพิ่ม ลบและแก้ไข ข้อมูลชื่อและรหัสนักศึกษาได้
สร้าง QR code	อาจารย์สามารถสร้าง QR code รวมถึงการตั้งค่า QR code เช่น เวลาในการสแกน QR code เป็นต้น
จัดการคะแนน	หน้าแสดงรายชื่อ และคะแนนแต่ละสัปดาห์ของนักศึกษา โดยอาจารย์สามารถจัดการกับคะแนนในแต่ละสัปดาห์ของนักศึกษาแต่ละคน
ดูประวัติการเข้าเรียน	อาจารย์สามารถตรวจสอบการเข้าเรียนของนักศึกษาทั้งหมดในชั้นเรียนนั้น โดยจะแสดงเป็นรายสัปดาห์
สแกน QR code	นักศึกษาสามารถสแกน QR code ของอาจารย์เพื่อเข้าสู่หน้ากรอกรหัสนักศึกษาเพื่อเป็นการยืนยันตัวตนในการมาเรียน โดยการจะสแกนได้นั้นขึ้นอยู่กับเวลาของ QR code ของอาจารย์ด้วย ซึ่งถ้าหากเวลาของ QR code หมด นักศึกษาก็ไม่สามารถสแกน QR code ได้
อัพเดท url	อัพเดท url สำหรับให้นักศึกษาสแกน
กรอกรหัสนักศึกษา	แสดงหน้าให้นักศึกษากรอกรหัสนักศึกษา เพื่อเป็นการยืนยันตัวตน โดยการจะมาที่หน้านี้ได้ต้องสแกน QR code ของอาจารย์เท่านั้น

ตารางที่ 3.3: Use Case สร้างวิชา

Use Case Title : สร้างวิชา	Use case Id : 1
Primary Actor : อาจารย์	
Stakeholder Actor : -	
Main Flow : อาจารย์สามารถสร้างหรือเพิ่มชั้นเรียนใหม่	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่มีไฟล์รายชื่อนักศึกษาที่ดาวน์โหลดมาจากระบบของมหา-วิทยาลัย จะไม่สามารถสร้างหรือเพิ่มชั้นเรียนใหม่ได้	

ตารางที่ 3.4: Use Case อัพเดทข้อมูลนักศึกษา

Use Case Title : เพิ่ม/ลบ/แก้ไข นักศึกษา	Use case Id : 2
Primary Actor : อาจารย์	
Stakeholder Actor : นักศึกษา	
Main Flow : อาจารย์สามารถ เพิ่ม ลบและแก้ไขรายชื่อหรือรหัสของนักศึกษาในชั้นเรียนที่สร้างขึ้นได้	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่มีการเพิ่มหรือสร้างชั้นเรียน ก็ไม่สามารถ เพิ่ม ลบและแก้ไขรายชื่อหรือรหัสของนักศึกษาได้	

ตารางที่ 3.5: Use Case สร้าง QR code

Use Case Title : สร้าง QR code	Use case Id : 3
Primary Actor : อาจารย์	
Stakeholder Actor : นักศึกษา	
Main Flow : อาจารย์สามารถสร้าง QR code เพื่อให้นักศึกษาใช้สแกนเพื่อเข้าสู่หน้ากรอกรหัสนักศึกษาได้	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่มีการเพิ่มหรือสร้างชั้นเรียน จะไม่อาจารย์สามารถสร้าง QR codeได้	
Exceptional Flow ที่ 2 : หากผู้ใช้ไม่เชื่อมต่ออินเทอร์เน็ต จะไม่อาจารย์สามารถสร้าง QR code-ได้	

ตารางที่ 3.6: Use Case จัดการคะแนน

Use Case Title : จัดการคะแนน	Use case Id : 4
Primary Actor : อาจารย์	
Stakeholder Actor : นักศึกษา	
Main Flow : อาจารย์สามารถจัดการคะแนนของนักศึกษาแต่ละสัปดาห์ได้	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่มีการเพิ่มหรือสร้างชั้นเรียน จะไม่จัดการคะแนนของนัก-ศึกษาได้	

ตารางที่ 3.7: Use Case ดูประวัติการเข้าเรียน

Use Case Title : ดูประวัติการเข้าเรียน	Use case Id : 5
Primary Actor : อาจารย์	
Stakeholder Actor : -	
Main Flow : หน้าแสดงข้อมูลการเข้าเรียนของนักศึกษาทั้งหมด	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่มีการเพิ่มหรือสร้างชั้นเรียน จะไม่สามารถดูข้อมูลการเข้าเรียนของนักศึกษาได้	

ตารางที่ 3.8: Use Case สแกน QR code

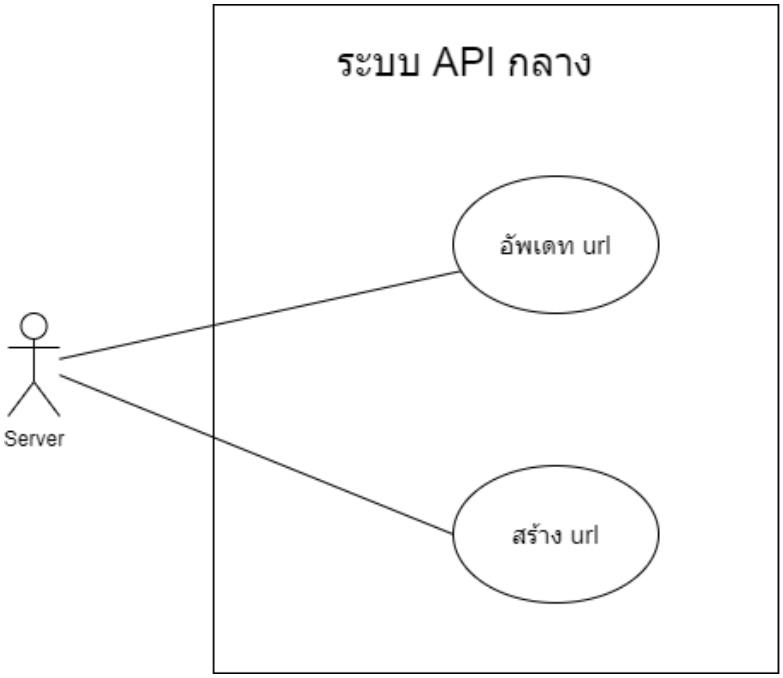
Use Case Title : สแกน QR code	Use case Id : 6
Primary Actor : นักศึกษา	
Stakeholder Actor : อาจารย์	
Main Flow : สแกน QR code ที่อาจารย์สร้างขึ้น	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่เชื่อมต่อ Wifi เดียวกันกับอาจารย์ จะไม่สามารถสแกน QR code	

ตารางที่ 3.9: Use Case อัพเดท url

Use Case Title : อาจารย์	Use case Id : 7
Primary Actor : อาจารย์	
Stakeholder Actor : -	
Main Flow : อัพเดท url ของคิวอาร์โค้ด	
Exceptional Flow ที่ 1 : ผู้ใช้ต้องทำการสร้างคิวอาร์ก่อน จึงจะสามารถอัพเดท url ได้	
Exceptional Flow ที่ 2 : ผู้ใช้จำเป็นต้องเชื่อมต่ออินเตอร์เน็ต จึงจะทำการอัพเดท url ได้	

ตารางที่ 3.10: Use Case กรอกรหัสนักศึกษา

Use Case Title : กรอกรหัสนักศึกษา	Use case Id : 8
Primary Actor : นักศึกษา	
Stakeholder Actor : -	
Main Flow : หน้ากรอกข้อมูลรหัสนักศึกษาเพื่อยืนยันตัวตน	
Exceptional Flow ที่ 1 : หากผู้ใช้สแกน QR code ของอาจารย์ไม่ผ่าน จะไม่สามารถเมาที่หน้ากรอกรหัสนักศึกษาได้	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่เชื่อมต่อ Wifi เดียวกันกับอาจารย์ จะไม่สามารถเมาที่หน้ากรอกรหัสนักศึกษาได้	



รูปที่ 3.11: Use Case Diagram ของระบบ API กลาง

ตารางที่ 3.11: อธิบาย Use Case หน้าที่ของระบบ ในภาพที่ 3.11

Use Case	คำอธิบาย
อัพเดท url	เมื่ออาจารย์สร้างคิวอาร์โค้ดระบบจะส่ง url ไปเก็บไว้ในฐานข้อมูลเพื่อให้นักศึกษาสแกน
สร้าง url	เมื่ออาจารย์สร้างคิวอาร์โค้ดครั้งแรกระบบจะส่ง url ไปเก็บไว้ในฐานข้อมูลเพื่อให้นักศึกษาสแกน

ตารางที่ 3.12: Use Case อัพเดท url

Use Case Title : อัพเดท url	Use case Id : 9
Primary Actor : อาจารย์	
Stakeholder Actor : -	
Main Flow : อัพเดท url ของคิวอาร์โค้ด	
Exceptional Flow ที่ 1 : ผู้ใช้ต้องทำการสร้างคิวอาร์ก่อน จึงจะสามารถอัพเดท url ได้	
Exceptional Flow ที่ 2 : ผู้ใช้จำเป็นต้องเชื่อมต่ออินเตอร์เน็ต จึงจะทำการอัพเดท url ได้	

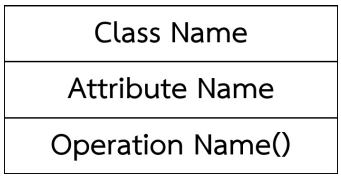
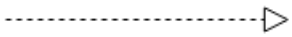
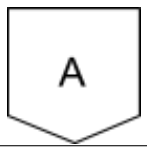
ตารางที่ 3.13: Use Case สร้าง url

Use Case Title : สร้าง url	Use case Id : 10
Primary Actor : อาจารย์	
Stakeholder Actor : -	
Main Flow : อัพเดท url ของคิวอาร์โค้ด	
Exceptional Flow ที่ 1 : ผู้ใช้ต้องทำการสร้างคิวอาร์ก่อน จึงจะสามารถสร้าง url ได้	
Exceptional Flow ที่ 2 : ผู้ใช้จำเป็นต้องเชื่อมต่ออินเตอร์เน็ต จึงจะทำการสร้าง url ได้	

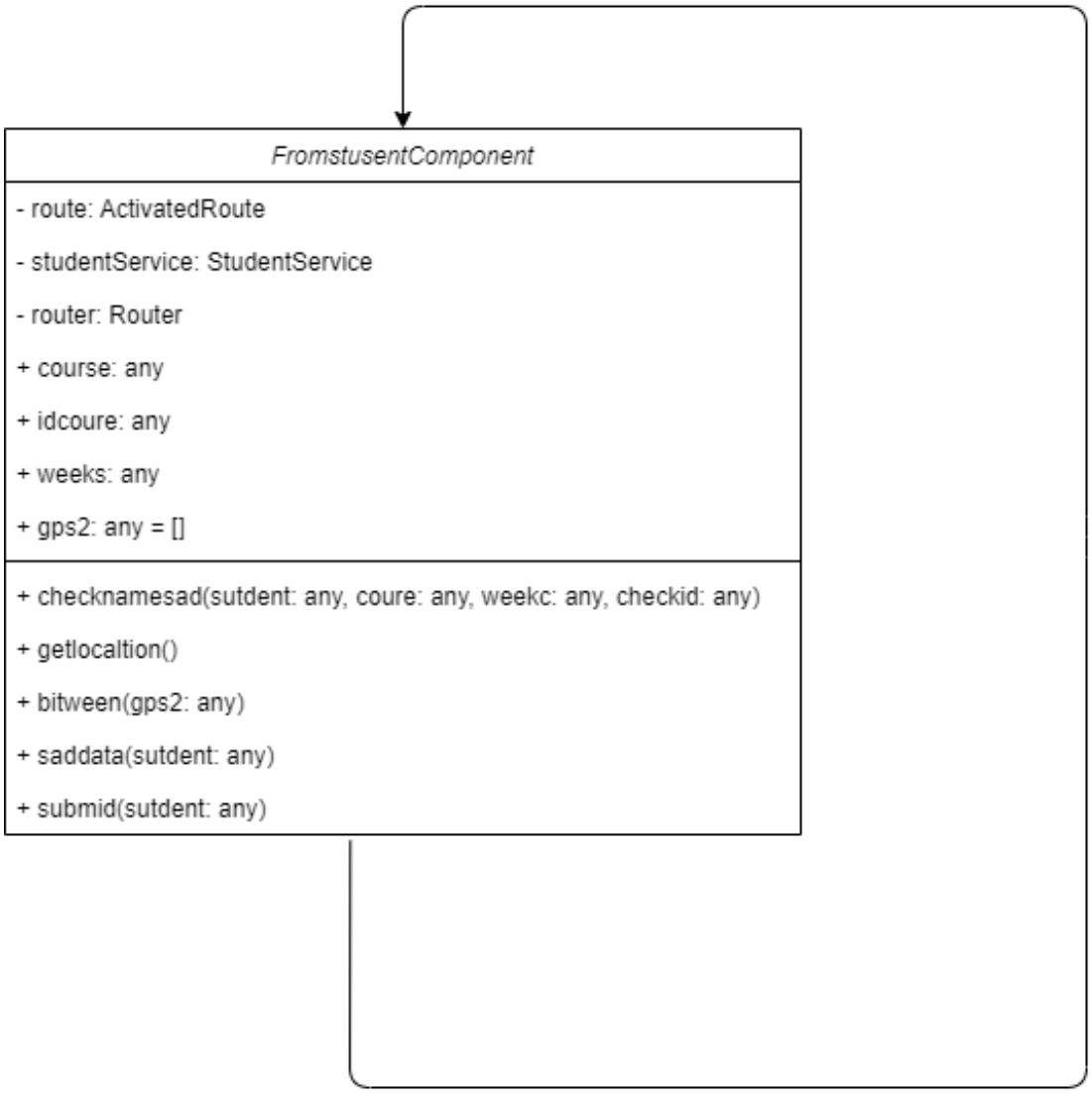
3.5 Class Diagram

Class Diagram คือแผนภาพที่ใช้แสดงคลาสและความสัมพันธ์ในแบบต่างๆ ระหว่างคลาส สัญลักษณ์ที่ใช้ในการเขียน Class Diagram แสดงในตารางที่ 3.14

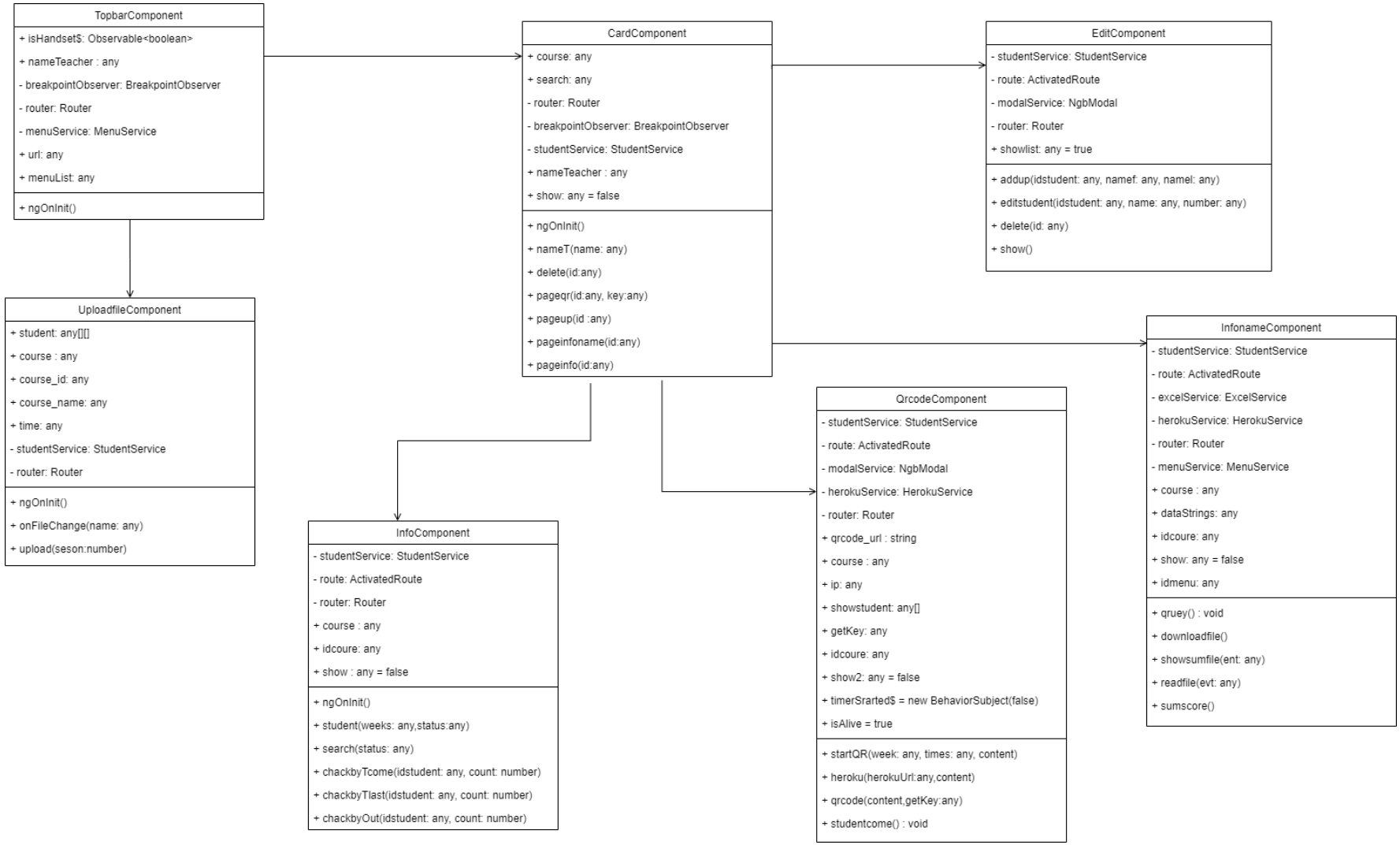
ตารางที่ 3.14: สัญลักษณ์ของ Class Diagram

สัญลักษณ์	การใช้งาน
	คลาส สัญลักษณ์แทนด้วยสี่เหลี่ยมแบ่งเป็น 3 ส่วน ส่วนบน เป็นชื่อของ class ส่วนกลาง เป็นชื่อ Attribute และส่วนล่างเป็น Operation Name หรือ Method ใช้สำหรับเขียนฟังก์ชันในการทำงานของคลาสนั้น ๆ ชนิดของ Visibility ของ Method และ Attribute แบ่งเป็น 3 ชนิด ได้แก่ 1. Public แทนสัญลักษณ์ด้วยเครื่องหมายบวก (+) 2. Private แทนสัญลักษณ์ด้วยเครื่องหมายลบ (-) 3. Protected แทนสัญลักษณ์ด้วยเครื่องหมายชาร์ป ()
	Dependency Relationship หมายความว่า คลาสที่อยู่ฝั่งต้นลูกศรสามารถเรียกใช้คลาสที่อยู่ฝั่งหัวลูกศร
	Composition Relationship เป็นความสัมพันธ์ระหว่างออบเจ็กต์หรือคลาสแบบขึ้นต่อกันและมีความเกี่ยวข้องกันเสมอ
	Realization Relationship เป็นความสัมพันธ์ระหว่าง Object หรือ Class ในลักษณะของการสืบทอดคุณสมบัติจาก Class หนึ่ง (Super class) ไปยังอีก Class หนึ่ง (Subclass)
	Connector เป็นสัญลักษณ์แทนด้วยรูปห้าเหลี่ยมและมีชื่ออยู่ตรงกลาง จะสร้างสัญลักษณ์นี้ไว้เมื่อต้องการเชื่อมต่อคลาสที่อยู่คนละหน้า

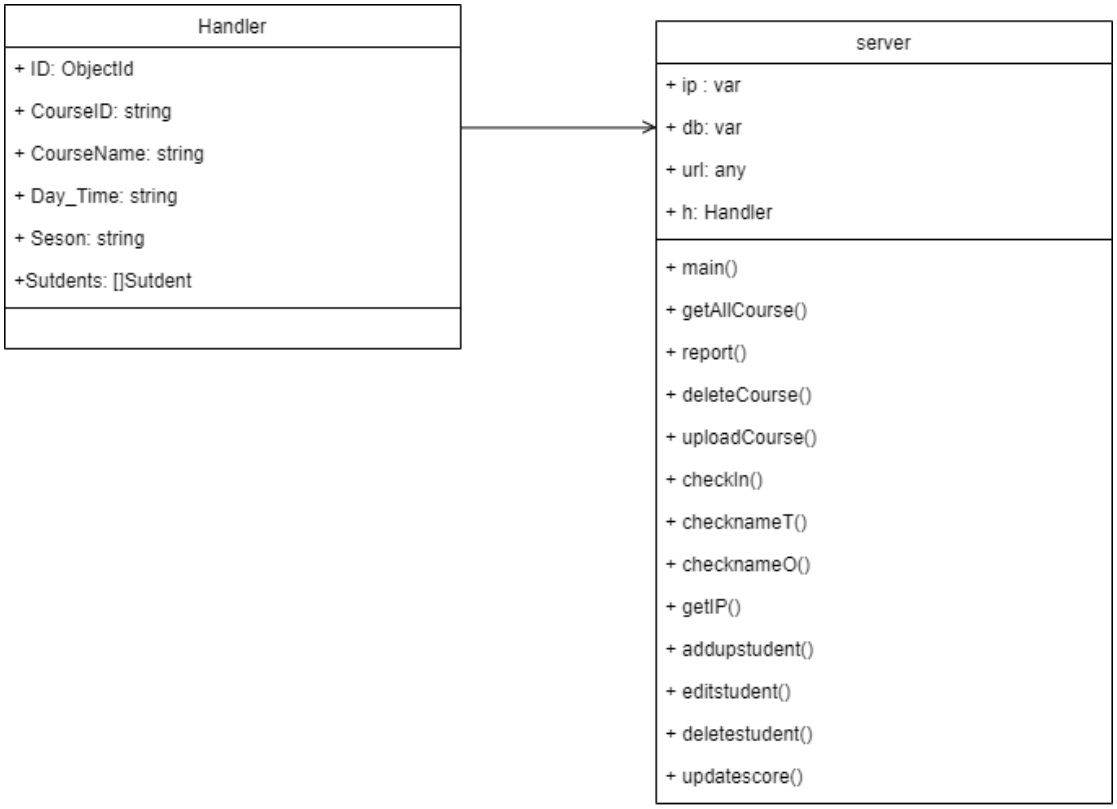
Class Diagram แสดงความสัมพันธ์ในรูปแบบต่างๆ ระหว่างคลาสของเว็บแอปพลิเคชันระบบเช็คชื่อด้วยคิวอาร์โค้ด อธิบายได้ตามภาพที่ 3.12 ดังต่อไปนี้



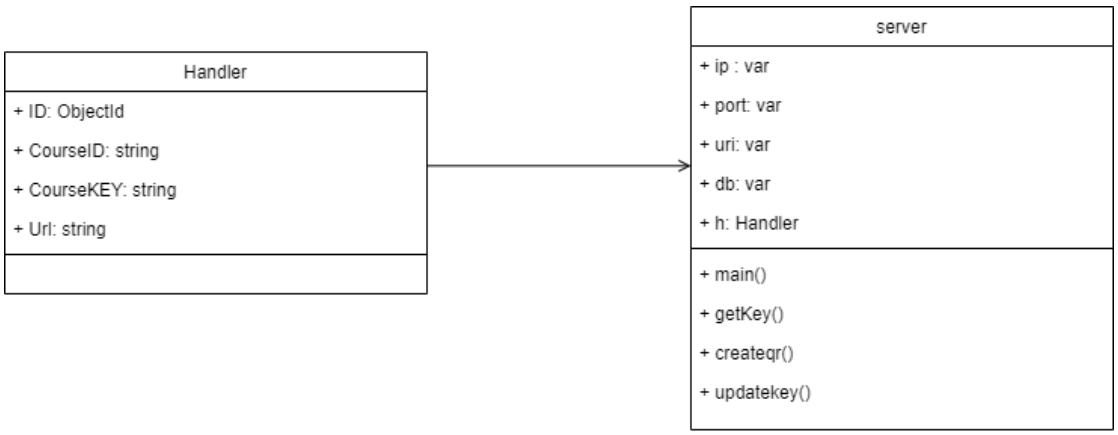
รูปที่ 3.12: Class Diagram ของแอปพลิเคชันระบบ เช็คชื่อด้วยคิวอาร์โค้ด



รูปที่ 3.13: Class Diagram ของแอปพลิเคชันระบบ เช็คชื่อด้วยคิวอาร์โค้ด



รูปที่ 3.14: Class Diagram ของแอปพลิเคชันระบบ API



รูปที่ 3.15: Class Diagram ของแอปพลิเคชันระบบ API กลาง

จากรูปภาพที่ 3.12 สามารถอธิบายแผนภาพ Class Diagram ได้ดังนี้

ตารางที่ 3.15: อธิบาย Class Diagram ของคลาสส่วนของนักศึกษา

Class Diagram	คำอธิบาย
FromstusentComponent	คลาส FromstusentComponent จะถูกเรียกใช้งานทุกครั้งเมื่อนักศึกษาสามารถสแกนคิวอาร์โค้ดของอาจารย์ได้ โดยวัตถุประสงค์การทำงานของคลาสนี้คือ เพื่อให้นักศึกษายืนยันตัวตนเพื่อรับคะ-แนนในการเข้าเรียน

จากรูปภาพที่ 3.13 สามารถอธิบายแผนภาพ Class Diagram ได้ดังนี้

ตารางที่ 3.16: อธิบาย Class Diagram ของคลาสส่วนของอาจารย์

Class Diagram	คำอธิบาย
TopbarComponent	คลาส TopbarComponent จะถูกเรียกใช้งานทุกครั้งเมื่ออาจารย์เข้าสู่เว็บแอพพลิเคชั่น โดยวุตถุประสงค์ของการทำงานของคลาสนี้คือ จัดการเกี่ยวกับการแสดงแถบเมนูส่วนบนของเว็บแอพพลิเคชั่น ซึ่งสามารถเรียกใช้งานคลาส CardComponent และ UploadfileComponent ได้
UploadfileComponent	คลาส UploadfileComponent เป็นคลาสที่มีหน้าที่ในการทำงานเกี่ยวกับการเพิ่มไฟล์รายชื่อนักศึกษา โดยวัตถุประสงค์ของคลาสนี้คือ ใช้ในการเพิ่มชั้นเรียนใหม่ ซึ่งการเพิ่มชั้นเรียนต้องเพิ่มไฟล์ที่ได้จากระบบของมหาลัยเท่านั้น นอกจากนี้ยังแสดงรายชื่อทั้งหมดในไฟล์ที่นำเข้าอีกด้วย
CardComponent	คลาส CardComponent จะถูกเรียกใช้งานทุกครั้งเมื่ออาจารย์เข้าสู่เว็บแอพพลิเคชั่น โดยวุตถุประสงค์ของการทำงานของคลาสนี้คือ แสดงชั้นเรียนทั้งหมดที่มี และแสดงเมนูสำหรับการเข้าถึงคลาสต่างๆ ดังนี้ InfoComponent QrcodeComponent EditComponent และ InfonameComponent
EditComponent	คลาส EditComponent เป็นคลาสที่มีหน้าที่ในการทำงานเกี่ยวกับการจัดการข้อมูลนักศึกษา เช่น การเพิ่ม ลบและแก้ไข ชื่อหรือรหัสนักศึกษา
QrcodeComponent	คลาส QrcodeComponent เป็นคลาสที่มีหน้าที่ในการทำงานเกี่ยวกับการสร้างคิวอาร์โค้ด โดยวัตถุประสงค์ของคลาสนี้คือ เพื่อตั้งค่าข้อมูลของคิวอาร์โค้ด และแสดงคิวอาร์โค้ดที่สร้างให้นักศึกษาสามารถสแกนได้

ตารางที่ 3.17: อธิบาย Class Diagram ของคลาสส่วนของอาจารย์

Class Diagram	คำอธิบาย
InfoComponent	คลาส InfoComponent เป็นคลาสที่มีหน้าที่ในการทำงานเกี่ยวกับการแสดงข้อมูลการเข้าเรียนทั้งหมดของนักศึกษา นอกจากนี้คลาสนี้ยังทำหน้าที่ในการดาวน์โหลดไฟล์รายชื่อ และรวมไฟล์รายชื่อเพื่อรวมคะแนนของนักศึกษาในกรณีที่มีอาจารย์มากกว่าหนึ่งคน
InfonameComponent	คลาส InfonameComponent เป็นคลาสที่มีหน้าที่ในการทำงานเกี่ยวกับการจัดการคะแนนของนักศึกษา โดยวัตถุประสงของคลาสนี้คือ การจัดการคะแนนของนักศึกษาทั้งหมดในรายวิชานั้นๆ

จากรูปภาพที่ 3.14 สามารถอธิบายแผนภาพ Class Diagram ได้ดังนี้

ตารางที่ 3.18: อธิบาย Class Diagram ของคลาสส่วนของอาจารย์

Class Diagram	คำอธิบาย
server	คลาส server จะถูกเรียกใช้งานทุกครั้งเมื่ออาจารย์เข้าสู่เว็บแอพพลิเคชั่น โดยวุตถุประสงค์ของการทำงานของคลาสนี้คือให้บริการ API และตรวจสอบข้อมูลต่างๆ
Handler	คลาส Handler เป็นคลาสที่มีหน้าที่ในการเป็นโมเดลของฐานข้อมูล

จากรูปภาพที่ 3.15 สามารถอธิบายแผนภาพ Class Diagram ได้ดังนี้

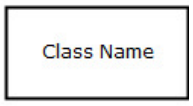
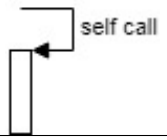
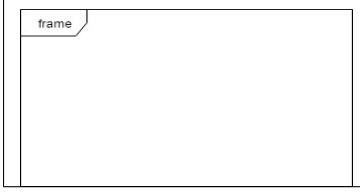
ตารางที่ 3.19: อธิบาย Class Diagram ของคลาสส่วนของอาจารย์

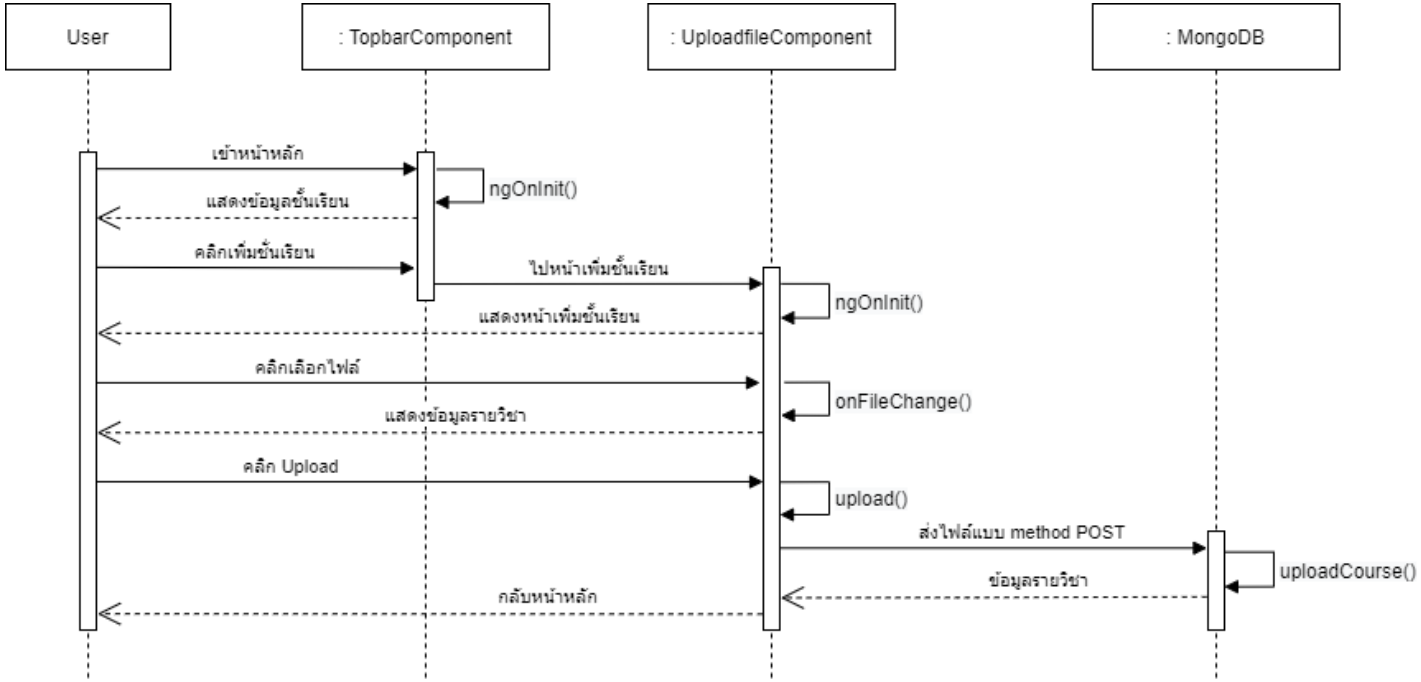
Class Diagram	คำอธิบาย
server	คลาส server จะถูกเรียกใช้งานทุกครั้งเมื่ออาจารย์เข้าสู่เว็บแอพพลิเคชั่น โดยวุตถุประสงค์ของการทำงานของคลาสนี้คือให้บริการ API และตรวจสอบข้อมูลต่างๆ
Handler	คลาส Handler เป็นคลาสที่มีหน้าที่ในการเป็นโมเดลของฐานข้อมูล

3.6 Sequence Diagram

Sequence Diagram เป็น Diagram ที่แสดงขั้นตอนการทำงานของแต่ละ Use Case ระหว่าง Object ต่างๆ ที่ส่งข้อความถึงกันและกัน โดย Sequence Diagram จะช่วยให้มองเห็นการทำงานของภาพรวมของระบบ ส่วนประกอบสัญลักษณ์ที่ใช้ในการเขียน Sequence Diagram แสดงดังตารางที่ 3.20

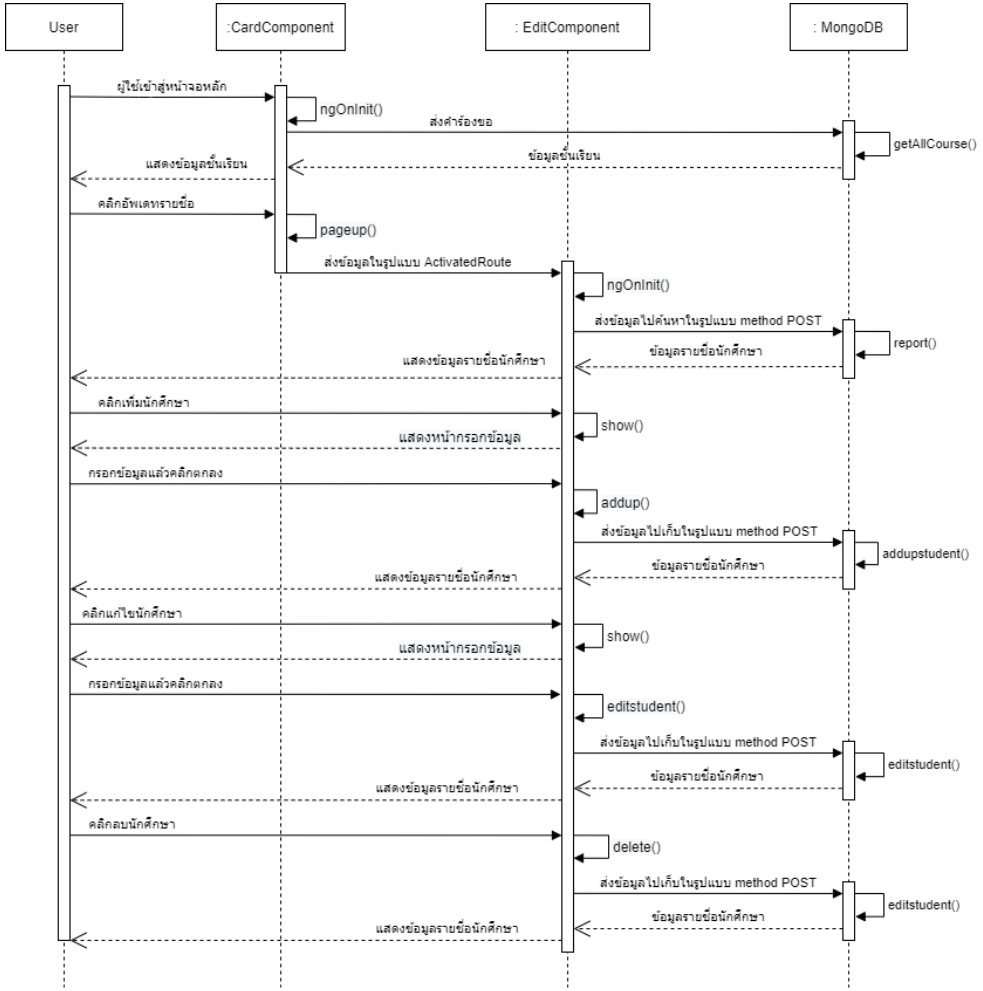
ตารางที่ 3.20: สัญลักษณ์ของ Sequence Diagram

สัญลักษณ์	การใช้งาน
	Class แสดงถึงการทำงานของ Use Case ในการส่งหรือรับข้อความ แทนด้วยสัญลักษณ์สี่เหลี่ยมมีชื่อคลาสอยู่ภายใน
	Lifeline หรือเส้นอายุขัย แสดงช่วงเวลาตั้งแต่เริ่มสร้าง object ในคลาสนั้น จนกระทั่ง object นั้นถูกทำลาย สัญลักษณ์แทนด้วยเส้นประ
	Focus of control หรือจุดควบคุม เป็นจุดควบคุมที่ object ใช้ทำการส่งหรือรับข้อความ สัญลักษณ์แทนด้วยสี่เหลี่ยม
	Message คือ ข้อความที่รับส่งระหว่าง Object สัญลักษณ์แทนด้วยลูกศรและประกอบด้วย 2 ส่วน คือ ข้อมูล (Data) และฟังก์ชัน (Function)
	Return Message เป็นข้อมูลที่ส่งกลับหลังจากทำงานเสร็จ
	Self call เป็นการเรียกฟังชันก์การทำงานภายในตัวเอง
	สร้างกรอบการทำงานของโปรแกรม เพื่อให้รู้ขอบเขตของการทำงานเช่น ลูป(loop)



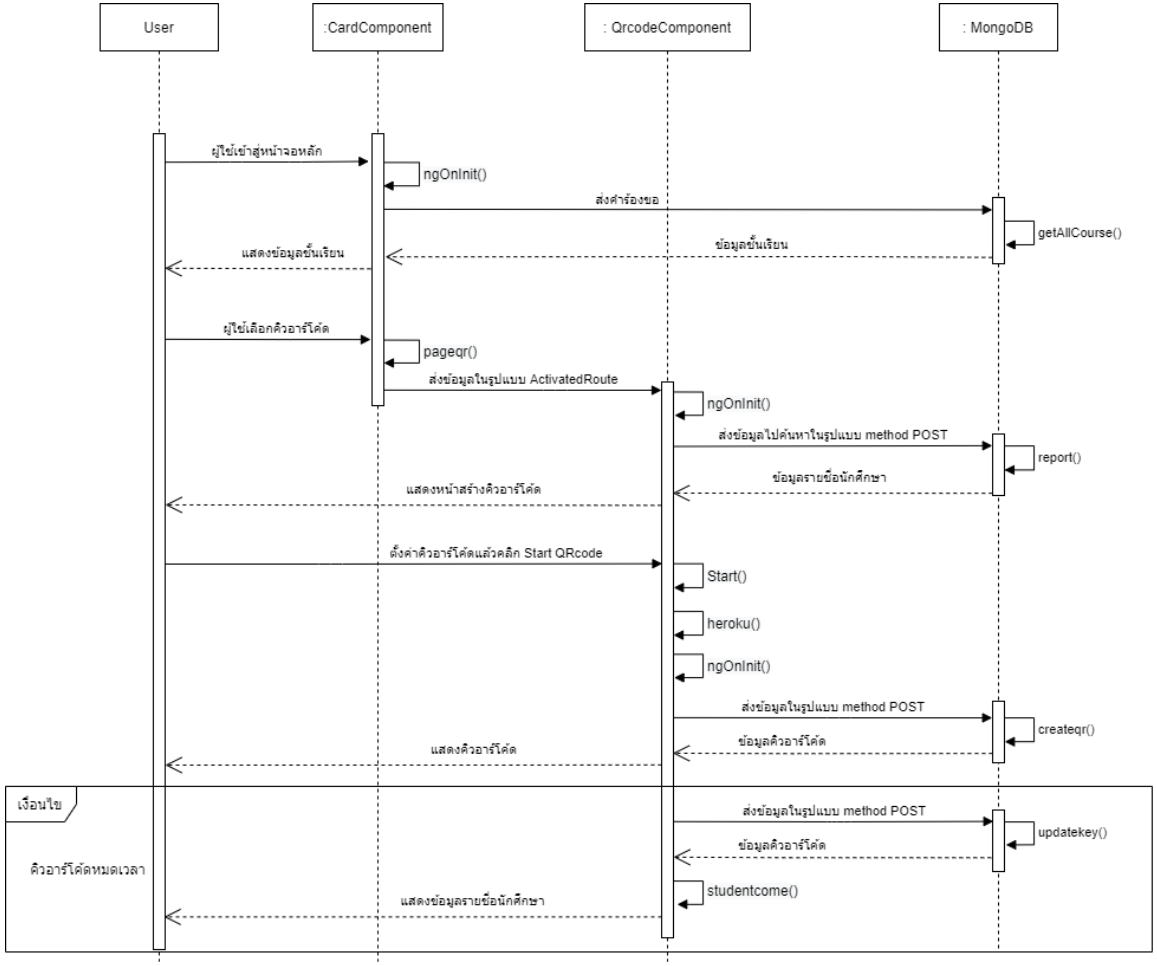
รูปที่ 3.16: Sequence Diagram สร้างชั้นเรียน

จากภาพที่ 3.16 สามารถอธิบายแผนภาพ Sequence Diagram สร้างคิวอาร์โค้ด ได้ดังนี้ เมื่อ ผู้ใช้เปิดโปรแกรมระบบจะเรียกใช้เมธอด ngOnInit() ที่คลาส TopbarComponent ระบบจะคืนค่าข้อมูลชั้นเรียนโดยจะแสดงเป็นการ์ด เมื่อผู้ใช้คลิกปุ่มเพิ่มชั้นเรียนระบบจะทำการไปที่หน้าเพิ่มชั้นเรียนจากนั่นจะเรียกใช้เมธอด ngOnInit() เพิ่มแสดงหน้าเพิ่มชั้นเรียน ผู้ใช้คลิกเลือกไฟล์ ระบบจะเรียกใช้เมธอด onFileChange() ในคลาส UploadfileComponent โดยเมธอดนี้จะทำหน้าที่อ่านไฟล์ที่เลือกเข้ามาและเตรียมข้อมูลสำหรับอัพโหลด และแสดงข้อมูล เมื่อผู้ใช้คลิกปุ่ม Upload ระบบจะเรียกใช้เมธอเ Upload() ที่คลาส UploadfileComponent ระบบจะส่งข้อมูลที่เตรียมไว้ส่งข้อมูลในรูปแบบ POST ไปที่ฐานข้อมูล ฐานข้อมูลจะเรียกใช้เมธอด uploadCourse() ซึ่งเมธอดนี้จะทำหน้าที่บันทึกข้อมูลลงฐานข้อมูล เมื่อบันทึกเสร็จจะคืนค่าข้อมูลชั้นเรียนจากนั่น ระบบจะกลับสู่หน้าแรก



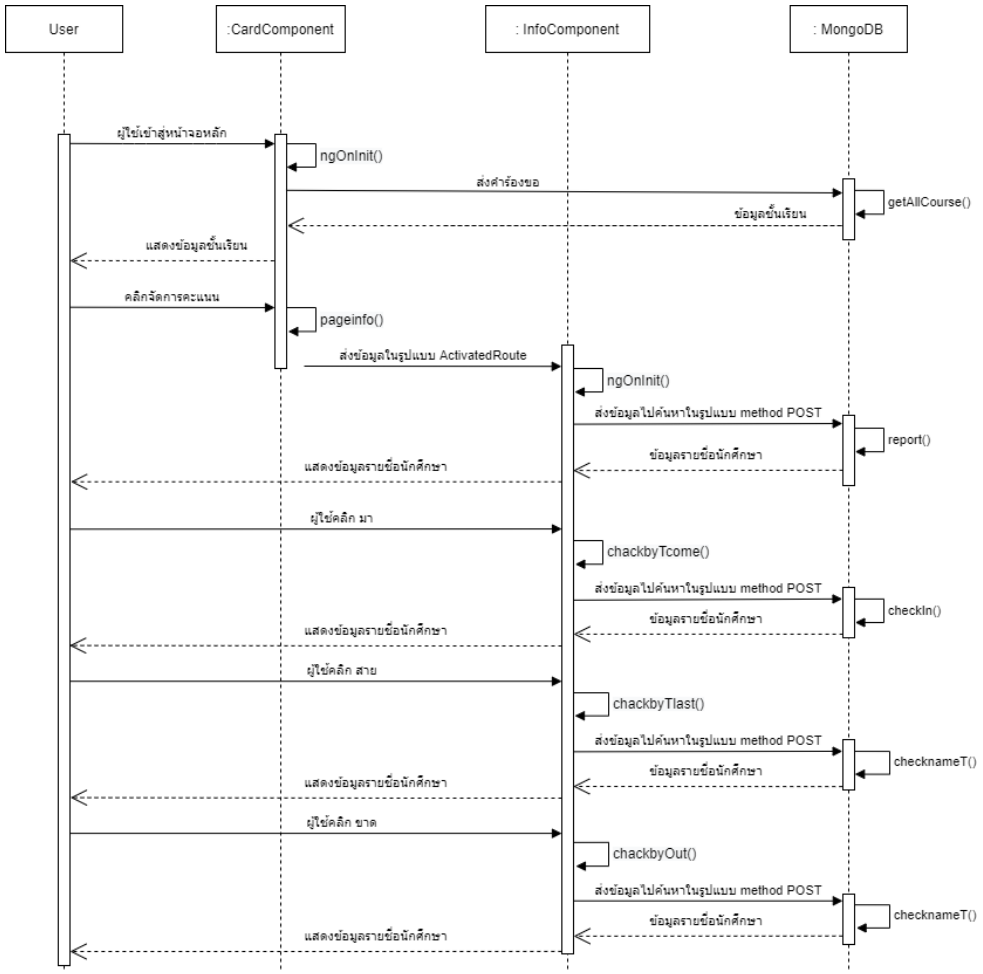
รูปที่ 3.17: Sequence Diagram อัพเดทข้อมูลนักศึกษา

จากภาพที่ 3.17 สามารถอธิบายแผนภาพ Sequence Diagram สร้างคิวอาร์โค้ด ได้ดังนี้ เมื่อ ผู้ใช้เปิดโปรแกรมระบบจะเรียกใช้เมธอด ngOnInit() ที่คลาส CardComponent ระบบจะส่งคำร้องขอไปที่ฐานข้อมูล ซึ่งฐานข้อมูลจะเรียนใช้เมธอด getAllCourse() โดยจะทำหน้าที่ค้นหาและคืนค่าข้อมูลชั้นเรียนที่มีในฐานข้อมูล จากนั่นระบบจะรับค่าที่คืนมาจากฐานข้อมูลโดยจะแสดงเป็นการ์ด เมื่อผู้ใช้เลือคิวอาร์โค้ดระบบจะเรียกใช้งานเมธอด pageup() ระบบจะไปที่หน้าสร้างคิวอาร์โค้ดโดยส่งข้อมูล id ของชั้นเรียนไปในรูปแบบ ActivatedRoute เมื่อมาที่หน้าสร้างคิวอาร์โค้ดแล้วระบบจะเรียกใช้เมธอด ngOnInit() ที่คลาส EditComponent ระบบจะส่งข้อมูล id ไปที่ฐานข้อมูลไปคนหาในรูปแบบ POST ในเมธอด report() โดยเมธอดจะค้นหาชั้นเรียนตาม id ที่ได้มาและคืนค่าข้อมูลชั้นเรียนที่หาพบและระบบจะแสดงหน้าอัพเดทข้อมูลนักศึกษา ผู้ใช้งานคลิกเพิ่มนักศึกษาระบบจะเรียกใช้เมธอด show() ในคลาส EditComponent โดยเมธอดนนี้จะทำการแสดงหน้ากรอกข้อมูลนักศึกษา เมื่อกรอกข้อมูลนักศึกษาแล้วกดปุ่มตกลงระบบจะเรียนใช้เมธอด addup() ในคลาส EditComponent โดยเมธอดี้จะทำหน้าส่งข้อมูลที่กรอกเข้ามาในหน้ากรอกข้อมูลนักศึกษาส่งไปในรูปแบบ POST เพื่อเก็บในฐานข้อมูล ซึ่งจะเรียกใช้เมธอด addupstudent() ในการบันทึกลงในฐานข้อมูลและจะคืนค่าข้อมูลนักศึกษาที่เพิ่มใหม่มาที่คลาส EditComponent จากนั่นจะแสดงข้อมูลรายชื่อนักศึกษา ต่อมาผู้ใช้งานคลิกปุ่มแก้ไขนักศึกษาระบบจะเรียกใช้เมธอด show() ในคลาส EditComponent โดยเมธอดนนี้จะทำการแสดงหน้ากรอกข้อมูลนักศึกษา เมื่อกรอกข้อมูลนักศึกษาแล้วกดปุ่มตกลงระบบจะเรียนใช้เมธอด editstudent() ในคลาส EditComponent โดยเมธอดี้จะทำหน้าส่งข้อมูลที่กรอกเข้ามาในหน้ากรอกข้อมูลนักศึกษาส่งไปในรูปแบบ POST เพื่อเก็บในฐานข้อมูล ซึ่งจะเรียกใช้เมธอด editstudent() ในการบันทึกลงในฐานข้อมูลและจะคืนค่าข้อมูลนักศึกษามาที่คลาส EditComponent จากนั่นจะแสดงข้อมูลรายชื่อนักศึกษา สุดท้ายผู้ใช้งานคลิกลบนักศึกษาระบบจะเรียกใช้เมธอด delete() ในคลาส EditComponent โดยเมธอดี้จะทำหน้าส่งข้อมูล id ชั้นเรียนและรหัสนักศึกษาไปในรูปแบบ POST เพื่อค้นหาและทำการลบข้อมูลในฐานข้อมูล ซึ่งจะเรียกใช้เมธอด deletestudent() ในการลบข้อมูลในฐานข้อมูลและจะคืนค่าข้อมูลนักศึกษามาที่คลาส EditComponent จากนั่นจะแสดงข้อมูลรายชื่อนักศึกษา



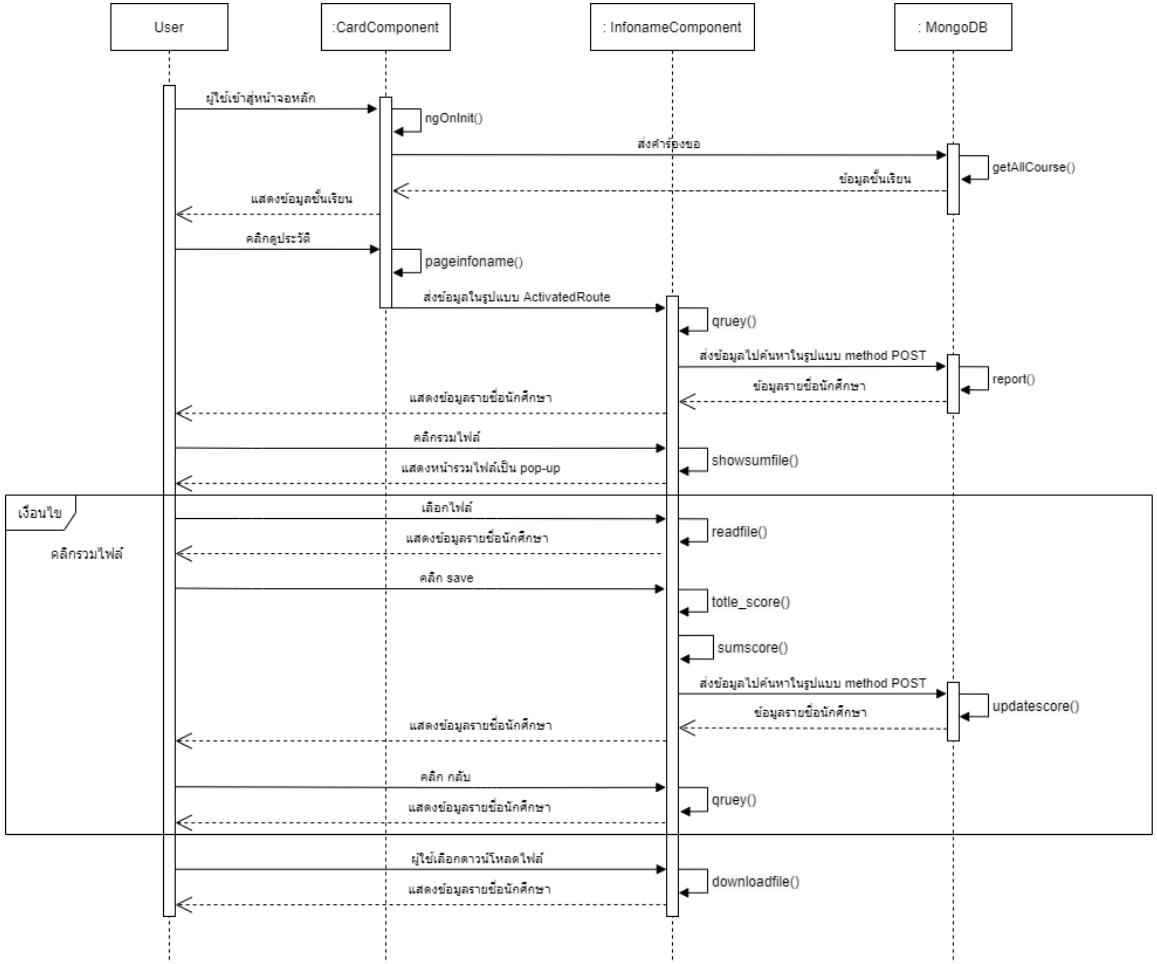
รูปที่ 3.18: Sequence Diagram สร้างคิวอาร์โค้ด

จากภาพที่ 3.18 สามารถอธิบายแผนภาพ Sequence Diagram สร้างคิวอาร์โค้ด ได้ดังนี้ เมื่อ ผู้ใช้เปิดโปรแกรมระบบจะเรียกใช้เมธอด ngOnInit() ที่คลาส CardComponent ระบบจะส่งคำร้องขอไปที่ฐานข้อมูล ซึ่งฐานข้อมูลจะเรียนใช้เมธอด getAllCourse() โดยจะทำหน้าที่ค้นหาและคืนค่าข้อมูลชั้นเรียนที่มีในฐานข้อมูล จากนั่นระบบจะรับค่าที่คืนมาจากฐานข้อมูลโดยจะแสดงเป็นการ์ด เมื่อผู้ใช้เลือคิวอาร์โค้ดระบบจะเรียกใช้งานเมธอด pageqr() ที่คลาส CardComponent ระบบจะไปที่หน้าสร้างคิวอาร์โค้ดโดยส่งข้อมูล id ของชั้นเรียนไปในรูปแบบ ActivatedRoute เมื่อมาที่หน้าสร้างคิวอาร์โค้ดแล้วระบบจะเรียกใช้เมธอด ngOnInit() ที่คลาส UploadfileComponent ระบบจะส่งข้อมูล id ไปที่ฐานข้อมูลไปคนหาในรูปแบบ POST ในเมธอด report() โดยเมธอดจะค้นหาชั้นเรียนตาม id ที่ได้มาและคืนค่าข้อมูลชั้นเรียนที่หาพบและระบบจะแสดงหน้าสร้างคิวอาร์โค้ด จากนั่นเมื่อผู้ใช้งานตั้งค่าคิวอาร์โค้ดแล้วกด Start QRcode ระบบจะเรียกใช้เมธอด Start() ซึ้งเมธอดนี้จะทำหน้าที่ในการสร้าง url ของคิวอาร์โค้ดและเรียกใช้เมธอด heroku() โดยเมธอดนี้จะทำหน้าที่อัพเดท url ที่เก็บไว้ในฐานข้อมูล จากนั่นระบบจะเรียกใช้งานเมธอด ngOnInit() ซึ้งจะหน้าที่ในการนับเวลาของคิวอาร์โค้ด เมื่อเวลาเริ่มนับถ่อยหลังระบบจะส่งข้อมูล url ในรูปแบบ POST ไปเก็บไว้ในฐานข้อมูลและคืนค่า url ที่บันทึกในฐานข้อมูล จากนั่นระบบจะแสดงคิวอาร์โค้ด เมื่อเวลานับถ่อยหลังหมดลงระบบจะส่ง url ไปอัพเดทในฐานข้อมูลและคืนค่า url ที่บันทึกในฐานข้อมูล จากนั่นจะแสดงรายชื่อนักศึกษาที่เข้าเรียน



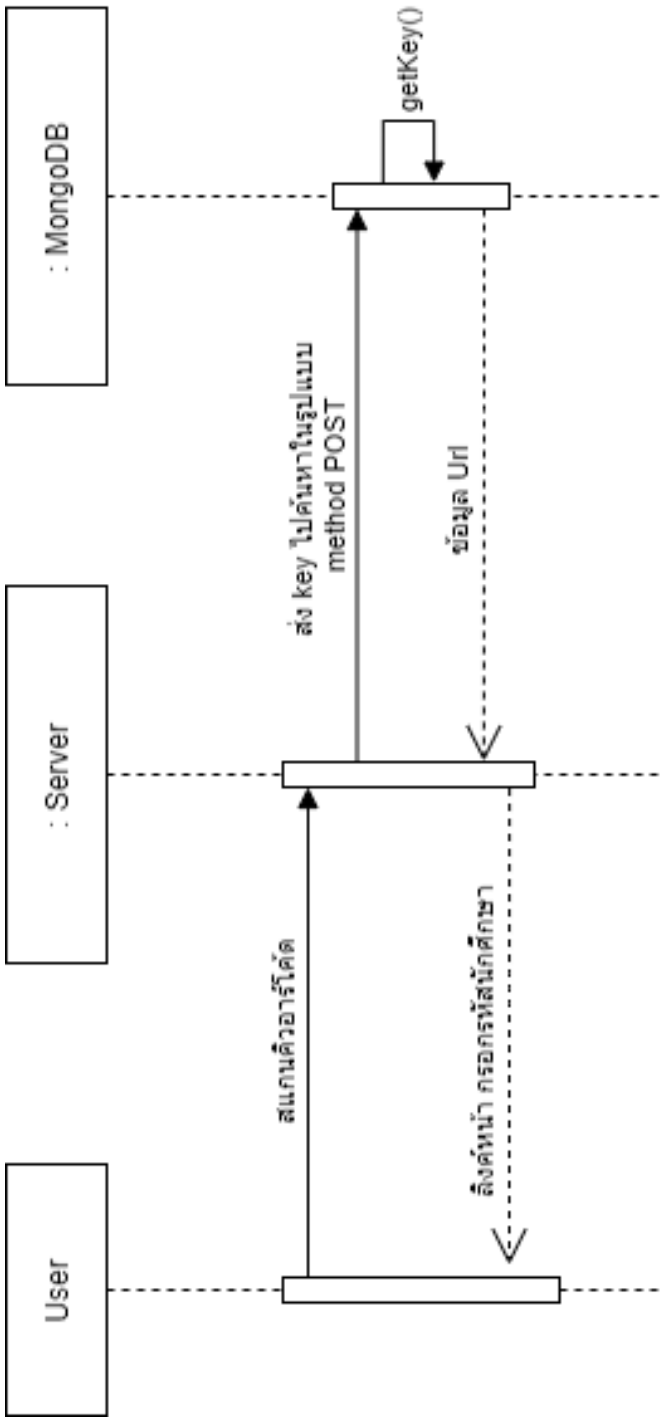
รูปที่ 3.19: Sequence Diagram จัดการคะแนน

จากภาพที่ 3.19 สามารถอธิบายแผนภาพ Sequence Diagram จัดการคะแนน ได้ดังนี้ เมื่อ ผู้ใช้เปิดโปรแกรมระบบจะเรียกใช้เมธอด ngOnInit() ที่คลาส CardComponent ระบบจะส่งคำร้องขอไปที่ฐานข้อมูล ซึ่งฐานข้อมูลจะเรียนใช้เมธอด getAllCourse() โดยจะทำหน้าที่ค้นหาและคืนค่าข้อมูลชั้นเรียนที่มีในฐานข้อมูล จากนั่นระบบจะรับค่าที่คืนมาจากฐานข้อมูลโดยจะแสดงเป็นการ์ด เมื่อผู้ใช้คลิกจัดการคะแนนระบบจะเรียกใช้งานเมธอด pageinfo() ที่คลาส CardComponent ระบบจะไปที่หน้าสร้างคิวอาร์โค้ดโดยส่งข้อมูล id ของชั้นเรียนไปในรูปแบบ ActivatedRoute เมื่อมาที่หน้าจัดการคะแนนแล้วระบบจะเรียกใช้เมธอด ngOnInit() ที่คลาส InfoComponent ระบบจะส่งข้อมูล id ไปที่ฐานข้อมูลไปคนหาในรูปแบบ POST ในเมธอด report() โดยเมธอดจะค้นหาชั้นเรียนตาม id ที่ได้มาและคืนค่าข้อมูลชั้นเรียนที่หาพบและระบบจะแสดงหน้าจัดการคะแนน โดยหน้าจัดการคะแนนจะโชว์รายชื่อและข้อมูลคะแนนของนักศึกษา เมื่อผู้ใช้งานคลิกปุ่มมาระบบจะเรียกใช้เมธอด checkbyTcome() ในคลาส InfoComponent ซึ่งเมธอดนี้จะทำการส่งข้อมูล id ชั้นเรียนและรหัสนักศึกษาไปที่ฐานข้อมูลในรูปแบบ POST โดยฐานข้อมูลจะเรียกใช้เมธอด checkIn() เพื่อให้คะแนนและจะคือค่าที่อัพเดทคะแนนกลับมาที่คลาส InfoComponent จากนั่นก็จะแสดงรายชื่อพร้อมคะแนนที่อัพเดท ต่อมาเมื่อผู้ใช้งานคลิกปุ่มสายระบบจะเรียกใช้เมธอด checkbyTlast() ในคลาส InfoComponent ซึ่งเมธอดนี้จะทำการส่งข้อมูล id ชั้นเรียนและรหัสนักศึกษาไปที่ฐานข้อมูลในรูปแบบ POST โดยฐานข้อมูลจะเรียกใช้เมธอด checknameT() เพื่อให้คะแนนและจะคือค่าที่อัพเดทคะแนนกลับมาที่คลาส InfoComponent จากนั่นก็จะแสดงรายชื่อพร้อมคะแนนที่อัพเดท สุดท้ายเมื่อผู้ใช้งานคลิกปุ่มมาระบบจะเรียกใช้เมธอด checkbyOut() ในคลาส InfoComponent ซึ่งเมธอดนี้จะทำการส่งข้อมูล id ชั้นเรียนและรหัสนักศึกษาไปที่ฐานข้อมูลในรูปแบบ POST โดยฐานข้อมูลจะเรียกใช้เมธอด checknameO() เพื่อให้คะแนนและจะคือค่าที่อัพเดทคะแนนกลับมาที่คลาส InfoComponent จากนั่นก็จะแสดงรายชื่อพร้อมคะแนนที่อัพเดท



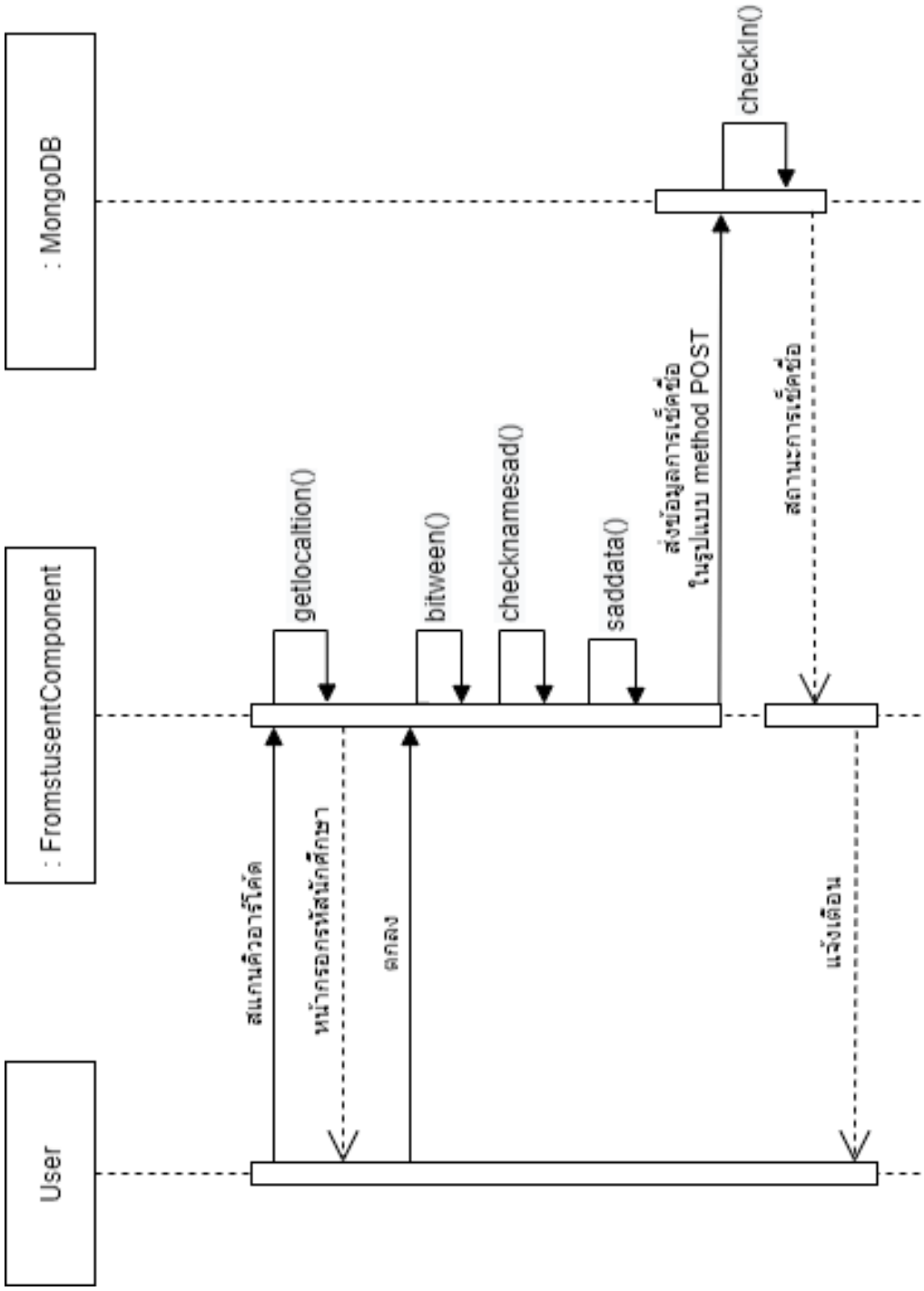
รูปที่ 3.20: Sequence Diagram ดูประวัติ

จากภาพที่ 3.20 สามารถอธิบายแผนภาพ Sequence Diagram ดูประวัติ ได้ดังนี้ เมื่อ ผู้ใช้เปิดโปรแกรมระบบจะเรียกใช้เมธอด ngOnInit() ที่คลาส CardComponent ระบบจะส่งคำร้องขอไปที่ฐานข้อมูล ซึ่งฐานข้อมูลจะเรียนใช้เมธอด getAllCourse() โดยจะทำหน้าที่ค้นหาและคืนค่าข้อมูลชั้นเรียนที่มีในฐานข้อมูล จากนั่นระบบจะรับค่าที่คืนมาจากฐานข้อมูลโดยจะแสดงเป็นการ์ด เมื่อผู้ใช้คลิกดูประวัติระบบจะเรียกใช้งานเมธอด pageinfoname() ที่คลาส CardComponent ระบบจะไปที่หน้าสร้างคิวอาร์โค้ดโดยส่งข้อมูล id ของชั้นเรียนไปในรูปแบบ ActivatedRoute เมื่อมาที่หน้าจัดการคะแนนแล้วระบบจะเรียกใช้เมธอด ngOnInit() ที่คลาส InfonameComponent ระบบจะส่งข้อมูล id ไปที่ฐานข้อมูลไปคนหาในรูปแบบ POST ในเมธอด report() โดยเมธอดจะค้นหาชั้นเรียนตาม id ที่ได้มาและคืนค่าข้อมูลชั้นเรียนที่หาพบและระบบจะแสดงหน้าประวัติ โดยหน้าประวัติจะโชว์ข้อมูลรายชื่อและคะแนนแต่ละสัปดาห์ของนักศึกษา เมื่อผู้ใช้คลิกปุ่มรวมไฟล์ระบบจะเรียกใช้งานเมธอด showsumfile() ในคลาส InfonameComponent ซึ่งเมธอดนี้ทำหน้าที่แสดงหน้ารวมไฟล์ เมื่อผู้ใช้เลือกไฟล์ระบบขะเรียกใช้งานเมธอด readfile() ในคลาส InfonameComponent ซึ่งเมธอดนี้จะทำหน้าที่อ่านไฟล์และตรวจสอบ ถ้าผู้ใช้กด save ระบบจะเรียกใช้งานเมธอด totlescore() ในคลาส InfonameComponent ซึ่งเมธอดนี้จะทำการรวมคะแนนจากไฟล์ที่นำเข้าและคะแนนเดิมจากนั่นจะเรียกใช้เมธอด sumscore() โดยเมธอดนี้จะทำการส่งข้อมูลไปที่ฐานข้อมูลที่อัพดกทไปเก็บในฐานข้อมูล ซึ่งฐานข้อมูลจะเรียกใช้เมธอด updatescore() เพื่อบันทึกข้อมูล จากนั่นจะส่งค่าข้อมูลที่อัพเดทกลับมาที่คลาส InfonameComponent แล้วจึงแสดงข้อมูลที่อัพเดท แต่ถ้าผู้ใช้งานกดกลับระบบจะกลับสู่หน้าประวัติ เมื่อผู้ใช้งานกดปุ่มดาวน์โหลดระบบจะเรียกใช้งานเมธอด downloadfile() ในคลาส InfonameComponent ซึ่งเมธอดนี้ทำหน้าที่ในการดาวน์โหลดข้อมูลรายชื่อและคะแนนของนักศึกษาเป็นไฟล์ excel



รูปที่ 3.21: Sequence Diagram สแกน QRcode

จากภาพที่ 3.21 สามารถอธิบายแผนภาพ Sequence Diagram สแกน QRcode ได้ดังนี้ เมื่อนักศึกษาสแกนคิวอาร์โค้ด คิวอาร์โค้ดจะลิงค์ไปยัง Server โดยจะส่งค่า key ที่เก็บไว้ในคิวอาร์โค้ดไปเพื่อนำไปค้นหารายชั้นเรียน จากนั่น Server จะเรียกใช้เมธอด getKey() ซึ่งเมธอดนี้จะทำหน้าที่คืนค่า url สำหรับไปที่หน้ากรอกรหัสนักศึกษา



รูปที่ 3.22: Sequence Diagram กรอกรหัสนักศึกษา

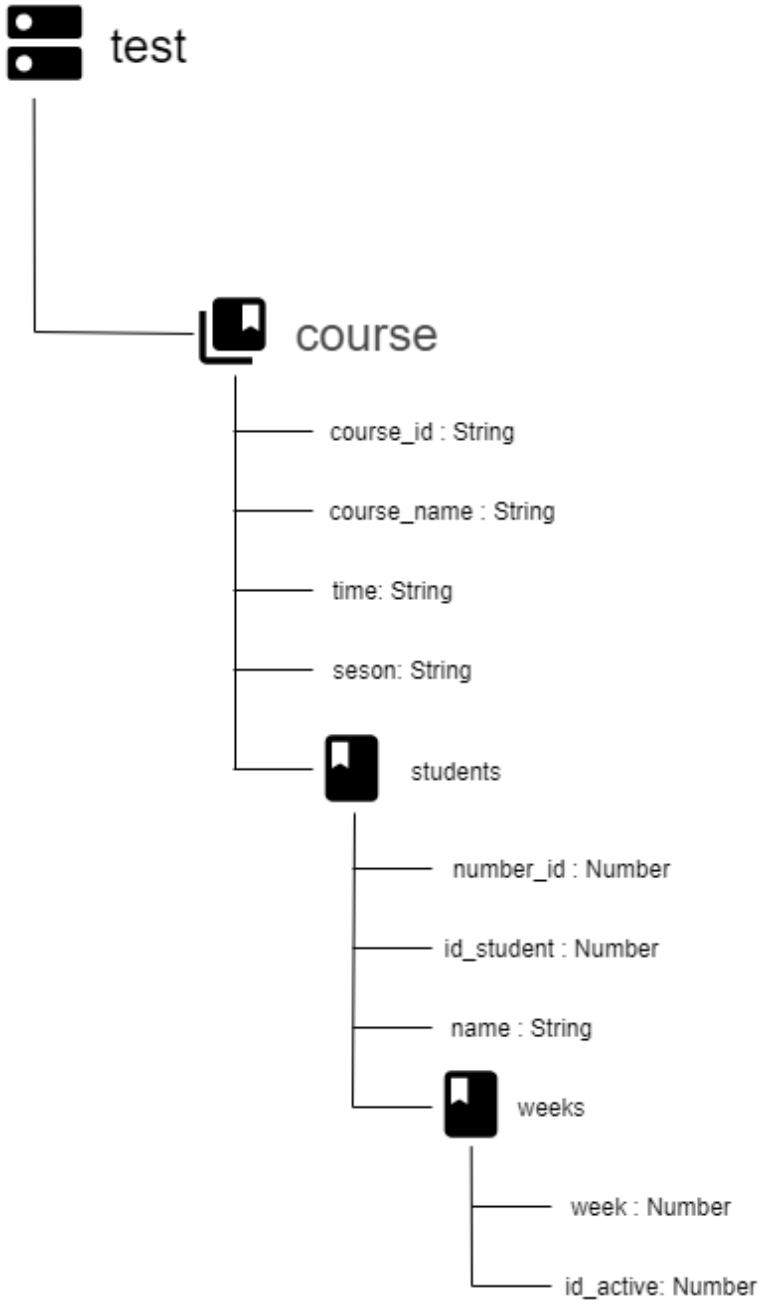
จากภาพที่ 3.22 สามารถอธิบายแผนภาพ Sequence Diagram แสดงส่งเอกสารตรวจสอบได้ดังนี้ เมื่อนักศึกษาสแกนคิวอาร์โค้ดผ่านจะเข้ามาที่หน้ากรอกรหัสนักศึกษาโดยระบบจะให้กรอกรัหัสนักศึกษาเพื่อเป็นการยื่นยันตัวตนจากนั่นระบบจะเรียกใช้งานเมธอด getlocaltion() ในคลาส FromstusentComponent เพื่อระบบบุตำแหน่งของนักศึกษา เมื่อกดตกลงระบบจะเรียกใช้เมธอด bitween() เพื่อคำนวนหาระยะหางระหว่างอาจารย์กับนักศึกษา จากนั่นจะเรียกใช้เมธอด checknamesad() เพื่อตรวจสอบว่านักศึกษาเช็คชื่อไปแล้วหรือยัง แล้วจึงเรียกใช้เมธอด saddata() เพื่อส่งข้อมูลไปที่ฐานข้อมูลในรูปแบบ POST โดยฐานข้อมูลจะเรียกใช้เมธอด checkIn() เพื่อจัดการคะแนนของนักศึกษา จากนั่นจะคืนค่าสถานะกลับมาที่คลาส FromstusentComponent ซึ่งคลาสจะทำการแจ้งเตือนการเช็คชื่อของนักศึกษา

3.7 โครงสร้างฐานข้อมูลไฟร์เบส(MongoDB Database Stucture)

MongoDB Database นั้นเป็น Database แบบ NoSQL และเป็น JSON database ที่มีโครงสร้างที่เป็น Key และ Value จัดเก็บข้อมูลในลักษณะโหนด หากต้องการเรียกงานจะเรียกใช้โดย การท่องไปยังโหนดที่ต้องการ ส่วนประกอบสัญลักษณ์ที่ใช้ในการเขียนโครงสร้างฐานข้อมูลแบบ MongoDB แสดงดังตารางที่ 3.21

ตารางที่ 3.21: สัญลักษณ์ของโครงสร้างฐานข้อมูลแบบ MongoDB

สัญลักษณ์	คำอธิบาย
	Database เป็นการเรียกชื่อแทนโหนด(Node)บนสุดที่ใช้ในการเก็บข้อมูล
	Collection เป็นการเรียกชื่อแทนของการเก็บหลาย ๆ เอกสารไว้ด้วยกัน
	Document เป็นการ เรียก ชื่อ แทน หน่วย การ เก็บ ของ ข้อมูล ใน Cloud Firestore ภายในจะประกอบไปด้วย ชื่อของ Document ชื่อของคีย์ (key) และ ค่าข้อมูล (value) โดยชื่อของ Document ห้ามซ้ำกัน ซึ่งใน Cloud Firestore สามารถระบุประเภทของข้อมูลได้ 9 ประเภทได้แก่ boolean, number, string, geo point, timestamp, array, object, reference และ null

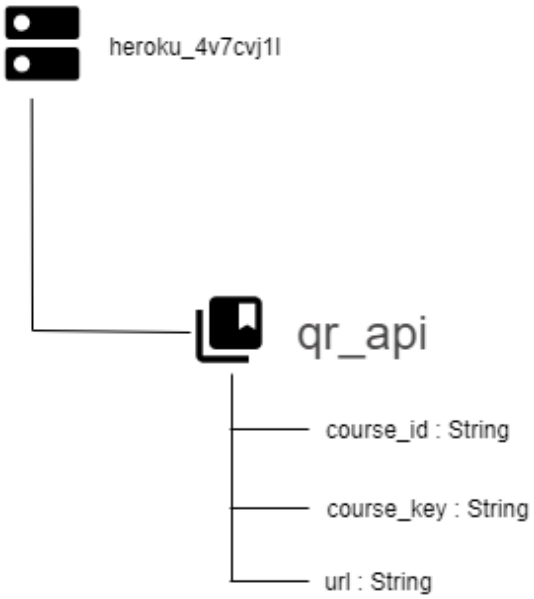


รูปที่ 3.23: โครงสร้างฐานข้อมูลแบบ MongoDB

จากรูที่ 3.23 สามารถอธิบายโครงสร้างของข้อมูลได้ดังนี้

ตารางที่ 3.22: อธิบายส่วนที่ใช้เก็บข้อมูล

Key	คำอธิบาย
course	โหนดสำหรับเก็บข้อมูลชั้นเรียนทั้งหมด
course$_id$	สำหรับเก็บข้อมูลไอดีของชั้นเรียน
course$_name$	สำหรับเก็บข้อมูลชื่อของชั้นเรียน
time	สำหรับเก็บข้อมูลเวลาของชั้นเรียน
seson	สำหรับเก็บข้อมูลเทอมของชั้นเรียน
students	โหนดสำหรับเก็บข้อมูลนักศึกษาทั้งหมด
number$_id$	สำหรับเก็บเลขที่ของนักศึกษา
id$_student$	สำหรับเก็บรหัสนักศึกษา
name	สำหรับเก็บชื่อของศึกษา
weeks	โหนดสำหรับเก็บข้อมูลคะแนนแต่ละอาทิต
week	สำหรับเก็บคะแนนศึกษา
id$_active$	สำหรับเก็บสถานะ



รูปที่ 3.24: โครงสร้างฐานข้อมูลแบบ MongoDB

จากรูที่ 3.24 สามารถอธิบายโครงสร้างของข้อมูลได้ดังนี้

ตารางที่ 3.23: อธิบายส่วนที่ใช้เก็บข้อมูลใน Heroku Server

Key	คำอธิบาย
qr$_{a}pi$	โหนดสำหรับเก็บข้อมูลของคิวอาร์โค้ดทั้งหมด
course$_{i}d$	สำหรับเก็บไอดีของชั้นเรียน
course$_{k}ey$	สำหรับเก็บคีย์ของคิวอาร์โค้ด
url	สำหรับเก็บ url ของคิวอาร์โค้ด

บทที่ 4

การพัฒนาระบบ

หลังจากที่ได้มีการเตรียมความพร้อมสำหรับการพัฒนาในด้านต่าง ไม่ว่าจะเป็นที่มาและความสำคัญของปัญหา เทคโนโลยีที่มีความเหมาะสมกับระบบ และการออกแบบระบบการทำงาน รวมไปถึงโครงสร้างของข้อมูล ในบทนี้จะเป็นการพูดถึงการสร้างระบบที่ได้มีการออกแบบไว้ในบทที่แล้วจะถูกนำเสนอในบทนี้ โดยการพัฒนาระบบแบ่งได้เป็นส่วนต่าง ๆ ดังนี้

4.1 การพัฒนาเว็บเซิร์ฟเวอร์

4.2 การพัฒนาเว็บแอปพลิเคชัน

4.1 การพัฒนาเว็บเซิร์ฟเวอร์

การพัฒนาระบบเช็คชื่อด้วยคิวอาร์โค้ดสำหรับเว็บเซิร์ฟเวอร์วัตถุประสงค์หลักของเว็บเซิร์ฟ-เวอร์คือให้บริการฐานข้อมูลและการเชื่อมต่อข้อมูลต่างๆของระบบเช็คชื่อด้วยคิวอาร์โค้ด โดยตัวเว็บเว็บเซิร์ฟเวอร์นี้ถูกพัฒนาขึ้นด้วย echo golang มีรายละเอียดการทำงานดังนี้

4.1.1 โครงสร้างและโค้ดของการให้บริการ API ของระบบระบบเช็คชื่อด้วยคิวอาร์โค้ด

การพัฒนาเว็บเซิร์ฟเวอร์ระบบเช็คชื่อด้วยคิวอาร์โค้ด มีวัตถุประสงค์หลักเพื่อเก็บข้อมูลและให้บริการต่างๆ เกี่ยวกับการเชื่อต่อกับข้อมูลที่เก็บไว้

```
1  import (
2  "github.com/labstack/echo"
3  "gopkg.in/mgo.v2"
4  )
5
6  func main(){
7        db, err := mgo.Dial("localhost")
8        if err != nil {
9        e.Logger.Fatal(err)
10       }
11       h := &Handler{DB: db}
12 }
```

รูปที่ 4.1: ไฟล์ server.go

จากภาพที่ 4.1 สามารถอธิบายการทำงานโค๊ดส่วนการเชื่อมต่อกับ mongodb โดยไฟล์ server.go สามารถอธิบายได้ดังนี้

- บรรทัดที่ 1-3 เป็นการนำเข้าไลบรารีของ mongodb และ echo framework
- บรรทัดที่ 7 เป็นการเชื่อต่อกับฐานข้อมูล mongodb โดยที่อยู่ของฐานข้อมูลคือ localhost
- บรรทัดที่ 8-9 เป็นการตรวจสอบการเชื่อต่อกับฐานข้อมูลว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 11 เป็นการประกาศตัวแปล h เพื่อเก็บข้อมูลจากไฟล์ Handler ซึ่ง Handler จะเก็บข้อมูลจากฐานข้อมูล

```
1 import (
2 "gopkg.in/mgo.v2"
3 )
4
5 type (
6 Handler struct {
7 DB *mgo.Session
8 }
9 )
```

รูปที่ 4.2: ไฟล์ handler.go

จากภาพที่ 4.2 สามารถอธิบายการทำงานโค๊ดส่วนที่สร้างตัวแปลเพื่อรับค่าจากฐานข้อมูล โดยไฟล์ handler.go ได้ดังนี้

- บรรทัดที่ 1-3 เป็นการนำเข้าไลบรารีของ mongodb
- บรรทัดที่ 5-8 เป็นการสร้างตัวแปลเพื่อรับค่าจากฐานข้อมูล mongodb

```
1       import (
2       "gopkg.in/mgo.v2/bson"
3       )
4
5       type (
6       Course struct {
7       ID bson.ObjectId `json:"id" bson:"_id,
           omitempty"`
8       CourseID string `json:"course_id" bson:"
           course_id"`
9       CourseName string `json:"course_name" bson:"
           course_name"`
10      Day_Time string `json:"time" bson:"time"`
11      Seson string `json:"seson" bson:"seson"`
12      Sutdents []Sutdent `json:"students" bson:"
           students"`
13      }
14      Sutdent struct {
15      NumberID int `json:"number_id" bson:"number_id
           "`
16      Idstudent string `json:"id_student" bson:"
           id_student"`
17      Name string `json:"name" bson:"name"`
18      Weeks Week `json:"weeks" bson:"weeks"`
19      }
20      Week struct {
21      Week1 Active `json:"week1" bson:"week1"`
22      Week2 Active `json:"week2" bson:"week2"`
23      Week3 Active `json:"week3" bson:"week3"`
24      Week4 Active `json:"week4" bson:"week4"`
25      Week5 Active `json:"week5" bson:"week5"`
26      Week6 Active `json:"week6" bson:"week6"`
27      Week7 Active `json:"week7" bson:"week7"`
28      Week8 Active `json:"week8" bson:"week8"`
29      Week9 Active `json:"week9" bson:"week9"`
30      Week10 Active `json:"week10" bson:"week10"`
31      Week11 Active `json:"week11" bson:"week11"`
32      Week12 Active `json:"week12" bson:"week12"`
33      Week13 Active `json:"week13" bson:"week13"`
34      Week14 Active `json:"week14" bson:"week14"`
35      Week15 Active `json:"week15" bson:"week15"`
36      Week16 Active `json:"week16" bson:"week16"`
37      }
38      Active struct {
39      Weeks float32 `json:"week" bson:"week"`
40      Idactive string `json:"id_active" bson:"
           id_active"`
41      })
```

รูปที่ 4.3: ไฟล์ model-Course.go

จากภาพที่ 4.3 สามารถอธิบายการทำงานโค๊ดส่วนที่สร้างโมเดลของฐานข้อมูล โดยไฟล์ model-Course.go ได้ดังนี้

- บรรทัดที่ 1-3 เป็นการนำเข้าไลบรารีของ mongodb
- บรรทัดที่ 5-41 เป็นการสร้างโมเดลของฐานข้อมูล mongodb

```
1  import (
2  "bytes"
3  "io/ioutil"
4  "net"
5  "net/http"
6  "os"
7
8  "github.com/labstack/echo"
9  "gopkg.in/mgo.v2"
10 "gopkg.in/mgo.v2/bson"
11
12 )
```

รูปที่ 4.4: ไฟล์ student.go

จากภาพที่ 4.4 สามารถอธิบายการทำงานโค๊ดส่วนที่นำเข้าไลบรารีต่างๆในไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 2 เป็นการนำเข้าไลบรารี bytes สำหรับเป็น type
- บรรทัดที่ 3 เป็นการนำเข้าไลบรารี io/ioutil สำหรับอ่านตัวแปลประเภท bytes
- บรรทัดที่ 4 เป็นการนำเข้าไลบรารี net สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ net
- บรรทัดที่ 5 เป็นการนำเข้าไลบรารี net/http สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ http
- บรรทัดที่ 6 เป็นการนำเข้าไลบรารี os สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ os
- บรรทัดที่ 8 เป็นการนำเข้าไลบรารี github.com/labstack/echo สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ echo
- บรรทัดที่ 9 เป็นการนำเข้าไลบรารี gopkg.in/mgo.v2 สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ mongodb
- บรรทัดที่ 10 เป็นการนำเข้าไลบรารี gopkg.in/mgo.v2/bson สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ bson ใน mongodb

```
1  func (h *Handler) getAllCourse(c echo.Context) (err
     error) {
2  users := []*Course{}
3  db := h.DB.Clone()
4
5  if err = db.DB("test").C("course").Find(nil).All(&
     users); err != nil {
6  return c.JSON(http.StatusBadRequest, err)
7  }
8  defer db.Close()
9  return c.JSON(http.StatusOK, users)
10 }
```

รูปที่ 4.5: ไฟล์ student.go

จากภาพที่ 4.5 สามารถอธิบายการทำงานโค๊ดส่วนฟังชั่นก์ดึงข้อมูลของชั้นเรียนทั้งหมดในไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ getAllCourse
- บรรทัดที่ 2 เป็นการประกาศตัวแปลชื่อ users เพื่อรับค่าจากโมเดล Course
- บรรทัดที่ 3 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปลชื่อ db
- บรรทัดที่ 5 เชื่อมต่อกับฐานข้อมูลและค้นหาข้อมูลของชั้นเรียนพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 6 แสดงข้อผิดพลาดถ้ามีในการเชื่อมต่อกับฐานข้อมูล
- บรรทัดที่ 8 ปิดการเชื่อมต่อกับฐานข้อมูล
- บรรทัดที่ 9 คือค่าที่ได้จากการค้นหา

```
1  func (h *Handler) uploadCourse(c echo.Context) (err
     error) {
2  user := new(Course)
3
4  err = c.Bind(user)
5  if err != nil {
6  return c.JSON(http.StatusBadRequest, err)
7  }
8
9  db := h.DB.Clone()
10 defer db.Close()
11
12 if err = db.DB("test").C("course").Insert(user); err
       != nil {
13 return c.JSON(http.StatusBadRequest, err)
14 }
15 if err != nil {
16 if mgo.IsDup(err) {
17 log.Panicln(err)
18 }
19
20 }
21 return c.JSON(http.StatusOK, &user)
22
23 }
```

รูปที่ 4.6: ไฟล์ student.go

จากภาพที่ 4.6 สามารถอธิบายการทำงานโค๊ดส่วนฟังชั่นก์สร้างชั้นเรียน โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ uploadCourse
- บรรทัดที่ 2 เป็นการสร้างตัวแปลประเภทชื่อ user เพื่อจำลองโมเดล Course
- บรรทัดที่ 4-6 เป็นการเป็นการตรวจสอบว่าจำลอง Course ได้หรือไม่ ถ้าเกิดข้อผิดพลาดก็จะแสดงข้อผิดพลาด
- บรรทัดที่ 9 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปลชื่อ db
- บรรทัดที่ 10 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 12-13 เชื่อมต่อกับฐานข้อมูลและเพิ่มข้อมูลของชั้นเรียนพร้อมทั้งตรวจสอบการ

เชื่อมต่อว่ามีข้อผิดพลาดหรือไม่

- บรรทัดที่ 15-17 เป็นการตรวจสอบการเชื่อมต่อกับฐานข้อมูล ถ้าการเชื่อมต่อผิดพลาดจะแสดงข้อผิดพลาด
- บรรทัดที่ 21 คือค่าที่ได้จากการเพิ่มชั้นเรียน

```
1  func (h *Handler) deleteCourse(c echo.Context) (err
      error) {
2  db := h.DB.Clone()
3  defer db.Close()
4
5  id := c.Param("id")
6
7  err = db.DB("test").C("course").Remove(bson.M{"_id":
       bson.ObjectIdHex(id)})
8  if err != nil {
9  return c.JSON(http.StatusBadRequest, err)
10 }
11
12 return c.NoContent(http.StatusNoContent)
```

รูปที่ 4.7: ไฟล์ student.go

จากภาพที่ 4.7 สามารถอธิบายการทำงานโค๊ดส่วนฟังชั่นก์ลบชั้นเรียน โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ deleteCourse
- บรรทัดที่ 2 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปลชื่อ db
- บรรทัดที่ 3 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 5 เป็นกราประกาศตัวแปลเพื่อรับข้อมูล id ของชั้นเรียนที่จะลบ
- บรรทัดที่ 7-10 เชื่อมต่อกับฐานข้อมูลและลบข้อมูลของชั้นเรียนที่ค้นหาเจอพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 12 คือค่าสถานะของการลบ

```
1  func (h *Handler) report(c echo.Context) error {
2  users := []*Course{}
3  db := h.DB.Clone()
4  defer db.Close()
5  id := c.Param("id")
6
7  err := db.DB("test").C("course").Find(bson.M{"_id":
      bson.ObjectIdHex(id)}).All(&users)
8  if err != nil {
9  return c.JSON(http.StatusBadRequest, err.Error())
10 }
11
12 return c.JSON(http.StatusOK, users)
13 }
```

รูปที่ 4.8: ไฟล์ student.go

จากภาพที่ 4.8 สามารถอธิบายการทำงานโค๊ดส่วนฟังชั่นก์ดึงข้อมูลของชั้นเรียนในไฟล์ โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ report
- บรรทัดที่ 2 เป็นการสร้างตัวแปลประเภทชื่อ user เพื่อจำลองโมเดล Course
- บรรทัดที่ 3 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปลชื่อ db
- บรรทัดที่ 4 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 5 เป็นกราประกาศตัวแปลเพื่อรับข้อมูล id ของชั้นเรียนที่จะค้นหา
- บรรทัดที่ 7 เชื่อมต่อกับฐานข้อมูลและค้นหาข้อมูลของชั้นเรียนพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 8-9 เป็นการตรวจสอบการเชื่อมต่อกับฐานข้อมูล ถ้าการเชื่อมต่อผิดพลาดจะแสดงข้อผิดพลาด
- บรรทัดที่ 12 คือค่าที่ได้จากการค้นหาชั้นเรียน

```
1  func (h *Handler) addupstudent(c echo.Context) error
       {
2  u := new(Sutdent)
3  id := c.Param("id")
4
5  if err := c.Bind(u); err != nil {
6  return c.JSON(http.StatusBadRequest, err)
7  }
8  db := h.DB.Clone()
9  defer db.Close()
10
11 if err := db.DB("test").C("course").
12 UpdateId(bson.ObjectIdHex(id), bson.M{"$addToSet":
     bson.M{"students": u}}); err != nil {
13 if err == mgo.ErrNotFound {
14 return echo.ErrNotFound
15 }
16 return c.JSON(http.StatusBadRequest, err)
17 }
18 return c.JSON(http.StatusOK, u)
19 }
```

รูปที่ 4.9: ไฟล์ student.go

จากภาพที่ 4.9 สามารถอธิบายการทำงานโค๊ดส่วนฟังชั่นก์เพิ่มนักศึกษาในชั้นเรียน โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ addupstudent
- บรรทัดที่ 2 เป็นการสร้างตัวแปลประเภทชื่อ u เพื่อจำลองโมเดล Sutdent
- บรรทัดที่ 3 เป็นกราประกาศตัวแปลเพื่อรับข้อมูล id ของนักศึกษาที่จะค้นหา
- บรรทัดที่ 5-6 เป็นการตรวจสอบการจำลองโมเดลว่าจำลองได้หรือไม่ ถ้าไม่ได้จะแสดงข้อผิดพลาด
- บรรทัดที่ 8 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปลชื่อ db
- บรรทัดที่ 9 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 11-17 เชื่อมต่อกับฐานข้อมูลและเพิ่มข้อมูลของนักศึกษาพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 18 คือค่าข้อมูลนักศึกษาที่เพิ่ม

```
1  func (h *Handler) editstudent(c echo.Context) error
      {
2  u := new(Sutdent)
3  id := c.Param("id")
4
5  if err := c.Bind(u); err != nil {
6  return c.JSON(http.StatusBadRequest, err)
7  }
8  db := h.DB.Clone()
9  defer db.Close()
10
11 query := bson.M{"_id": bson.ObjectIdHex(id), "
      students.id_student": u.Idstudent}
12 update := bson.M{
13 "$set": bson.M{
14 "students.$": u,
15 },
16 }
17
18 err := db.DB("test").C("course").Update(query,
      update)
19 if err != nil {
20 return c.JSON(http.StatusBadRequest, err.Error())
21 }
22 return c.JSON(http.StatusOK, u.Idstudent)
23 }
```

รูปที่ 4.10: ไฟล์ student.go

จากภาพที่ 4.10 สามารถอธิบายการทำงานโค๊ดส่วนฟังชั่นก์แก้ไขข้อมูลนักศึกษาในชั้นเรียน โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ editstudent
- บรรทัดที่ 2 เป็นการสร้างตัวแปลประเภทชื่อ u เพื่อจำลองโมเดล Sutdent
- บรรทัดที่ 3 เป็นกราประกาศตัวแปลเพื่อรับข้อมูล id ของนักศึกษาที่จะค้นหา
- บรรทัดที่ 5-6 เป็นการตรวจสอบการจำลองโมเดลว่าจำลองได้หรือไม่ ถ้าไม่ได้จะแสดงข้อผิดพลาด
- บรรทัดที่ 8 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปลชื่อ db
- บรรทัดที่ 9 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb

- บรรทัดที่ 11 เป็นการสร้างตัวแปล query เพื่อใช้ในฟังก์ชั่น Updateโดยตัวแปลนี้จะเก็บข้อมูลที่จะค้นหานักเรียนที่จะทำการแก้ไข้
- บรรทัดที่ 12-14 เป็นการสร้างตัวแปล update เพื่อใช้ในฟังก์ชั่น Update โดยตัวแปลนี้จะเก็บข้อมูลที่จะอัพเดทข้อมูลของนักศึกษา
- บรรทัดที่ 17-19 เชื่อมต่อกับฐานข้อมูลและอัพเดทข้อมูลของนักศึกษาพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 17 คือค่าข้อมูลนักศึกษาที่อัพเดท

```
 1 func (h *Handler) deletestudent(c echo.Context)
      error {
 2 u := new(Sutdent)
 3 id := c.Param("id")
 4
 5 if err := c.Bind(u); err != nil {
 6 return c.JSON(http.StatusBadRequest, err)
 7 }
 8 db := h.DB.Clone()
 9 defer db.Close()
10
11
12 query := bson.M{"_id": bson.ObjectIdHex(id)}
13 update := bson.M{"$pull": bson.M{"students": bson.M
      {"id_student": u.Idstudent}}}
14
15 err := db.DB("test").C("course").Update(query,
      update)
16 if err != nil {
17 return c.JSON(http.StatusBadRequest, err.Error())
18 }
19 return c.JSON(http.StatusOK, u)
20 }
```

รูปที่ 4.11: ไฟล์ student.go

จากภาพที่ 4.11 สามารถอธิบายการทำงานโค๊ดส่วนฟังชั่นก์ลบนักศึกษาในชั้นเรียน โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ deletestudent

- บรรทัดที่ 2 เป็นการสร้างตัวแปลประเภทชื่อ u เพื่อจำลองโมเดล Sutdent
- บรรทัดที่ 3 เป็นกราประกาศตัวแปลเพื่อรับข้อมูล id ของนักศึกษาที่จะค้นหา
- บรรทัดที่ 5-6 เป็นการตรวจสอบการจำลองโมเดลว่าจำลองได้หรือไม่ ถ้าไม่ได้จะแสดงข้อผิดพลาด
- บรรทัดที่ 8 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปลชื่อ db
- บรรทัดที่ 9 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 12 เป็นการสร้างตัวแปล query เพื่อใช้ในฟังก์ชั่น Updateโดยตัวแปลนี้จะเก็บข้อมูลที่จะค้นหานักเรียนที่จะทำการลบ
- บรรทัดที่ 13 เป็นการสร้างตัวแปล update เพื่อใช้ในฟังก์ชั่น Update โดยตัวแปลนี้จะเก็บข้อมูลที่จะลบข้อมูลของนักศึกษา
- บรรทัดที่ 15-16 เชื่อมต่อกับฐานข้อมูลและอัพเดทข้อมูลของนักศึกษาพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 19 คือค่าข้อมูลนักศึกษาที่อัพเดท

```
1  func (h *Handler) updatescore(c echo.Context) error
       {
2  user := new(Course)
3  id := c.Param("id")
4
5  err := c.Bind(user)
6  if err != nil {
7  return c.JSON(http.StatusBadRequest, err)
8  }
9
10 db := h.DB.Clone()
11 defer db.Close()
12
13 query := bson.M{"_id": bson.ObjectIdHex(id)}
14 update1 := bson.M{"$set": bson.M{"students": user.
       Sutdents}}
15
16 err = db.DB("test").C("course").Update(query,
       update1)
17 if err != nil {
18 return c.JSON(http.StatusBadRequest, err.Error())
19 }
20 return c.JSON(http.StatusOK, user)
21 }
```

รูปที่ 4.12: ไฟล์ student.go

จากภาพที่ 4.12 สามารถอธิบายการทำงานโค๊ดส่วนฟังชั่นก์รวมคะแนนของชั้นเรียน โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ updatescore
- บรรทัดที่ 2 เป็นการสร้างตัวแปลประเภทชื่อ user เพื่อจำลองโมเดล Course
- บรรทัดที่ 3 เป็นกราประกาศตัวแปลเพื่อรับข้อมูล id ของนักศึกษาที่จะค้นหา
- บรรทัดที่ 5-7 เป็นการตรวจสอบการจำลองโมเดลว่าจำลองได้หรือไม่ ถ้าไม่ได้จะแสดงข้อผิดพลาด
- บรรทัดที่ 10 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปลชื่อ db
- บรรทัดที่ 11 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 13 เป็นการสร้างตัวแปล query เพื่อใช้ในฟังก์ชั่น Updateโดยตัวแปลนี้จะเก็บ

ข้อมูลที่จะค้นหานักเรียนที่จะทำการค้นหาเพื่อจะอัพเดท

- บรรทัดที่ 14 เป็นการสร้างตัวแปล update เพื่อใช้ในฟังก์ชั่น Update โดยตัวแปลนี้จะเก็บข้อมูลที่จะอัพเดทข้อมูลของนักศึกษา
- บรรทัดที่ 16-18 เชื่อมต่อกับฐานข้อมูลและอัพเดทข้อมูลของนักศึกษาพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 20 คือค่าข้อมูลนักศึกษาที่อัพเดท

```
1 import (
2 "net/http"
3
4 "github.com/labstack/echo"
5 "gopkg.in/mgo.v2/bson"
6
7 )
```

รูปที่ 4.13: ไฟล์ checkname.go

จากภาพที่ 4.13 สามารถอธิบายการทำงานโค๊ดส่วนที่นำเข้าไลบรารีต่างๆในไฟล์ โดยไฟล์ checkname.go ได้ดังนี้

- บรรทัดที่ 2 เป็นการนำเข้าไลบรารี net/http เพื่อใช้งานฟังก์ชั่น http
- บรรทัดที่ 4 เป็นการนำเข้าไลบรารี github.com/labstack/echo สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ echo
- บรรทัดที่ 5 เป็นการนำเข้าไลบรารี gopkg.in/mgo.v2/bson สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ bson ใน mongodb

```
1  func (h *Handler) checkIn(c echo.Context) error {
2  u := new(Course)
3  id := c.Param("id")
4  stdid := c.FormValue("stdid")
5  week := c.FormValue("week")
6        db := h.DB.Clone()
7        defer db.Close()
8        query := bson.M{"_id": bson.ObjectIdHex(id), "
           students.id_student": stdid}
9        update := bson.M{}
10 switch week {
11 case "1":
12 update = bson.M{
13 "$set": bson.M{
14 "students.$.weeks.week1.week": 1,
15 "students.$.weeks.week1.id_active": stdid,
16 },
17 }
18 case "2":
19 update = bson.M{
20 "$set": bson.M{
21 "students.$.weeks.week2.week": 1,
22 "students.$.weeks.week2.id_active": stdid,
23 },
24       .
25       .
26       .
27 }
28 case "15":
29 update = bson.M{
30 "$set": bson.M{
31 "students.$.weeks.week16.week": 1,
32 "students.$.weeks.week16.id_active": stdid,
33 },
34 }
35 case "16":
36 update = bson.M{
37 "$set": bson.M{
38 "students.$.weeks.week16.week": 1,
39 "students.$.weeks.week16.id_active": stdid,
40 },
41 }
42 default:
43 return c.JSON(http.StatusBadRequest, "week invalid")
44 }
45 err := db.DB("test").C("course").Update(query,
     update)
46 if err != nil {
47 return c.JSON(http.StatusBadRequest, err.Error())
48 }
49 return c.JSON(http.StatusOK, u)
50 }
```

รูปที่ 4.14: ไฟล์ checkname.go

จากภาพที่ 4.14 สามารถอธิบายการทำงานโค๊ดส่วนฟังชั่นก์เช็คชื่อ โดยไฟล์ checkname.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ checkIn
- บรรทัดที่ 2 เป็นการสร้างตัวแปลประเภทชื่อ u เพื่อจำลองโมเดล Course
- บรรทัดที่ 3 เป็นกราประกาศตัวแปลชื่อ id เพื่อรับข้อมูลของรายวิชาที่จะค้นหา
- บรรทัดที่ 4 เป็นกราประกาศตัวแปลชื่อ stdid เพื่อรับข้อมูลรหัสนักศึกษาที่จะค้นหา
- บรรทัดที่ 5 เป็นกราประกาศตัวแปลชื่อ week เพื่อรับข้อมูลสัปดาห์ของรายวิชาที่จะค้นหา
- บรรทัดที่ 6 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปลชื่อ db
- บรรทัดที่ 7 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 8 เป็นการสร้างตัวแปล query เพื่อใช้ในฟังก์ชั่น Updateโดยตัวแปลนี้จะเก็บข้อมูลที่จะค้นหานักเรียนที่จะทำการค้นหาเพื่อจะอัพเดท
- บรรทัดที่ 9 เป็นการสร้างตัวแปล update เพื่อใช้ในฟังก์ชั่น Update โดยตัวแปลนี้จะเก็บข้อมูลที่จะอัพเดทข้อมูลของนักศึกษา
- บรรทัดที่ 10-44 เป็นการสร้างเงื่อนไขตรวจสอบการเช็คชื่อ จะนำข้อมูลสัปดาห์เป็นตัวตรวจสอบ ถ้าข้อมูลตรงกันจะให้คะแนน แต่ถ้าข้อมูลไม่ตรงกันจะคืนค่าสตริง week invalid
- บรรทัดที่ 45-47 เชื่อมต่อกับฐานข้อมูลและอัพเดทข้อมูลของนักศึกษาพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 49 คือค่าข้อมูลนักศึกษาที่อัพเดท

4.1.2 โครงสร้างของ API ใน heroku Server

```
1 import (
2 "net/http"
3 "os"
4 "github.com/labstack/echo/v4"
5 "github.com/labstack/echo/v4/middleware"
6 "gopkg.in/mgo.v2"
7
8 )
```

รูปที่ 4.15: ไฟล์ server.go

จากภาพที่ 4.36 สามารถอธิบายการทำงานโค๊ดส่วนที่นำเข้าไลบรารีต่างๆในไฟล์ โดยไฟล์ server.go ได้ดังนี้

- บรรทัดที่ 2 เป็นการนำเข้าไลบรารี net/http เพื่อใช้งานฟังก์ชั่น http
- บรรทัดที่ 3 เป็นการนำเข้าไลบรารี os เพื่อใช้งานฟังก์ชั่นของ os
- บรรทัดที่ 4 เป็นการนำเข้าไลบรารี github.com/labstack/echo/v4 สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ echo v4
- บรรทัดที่ 5 เป็นการนำเข้าไลบรารี github.com/labstack/echo/v4/middleware สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ middleware ใน echo v4
- บรรทัดที่ 6 เป็นการนำเข้าไลบรารี gopkg.in/mgo.v2 สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ mongodb

```
1  func main() {
2  e := echo.New()
3  port := os.Getenv("PORT")
4  uri := os.Getenv("MONGODB_URI")
5
6  e.Use(middleware.Logger())
7  e.Use(middleware.Recover())
8  e.Use(middleware.CORSWithConfig(middleware.CORSConfig{
9  AllowOrigins: []string{"*"},
10 AllowMethods: []string{http.MethodGet, http.
      MethodPut, http.MethodPost, http.MethodDelete},
11 }))
12
13 db, err := mgo.Dial(uri)
14 if err != nil {
15 e.Logger.Fatal(err)
16 }
17
18 h := &Handler{DB: db}
19
20 }
```

รูปที่ 4.16: ไฟล์ server.go

จากภาพที่ 4.37 สามารถอธิบายการทำงานโค๊ดส่วนที่เชื่อมต่อกับฐานข้อมูลจากฐานข้อมูล

โดยไฟล์ server.go ได้ดังนี้

- บรรทัดที่ 1 สร้างฟังก์ชั่นชื่อว่า main
- บรรทัดที่ 2 สร้างตัวแปล e เพื่อเรียกใช้ฟังก์ชั่น New() โดยฟังก์ชั่นนี้จะสามารถใช้งานฟังก์ชั่นต่างๆใน echo v4 ได้
- บรรทัดที่ 3 สร้างตัวแปล port เพื่อเก็บค่าของการสุ่ม PORT
- บรรทัดที่ 4 สร้างตัวแปล uri เพื่อเก็บประเภทของฐานขอมูลที่จะเชื่อมต่อ
- บรรทัดที่ 6-10 เป็นการอนุญาติให้สามารถเข้าถึงฐานข้อมูลได้ โดยสิ่งที่ทำได้คือ get put post และ delete
- บรรทัดที่ 12 เป็นการสร้างตัวแปล db เพื่อเชื่อมต่อกับฐานข้อมูล
- บรรทัดที่ 13-15 เป็นการตรวจสอบว่าเชื่อต่อกับฐานข้อมูลได้หรือไม่ ถ้าไม่ก็จะคือค่า error
- บรรทัดที่ 18 เป็นการประกาศตัวแปล h เพื่อเก็บข้อมูลจากไฟล์ handler.go ซึ่ง Handler จะเก็บข้อมูลจากฐานข้อมูล

```
1  import (
2  "gopkg.in/mgo.v2"
3
4  )
5
6  type (
7  Handler struct {
8  DB *mgo.Session
9  }
10 )
```

รูปที่ 4.17: ไฟล์ handler.go

จากภาพที่ 4.38 สามารถอธิบายการทำงานโค๊ดส่วนที่นำเข้าไลบรารีต่างๆในไฟล์ โดยไฟล์ handler.go ได้ดังนี้

- บรรทัดที่ 2 เป็นการนำเข้าไลบรารีของ mongodb
- บรรทัดที่ 6-9 เป็นการสร้างตัวแปลเพื่อรับค่าจากฐานข้อมูล mongodb

```
1  import (
2  "gopkg.in/mgo.v2/bson"
3
4  )
5
6  type (
7        DataQR struct {
8              ID bson.ObjectId `json:"id" bson:"_id,
                  omitempty"`
9              CourseID string `json:"course_id" bson:"
                  course_id"`
10             CourseKEY string `json:"course_key" bson
                  :"course_key"`
11             Url string `json:"url" bson:"url"`
12             }
13       )
```

รูปที่ 4.18: ไฟล์ module.go

จากภาพที่ 4.39 สามารถอธิบายการทำงานโค๊ดส่วนที่สร้างโมเดลของฐานข้อมูล โดยไฟล์ module.go ได้ดังนี้

- บรรทัดที่ 2 เป็นการนำเข้าไลบรารีของ mongodb
- บรรทัดที่ 6-12 เป็นการสร้างโมเดลของฐานข้อมูล mongodb ชื่อ DataQR

```
1  import (
2  "net/http"
3  "github.com/labstack/echo/v4"
4  "gopkg.in/mgo.v2/bson"
5
6  )
```

รูปที่ 4.19: ไฟล์ checkstatus.go

จากภาพที่ 4.19 สามารถอธิบายการทำงานโค๊ดส่วนที่นำเข้าไลบรารีต่างๆในไฟล์ โดยไฟล์ checkstatus.go ได้ดังนี้

- บรรทัดที่ 2 เป็นการนำเข้าไลบรารี net/http สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ http

- บรรทัดที่ 3 เป็นการนำเข้าไลบรารี github.com/labstack/echo/v4 สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ echo v4
- บรรทัดที่ 4 เป็นการนำเข้าไลบรารี gopkg.in/mgo.v2/bson สำหรับเรียกใช้งานฟังชั่นก์ต่างๆของ bson ใน mongodb

```
1  func (h *Handler) createqr(c echo.Context) (err
      error) {
2
3  user := &DataQR{ID: bson.NewObjectId()}
4  err = c.Bind(user)
5  if err != nil {
6  return c.JSON(http.StatusBadRequest, err.Error())
7  }
8
9  db := h.DB.Clone()
10 defer db.Close()
11
12 if err = db.DB(”heroku_4v7cvj1l”).C(”qr_api”).Insert
      (user); err != nil {
13 return c.JSON(http.StatusBadRequest, err)
14 }
15 return c.JSON(http.StatusOK, user)
16 }
```

รูปที่ 4.20: ไฟล์ checkstatus.go

จากภาพที่ 4.20 สามารถอธิบายการทำงานโค๊ดส่วนที่สร้างข้อมูลของคิวอาร์โค้ด โดยไฟล์ checkstatus.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ createqr
- บรรทัดที่ 3 เป็นการประกาศตัวแปลชื่อ users เพื่อรับค่าจากโมเดล DataQR ซึ่งจะทำการเซ็ตค่า ID ตั้งแต่ต้น
- บรรทัดที่ 4-6 เป็นการตรวจสอบการรับค่าโมเดลว่าถูกต้องหรือไม่ ถ้าผิดพลาดจะแสดงข้อผิดพลาด
- บรรทัดที่ 9 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปลชื่อ db
- บรรทัดที่ 10 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb

- บรรทัดที่ 12-13 เชื่อมต่อกับฐานข้อมูลและเพิ่มข้อมูลของชั้นเรียนพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 15 คืนค่าที่ได้จากการเพิ่มข้อมูล

```
1  func (h *Handler) updatekey(c echo.Context) (err
     error) {
2  users := new(DataQR)
3  if err = c.Bind(users); err != nil {
4  return
5  }
6  courseid := c.FormValue("courseid")
7  coursekey := c.FormValue("coursekey")
8  urls := c.FormValue("url")
9
10 db := h.DB.Clone()
11 defer db.Close()
12
13 query := bson.M{"course_id": courseid}
14 update := bson.M{"$set": bson.M{"course_key":
     coursekey, "url": urls}}
15
16 if err = db.DB("heroku_4v7cvj1l").C("qr_api").Update
     (query, update); err != nil {
17 return c.JSON(http.StatusBadRequest, err.Error())
18 }
19 if err = db.DB("heroku_4v7cvj1l").C("qr_api").Find(
     bson.M{"course_id": courseid}).One(&users); err
     != nil {
20 return c.JSON(http.StatusBadRequest, err)
21 }
22
23 return c.JSON(http.StatusOK, users.ID)
24 }
```

รูปที่ 4.21: ไฟล์ checkstatus.go

จากภาพที่ 4.21 สามารถอธิบายการทำงานโค๊ดส่วนที่อัพเดทข้อมูลของงคิวอาร์โค้ด โดยไฟล์ checkstatus.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ createqr
- บรรทัดที่ 2 เป็นการประกาศตัวแปลชื่อ users เพื่อรับค่าจากโมเดล DataQR

- บรรทัดที่ 3-4 เป็นการตรวจสอบการรับค่าโมเดลว่าถูกต้องหรือไม่ ถ้าผิดพลาดจะแสดงข้อผิดพลาด
- บรรทัดที่ 6 เป็นการประกาศตัวแปลชื่อ courseid เพื่อรับค่า id สำหรับค้นหาข้อมูลใน DataQR
- บรรทัดที่ 7 เป็นการประกาศตัวแปลชื่อ coursekey เพื่อรับค่า Key สำหรับค้นหาข้อมูลที่จะอัพเดท
- บรรทัดที่ 8 เป็นการประกาศตัวแปลชื่อ urls เพื่อรับค่า Url สำหรับอัพเดท url
- บรรทัดที่ 10 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปลชื่อ db
- บรรทัดที่ 11 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 13 เป็นการสร้างตัวแปล query เพื่อใช้ในฟังก์ชั่น Updateโดยตัวแปลนี้จะเก็บข้อมูลที่จะค้นหานักเรียนที่จะทำการค้นหาเพื่อจะอัพเดท
- บรรทัดที่ 14 เป็นการสร้างตัวแปล update เพื่อใช้ในฟังก์ชั่น Update โดยตัวแปลนี้จะเก็บข้อมูลที่จะอัพเดทข้อมูลของนักศึกษา
- บรรทัดที่ 16-17 เชื่อมต่อกับฐานข้อมูลและอัพเดทข้อมูลของชั้นเรียนพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 19-20 เชื่อมต่อกับฐานข้อมูลและค้นหาข้อมูลของชั้นเรียนพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 23 คืนค่าที่ Key ได้จากการอัพเดทข้อมูล

```
1  func (h *Handler) getKey(c echo.Context) (err error)
       {
2  users := new(DataQR)
3  if err = c.Bind(users); err != nil {
4  return
5  }
6
7  id := c.Param("id")
8
9  db := h.DB.Clone()
10 defer db.Close()
11
12 if err = db.DB("heroku_4v7cvj1l").C("qr_api").Find(
     bson.M{"_id": bson.ObjectIdHex(id)}).One(&users);
       err != nil {
13 return c.JSON(http.StatusBadRequest, err)
14 }
15
16 return c.Redirect(http.StatusMovedPermanently, users
     .Url)
17 }
```

รูปที่ 4.22: ไฟล์ checkstatus.go

จากภาพที่ 4.22 สามารถอธิบายการทำงานโค๊ดส่วนที่ดึงข้อมูลคิวอาร์โค้ดไปใช้ โดยไฟล์ checkstatus.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ getKey
- บรรทัดที่ 2 เป็นการประกาศตัวแปลชื่อ users เพื่อรับค่าจากโมเดล DataQR
- บรรทัดที่ 3-4 เป็นการตรวจสอบการรับค่าโมเดลว่าถูกต้องหรือไม่ ถ้าผิดพลาดจะแสดงข้อผิดพลาด
- บรรทัดที่ 7 เป็นการประกาศตัวแปลชื่อ id เพื่อรับค่า id สำหรับค้นหาข้อมูล
- บรรทัดที่ 9 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปลชื่อ db
- บรรทัดที่ 10 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 12-13 เชื่อมต่อกับฐานข้อมูลและค้นหาข้อมูลของชั้นเรียนพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 16 คืนค่าที่ Url ได้จากการค้นหาข้อมูล

4.2 การพัฒนาเว็บแอปพลิเคชัน

การพัฒนาระบบเช็คชื่อด้วยคิวอาร์โค้ดสำหรับเว็บแอปพลิเคชันนั้นวัตถุประสงค์หลังเพื่อสร้างความสะดวกต่อการใช้งานระบบเช็คชื่อด้วยคิวอาร์โค้ด โดยตัวเว็บแอปพลิเคชันนี้ถูกพัฒนาขึ้นด้วย Angular มีรายละเอียดการทำงานดังนี้

4.2.1 โครงสร้างและโค้ด Service เพื่อใช้บริการ API

ในการเชื่อมต่อเว็บแอปพลิเคชันกับ API จำเป็นต้องสร้างโฟเดอร์ Service เพื่อใช้บริการต่างๆ ของ API ทำได้ดังนี้

```
1 import { Injectable, OnInit } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { Local } from 'protractor/built/
     driverProviders';
4 import { Subject } from 'rxjs';
5 import { delay, map } from 'rxjs/operators';
6
7 apiURL = 'http://10.0.0.27:443'
```

รูปที่ 4.23: ไฟล์ student.service.ts

จากภาพที่ 4.23 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการนำเข้าไลบรารี Injectable, OnInit
- บรรทัดที่ 2 เป็นการนำเข้าไลบรารี HttpClient
- บรรทัดที่ 3 เป็นการนำเข้าไลบรารี Local
- บรรทัดที่ 4 เป็นการนำเข้าไลบรารี rxjs
- บรรทัดที่ 5 เป็นการนำเข้าไลบรารี operators
- บรรทัดที่ 7 เป็นการประกาศตัวแปลชื่อ apiURL เพื่อเก็บโฮดของ API ที่จะใช้บริการ

```
1 getStudent() {
2 return this.httpClient.get(`${this.apiURL}/getCourse
     `);
3 }
```

รูปที่ 4.24: ไฟล์ student.service.ts

จากภาพที่ 4.24 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ getStudent
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาท getCourse

```
1 getIP() {
2 return this.httpClient.get(`${this.apiURL}/getIP`)
3 }
```

รูปที่ 4.25: ไฟล์ student.service.ts

จากภาพที่ 4.25 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ getIP
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาท getIP

```
1   chakname(id: any, stdids: any, weeks: any) {
2 var Data: any = new FormData();
3 Data.append("stdid", stdids);
4 Data.append("week", weeks);
5 return this.httpClient.post(`${this.apiURL}/
     chackname/${id}`, Data);
6 }
```

รูปที่ 4.26: ไฟล์ student.service.ts

จากภาพที่ 4.26 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ chakname โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า id stids และ weeks มา

- บรรทัดที่ 2 เป็นการสร้างตัวแปลชื่อ Data เพื่อเซ็ตข้อมูลที่จะส่งไปหา API เป็นรูปแบบ FormData
- บรรทัดที่ 3 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า stdid
- บรรทัดที่ 4 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า week
- บรรทัดที่ 5 เป็นการนำเรียกใช้ API โดยใช้พาท chackname โดยส่ง Data ไปด้วย

```
1  chacknameby_t(id: any, stdids: any, weeks: any) {
2 var Data: any = new FormData();
3 Data.append("stdid", stdids);
4 Data.append("week", weeks);
5 return this.httpClient.post(`${this.apiURL}/
     chacknameT/${id}`, Data);
6 }
```

รูปที่ 4.27: ไฟล์ student.service.ts

จากภาพที่ 4.27 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ chacknameby-t โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า id stids และ weeks มา
- บรรทัดที่ 2 เป็นการสร้างตัวแปลชื่อ Data เพื่อเซ็ตข้อมูลที่จะส่งไปหา API เป็นรูปแบบ FormData
- บรรทัดที่ 3 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า stdid
- บรรทัดที่ 4 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า week
- บรรทัดที่ 5 เป็นการนำเรียกใช้ API โดยใช้พาท chacknameT โดยส่ง Data ไปด้วย

```
1  checknameOut_t(id: any, stdids: any, weeks: any) {
2 var Data: any = new FormData();
3 Data.append("stdid", stdids);
4 Data.append("week", weeks);
5 return this.httpClient.post(`${this.apiURL}/
     checknameO/${id}`, Data);
6 }
```

รูปที่ 4.28: ไฟล์ student.service.ts

จากภาพที่ 4.28 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ checknameOut-t โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า id stids และ weeks มา
- บรรทัดที่ 2 เป็นการสร้างตัวแปลชื่อ Data เพื่อเซ็ตข้อมูลที่จะส่งไปหา API เป็นรูปแบบ FormData
- บรรทัดที่ 3 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า stdid
- บรรทัดที่ 4 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า week
- บรรทัดที่ 5 เป็นการนำเรียกใช้ API โดยใช้พาท checknameO โดยส่ง Data ไปด้วย

```
1  uploadStudent(course_id: any, course_names: any,
     times: any, sesons: any, student: any) {
2  return this.httpClient.post(`${this.apiURL}/upload`,
     {
3  course_id: course_id+"-"+sesons,
4  course_name: course_names,
5  time: times,
6  seson: sesons,
7  students: student
8  });
9  }
```

รูปที่ 4.29: ไฟล์ student.service.ts

จากภาพที่ 4.29 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ uploadStudent โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า courseid coursenames times sesons และ student มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาท upload
- บรรทัดที่ 3-7 เป็นการเซ็ตข้อมูลที่จะส่งไปให้กับ API

```
1  updateStudent(id: any, dataupdate: any) {
2  return this.httpClient.post(`${this.apiURL}/
     updatescore/${id}`, {
3  students: dataupdate
4  });
5  }
```

รูปที่ 4.30: ไฟล์ student.service.ts

จากภาพที่ 4.30 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ updateStudent โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า course-id และ dataupdate มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาท updatescore
- บรรทัดที่ 3 เป็นการเซ็ตข้อมูลที่จะส่งไปให้กับ API

```
1  addstudent(idcoure: any, idstudent: any, names:
     any, number: any) {
2  return this.httpClient.post(`${this.apiURL}/
     addupstudent/${idcoure}`, {
3  number_id: number,
4  id_student: idstudent,
5  name: names
6  })
7  }
```

รูปที่ 4.31: ไฟล์ student.service.ts

จากภาพที่ 4.31 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ addstudent โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า idcoure idstudent names และ number มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาท addupstudent
- บรรทัดที่ 3-5 เป็นการเซ็ตข้อมูลที่จะส่งไปให้กับ API

```
1  editstudent(idcoure: any, idstudent: any, names:
     any, number: any) {
2 return this.httpClient.post(`${this.apiURL}/
    editstudent/${idcoure}`, {
3 number_id: number,
4 id_student: idstudent,
5 name: names
6 })
7 }
```

รูปที่ 4.32: ไฟล์ student.service.ts

จากภาพที่ 4.32 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ editstudent โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า idcoure idstudent names และ number มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาท editstudent
- บรรทัดที่ 3-5 เป็นการเซ็ตข้อมูลที่จะส่งไปให้กับ API

```
1  deletestudent(idcoure: any, idstudent: any) {
2 return this.httpClient.post(`${this.apiURL}/
    deletestudent/${idcoure}`, {
3 id_student: idstudent
4 })
5 }
```

รูปที่ 4.33: ไฟล์ student.service.ts

จากภาพที่ 4.33 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ deletestudent โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า idcoure และ idstudent มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาท deletestudent
- บรรทัดที่ 3-5 เป็นการเซ็ตข้อมูลที่จะส่งไปให้กับ API

```
1  deleteCourse(id: any) {
2 return this.httpClient.delete(`${this.apiURL}/
     deleteCourse/${id}`);
3 }
```

รูปที่ 4.34: ไฟล์ student.service.ts

จากภาพที่ 4.34 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ deleteCourse โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า id มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาท deleteCourse

```
1  reportCourse(id: any) {
2 return this.httpClient.get(`${this.apiURL}/report/${
     id}`);
3 }
```

รูปที่ 4.35: ไฟล์ student.service.ts

จากภาพที่ 4.35 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ reportCourse โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า id มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาท report

```
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3
4 apiURL : string = "https://testmypro-01.herokuapp.
     com"
```

รูปที่ 4.36: ไฟล์ heroku.service.ts

จากภาพที่ 4.36 โครงสร้างของไฟล์ heroku.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการนำเข้าไลบรารี Injectable
- บรรทัดที่ 2 เป็นการนำเข้าไลบรารี HttpClient
- บรรทัดที่ 4 เป็นการสร้างตัวแปลชื่อ apiURL เพื่อเก็บโฮดของ API ที่จะใช้งาน

```
1   getKey(key:any){
2 return this.httpClient.get(`${this.apiURL}/getKey/`+
      key)
3 }
```

รูปที่ 4.37: ไฟล์ heroku.service.ts

จากภาพที่ 4.37 โครงสร้างของไฟล์ heroku.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ getKey โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า key มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาท getKey

```
1   craeteQR(idcours:any,keycours:any,herokuUrl:any){
2 return this.httpClient.post(`${this.apiURL}/createqr
      `,{
3 course_id:idcours,
4 course_key:keycours,
5 url:herokuUrl
6 });
7 }
```

รูปที่ 4.38: ไฟล์ heroku.service.ts

จากภาพที่ 4.38 โครงสร้างของไฟล์ heroku.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ craeteQR โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า idcours keycours และ herokuUrl มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาท createqr
- บรรทัดที่ 3-5 เป็นการเซ็ตข้อมูลที่จะส่งไปให้กับ API

```
1   updateQR(idcours:any,keycours:any,herokuUrl:any){
2 var Data:any = new FormData();
3 Data.append("courseid",idcours)
4 Data.append("coursekey",keycours)
5 Data.append("url",herokuUrl)
6 return this.httpClient.post(`${this.apiURL}/updateqr
    `,Data)
7 }
```

รูปที่ 4.39: ไฟล์ heroku.service.ts

จากภาพที่ 4.39 โครงสร้างของไฟล์ heroku.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ updateQR โดยการจะเรียกใช้ฟังก์ชั่นนี้ได้ต้องส่งค่า idcours keycours และ herokuUrl มา
- บรรทัดที่ 2 เป็นการสร้างตัวแปลชื่อ Data เพื่อเซ็ตข้อมูลที่จะส่งไปหา API เป็นรูปแบบ FormData
- บรรทัดที่ 3 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า courseid
- บรรทัดที่ 4 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า coursekey
- บรรทัดที่ 5 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า url
- บรรทัดที่ 6 เป็นการนำเรียกใช้ API โดยใช้พาท updateqr โดยส่ง Data ไปด้วย

4.2.2 โครงสร้างของการสร้างชั้นเรียน

```
<input class="btn btn-info " type="file" (change)="
    onFileChange($event)" multiple="false" />

<div class="row">

<div class="col-md-6"></div>
<div class="input-group col-md-4">
</div>
<div class="col-md-2 container">
<select class="form-control" #seson name="seson" id
    ="">
<option selectedเลือกเทอม></option>
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
</select>
</div>
</div>
<pรหัสวิชา> :{{course_id}}</p>
<pชื่อวิชา> : {{course_name}}</p>
<pวันเวลาเรียน> : {{time}}</p>
<div class="text-right">
<button class="btn btn-success " (click)="upload(seson.value)">Upload
    !</button>
</div>
<hr>
<table class="table">
<thead>
<tr>
<th scope="colเลขที่"></th>
<th scope="colรหัสนักศึกษา"></th>
<th scope="colชื่อนานสกุล">-</th>
</tr>
</thead>
<tbody *ngFor="let s of student | paginate: { itemsPerPage: 20,
    currentPage: p }">
<tr>
<th>{{s.number}}</th>
<td>{{s.id_student}}</td>
<td>{{s.name}}</td>
</tr>
</tbody>
</table>
```

รูปที่ 4.40: การสร้างหน้าจอส่วนสร้างชั้นเรียน formfile.component.html

จากภาพที่ 4.40 โครงสร้างของการสร้างหน้าจอส่วนสร้างชั้นเรียน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างช่องรับข้อมูลไฟล์สำหรับสร้างชั้นเรียน
- บรรทัดที่ 8-13 เป็นการสร้างตัวเลือกให้อาจารย์ใช้ในการเลือกเทอมที่จะสร้างชั้นเรียน
- บรรทัดที่ 16- เป็นการโชว์ข้อมูลที่รับเข้ามา

```
onFileChange(evt: any) {
try {
/* wire up file reader */
const target: DataTransfer = <DataTransfer>(evt.
   target);
if (target.files.length !== 1) throw new Error('
   Cannot use multiple files');
const reader: FileReader = new FileReader();
reader.onload = (e: any) => {
/* read workbook */
const bstr: string = e.target.result;
const wb: XLSX.WorkBook = XLSX.read(bstr, { type: '
   binary' });

/* grab first sheet */
const wsname: string = wb.SheetNames[0];
const ws: XLSX.WorkSheet = wb.Sheets[wsname];
this.wss = [ws.B4.w, ws.B6.w, ws.M4.w]

/* save data */
this.data = <any>(XLSX.utils.sheet_to_json(ws, {
   header: 1 }));
this.course = ws.B4.w;
this.time = ws.B6.w;
for (let i = 9; i <= this.data.length - 3; i++) {
this.student[i - 9] = this.data[i];
for (let j = 1; j <= 3; j++) {
this.student[i - 9]['number'] = this.data[i][1];
this.student[i - 9]['id_student'] = this.data[i][2];
this.student[i - 9]['name'] = this.data[i][3];
}
this.student[i - 9].splice(0)
}
this.course_id = this.course.substr(10,7);
this.course_name=this.course.substr(19);
this.time=this.time.substr(13);
};
return reader.readAsBinaryString(target.files[0]);
} catch (error) {
alert('fire ไม่ถูกต้อง')
}
}
```

รูปที่ 4.41: การสร้างหน้าจอส่วนสร้างชั้นเรียน formfile.component.ts

จากภาพที่ 4.42 โครงสร้างของการสร้างหน้าจอส่วนสร้างชั้นเรียน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ onFileChange โดยจับรับไฟล์ข้อมูลมา
- บรรทัดที่ 4-34 เป็นการอ่านไฟล์ที่รับเข้ามา และเก็บข้อมูลที่ได้จากการอ่านไว้ใน student ซึ่งโค้ดการอ่านไฟล์นี้จะถูกครอบด้วย Try catch
- บรรทัดที่ 35-37 เป็นการเช็ตว่าการอ่านข้อมูลถูกต้องหรือไม่ถ้าไม่จะแสดงข้อความ fire ไม่ถูกต้อง

```
1  upload(seson:number) {
2  var data =[];
3  for (var key in this.student) {
4  data.push({
5  'number_id': this.student[key]['number'],
6  'id_student': this.student[key]['id_student'],
7  'name': this.student[key]['name']
8  });
9  }
10
11 if (this.student.length<=1) {
12 alertยังไม่ได้เลือก(" fire");
13 }else{
14 if (localStorage.getItem('name')!=null) {
15 this.nameT=localStorage.getItem('name');
16 if (seson==1||seson==2||seson==3) {
17 console.log("work seson");
18 this.studentService.uploadStudent(this.course_id,this.course_name,this.
       time,seson, data).subscribe(
19 () => {
20 this.router.navigate(['/'])
21 }
22 )
23 }else{
24 alertเลือกเทอม('')
25 }
26 }else{
27 alertยังไม่มีชื่ออาจารย์('')
28 }
29 }
30 }
```

รูปที่ 4.42: การสร้างหน้าจอส่วนสร้างชั้นเรียน formfile.component.ts

จากภาพที่ 4.42 โครงสร้างของการสร้างหน้าจอส่วนสร้างชั้นเรียน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ upload โดยจับรับข้อมูล seson มา
- บรรทัดที่ 2-9 เป็นการสร้างตัวแปลชื่อ Data เพื่อเตรียมข้อมูลสำหรับอัพโหลด
- บรรทัดที่ 11-12 เป็นการเช็ตว่าใน student มีข้อมูลหรือไม่

– บรรทัดที่ 14-28 เป็นการเช็ตชื่อ อาจารย์ และเทอม ว่ามีข้อมูลหรือไม่ ถ้ามีก็จะเรียกใช้งานฟังก์ชั่น uploadStudent ในไฟล์ student.service.ts ซึ่งจะส่งข้อมูล course-id course-name time seson และ data ไปด้วย

4.2.3 โครงสร้างของการอัพเดทข้อมูลนักศึกษา

```
1  <button [hidden]="showB" class="btn-lg btn-block btn
       -info" (click)="show()" style="line-height: เพิ่ม
       นักศึกษา1.5;"></button>
2  <table class="table">
3          <thead>
4                  <tr>
5                  <th scope="colเลขที่"></th>
6                  <th scope="colรหัสนักศึกษา"></th>
7                  <th scope="colชื่อ"></th>
8                  <th colspan="2" class="text-center">Update</th>
9                  </tr>
10         </thead>
11         <tbody>
12                 <tr *ngFor="let s of course.students | formFilter:search
                       ">
13                 <td class="text-center">{{s.number_id}}</td>
14                 <td>{{s.id_student}}</td>
15                 <td>{{s.name}}</td>
16                 <td><button class="btn-lg btn-block btn-success" (
                       click)="showadd(content)แก้ไข"></button></td>
17                 <td><button type="submit" class="btn-lg btn-block
                       btn-danger" (click)="delete(s.id_student)ลบ"></
                       button></td>
18 </tr>
19 </tbody>
20 </table>
```

รูปที่ 4.43: การสร้างหน้าจอส่วนอัพเดทข้อมูลนักศึกษา formup.component.html

จากภาพที่ 4.43 โครงสร้างของการสร้างหน้าจอส่วนอัพเดทข้อมูลนักศึกษา สามารถอธิบายการทำงานได้ดังนี้

– บรรทัดที่ 1 เป็นการสร้างปุ่มสำหรับเพิ่มนักศึกษา

– บรรทัดที่ 3-15 เป็นการโชว์ข้อมูลรายชื่อนักศึกษาในชั้นเรียนทั้งหมด

- บรรทัดที่ 16 เป็นการสร้างปุ่มสำหรับแก้ไขข้อมูลนักศึกษา
- บรรทัดที่ 17 เป็นการสร้างปุ่มสำหรับลบนักศึกษา

```
1  addup(idstudent: any, namef: any, namel: any) {
2  var name = namef + " " + namel
3  this.number += 1;
4  this.studentService.addstudent(this.idcoure,
      idstudent, name, this.number).subscribe(
5  () => {
6  alert("status OK")
7  this.studentService.reportCourse(this.idcoure).
      subscribe(
8  (data) => {
9  this.course = data[0];
10 }
11 )
12 document.getElementById('close1').click()
13 }
14 )
15 }
```

รูปที่ 4.44: การสร้างหน้าจอส่วนอัพเดทข้อมูลนักศึกษา formup.component.ts

จากภาพที่ 4.44 โครงสร้างของการสร้างหน้าจอส่วนอัพเดทข้อมูลนักศึกษา สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ addup โดยจับรับข้อมูล idstudent namef และ namel มา
- บรรทัดที่ 2 เป็นการรวมข้อมูลระหว่าง namef และ namel โดย namef คือชื่อ และ namel คือนามสกุล
- บรรทัดที่ 3 เป็นการนำเอาเลขที่สุดท้ายบวกหนึ่งเพื่อจะเป็นเลขที่ใหม่สำหรับนักศึกษาที่ถูกเพิ่ม
- บรรทัดที่ 4-14 เป็นการเรียนใช้ฟังก์ชั้น addstudent ในไฟล์ student.service.ts ซึ่งเป็นฟังก์ชั้นสำหรับเพิ่มนักศึกษา

```
1   editstudent(idstudent: any, name: any, number: any
      ) {
2  this.studentService.editstudent(this.idcoure,
     idstudent, name, number).subscribe(
3  () => {
4  alert("status OK")
5  this.studentService.reportCourse(this.idcoure).
     subscribe(
6  (data) => {
7  this.course = data[0];
8  }
9  )
10 document.getElementById('close2').click()
11 }
12 )
13 }
```

รูปที่ 4.45: การสร้างหน้าจอส่วนอัพเดทข้อมูลนักศึกษา formup.component.ts

จากภาพที่ 4.45 โครงสร้างของการสร้างหน้าจอส่วนอัพเดทข้อมูลนักศึกษา สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ editstudent โดยจับรับข้อมูล idstudent name และ number มา
- บรรทัดที่ 2-12 เป็นการเรียนใช้ฟังก์ชั้น editstudent ในไฟล์ student.service.ts ซึ่งเป็นฟังก์ชั้นสำหรับแก้ไขข้อมูลนักศึกษา

```
1   delete(id: any) {
2  this.studentService.deletestudent(this.idcoure, id).
     subscribe(
3  () => {
4  this.studentService.reportCourse(this.idcoure).
     subscribe(
5  (data) => {
6  this.course = data[0];
7  }
8  )
9  }
10 );
11 }
```

รูปที่ 4.46: การสร้างหน้าจอส่วนอัพเดทข้อมูลนักศึกษา formup.component.ts

จากภาพที่ 4.46 โครงสร้างของการสร้างหน้าจอส่วนอัพเดทข้อมูลนักศึกษา สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ delete โดยจับรับข้อมูล id มา
- บรรทัดที่ 2-10 เป็นการเรียนใช้ฟังก์ชั้น deletestudent ในไฟล์ student.service.ts ซึ่งเป็นฟังก์ชั้นสำหรับลบข้อมูลนักศึกษา

4.2.4 โครงสร้างของการสร้างคิวอาร์โค้ด

```
<button type="button" class="btn btn-info btn-lg
   text-center"
(click)="student(weeks.value,times.value,content);
   timerSrarted$.next(true);">Start QR
code</button>
<ng-template #content let-modal>
<div class="modal-header">
<h4 class="modal-title text-centerสัปดาห์
   ที่">{{weeks.value}}</h4>
<a type="button" class="close" aria-label="Close" id="close"
(click)="timerSrarted$.next(false); modal.dismiss('Cross click')">
<span aria-hidden="true">&times;</span>
</a>
</div>
<div class="modal-body">
<div class="text-center">
<!-- <p>week at {{weeks.value}}</p> -->
<h1>
<countdown #countdown [config]="{leftTime: time, demand: ti}">$!m!:$!s
   !</countdown>
</h1>
</div>
<div class="container col-md-8">
<ngx-qrcode
[qrc-value]="qrcode_url"
qrc-element-type="url"
qrc-class="aclass"
qrc-errorCorrectionLevel="L"
qrc-scanQrcode="timeqrcode"
>
</ngx-qrcode>
</div>
</div>
</ng-template>
```

รูปที่ 4.47: การสร้างหน้าจอส่วนสร้างคิวอาร์โค้ด qrcode.component.html

จากภาพที่ 4.47 โครงสร้างของการสร้างหน้าจอส่วนสร้างคิวอาร์โค้ด สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-3 เป็นการสร้างปุ่มสำหรับสร้างคิวอาร์โค้ด
- บรรทัดที่ 4-30 เป็นการแสดงคิวอาร์โค้ดที่สร้างรวมถึงเวลาของคิวอาร์โค้ด

```
1  Start(week: any, times: any, content) {
2  this.weeks = week;
3  this.time = times;
4
5  var herokuUrl = 'http://'+this.ip+':80/student/' +
     this.idcoure + ',' + this.weeks;
6
7  this.heroku(herokuUrl,content)
8  this.qruery()
9
10 }
```

รูปที่ 4.48: การสร้างหน้าจอส่วนสร้างคิวอาร์โค้ด qrcode.component.ts

จากภาพที่ 4.48 โครงสร้างลอจิกของหน้าดูรายละเอียดของข่าวสาร สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ Start โดยจับรับข้อมูล week times และ content มา
- บรรทัดที่ 2 เป็นการเก็บข้อมูล week ไว้ใน weeks
- บรรทัดที่ 3 เป็นการเก็บข้อมูล times ไว้ใน time
- บรรทัดที่ 5 เป็นการประกาศตัวแปลชื่อ herokuUrl เพื่อเก็บข้อมูลพาทที่จะไปยังหน้ากรอกรหัสนักศึกษาเพื่อเช็คชื่อ
- บรรทัดที่ 7 เป็นการเรียกใช้ฟังก์ชั่น heroku โดยฟังก์ชั่นนี้จะทำหน้าที่ส่งค่า url ไปเก็บไว้ใน heroku server
- บรรทัดที่ 8 เป็นการเรียกใช้ฟังก์ชั่น qruery โดยฟังก์ชั่นนี้จะทำหน้าที่สร้างคิวอาร์โค้ด

4.2.5 โครงสร้างของการจัดการคะแนน

```
<select class="form-control" #status name="status"
   style="width: 20%;">
<option selectedทั้งหมด></option>
<option valueมา="1"></option>
<option valueสาย="0.5"></option>
<option valueขาด="0"></option>
</select>
<table class="table table-bordered">
<thead>
<tr class="text-center">
<th scope="colเลขที่"></th>
<th scope="colรหัสนักศึกษา"></th>
<th scope="colชื่อ"></th>
<th colspanสัปดาห์ที่="4"> {{twee}}</th>
<th scope="colรวม"></th>
</tr>

</thead>
<tbody>
<tr *ngFor="let s of course.students | studentFilter:[{week:twee},{status:
    statusShow}] | paginate: { itemsPerPage: 20, currentPage: p }" >
<td class="text-center">{{s.number_id}}</td>
<td>{{s.id_student}}</td>
<td>{{s.name}}</td>
<td class="text-center" style="font-size: 1.2rem;">
<span >{{retest(s.weeks)}}</span>
</td>
<td class="text-center"><button class="btn-lg btn-block btn-info" id
    ="click1" (click)="chackbyTcome(s.id_student,1)มา"></button></td>
<td class="text-center"><button class="btn-lg btn-block btn-warning
    " (click)="chackbyTlast(s.id_student,0.5)สาย"></button></td>
<td class="text-center"><button class="btn-lg btn-block btn-danger
    " id="click1" (click)="chackbyOut(s.id_student,1)ขาด"></button></td
    >
<td class="text-center" style="font-size: 1.2rem;">{{getTotal(s.weeks)
    }}</td>
</tr>
</tbody>
</table>
```

รูปที่ 4.49: การสร้างหน้าจอส่วนจัดการคะแนน info.component.html

จากภาพที่ 4.49 โครงสร้างลอจิกของหน้าจัดการคะแนน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-6 เป็นการสร้างตัวเลือกสำหรับค้นหาคนที่ มา สายและขาด
- บรรทัดที่ 7-32 เป็นการโชว์ข้อมูลคะแนนของนักศึกษาแต่ละอาทิต และปุ่มสำหรับให้คะแนน

```
1   chackbyTcome(idstudent: any, count: number) {
2  this.studentService.chakname(this.idcoure, idstudent
     , this.twee).subscribe(
3  () => {
4  this.studentService.reportCourse(this.idcoure).
     subscribe(
5  (data) => {
6  this.course = data[0];
7  });
8  }
9  )
10 }
```

รูปที่ 4.50: การสร้างหน้าจอส่วนจัดการคะแนน info.component.ts

จากภาพที่ 4.50 โครงสร้างลอจิกของหน้าจัดการคะแนน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ chackbyTcome โดยจะรับข้อมูล idstudent และ count มา
- บรรทัดที่ 2-10 เป็นการเรียนใช้ฟังก์ชั้น chakname ในไฟล์ student.service.ts ซึ่งเป็นฟังก์ชั้นสำหรับให้คะแนนนักศึกษา

```
1   chackbyTlast(idstudent: any, count: number) {
2  this.studentService.chacknamebyt(this.idcoure,
     idstudent, this.twee).subscribe(
3  () => {
4  this.studentService.reportCourse(this.idcoure).
     subscribe(
5  (data) => {
6  this.course = data[0];
7  });
8  }
9  )
10 }
```

รูปที่ 4.51: การสร้างหน้าจอส่วนจัดการคะแนน info.component.ts

จากภาพที่ 4.51 โครงสร้างลอจิกของหน้าจัดการคะแนน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ chackbyTlast โดยจะรับข้อมูล idstudent และ count มา
- บรรทัดที่ 2-10 เป็นการเรียนใช้ฟังก์ชั้น chacknamebyt ในไฟล์ student.service.ts ซึ่งเป็นฟังก์ชั้นสำหรับให้คะแนนนักศึกษา

```
1   chackbyOut(idstudent: any, count: number) {
2  this.studentService.checknameOut(this.idcoure,
     idstudent, this.twee).subscribe(
3  () => {
4  this.studentService.reportCourse(this.idcoure).
     subscribe(
5  (data) => {
6  this.course = data[0];
7  });
8  }
9  )
10 }
```

รูปที่ 4.52: การสร้างหน้าจอส่วนจัดการคะแนน info.component.ts

จากภาพที่ 4.52 โครงสร้างลอจิกของหน้าจัดการคะแนน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ chackbyOut โดยจะรับข้อมูล idstudent และ count มา
- บรรทัดที่ 2-10 เป็นการเรียนใช้ฟังก์ชั้น checknameOut ในไฟล์ student.service.ts ซึ่งเป็นฟังก์ชั้นสำหรับให้คะแนนนักศึกษา

4.2.6 โครงสร้างของการหน้าดูประวัติการเข้าเรียน

```
1  <button class="btn btn-success" (click)="downloadfile
       ()">Download File</button>
2  <button class="btn btn-secondary" (click)="
       showsumfile(content)รวม
       คะแนน"></button>
3
4  <table class="table table-bordered table-hover">
5  <thead>
6  <tr>
7  <th scope="col" class="text-centerเลขที่"></th>
8  <th scope="col" class="text-centerรหัสนักศึกษา"></th>
9  <th scope="col" class="text-centerชื่อ"></th>
10 <th colspan="16" class="text-centerสัปดาห์"></th>
11 </tr>
12 </thead>
13 <tbody>
14 <tr *ngFor="let s of course.students | paginate: { itemsPerPage: 20,
       currentPage: p } ">
15 <td class="text-center">{{s.number_id}}</td>
16 <td>{{s.id_student}}</td>
17 <td>{{s.name}}</td>
18  <td *ngFor="let w of weeks">
19 <span>{{w.week}}</span>
20 </td>
21 </td>
22 </tr>
23 </tbody>
24 </table>
```

รูปที่ 4.53: การสร้างหน้าจอส่วนหน้าดูประวัติการเข้าเรียน infoname.component.html

จากภาพที่ 4.53 โครงสร้างของการสร้างหน้าจอดูประวัติการเข้าเรียน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างปุ่มสำหรับการดาวน์โหลดไฟล์รายชื่อนักศึกษา
- บรรทัดที่ 2 เป็นการสร้างปุ่มสำหรับรวมคะแนน
- บรรทัดที่ 4-24 เป็นตารางแสดงรายชื่อและคะแนนขแงนักศึกษาของชั้นเรียน

```
1          qruey() {
2          this.route.paramMap.subscribe(params =>
              {
3          this.id = params.get('id');
4          this.studentService.reportCourse(params.
              get('id')).subscribe(
5          (data) => {
6          this.course = data[0];
7          }
8          )
9          });
10         }
```

รูปที่ 4.54: การสร้างหน้าจอส่วนหน้าดูประวัติการเข้าเรียน infoname.component.ts

จากภาพที่ 4.54 โครงสร้างของการสร้างหน้าจอดูประวัติการเข้าเรียน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ qruey
- บรรทัดที่ 3 เป็นการประกาศตัวแปลชื่อ id สำหรับเก็บข้อมูล id ของรายวิชาซึ่งดึงมาจากพาท
- บรรทัดที่ 4 เป็นการเรียกใช้ฟังก์ชั่น reportCourse ในไฟล์ student.service.ts โดยฟังก์ชั่นนี้จะทำหน้าที่ดึงข้อมูลชั้นเรียนที่มี id ตรงกับ id ที่ส่งไป
- บรรทัดที่ 6 เป็นการเก็บข้อมูลที่ได้จากการเรียกใช้ฟังก์ชั่น reportCourse ไว้ในตัวแปลชื่อ course

```
1   downloadfile() {
2  var excelfile = [[]];
3  for (let index = 0; index < this.course['students'].
      length; index++) {
4  excelfile[index] = this.course['students'][index];
5
6  for (let j = 1; j <= 16; j++) {
7  excelfile[index]['week' + j] = this.course['students
      '][index].weeks['week' + j].week;
8  }
9  excelfile[index]['course_id']=this.course['course_id
      '];
10 delete excelfile[index]['weeks']
11 }
12 var nameCoure = this.course['course_name'].split('
      ').shift() + เทอม"—:" + this.course['seson'] + "—sec—" + this.
      course['course_name'].split(' ').pop() + "—" + localStorage.getItem('
      name');
13 this.excelService.exportAsExcelFile(excelfile, nameCoure);
14 this.qruey()
15 }
```

รูปที่ 4.55: การสร้างหน้าจอส่วนหน้าดูประวัติการเข้าเรียน infoname.component.ts

จากภาพที่ 4.55 โครงสร้างของการสร้างหน้าจอดูประวัติการเข้าเรียน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ downloadfile
- บรรทัดที่ 2 เป็นการประกาศตัวแปลชื่อ excelfile สำหรับเก็บข้อมูล
- บรรทัดที่ 3-11 เป็นการเตรียมข้อมูลสำหรับการดาวน์โหลด
- บรรทัดที่ 12 เป็นการประกาศตัวแปล nameCoure เพื่อเก็บชื่อของไฟล์ที่จะดาวน์โหลด
- บรรทัดที่ 13 เป็นการเรียกใช้ฟังก์ชั่น exportAsExcelFile ในไฟล์ excel.service.ts โดยฟังก์ชั่นนี้จะทำหน้าที่แปลงข้อมูลอาเรย์ให้เป็นไฟล excel แล้วดาวน์โหลดไฟล์
- บรรทัดที่ 14 เป็นการเรียกใช้ฟังก์ชั่น qruey เพื่อดึงข้อมูลชั้นเรียนทั้งหมด

```
1  totlescore() {
2 var a = this.sumscore();
3 if (a!=null) {
4 this.studentService.updateStudent(this.id, a).
     subscribe();
5 }else{
6 alert('fire ไม่รงกัน')
7 }
8 }
```

รูปที่ 4.56: การสร้างหน้าจอส่วนหน้าดูประวัติการเข้าเรียน infoname.component.ts

จากภาพที่ 4.56 โครงสร้างของการสร้างหน้าจอดูประวัติการเข้าเรียน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ totlescore
- บรรทัดที่ 2 เป็นการประกาศตัวแปลชื่อ a สำหรับเก็บข้อมูลที่คืนค่ามาจากฟังก์ชั่น sumscore โดยฟังก์ชั่นจะคืนค่าข้อมูลการรวมคะแนนเป็นอาเรย์
- บรรทัดที่ 3-7 สร้างเงื่อนไขตรวจสอบว่า a มีข้อมูลหรือไม่ถ้ามีจะเรียกใช้ฟังก์ชั่น updateStudent โดยฟังก์ชั่นนี้จะอัพเดทคะแนนของนักศึกษา ถ้าไม่มีข้อมูลจะแสดงข้อความ fire ไม่รงกัน

4.2.7 โครงสร้างของการกรอกรหัสนักศึกษา

```
1  <div class="container">
2  <h5 class="text-centerกรอกรหัส"> นักศึกษา</h5>
3  <form>
4  <div class="container col-md-6">
5  <input type="text" class="form-control" #idstudent name="idstudent"
       placeholderกรอกรหัส
       นักศึกษา="" >
6  <br>
7  <button type="button" class="btn btn-primary" (click)="submid(
       idstudent.value)ตกลง"></button>
8  </div>
9  </form>
10 </div>
```

รูปที่ 4.57: การสร้างหน้าจอส่วนกรอกรหัสนักศึกษา fromstusent.component.html

จากภาพที่ 4.57 โครงสร้างของการสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าสร้างประกาศ สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 5 เป็นการสร้างช่องกรอกรหัสนักศึกษาเพื่อรับข้อมูลรหัสนักศึกษา
- บรรทัดที่ 7 เป็นการสร้างปุ่มเพื่อกดยืนยันรหัสนักศึกษาที่กรอก

```
1   submid(sutdent: any) {
2  var coure = localStorage.getItem('idcoure');
3  var weekc = localStorage.getItem('week');
4  var checkid = localStorage.getItem('id_active');
5
6  var between = this.getlocaltion();
7  if (between <= 150) {
8  this.checknamesad(sutdent, coure, weekc, checkid);
9  } else {
10 alertไม่สามารถเช็คชื่อได้("")
11 }
12 }
```

รูปที่ 4.58: การสร้างหน้าจอส่วนกรอกรหัสนักศึกษา fromstusent.component.html

จากภาพที่ 4.58 โครงสร้างของการสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าสร้างประกาศ สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชั่นชื่อ submid โดยฟังก์ชั่นนี้จะรับข้อมูลรหัสนักศึกษา
- บรรทัดที่ 2 เป็นการประกาศตัวแปลชื่อ coure สำหรับเก็บข้อมูลที่ดึงจาก localStorage
- บรรทัดที่ 3 เป็นการประกาศตัวแปลชื่อ weekc สำหรับเก็บข้อมูลที่ดึงจาก localStorage
- บรรทัดที่ 4 เป็นการประกาศตัวแปลชื่อ checkid สำหรับเก็บข้อมูลที่ดึงจาก localStorage
- บรรทัดที่ 6 เป็นการประกาศตัวแปลชื่อ between สำหรับเก็บข้อมูลที่คืนค่าจากฟังก์ชั่น getlocaltion โดยฟังก์ชั่นนี้จะคำนวนระยะห่างระหว่างอุปกรณ์ของนักศึกษาและอุปกรณ์ของอาจารย์
- บรรทัดที่ 7 เป็นการสร้างเงื่อนไขตรวจสอบระยะห่างของอุปกรณ์ระหว่างอาจารย์และนักศึกษา โดยห่างไม่เกิน 150 เมตร ถ้าไม่เกินจะเรียกใช้ฟังก์ชั่น checknamesad ซึ่งฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับเช็คชื่อและตรวจสอบข้อมูลว่านักศึกษาเช็คชื่อไปแล้วหรือไม่ ถ้าเกินระยะจะแสดงข้อความ ไม่สามารถเช็คชื่อได้

บทที่ 5

การทดสอบระบบ

การทดสอบการทำงานของเว็บแอปพลิเคชันระบบเช็คชื่อด้วยคิวอาร์โค้ด โดยทำการทดสอบในลักษณะ Black-box Testing [9] หรือ Data-Driven testing ซึ่งเป็นการเทสแบบที่ไม่สนใจโปรเซส (Process) การทำงานภายในของโปรแกรมว่าทำงานอย่างไร แต่จะเน้นไปที่ Input และ Result ที่ได้มากกว่าว่าการทำงานต่าง ๆ ถูกต้องตามความต้องการ (Requirement) หรือไม่ ซึ่งการทดสอบการใช้งานเว็บแอปพลิเคชัน ได้ผลดังนี้

5.1 การทดสอบการใช้งานเว็บแอปพลิเคชัน

5.1.1 การทดสอบหน้าสร้างชั้นเรียน

การทดสอบหน้าสร้างชั้นเรียนของเว็บแอปพลิเคชันในการสร้างชั้นเรียนของอาจารย์ ซึ่งการทำงานประกอบด้วย เมนูเลือกไฟล์ เมนูอัพโหลดไฟล์ และรายละเอียดของชั้นรียนที่อัพโหลด 5.1

ตารางที่ 5.1: ผลการทดสอบการสร้างชั้นเรียน

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
สร้างชั้นเรียน	กดปุ่มเมนูเลือกไฟล์	ระบบ แสดง ผล หน้า จอ เลือกไฟล์ที่จะอัพโหลด
	เลือก ไฟล์ excel ที่ ถูก เทม-เพลต	ระบบ แสดง ราย ละเอียด ข้อ-มูลของชั้นเรียน ถ้าไฟล์ไม่ถูกเทมเพลตระบบแสดงข้อความ ”ไฟล์ไม่ถูกต้อง”
	กดปุ่มเมนูเลือกเทอม	ระบบแสดงตัวเลือก 1 2 และ 3
	กดปุ่มเมนูอัพโหลดไฟล์	ระบบ จะ ทำการ อัพโหลด ไฟล์ไปที่ฐานข้อมูล และกลับไปที่หน้าหลัก

5.1.2 การทดสอบหน้าอัพเดทข้อมูลนักศึกษา

ในการแสดงผลหน้าอัพเดทข้อมูลนักศึกษานั้นจะประกอบไปด้วย รายละเอียดรายชื่อนักศึกษา เมนูเพิ่มนักศึกษา เมนูแก้ไขและเมนูลบ ผลการทดสอบดังตารางที่ 5.2

ตารางที่ 5.2: ผลการทดสอบหน้าอัพเดทข้อมูลนักศึกษา

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้าอัพเดทข้อมูลนักศึกษา	กดปุ่มเพิ่มนักศึกษา	ระบบ แสดง ผล หน้า สำหรับกรอกข้อมูลนักศึกษา
	กดปุ่มแก้ไข	ระบบ แสดง ผล หน้า สำหรับกรอกข้อมูลนักศึกษา
	กดปุ่มลบ	ระบบลบข้อมูลนักศึกษา
	กดปุ่มกลับ	ระบบแสดงผลหน้าจอหลัก

5.1.3 การทดสอบหน้าสร้างคิวอาร์โค้ด

ในการแสดงผลหน้าสร้างคิวอาร์โค้ดนั้นจะประกอบไปด้วย เมนูเลือกสัปดาห์ เมนูเลือกเวลาและเมนูสร้างคิวอาร์โค้ด ผลการทดสอบดังตารางที่ 5.3

ตารางที่ 5.3: ผลการการทดสอบหน้าสร้างคิวอาร์โค้ด

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
สร้างคิวอาร์โค้ด	กดปุ่มเมนูเลือกสัปดาห์	ระบบแสดงตัวเลือก 1-16 ซึ่งเป็นสัปดาห์ที่เช็คชื่อ
	กดปุ่มเมนูเลือกเวลา	ระบบแสดงตัวเลือก 5 10 15 และ 30 ซึ่งเป็นเวลาของคิวอาร์โค้ดที่เช็คชื่อ
	กดปุ่ม Start QR code	ระบบแสดงคิวอาร์โค้ดพร้อมกับ เวลา และ สัปดาห์ ของ คิวอาร์โค้ดที่ตั้งค่า

5.1.4 การทดสอบหน้าจัดการคะแนน

ในการแสดงผลหน้าจัดการคะแนนนั้นจะประกอบไปด้วย เมนูเลือกสัปดาห์ เมนูเลือกสถานะ รายชื่อและคะแนนนักศึกษา เมนูมา เมนูสายและเมนูขาด ผลการทดสอบดังตารางที่ 5.4

ตารางที่ 5.4: ผลการการทดสอบหน้าจัดการคะแนน

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
จัดการคะแนน	กดปุ่มเมนูเลือกสัปดาห์	ระบบแสดงตัวเลือก 1-16 ซึ่งเป็นสัปดาห์ที่เช็คชื่อ
	กดปุ่มเมนูเลือกสถานะ	ระบบแสดงตัวเลือก มา ขาด และ สาย ซึ่งเป็นสถานะของนักศึกษา
	กดปุ่มเมนูมา	ระบบส่งคะแนนไปเก็บในฐานข้อมูลและแสดงคะแนน
	กดปุ่มเมนูสาย	ระบบส่งคะแนนไปเก็บในฐานข้อมูลและแสดงคะแนน
	เมนูขาด	ระบบส่งคะแนนไปเก็บในฐานข้อมูลและแสดงคะแนน

5.1.5 การทดสอบหน้าดูประวัติการเข้าเรียน

ในการแสดงผลหน้าดูประวัติการเข้าเรียนนั้นจะประกอบไปด้วย รายชื่อและคะแนนนักศึกษาทั้งหมด เมนูดาวน์โหลดไฟล์และเมนูรวมไฟล์ ผลการทดสอบดังตารางที่ 5.5

ตารางที่ 5.5: ผลการการทดสอบหน้าดูประวัติการเข้าเรียน

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
ดูประวัติการเข้าเรียน	กดปุ่มดูประวัติจากหน้าหลัก	ระบบแสดงผลหน้าจอตารางรายชื่อและคะแนนของนักศึกษาทั้งหมดในชั้นเรียน
	กดปุ่มดาวน์โหลดไฟล์	ระบบทำการดาวน์โหลดรายชื่อและคะแนนของนักศึกษาทั้งหมดในชั้นเรียน โดยดาวน์โหลดเป็นไฟล์ excel
	กดปุ่มรวมไฟล์	ระบบแสดงผลหน้าจอเลือกไฟล์ที่จะอัพโหลด โดยไฟล์ต้องเป็นไฟล์ที่ดาวน์โหลดจากระบบซึ่งมีคิวอาร์โค้ด ถ้าไม่ใช่ระบบจะแสดงข้อความ ”ไฟล์ไม่ถูกต้อง” เมื่อเลือกไฟล์แล้วระบบจะแสดงรายละเอียดข้อมูลของชั้นเรียน
	กดปุ่ม save	ระบบทำการอัพเดทข้อมูลคะแนนแต่ละสัปดาห์และบันทึกลงในฐานข้อมูล
	กดปุ่มกลับ	ระบบแสดงผลหน้าจอหลัก

5.1.6 การทดสอบหน้ากรอกรหัสนักศึกษา

ในการแสดงผลหน้ากรอกรหัสนักศึกษานั้นจะประกอบไปด้วย ช่องกรอกข้อมูล เมนูตกลง ผลการทดสอบดังตารางที่ 5.6

ตารางที่ 5.6: ผลการการทดสอบหน้ากรอกรหัสนักศึกษา

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
กรอกรหัสนักศึกษา	สแกนคิวอาร์โค้ดของอาจารย์ก่อนหมดเวลา	ระบบ แสดง หน้า กรอก ข้อมูล สำหรับรหัสนักศึกษา
	สแกนคิวอาร์โค้ดของอาจารย์หลังหมดเวลา	ระบบ แสดง หน้า หมด เวลา สแกน
	กดปุ่มเมนูตกลง	ระบบ ทำการ ส่ง ข้อมูล รหัสนักศึกษา ไป ตรวจ สอบ และบันทึกคะแนนลงในฐานข้อมูล

บทที่ 6

สรุปและข้อเสนอแนะ

การดำเนินโครงงานเพื่อพัฒนาระบบเช็คชื่อด้วยคิวอาร์โค้ด นี้ พบว่าระบบสามารถทำงานได้ตามที่วิเคราะห์และออกแบบไว้ แต่ก็พบปัญหาและอุปสรรคระหว่างการพัฒนา ในบทนี้ผู้พัฒนาจึงขอสรุปความสามารถของระบบ ชี้แจงปัญหาและอุปสรรค พร้อมเสนอแนวทางในการพัฒนาระบบเช็คชื่อด้วยคิวอาร์โค้ด ต่อ ตามลำดับ

6.1 สรุปความสามารถของระบบ

ระบบเช็คชื่อด้วยคิวอาร์โค้ด เว็บแอปพลิเคชันสามารถสรุปความสามารถที่ระบบทำได้ดังนี้

6.1.1 เว็บแอปพลิเคชัน

ความสามารถหลักของเว็บแอปพลิเคชันนั้นเน้นสร้างความสะดวกต่อการจัดการเอกสารเรื่องข้อมูลต่างๆ ที่เกี่ยวข้องกับระบบเช็คชื่อด้วยคิวอาร์โค้ด โดยแบ่งความสามารถของระบบตามประเภทของผู้ใช้งานดังนี้

1. อาจารย์
    - ตรวจสอบรายชื่อนักศึกในคลาสเรียน
    - เพิ่ม และลบ คลาสเรียน
    - จัดการคะแนนของนักศึกษา
    - เพิ่ม ลบและแก้ไข้ ชื่อนักศึกษาในคลาสเรียนได้
    - รวมไฟล์คะแนน ของคลาสเรียนเดียวกัน
    - สร้างคิวอาร์โค้ด สำหรับให้นักศึกษาสแกนเพื่อเช็คชื่อได้
2. นักศึกษา
    - สแกนคิวอาร์โค้ดของผู้สอนเพื่อเป็นการเช็คชื่อเข้าเรียน

6.2 ปัญหาและอุปสรรคในการพัฒนา

1. เนื่องจากตัวระบบเป็นเว็บแอปพลิเคชันจึงไม่สามารถดึงค่า IP ของเครื่องคอมพิวเตอร์ของอาจารย์ได้จึงจำเป็นต้องตั้งค่าเลข IP ก่อนจะใช้งานระบบ
   แนวทางการแก้ไข : ทำเป็นโปรแกรมเดสก์ท็อปที่สามารถติดตั้งบนเครื่องคอมพิวเตอร์ ซึ่งสามารถดึงค่า IP ของเครื่องคอมพิวเตอร์มาใช้งานได้

6.3 แนวทางการพัฒนาต่อ

1. สร้าง Web server ของระบบซึ่งเป็นโปรแกรมที่มีหน้าที่ให้บริการด้านการจัดการเว็บไซต์ และ Database server ซึ่งเป็นโปรแกรมที่ทำหน้าที่ให้บริการด้านการจัดการดูแลข้อมูลต่าง ๆ ภายในเว็บไซต์
2. พัฒนาระบบให้เป็นโปรแกรมบนเดสก์ท็อป โดยสามารถเปิดใช้งานผ่านเครื่องคอมพิวเตอร์ได้เลยไม่จำเป็นต้องเข้าผ่านเว็บเบราว์เซอร์

บรรณานุกรม

[1] KimJaeHa (2549). เว็บแอพพลิเคชั่น [ออนไลน์]. สืบค้นเมื่อ 1 พฤษภาคม 2562. จาก https://medium.com/artisan-digital-agency .

[2] Google Inc. (2549). echo golang [ออนไลน์]. สืบค้นเมื่อ 1 พฤษภาคม 2562. จาก https://echo.labstack.com/ .

[3] mindphp. (2555). Javascript คืออะไร [ออนไลน์]. สืบค้นเมื่อ 10 พฤษภาคม 2562. จาก https://goo.gl/FAeTb2 .

[4] Google Inc. (2010). Angular cli [ออนไลน์]. สืบค้นเมื่อ 3 พฤษภาคม 2562. จาก https://angular.io/ .

[5] MongoDB Inc. (2009). ฐานข้อมูล mongodb [ออนไลน์]. สืบค้นเมื่อ 2 พฤษภาคม 2562. จาก https://www.mongodb.com/ .

[6] Jstoppen (2013). plickers [ออนไลน์]. สืบค้นเมื่อ 20 ธันวาคม 2562. จาก https://www.plickers.com/library .

[7] Student Care Co., Ltd.. (2556). ระบบ student care [ออนไลน์]. สืบค้นเมื่อ 20 ธันวาคม 2562. จาก http://www.student.co.th/ .

[8] Kunchit Phiu-Nual. (2557). ความหมายและความสำคัญของ system architecture [ออนไลน์]. สืบค้นเมื่อ 12 พฤษภาคม 2562. จาก https://goo.gl/6ZhGQo .

[9] Atthaboon S. (2555). Black-box testing strategy [ออนไลน์]. สืบค้นเมื่อ 20 พฤษภาคม 2562. จาก http://everybitsconsult.com/blog/2015/06/22/black-box-testing.html .

ภาคผนวก

ภาคผนวก ก

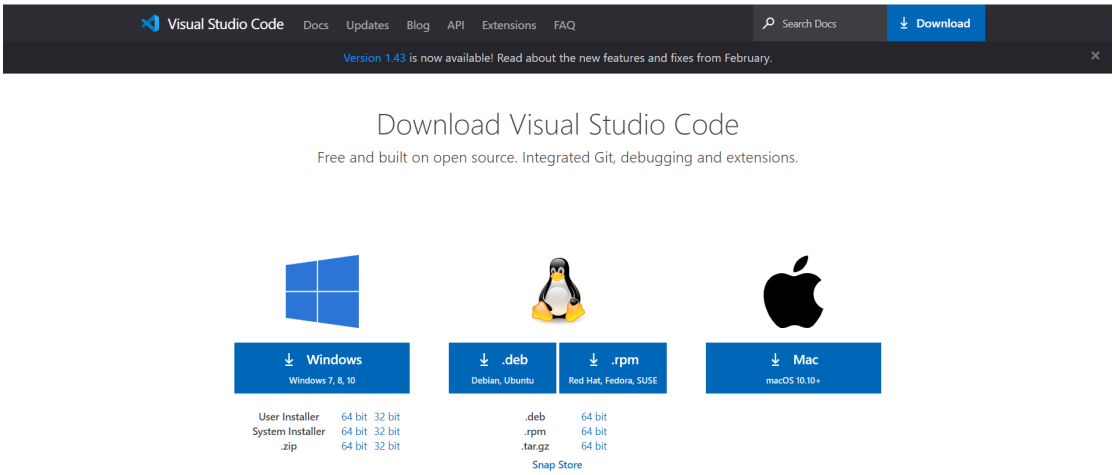
การติดตั้งเครื่องมือที่ใช้พัฒนาโปรแกรม

การติดตั้งเครื่องมือที่ใช้ในการพัฒนาเว็บแอปพลิเคชันระบบเช็คชื่อด้วยคิวอาร์โค้ด มีโปรแกรมที่จำเป็นในการพัฒนาระบบดังต่อไปนี้

- การติดตั้ง Visual Studio Code
- การติดตั้ง Go
- การติดตั้ง Angular Framwork

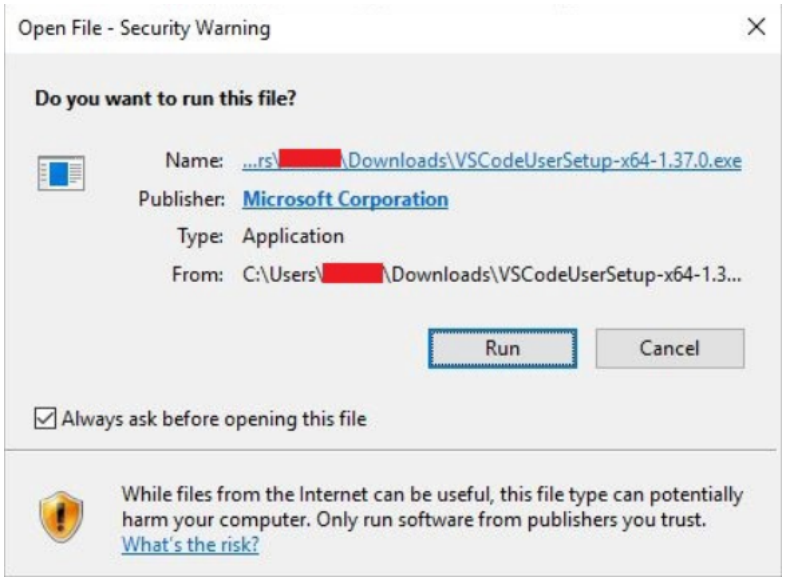
ก.1 การติดตั้ง Visual Studio Code

1. สามารถดาวน์โหลด Visual Studio Code ได้ที่ https://code.visualstudio.com/download ดังแสดงในรูปที่ ก.1



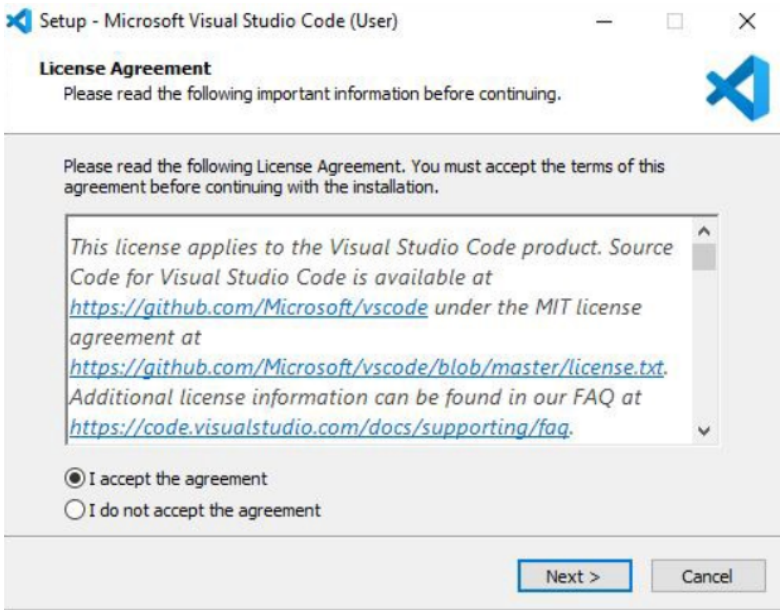
รูปที่ ก.1: หน้าเว็บดาวน์โหลด Visual Studio Code

2. แสดงหน้าต่างต้อนรับของ Visual Studio Code ทำการกด Run เพื่อเริ่มกระบวนการติดตั้ง ดังแสดงในรูปที่ ก.2



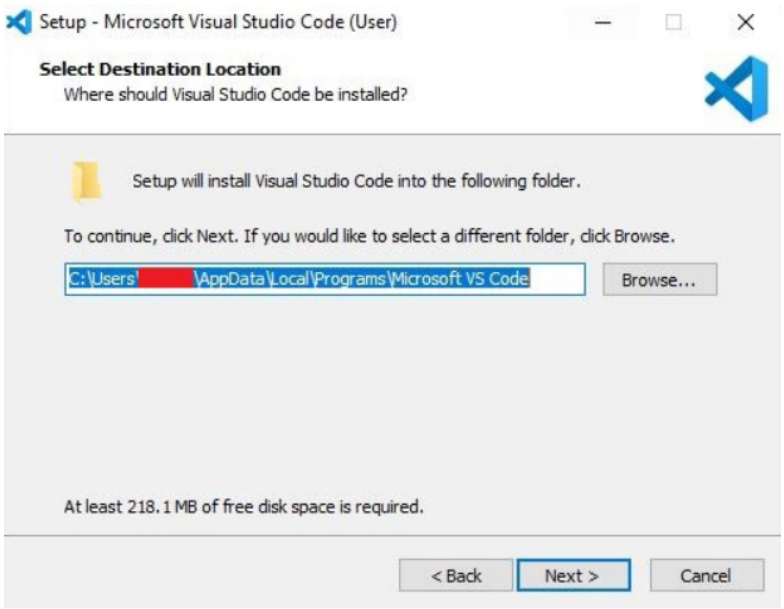
รูปที่ ก.2: หน้าต่างต้อนรับของ Visual Studio Code

3. แสดงหน้าต่างข้อตกลงการใช้งาน Visual Studio Code เลือก “I accept the agreement” และคลิกปุ่ม Next > ดังแสดงในรูปที่ ก.3



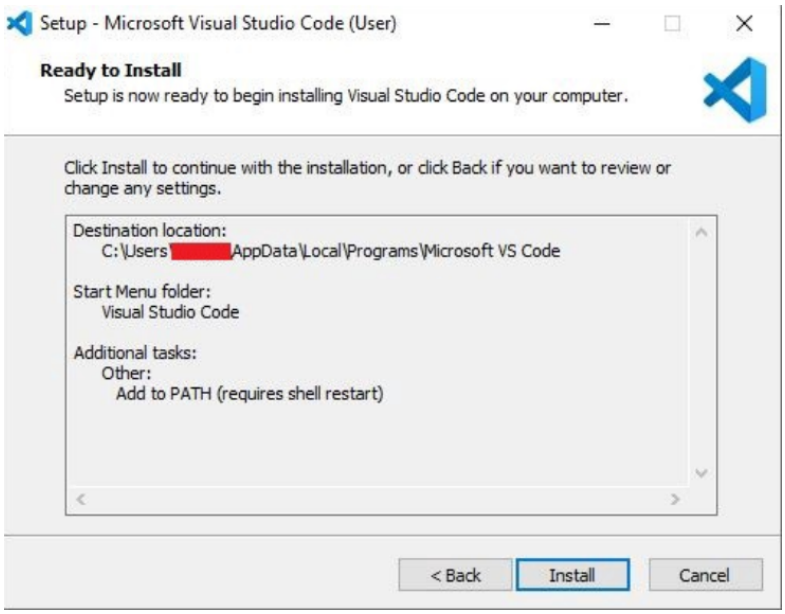
รูปที่ ก.3: หน้าต่างข้อตกลงการใช้งาน Visual Studio Code

4. แสดงหน้าต่างที่จัดเก็บไฟล์ต่างๆ ของ Visual Studio Code ทำการกด Next ดังแสดงในรูปที่ ก.4



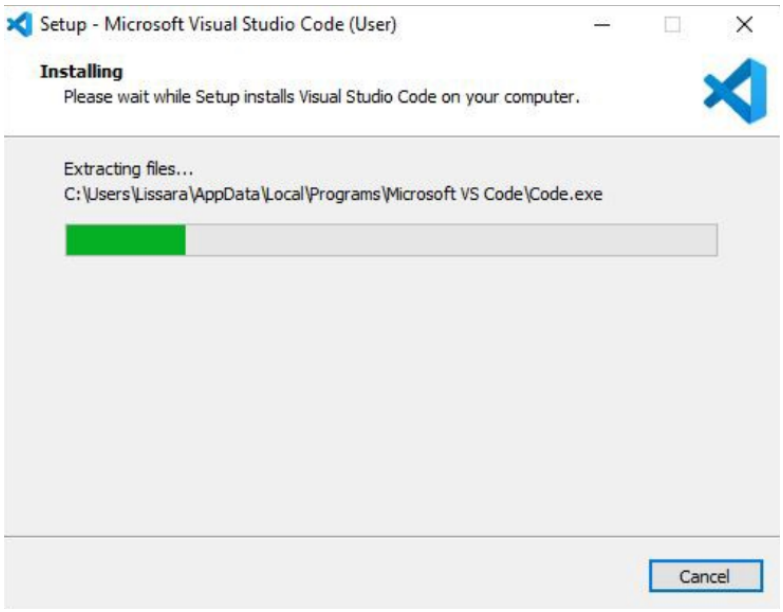
รูปที่ ก.4: หน้าต่างที่จัดเก็บไฟล์ต่างๆ ของ Visual Studio Code

5. หน้าต่างเริ่มทำการติดตั้งทำการกด Install ดังแสดงในรูปที่ ก.5



รูปที่ ก.5: หน้าต่างที่จัดเก็บไฟล์ต่างๆ ของ Visual Studio Code

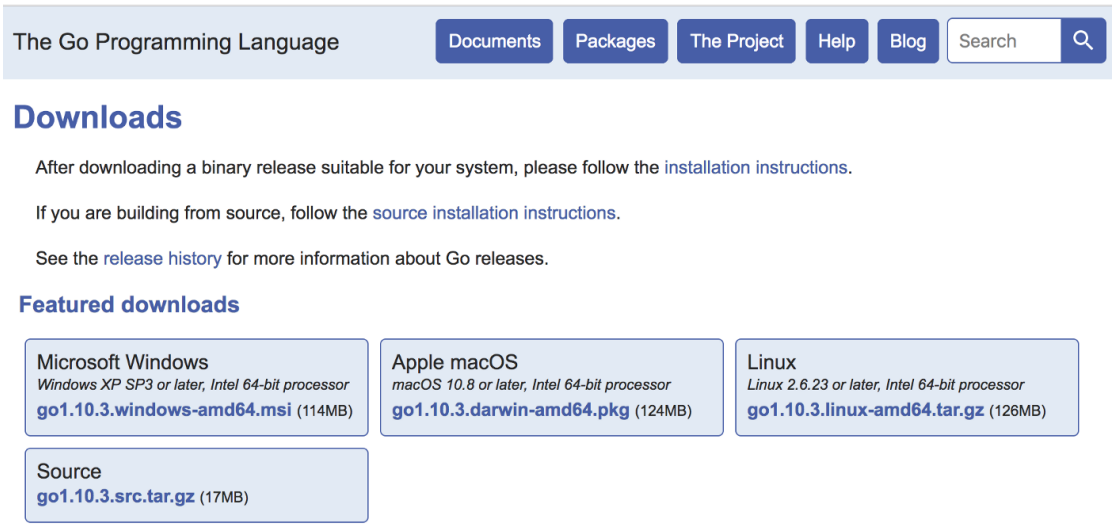
6. หน้าต่างผลการติดตั้ง Visual Studio Code ดังแสดงในรูปที่ ก.6



รูปที่ ก.6: หน้าต่างผลการติดตั้ง Visual Studio Code

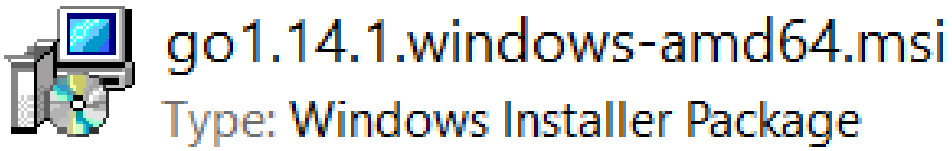
ก.2 การติดตั้ง Go

1. สามารถดาวน์โหลด Go installer package ได้ที่ https://golang.org/dl/ ดังแสดงในรูปที่ ก.7



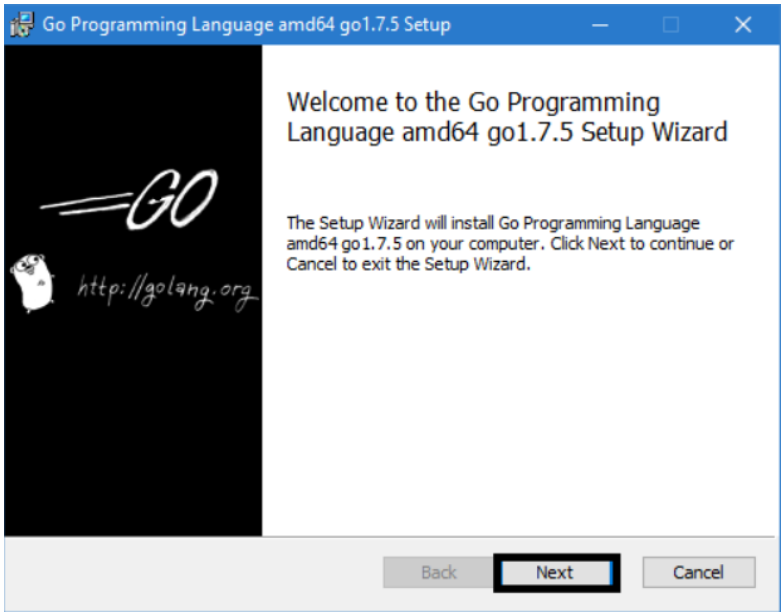
รูปที่ ก.7: หน้าเว็บดาวน์โหลด Go

2. เปิดไฟล์ติดตั้ง ชื่อ go1.14.1.windows-amd64.msi เพื่อทำการติดตั้ง ดังแสดงในรูปที่ ก. 8



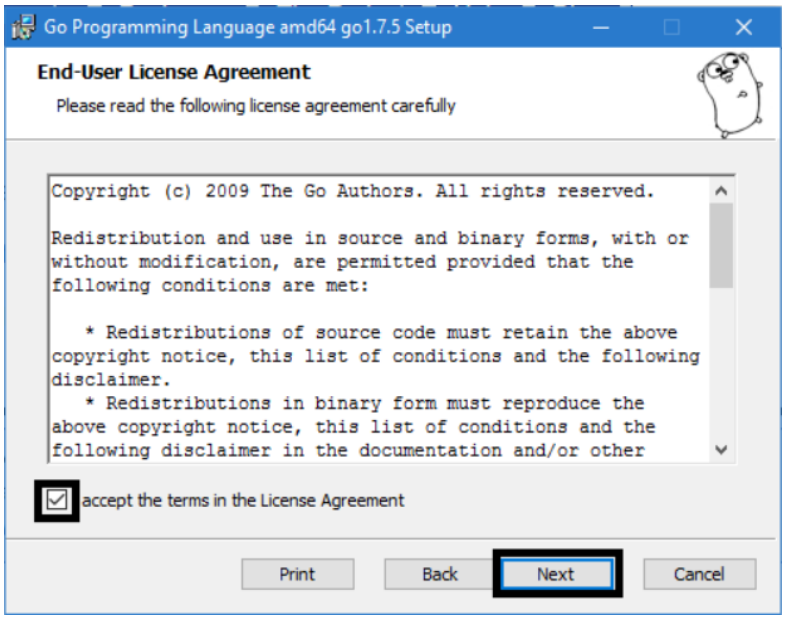
รูปที่ ก.8: ไฟล์ติดตั้งสำหรับติดตั้ง Go

3. แสดงหน้าต่างต้อนรับของ Go ทำการกด Next เพื่อเริ่มกระบวนการติดตั้ง ดังแสดงในรูปที่ ก.9



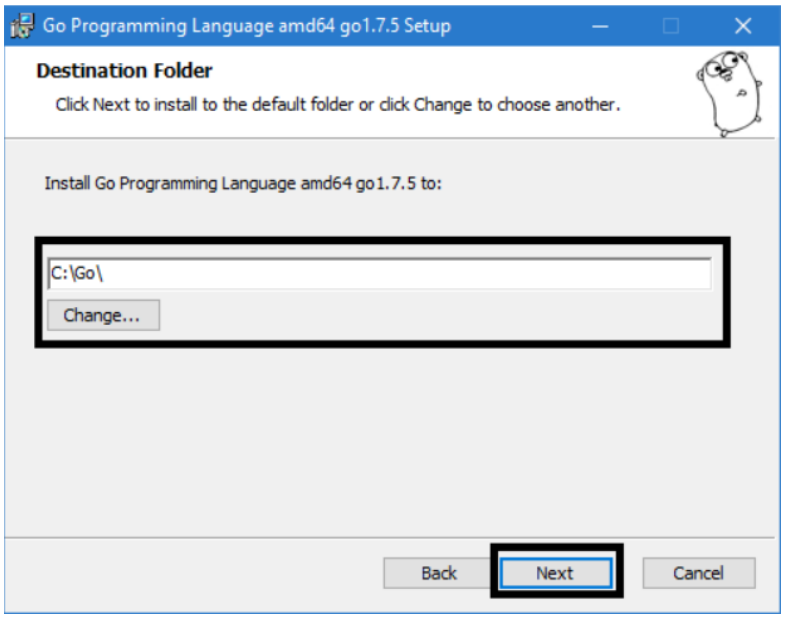
รูปที่ ก.9: หน้าต่างตอนรับของ Go

4. แสดงหน้าต่างข้อตกลงในการใช้ Go ให้เลือกช่อง I accept the terms in the License Agreement และกด Next ดังแสดงในรูปที่ ก.10



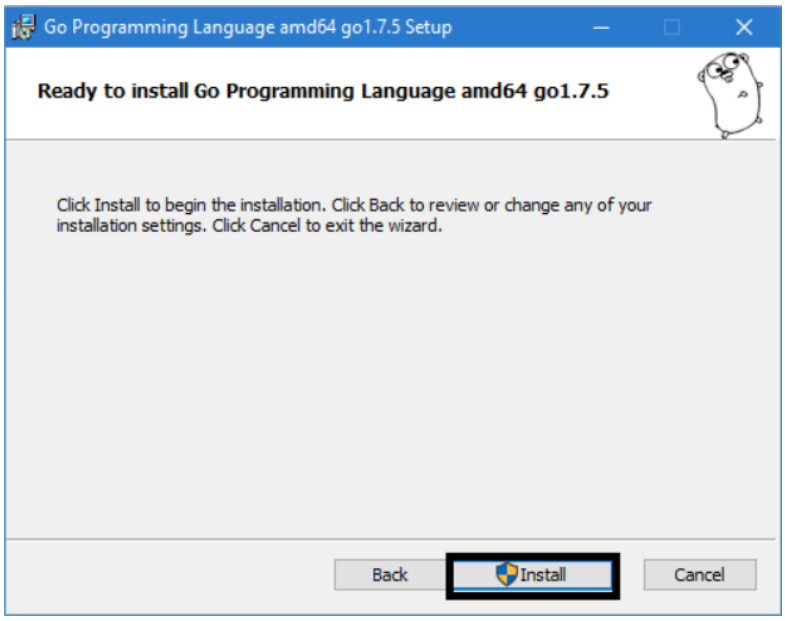
รูปที่ ก.10: หน้าต่างข้อตกลงในการใช้ Go

5. แสดงหน้าต่างเลือกโฟลเดอร์ที่จะทำการติดตั้ง ดังแสดงในรูปที่ ก.11



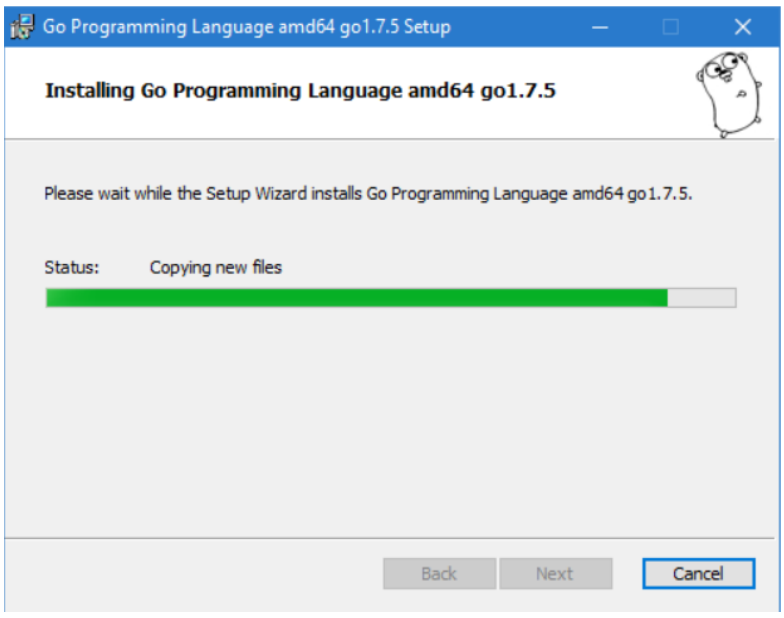
รูปที่ ก.11: หน้าต่างเลือกโฟลเดอร์ที่จะทำการติดตั้ง Go

6. แสดงหน้าต่างสำหรับติดตั้ง Go ทำการกด Install เพื่อทำการติดตั้ง ดังแสดงในรูปที่ ก.12



รูปที่ ก.12: หน้าต่างติดตั้ง Go

7. หน้าต่างผลการติดตั้ง Go ดังแสดงในรูปที่ ก.13



รูปที่ ก.13: หน้าต่างผลการติดตั้ง Go

ก.3 การติดตั้ง Angular Framwork

การติดตั้ง Angular Framwork สามารถทำผ่านคำสั่ง command line ได้ โดยจำเป็นต้องทำการติดตั้ง Node.js ก่อนเพื่อใช้ในกระบวนการติดตั้งนี้ ดังแสดงในรูปที่ ก.14

```
npm install -g @angular/cli
```

รูปที่ ก.14: คำสั่งสำหรับติดตั้ง Angular Framwork

ผลการติดตั้ง Angular Framwork ดังแสดงในรูปที่ ก.15

```
phatchaphansMBP:project phatpan$ npm install -g @angular/cli
/Users/phatpan/npm/bin/ng -> /Users/phatpan/npm/lib/node_modules/@angular/cli/bin/ng
/Users/phatpan/npm/lib
└── @angular/cli@1.0.3
```

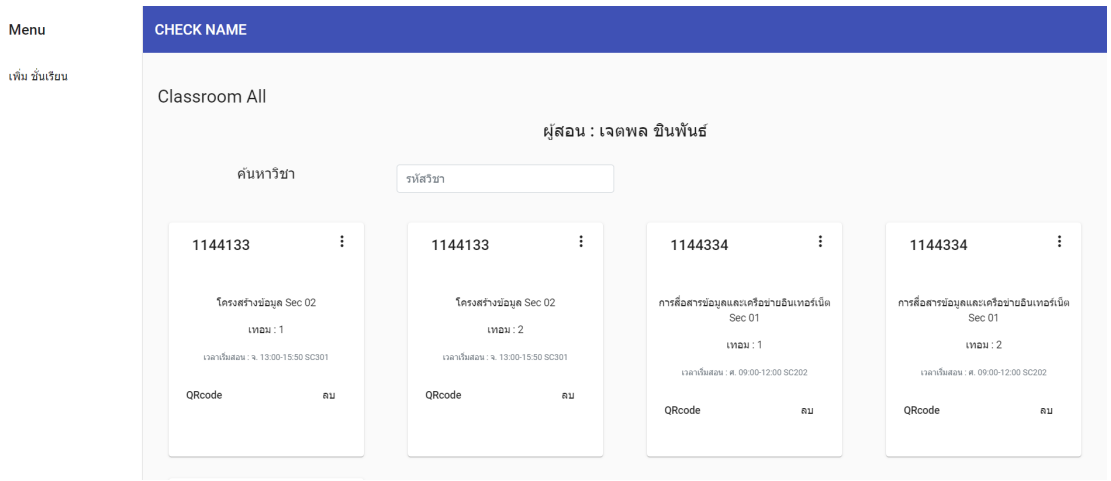
รูปที่ ก.15: หน้าต่างผลการติดตั้ง

ภาคผนวก ข

คู่มือการติดตั้งระบบ

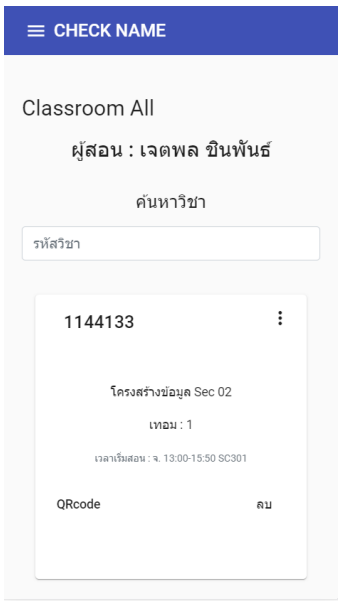
เนื่องด้วยระบบเช็คชื่อด้วยคิวอาร์โค้ดเป็นเว็บแอปพลิเคชั่น ถูกพัฒนาบนพื้นฐานการทำงานบนเว็บไซต์รองรับการใช้งานทั้งบนคอมพิวเตอร์และโทรสัพท์มือถือ ทำให้ผู้ใช้สามารถเข้าใช้งานผ่านเว็บบราวเซอร์ได้โดยไม่จำเป็นต้องทำการติดตั้งเพื่อใช้งาน

1. หน้าเว็บบราวเซอร์สำหรับหน้าจอคอมพิวเตอร์ ดังแสดงในรูปที่ ข.1



รูปที่ ข.1: หน้าเว็บแอปพลิเคชั่น ระบบเช็คชื่อด้วยคิวอาร์โค้ดมุมมองของคอมพิวเตอร์

2. หน้าเว็บบราวเซอร์สำหรับหน้าจอโทรสัพท์มือถือ ดังแสดงในรูปที่ ข.2



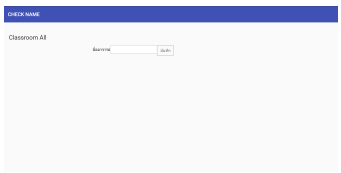
รูปที่ ข.2: หน้าเว็บแอปพลิเคชั่น ระบบเช็คชื่อด้วยคิวอาร์โค้ดมุมมองของโทรสัพท์มือถือ

ภาคผนวก ค

คู่มือการใช้งานระบบ

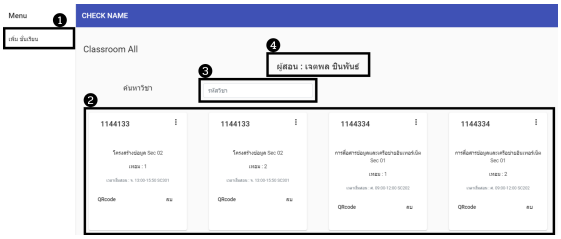
คู่มือการใช้งานทั้งหมดของระบบเช็คชื่อด้วยคิวอาร์โค้ด เบื่องต้น

1. ส่วนของหน้าเมนูเว็บแอปพลิเคชันสำหรับอาจารย์
   - หน้าจอหลักแสดงผลครั้งแรกเมื่อผู้ใช้ทำการเปิดใช้งานเว็บแอปพลิเคชัน ดังแสดงในรูปที่ ค.1



รูปที่ ค.1: หน้าจอหลักเมื่อเข้าใช้งานครั้งแรก

   - เมื่อทำการกรอกชื่อในระบบและกดตกลง ระบบจะทำการบันทึกชื่อลงใน localStorage จากนั่นระบบจะแสดงข้อมูลชั้นเรียนที่มีทั้งหมด ดังแสดงในรูปที่ ค.2

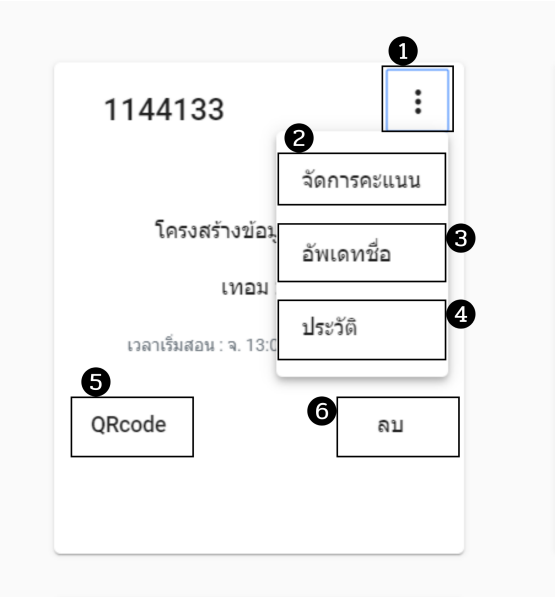


รูปที่ ค.2: หน้าหลัก

จากรูปที่ ค.2 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ ปุ่มสำหรับการไปยังหน้าเพิ่มชั้นเรียน
- หมายเลข 2 คือ แสดงชั้นเรียนทั้งหทดที่มีในฐานข้อมูล
- หมายเลข 3 คือ ช่องกรอกข้อมูลสำหรับค้นหาชั้นเรียนโดยกรอกรหัสวิชา
- หมายเลข 4 คือ แสดงชื่อของผู้ใช้งานระบบ

- ข้อมูลส่วนหนึ่งที่แสดงในหน้าหลัก เป็นส่วนการ์ดที่ใช้ในการเข้าถึงการจัดการข้อมูลต่างๆในชั้นเรียน ดังแสดงในรูปที่ ค.3

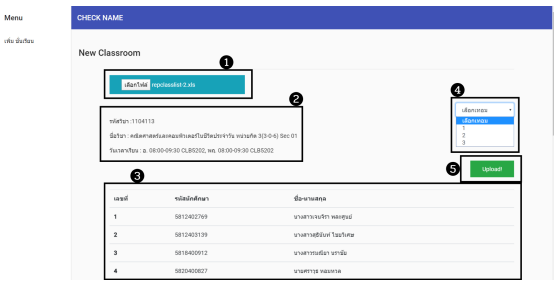


รูปที่ ค.3: การ์ดแสดงข้อมูลชั้นเรียน

จากรูปที่ ค.3 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ ปุ่มเป็นตัวเลือกแสดงตัวเลือกเพิ่มเติม
- หมายเลข 2 คือ ปุ่มสำหรับการไปยังหน้าจัดการคะแนน
- หมายเลข 3 คือ ปุ่มสำหรับการไปยังหน้าอัพเดทชื่อ
- หมายเลข 4 คือ ปุ่มสำหรับการไปยังหน้าประวัติ
- หมายเลข 5 คือ ปุ่มสำหรับการไปยังหน้าสร้างคิวอาร์โค้ด
- หมายเลข 6 คือ ปุ่มสำหรับลบชั้นเรียน

- เมื่อผู้ใช้กดปุ่ม เพิ่มชั้นเรียน ระบบจะแสดงหน้าเพิ่มชั้นเรียนซึ่งเป็นหน้าสำหรับเพิ่มชั้นเรียนใหม่ ดังแสดงในรูปที่ ค.4

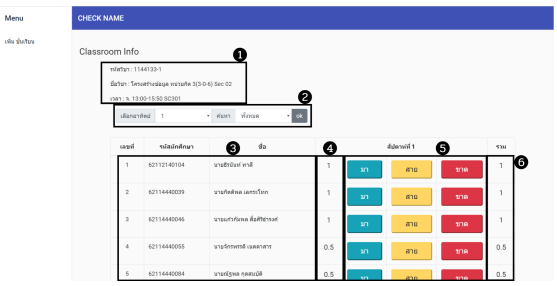


รูปที่ ค.4: หน้าเพิ่มชั้นเรียน

จากรูปที่ ค.4 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ ช่องสำหรับเลือกไฟล์ข้อมูลชั้นเรียนที่จะทำการเพิ่ม
- หมายเลข 2 คือ พื้นที่แสดงข้อมูลของชั้นเรียน
- หมายเลข 3 คือ ตารางแสดงรายชื่อทั้งหมดในชั้นเรียน
- หมายเลข 4 คือ ปุ่มแสดงตัวเลือกสำหรับเลือกเทอมการศึกษา
- หมายเลข 5 คือ ปุ่มสำหรับอัพโหลดข้อมูลชั้นเรียน

- เมื่อผู้ใช้กดปุ่ม จัดการคะแนน ระบบจะแสดงหน้าจัดการคะแนนนักศึกษาซึ่งเป็นหน้าสำหรับจัดการคะแนนของนักศึกษาในชั้นเรียน ดังแสดงในรูปที่ ค.5



รูปที่ ค.5: หน้าจัดการคะแนน

จากรูปที่ ค.5 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ พื้นที่แสดงข้อมูลของชั้นเรียน
- หมายเลข 2 คือ ปุ่มตัวเลือกสำหรับเลือกเพื่อค้นหาข้อมูล

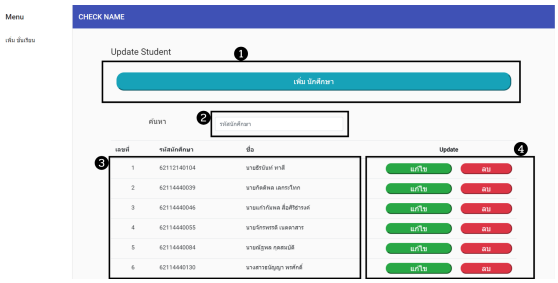
– หมายเลข 3 คือ ตารางแสดงรายชื่อทั้งหมดในชั้นเรียน

– หมายเลข 4 คือ แสดงคะแนนในอาทิตนั้นๆ

– หมายเลข 5 คือ ปุ่มสำหรับให้คะแนนนักศึกษา

– หมายเลข 6 คือ แสดงคะแนนรวมทั้งหมด

- เมื่อผู้ใช้กดปุ่ม อัพเดทชื่อ ระบบจะแสดงหน้าอัพเดทชื่อซึ่งเป็นหน้าสำหรับ เพิ่ม ลบ และแก้ไข ชื่อและรหัสนักศึกษา ดังแสดงในรูปที่ ค.6



รูปที่ ค.6: หน้าอัพเดทชื่อ

จากรูปที่ ค.6 สามารถอธิบายการใช้งานได้ดังนี้

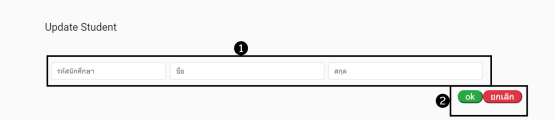
– หมายเลข 1 คือ ปุ่มสำหรับเพิ่มนักศึกษา

– หมายเลข 2 คือ ช่องกรอกข้อมูลสำหรับค้นหานักศึกษาโดยกรอกรหัสนักศึกษา

– หมายเลข 3 คือ ตารางแสดงรายชื่อทั้งหมดในชั้นเรียน

– หมายเลข 4 คือ ปุ่มสำหรับแก้ไขและลบ นักศึกษา

- เมื่อผู้ใช้กดปุ่ม เพิ่มนักศึกษา ระบบจะแสดงหน้าสำหรับกรอกข้อมูลนักศึกษา ดังแสดงในรูปที่ ค.7



รูปที่ ค.7: หน้าเพิ่มนักศึกษา

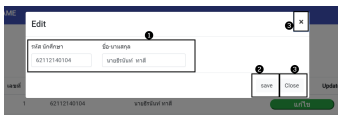
จากรูปที่ ค.7 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ ช่องสำหรับกรอกข้อมูลนักศึกษา

– หมายเลข 2 คือ ปุ่มตกลง สำหรับบันทึกข้อมูล

– หมายเลข 2 คือ ปุ่มยกเลิก สำหรับกลับสู่หน้าอัพเดทชื่อ

- เมื่อผู้ใช้กดปุ่ม แก้ไข ระบบจะแสดงหน้าสำหรับกรอกข้อมูลนักศึกษา ดังแสดงในรูปที่ ค.8



รูปที่ ค.8: หน้าจอต้อนรับ

จากรูปที่ ค.8 สามารถอธิบายการใช้งานได้ดังนี้

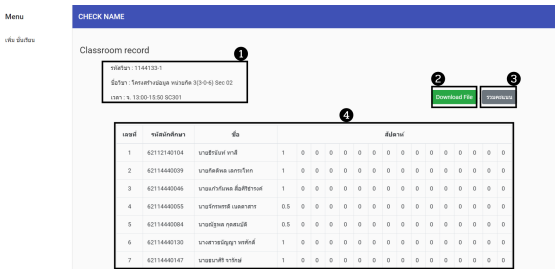
– หมายเลข 1 คือ ช่องสำหรับกรอกข้อมูลนักศึกษา

– หมายเลข 2 คือ ปุ่ม save สำหรับบันทึกข้อมูล

– หมายเลข 3 คือ ปุ่ม close สำหรับกลับสู่หน้าอัพเดทชื่อ

– หมายเลข 3 คือ ปุ่ม X สำหรับกลับสู่หน้าอัพเดทชื่อ

- เมื่อผู้ใช้กดปุ่ม ประวัติ ระบบจะแสดงหน้าประวัติซึ่งเป็นหน้าสำหรับแสดงข้อมูลของชั้นเรียนทั้งหมด ดังแสดงในรูปที่ ค.9



รูปที่ ค.9: หน้าประวัติ

จากรูปที่ ค.9 สามารถอธิบายการใช้งานได้ดังนี้

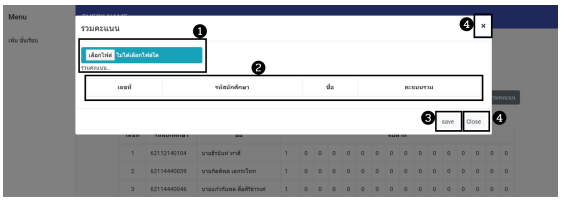
– หมายเลข 1 คือ พื้นที่แสดงข้อมูลของชั้นเรียน

– หมายเลข 2 คือ ปุ่มดาวน์โหลด เป็นปุ่มสำหรับดาวน์โหลดข้อมูลชั้นเรียน โดยจะดาวน์โหลดเป็นไฟล์ excel

– หมายเลข 3 คือ ปุ่มรวมคะแนน

– หมายเลข 4 คือ ตารางแสดงรายชื่อทั้งหมดในชั้นเรียน

- เมื่อผู้ใช้กดปุ่ม รวมคะแนน ระบบจะแสดงหน้ารวมคะแนน ดังแสดงในรูปที่ ค.10



รูปที่ ค.10: หน้ารวมคะแนน

จากรูปที่ ค.10 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ ช่องสำหรับเลือกไฟล์ข้อมูลชั้นเรียนที่จะทำการรวมคะแนนโดยจะต้องเป็นไฟล์ที่ดาวน์โหลดจะระบบเท่านั่น
- หมายเลข 2 คือ ตารางแสดงรายชื่อทั้งหมดในชั้นเรียน ของไฟล์ที่เลือก
- หมายเลข 3 คือ ปุ่ม save สำหรับบันทึกข้อมูล
- หมายเลข 4 คือ ปุ่ม close สำหรับกลับสู่หน้าอัพเดทชื่อ
- หมายเลข 4 คือ ปุ่ม X สำหรับกลับสู่หน้าอัพเดทชื่อ

- เมื่อผู้ใช้กดปุ่ม QRcode ระบบจะแสดงหน้าสร้างคิวอาร์โค้ดซึ่งเป็นหน้าสำหรับสร้างคิวอาร์โค้ดให้นักศึกษาสแกน ดังแสดงในรูปที่ ค.11

รูปที่ ค.11: หน้าสร้างคิวอาร์โค้ด

จากรูปที่ ค.11 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ ปุ่มตัวเลือกสำหรับตั้งค่าสัปดาห์และเวลาของคิวอาร์โค้ด
- หมายเลข 2 คือ ปุ่ม Start QR code เปนปุ่มสำหรับสร้างคิวอาร์โค้ด

- เมื่อผู้ใช้กดปุ่ม Start QR code ระบบจะแสดงคิวอาร์โค้ด ดังแสดงในรูปที่ ค.12

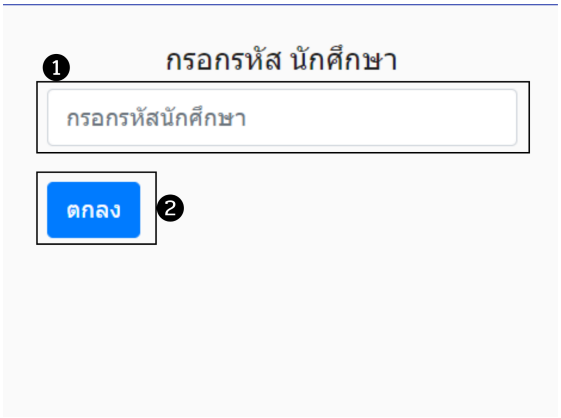
รูปที่ ค.12: หน้าแสดงคิวอาร์โค้ด

จากรูปที่ ค.12 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ แสดงสัปดาห์ของคิวอาร์โค้ด
- หมายเลข 2 คือ แสดงคิวอาร์โค้ด
- หมายเลข 3 คือ แสดงเวลาของคิวอาร์โค้ด

2. ส่วนของหน้าเมนูเว็บแอปพลิเคชันสำหรับนักศึกษา

- เมื่อผู้ใช้ทำการสแกนคิวอาร์โค้ดสำเร็จ ระบบจะแสดงหน้ากรอกรหัสนักศึกษา ดังแสดงในรูปที่ ค.13

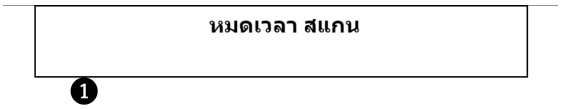


รูปที่ ค.13: หน้ากรอกรหัสนักศึกษา

จากรูปที่ ค.13 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ ช่องสำหรับกรอกรหัสนักศึกษาเพื่อยืนยันตัวตน
- หมายเลข 2 คือ ปุ่มตกลง สำหรับยืนยัน

- เมื่อผู้ใช้ทำการสแกนคิวอาร์โค้ดไม่สำเร็จ ระบบจะแสดงหน้าหมดเวลาสแกน ดังแสดงในรูปที่ ค.14



รูปที่ ค.14: หน้าสแกนคิวอาร์โค้ดไม่สำเร็จ

จากรูปที่ ค.14 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ แสดงข้อความ หมดเวลาสแกน

ประวัติผู้เขียน

ชื่อ-สกุล: นายเจตพล ชินพันธ์

รหัสประจำตัวนักศึกษา: 59110440112

วัดเกิด: 09 06 2540

ที่อยู่ที่สามารถติดต่อได้: 11 หมู่1 ตำบลหนองแค อำเภอราษีไศล จังหวัดศรีสะเกษ 33160

เบอร์โทรศัพท์: (+66) 0660643497

อิเมลล์: jadtaphon.ch.59@ubu.ac.th

ระดับมัธยมต้น: โรงเรียนราษีไศล จังหวัดศรีสะเกษ

ระดับมัธยมปลาย: โรงเรียนราษีไศล จังหวัดศรีสะเกษ

ระดับอุดมศึกษา: ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ สาขาวิทยาการ คอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี