

ระบบเช็คชื่อด้วยคิวอาร์โค้ด
QR Code check name

นายเจตพล ชินพันธ์

โครงงานนี้เป็นส่วนหนึ่งของการศึกษา
หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ คณะวิทยาศาสตร์
มหาวิทยาลัยอุบลราชธานี
ปีการศึกษา 2562
ลิขสิทธิ์ของมหาวิทยาลัยอุบลราชธานี

โครงการ : ระบบเช็คชื่อด้วยคิวอาร์โค้ด
QR Code check name
โดย : นายเจตพล ชินพันธ์
อาจารย์ที่ปรึกษา : ดร. เกรียงศักดิ์ ตรีประไพณ
ระดับการศึกษา : วิทยาศาสตร์บัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ปีการศึกษา : 2562

ได้รับการพิจารณาให้เป็นส่วนหนึ่งของการศึกษา
ตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์

คณะกรรมการสอบประเมินความรู้โครงการคอมพิวเตอร์

..... อาจารย์ที่ปรึกษา
(ดร. เกรียงศักดิ์ ตรีประไพณ)

..... กรรมการ
(ดร.ทศพร จูนิม)

..... กรรมการ
(ดร.ไพชยนต์ คงไชย)

..... หัวหน้าภาควิชา
(ผศ.ดร. สุพจน์ สืบตร)

วันที่ ... / ... / ...

กิตติกรรมประกาศ

การพัฒนาโครงการระบบเชื้อชื้อด้วยคิวอาร์โค้ด สำเร็จลุล่วงได้ด้วยความกรุณาและความช่วยเหลือจากหลายๆ ท่าน ข้าพเจ้าขอขอพระคุณทุกท่าน ที่มีส่วนร่วมในการพัฒนาโครงการนี้

ขอขอบพระคุณอาจารย์ที่ปรึกษา ดร.เกรียงศักดิ์ ตรีประพิน อาจารย์ที่ปรึกษาโครงการที่ได้แนะนำทฤษฎีและแนวทางในแก้ปัญหาต่าง ๆ ที่เกิดขึ้นระหว่างการพัฒนา ระบบ อีกครั้งยังคอยตรวจสอบความก้าวหน้าของการทำงานเป็นระยะ ๆ รวมทั้งสร้างกำลังใจให้ผู้พัฒนาอยู่เสมอ

ขอขอบพระคุณเจ้าหน้าที่ภาควิชาคณิตศาสตร์ สถิติและคอมพิวเตอร์ ที่คอยเอื้ออำนวยความสะดวกทั้งเรื่องอุปกรณ์และสถานที่ต่อการปฏิบัติงานของผู้พัฒนา

ขอกราบขอบพระคุณบิดา มารดา ที่คอยให้กำลังใจ คอยให้ความรักและความห่วงใยเสมอมา ตลอดจนคอยช่วยเหลือทุนทรัพย์ทางด้านการศึกษาและอุปกรณ์ในการพัฒนาโครงการ

นายเจตพล ชินพันธ์

22 เมษายน 63

| | | |
|------------------|---|--|
| โครงการ | : | ระบบเช็คชื่อด้วยคิวอาร์โค้ด |
| โดย | : | นายเจตพล ชินพันธ์ |
| อาจารย์ที่ปรึกษา | : | ดร. เกรียงศักดิ์ ตริประพัฒน์ |
| ระดับการศึกษา | : | วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ |
| ปีการศึกษา | : | 2562 |

บทคัดย่อ

โครงการนี้มีวัตถุประสงค์เพื่อพัฒนาการเรียนการสอนโดยสร้างเว็บแอปพลิเคชันด้วยแองกิวลา เพื่อเช็คชื่อในการเข้าเรียนซึ่งเป็นส่วนสำคัญในการให้คะแนนเสริมช่วยนักศึกษา และพัฒนานักศึกษาเกี่ยวกับวินัยในการมาเรียนให้ตรงเวลา โดยในเว็บแอปพลิเคชันนี้จะเป็นการจัดการข้อมูลคะแนนการมาเรียนของนักศึกษาซึ่งระบบจะมุ่งเน้นความรวดเร็วในการใช้งานและการเก็บข้อมูล นอกจากนี้ยังมีการนำเทคโนโลยีคิวอาร์โค้ด มาใช้ในระบบ โดยการให้อาจารย์เป็นผู้สร้างคิวอาร์โค้ด และให้นักศึกษาเป็นผู้สแกนซึ่งนักศึกษาเพียงแคสแกนคิวอาร์โค้ด ของอาจารย์และกรอกรหัสนักศึกษาเพื่อยืนยันตัวตน

คำสำคัญ: เว็บแอปพลิเคชัน เช็คชื่อเข้าเรียนด้วยคิวอาร์โค้ด

Topic : QR Code check name
Author : 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 |
|--|------|
| กิตติกรรมประกาศ | ค |
| บทคัดย่อภาษาไทย | ง |
| บทคัดย่อภาษาอังกฤษ | จ |
| สารบัญ | ฉ |
| สารบัญตาราง | ณ |
| สารบัญภาพ | ญ |
| บทที่ | |
| 1 บทนำ | 1 |
| 1.1 ที่มาและเหตุผล | 1 |
| 1.2 วัตถุประสงค์ | 2 |
| 1.3 ขอบเขตของโครงการ | 2 |
| 1.4 ประโยชน์ที่คาดว่าจะได้รับ | 2 |
| 1.5 เครื่องมือที่ใช้ในการพัฒนา (Development tools) | 3 |
| 1.5.1 ฮาร์ดแวร์ | 3 |
| 1.5.2 ซอฟต์แวร์ (Software) | 3 |
| 1.5.3 แผนการดำเนินการ | 4 |
| 2 ทฤษฎีที่เกี่ยวข้อง | 5 |
| 2.1 ความรู้พื้นฐานของเว็บแอปพลิเคชัน | 5 |
| 2.1.1 ความสำคัญของเว็บแอปพลิเคชัน | 5 |
| 2.1.2 ส่วนประกอบของเว็บแอปพลิเคชัน | 6 |
| 2.2 ความรู้พื้นฐานภาษาโก (Go) | 8 |
| 2.2.1 ชุดคำสั่งในภาษาโก | 9 |
| 2.3 ความรู้พื้นฐาน Java Script | 12 |
| 2.3.1 ลักษณะการทำงานของ JavaScript | 13 |
| 2.3.2 JavaScript กับ HTML | 14 |
| 2.3.3 โครงสร้างภาษา | 14 |
| 2.4 ความรู้พื้นฐาน Angular Fronted Framework | 19 |

| | | |
|-------|--|-----|
| 2.4.1 | การติดตั้ง | 19 |
| 2.4.2 | คุณลักษณะของแองกูลาร์ | 20 |
| 2.4.3 | ข้อดีของ Angular | 20 |
| 2.5 | ความรู้พื้นฐาน MongoDB | 21 |
| 2.5.1 | ทำไมถึงเลือกใช้ MongoDB | 21 |
| 2.5.2 | Schema-less คืออะไร | 21 |
| 2.6 | เอกสารและงานวิจัยที่เกี่ยวข้อง | 22 |
| 2.6.1 | เว็บไซต์ Plickers | 22 |
| 2.6.2 | เว็บไซต์ smart classroom QR-Code Check-in | 24 |
| 2.6.3 | โปรแกรม Student Care | 25 |
| 3 | การวิเคราะห์และออกแบบระบบ | 26 |
| 3.1 | โครงสร้างภาพรวมของระบบ | 27 |
| 3.2 | System Requirements | 28 |
| 3.2.1 | Functional Requirements | 28 |
| 3.2.2 | Non-functional Requirements | 28 |
| 3.3 | User Interface Design | 29 |
| 3.4 | Use Case Diagram | 34 |
| 3.5 | Class Diagram | 43 |
| 3.6 | Sequence Diagram | 51 |
| 3.7 | โครงสร้างฐานข้อมูลไฟร์เบส (MongoDB Database Stucture) | 66 |
| 4 | การพัฒนาระบบ | 70 |
| 4.1 | การพัฒนาเว็บเซิร์ฟเวอร์ | 70 |
| 4.1.1 | โครงสร้างและโค้ดของการให้บริการ API ของระบบเช็คชื่อด้วยคิวอาร์โค้ด | 70 |
| 4.1.2 | โครงสร้างของ API ใน heroku Server | 85 |
| 4.2 | การพัฒนาเว็บแอปพลิเคชัน | 93 |
| 4.2.1 | โครงสร้างและโค้ด Service เพื่อให้บริการ API | 93 |
| 4.2.2 | โครงสร้างของโค้ดใน Angular หน้าสร้างชั้นเรียน | 102 |
| 4.2.3 | โครงสร้างของโค้ดใน Angular หน้าอัปเดตข้อมูลนักศึกษา | 107 |
| 4.2.4 | โครงสร้างของโค้ดใน Angular หน้าสร้างคิวอาร์โค้ด | 111 |

| | | |
|-------|--|-----|
| 4.2.5 | โครงสร้างของโค้ดใน Angular หน้าจัดการคะแนน | 113 |
| 4.2.6 | โครงสร้างของโค้ดใน Angular หน้าดูประวัติการเข้าเรียน | 117 |
| 4.2.7 | โครงสร้างของโค้ดใน Angular หน้ากรอกรหัสนักศึกษา | 121 |
| 5 | การทดสอบระบบ | 123 |
| 5.1 | การทดสอบการใช้งานเว็บแอปพลิเคชัน | 123 |
| 5.1.1 | การทดสอบหน้าสร้างชั้นเรียน | 123 |
| 5.1.2 | การทดสอบหน้าอัปเดตข้อมูลนักศึกษา | 124 |
| 5.1.3 | การทดสอบหน้าสร้างคิวอาร์โค้ด | 125 |
| 5.1.4 | การทดสอบหน้าจัดการคะแนน | 126 |
| 5.1.5 | การทดสอบหน้าดูประวัติการเข้าเรียน | 127 |
| 5.1.6 | การทดสอบหน้ากรอกรหัสนักศึกษา | 128 |
| 6 | สรุปและข้อเสนอแนะ | 129 |
| 6.1 | สรุปความสามารถของระบบ | 129 |
| 6.1.1 | เว็บแอปพลิเคชัน | 129 |
| 6.2 | ปัญหาและอุปสรรคในการพัฒนา | 130 |
| 6.3 | แนวทางการพัฒนาต่อ | 130 |
| | บรรณานุกรม | 131 |
| | ภาคผนวก | 133 |
| | ภาคผนวก ก การติดตั้งเครื่องมือที่ใช้พัฒนาโปรแกรม | 133 |
| | ก.1 การติดตั้ง Visual Studio Code | 133 |
| | ก.2 การติดตั้ง Go | 136 |
| | ก.3 การติดตั้ง Angular Framework | 140 |
| | ภาคผนวก ข คู่มือการติดตั้งระบบ | 141 |
| | ภาคผนวก ค คู่มือการใช้งานระบบ | 143 |
| | ประวัติผู้พัฒนา | 151 |

สารบัญตาราง

| ตารางที่ | หน้า |
|---|------|
| 1.1 ขั้นตอนการดำเนินงาน | 4 |
| 3.1 สัญลักษณ์ของ Use case Diagram | 34 |
| 3.2 อธิบาย Use Case หน้าที่ของระบบ ในภาพที่ 3.10 | 36 |
| 3.3 Use Case สร้างวิชา | 37 |
| 3.4 Use Case อัปเดตข้อมูลนักศึกษา | 37 |
| 3.5 Use Case สร้าง QR code | 38 |
| 3.6 Use Case จัดการคะแนน | 38 |
| 3.7 Use Case ดูประวัติการเข้าเรียน | 39 |
| 3.8 Use Case สแกน QR code | 39 |
| 3.9 Use Case อัปเดต URL | 39 |
| 3.10 Use Case กรอกรหัสนักศึกษา | 40 |
| 3.11 อธิบาย Use Case หน้าที่ของระบบ ในภาพที่ 3.11 | 41 |
| 3.12 Use Case อัปเดต URL | 42 |
| 3.13 Use Case สร้าง URL | 42 |
| 3.14 สัญลักษณ์ของ Class Diagram | 43 |
| 3.15 อธิบาย Class Diagram ของคลาสส่วนของนักศึกษา | 47 |
| 3.16 อธิบาย Class Diagram ของคลาสส่วนของอาจารย์ | 48 |
| 3.17 อธิบาย Class Diagram ของคลาสส่วนของอาจารย์ | 49 |
| 3.18 อธิบาย Class Diagram ของคลาสส่วนของอาจารย์ | 50 |
| 3.19 อธิบาย Class Diagram ของคลาสส่วนของอาจารย์ | 50 |
| 3.20 สัญลักษณ์ของ Sequence Diagram | 51 |
| 3.21 สัญลักษณ์ของโครงสร้างฐานข้อมูลแบบ MongoDB | 66 |
| 3.22 อธิบายส่วนที่ใช้เก็บข้อมูล | 68 |
| 3.23 อธิบายส่วนที่ใช้เก็บข้อมูลใน Heroku Server | 69 |
| 5.1 ผลการทดสอบการสร้างชั้นเรียน | 123 |
| 5.2 ผลการทดสอบหน้าอัปเดตข้อมูลนักศึกษา | 124 |
| 5.3 ผลการทดสอบหน้าสร้างคิวอาร์โค้ด | 125 |
| 5.4 ผลการทดสอบหน้าจัดการคะแนน | 126 |
| 5.5 ผลการทดสอบหน้าดูประวัติการเข้าเรียน | 127 |
| 5.6 ผลการทดสอบหน้ากรอกรหัสนักศึกษา | 128 |

สารบัญภาพ

| รูปที่ | | หน้า |
|--------|--|------|
| 2.1 | โครงสร้างของระบบปฏิบัติการแอนดรอยด์ | 6 |
| 2.2 | โครงสร้างคำสั่ง if-else | 9 |
| 2.3 | โครงสร้างคำสั่ง func | 10 |
| 2.4 | โครงสร้างคำสั่ง go | 10 |
| 2.5 | โครงสร้างการประกาศตัวแปร | 11 |
| 2.6 | โครงสร้างคำสั่ง for | 11 |
| 2.7 | โครงสร้างคำสั่ง channel | 11 |
| 2.8 | ตัวอย่างคำสั่งภาษา TypeScript ด้วย minted | 13 |
| 2.9 | ตัวแปรอาร์เรย์ | 16 |
| 2.10 | การเปรียบเทียบว่าจริงหรือเท็จ | 18 |
| 2.11 | ตัวดำเนินการเชิงข้อความ | 18 |
| 2.12 | สร้างโปรเจคใหม่แบบ Ivy | 20 |
| 2.13 | ตัวอย่างข้อมูลใน Collection | 21 |
| 2.14 | หน้าหลักในการใช้งานระบบ plickers | 22 |
| 2.15 | คิวอาร์โค้ดของ plickers | 23 |
| 2.16 | หน้าแรกของเว็บไซต์ smart classroom QR-Code Check-in | 24 |
| 2.17 | โปรแกรม Student Care | 25 |
| 3.1 | System architecture ระบบเช็คชื่อด้วยคิวอาร์โค้ด | 27 |
| 3.2 | หน้าหลัก | 29 |
| 3.3 | หน้าจอสร้างชั้นเรียน | 30 |
| 3.4 | หน้าจอจัดการคะแนนนักศึกษา | 30 |
| 3.5 | หน้าจอประวัติการเข้าเรียน | 31 |
| 3.6 | หน้ารวมไฟล์ | 31 |
| 3.7 | หน้าสร้างคิวอาร์โค้ด | 32 |
| 3.8 | หน้าแสดงคิวอาร์โค้ด | 32 |
| 3.9 | หน้าจัดการนักศึกษา | 33 |
| 3.10 | Use Case Diagram ของระบบเช็คชื่อด้วยคิวอาร์โค้ด | 35 |
| 3.11 | Use Case Diagram ของระบบ API กลาง | 41 |
| 3.12 | Class Diagram ของแอปพลิเคชันระบบ เช็คชื่อด้วยคิวอาร์โค้ด | 44 |
| 3.13 | Class Diagram ของแอปพลิเคชันระบบ เช็คชื่อด้วยคิวอาร์โค้ด | 45 |
| 3.14 | Class Diagram ของแอปพลิเคชันระบบ API | 46 |
| 3.15 | Class Diagram ของแอปพลิเคชันระบบ API กลาง | 46 |
| 3.16 | Sequence Diagram สร้างชั้นเรียน | 52 |
| 3.17 | Sequence Diagram อัปเดตข้อมูลนักศึกษา | 54 |
| 3.18 | Sequence Diagram สร้างคิวอาร์โค้ด | 56 |

| | | |
|------|---|----|
| 3.19 | Sequence Diagram จัดการคะแนน | 58 |
| 3.20 | Sequence Diagram ดูประวัติ | 60 |
| 3.21 | Sequence Diagram สแกน QRcode | 62 |
| 3.22 | Sequence Diagram กรอกรหัสนักศึกษา | 64 |
| 3.23 | โครงสร้างฐานข้อมูลแบบ MongoDB | 67 |
| 3.24 | โครงสร้างฐานข้อมูลแบบ MongoDB | 69 |
| 4.1 | ไฟล์ server.go | 70 |
| 4.2 | ไฟล์ handler.go | 71 |
| 4.3 | ไฟล์ model-Course.go | 72 |
| 4.4 | ไฟล์ student.go | 73 |
| 4.5 | ไฟล์ student.go | 74 |
| 4.6 | ไฟล์ student.go | 75 |
| 4.7 | ไฟล์ student.go | 76 |
| 4.8 | ไฟล์ student.go | 77 |
| 4.9 | ไฟล์ student.go | 78 |
| 4.10 | ไฟล์ student.go | 79 |
| 4.11 | ไฟล์ student.go | 80 |
| 4.12 | ไฟล์ student.go | 82 |
| 4.13 | ไฟล์ checkname.go | 83 |
| 4.14 | ไฟล์ checkname.go | 84 |
| 4.15 | ไฟล์ server.go | 85 |
| 4.16 | ไฟล์ server.go | 86 |
| 4.17 | ไฟล์ handler.go | 87 |
| 4.18 | ไฟล์ module.go | 88 |
| 4.19 | ไฟล์ checkstatus.go | 88 |
| 4.20 | ไฟล์ checkstatus.go | 89 |
| 4.21 | ไฟล์ checkstatus.go | 90 |
| 4.22 | ไฟล์ checkstatus.go | 92 |
| 4.23 | ไฟล์ student.service.ts | 93 |
| 4.24 | ไฟล์ student.service.ts | 94 |
| 4.25 | ไฟล์ student.service.ts | 94 |
| 4.26 | ไฟล์ student.service.ts | 94 |
| 4.27 | ไฟล์ student.service.ts | 95 |
| 4.28 | ไฟล์ student.service.ts | 95 |
| 4.29 | ไฟล์ student.service.ts | 96 |
| 4.30 | ไฟล์ student.service.ts | 97 |
| 4.31 | ไฟล์ student.service.ts | 97 |
| 4.32 | ไฟล์ student.service.ts | 98 |

| | | |
|------|--|-----|
| 4.33 | ไฟล์ student.service.ts | 98 |
| 4.34 | ไฟล์ student.service.ts | 99 |
| 4.35 | ไฟล์ student.service.ts | 99 |
| 4.36 | ไฟล์ heroku.service.ts | 99 |
| 4.37 | ไฟล์ heroku.service.ts | 100 |
| 4.38 | ไฟล์ heroku.service.ts | 100 |
| 4.39 | ไฟล์ heroku.service.ts | 101 |
| 4.40 | โค้ดในไฟล์ formfile.component.html | 102 |
| 4.41 | โค้ดในไฟล์ formfile.component.ts | 104 |
| 4.42 | โค้ดในไฟล์ formfile.component.ts | 106 |
| 4.43 | โค้ดในไฟล์ formup.component.html | 107 |
| 4.44 | โค้ดในไฟล์ formup.component.ts | 108 |
| 4.45 | โค้ดในไฟล์ formup.component.ts | 109 |
| 4.46 | โค้ดในไฟล์ formup.component.ts | 110 |
| 4.47 | โค้ดในไฟล์ qrcode.component.html | 111 |
| 4.48 | โค้ดในไฟล์ qrcode.component.ts | 112 |
| 4.49 | โค้ดในไฟล์ info.component.html | 113 |
| 4.50 | โค้ดในไฟล์ info.component.ts | 114 |
| 4.51 | โค้ดในไฟล์ info.component.ts | 115 |
| 4.52 | โค้ดในไฟล์ info.component.ts | 116 |
| 4.53 | โค้ดในไฟล์ infoname.component.html | 117 |
| 4.54 | โค้ดในไฟล์ infoname.component.ts | 118 |
| 4.55 | โค้ดในไฟล์ infoname.component.ts | 119 |
| 4.56 | โค้ดในไฟล์ infoname.component.ts | 120 |
| 4.57 | โค้ดในไฟล์ fromstusent.component.html | 121 |
| 4.58 | โค้ดในไฟล์ fromstusent.component.ts | 121 |
| ก.1 | หน้าเว็บดาวน์โหลด Visual Studio Code | 133 |
| ก.2 | หน้าต่างต้อนรับของ Visual Studio Code | 134 |
| ก.3 | หน้าต่างข้อตกลงการใช้งาน Visual Studio Code | 134 |
| ก.4 | หน้าต่างที่จัดเก็บไฟล์ต่างๆ ของ Visual Studio Code | 135 |
| ก.5 | หน้าต่างที่จัดเก็บไฟล์ต่างๆ ของ Visual Studio Code | 135 |
| ก.6 | หน้าต่างผลการติดตั้ง Visual Studio Code | 136 |
| ก.7 | หน้าเว็บดาวน์โหลด Go | 136 |
| ก.8 | ไฟล์ติดตั้งสำหรับติดตั้ง Go | 137 |
| ก.9 | หน้าต่างต้อนรับของ Go | 137 |
| ก.10 | หน้าต่างข้อตกลงในการใช้ Go | 138 |
| ก.11 | หน้าต่างเลือกโพลเดอร์ที่จะทำการติดตั้ง Go | 138 |
| ก.12 | หน้าต่างติดตั้ง Go | 139 |

| | | |
|------|---|-----|
| ก.13 | หน้าตาผลการติดตั้ง Go | 139 |
| ก.14 | คำสั่งสำหรับติดตั้ง Angular Framework | 140 |
| ก.15 | หน้าตาผลการติดตั้ง | 140 |
| ข.1 | หน้าเว็บแอปพลิเคชัน ระบบเช็คชื่อกับคิวอาร์โค้ดมุมมองของคอมพิวเตอร์ | 141 |
| ข.2 | หน้าเว็บแอปพลิเคชัน ระบบเช็คชื่อกับคิวอาร์โค้ดมุมมองของโทรศัพท์มือถือ | 142 |
| ค.1 | หน้าจอหลักเมื่อเข้าใช้งานครั้งแรก | 143 |
| ค.2 | หน้าหลัก | 143 |
| ค.3 | การแสดงผลข้อมูลชั้นเรียน | 144 |
| ค.4 | หน้าเพิ่มชั้นเรียน | 145 |
| ค.5 | หน้าจัดการคะแนน | 145 |
| ค.6 | หน้าอัปเดตชื่อ | 146 |
| ค.7 | หน้าเพิ่มนักศึกษา | 146 |
| ค.8 | หน้าจอต้อนรับ | 147 |
| ค.9 | หน้าประวัติ | 147 |
| ค.10 | หน้ารวมคะแนน | 148 |
| ค.11 | หน้าสร้างคิวอาร์โค้ด | 148 |
| ค.12 | หน้าแสดงคิวอาร์โค้ด | 149 |
| ค.13 | หน้ากรอกรหัสนักศึกษา | 150 |
| ค.14 | หน้าสแกนคิวอาร์โค้ดไม่สำเร็จ | 150 |

บทที่ 1

บทนำ

1.1 ที่มาและเหตุผล

การเช็คชื่อเข้าเรียน เป็นสิ่งสำคัญในการตรวจสอบความกระตือรือร้นในการเข้าเรียนของนักศึกษา และเป็นตัวชี้วัดการให้คะแนนนักศึกษาของผู้สอนหรือเช็คยอดนักศึกษาที่เข้าเรียนในคลาสนั้น ๆ ซึ่งการเช็คชื่อเป็นสิ่งสำคัญที่ช่วยคณะกรรมการตรวจสอบ และเป็นหลักฐานในการให้คะแนนของผู้สอนที่จะแสดงต่อคณะกรรมการเมื่อมีการตรวจสอบแหล่งที่มาของคะแนน

การเช็คชื่อในคลาสนั้นส่วนมากเป็นการเรียกชื่อนักศึกษาแล้วให้นักศึกษาขานตอบ ซึ่งจะทำให้การเช็คชื่อนั้นมีความล่าช้าและทำให้เสียเวลาในการเรียนการสอน ซึ่งเวลาที่เสียไปในการเช็คชื่อนักศึกษาสามารถนำไปใช้ทำสิ่งต่าง ๆ ได้ เช่น พักครึ่ง ทำแบบฝึกหัด ทวี ตอบคำถาม และเตรียมตัวสำหรับคาบต่อไป เป็นต้น นอกจากนี้ผู้สอนบางท่านอาจจะใช้วิธีสุมเช็คชื่อเพื่อลดเวลาในที่ใช้ในการเช็คชื่อในบ้างคาบเรียน แต่นั่นทำให้นักศึกษาส่วนมากคิดว่าผู้สอนอาจจะไม่เช็คชื่อในคาบเรียนจึงไม่เข้าเรียน และมีผู้สอนบางท่านยังใช้กระดาษในการเช็คชื่อ ซึ่งเป็นการสิ้นเปลืองกระดาษและอาจทำให้กระดาษชำรุด หรือสูญหาย ส่งผลทำให้ไม่มีหลักฐานในการให้คะแนนนักศึกษาที่เข้าเรียนและยังไม่สามารถระบุแหล่งที่มาของคะแนนที่ให้นักศึกษาต่อคณะกรรมการอีกด้วย

จากปัจจัยที่กล่าวมา ผู้พัฒนาจึงมีความคิดที่จะพัฒนาระบบเช็คชื่อเข้าเรียน โดยจะเน้นให้ระบบใช้งานง่ายไม่ซับซ้อน เข้าถึงง่าย และมีความรวดเร็ว โดยใช้เทคโนโลยี Quick Response Code (QR Code) เพื่อให้ง่ายต่อการใช้งาน ซึ่งจะให้นักศึกษาสแกน QR Code ของผู้สอน โดยผู้สอนจะเปิด QR Code ให้นักศึกษาทุกคนในคลาสนั้นสแกนเพื่อเป็นการเช็คชื่อเข้าเรียน นอกจากนี้ นักศึกษาไม่จำเป็นต้องโหลดแอปพลิเคชันเพิ่มก็สามารถเช็คชื่อเข้าเรียนได้

1.2 วัตถุประสงค์

1. เพื่อพัฒนาระบบแอปพลิเคชันการเช็คชื่อในคลาสเรียนที่มีขนาดใหญ่
2. เพื่อเป็นแหล่งเก็บข้อมูลการเข้าเรียนของนักศึกษาและเป็นหลักฐานแสดงการมาเรียนของนักศึกษา ซึ่งจะเป็นแหล่งที่มาของคะแนนการเข้าเรียน
3. เพื่อลดการใช้กระดาษที่ใช้ในการเช็คชื่อ และป้องกันการสูญหายหรือชำรุดของข้อมูล

1.3 ขอบเขตของโครงการ

1.3.1 ผู้สอน

- สามารถเพิ่ม และลบ คลาสเรียน
- สามารถตรวจสอบรายชื่อนักศึกษาในคลาสเรียน
- สามารถจัดการคะแนนของนักศึกษา
- สามารถสร้าง QR Code สำหรับให้นักศึกษาสแกนเพื่อเช็คชื่อได้
- สามารถเพิ่ม ลบ และแก้ไข ชื่อนักศึกษาในคลาสเรียนได้
- สามารถรวมไฟล์คะแนน ของคลาสเรียนเดียวกัน

1.3.2 นักศึกษา

- สแกน QR Code ของผู้สอนเพื่อเป็นการเช็คชื่อเข้าเรียน

1.4 ประโยชน์ที่คาดว่าจะได้รับ

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

1.5 เครื่องมือที่ใช้ในการพัฒนา (Development tools)

1.5.1 ฮาร์ดแวร์

1. สมาร์ทโฟน (Smart phone)

- ทำงานบนระบบปฏิบัติการแอนดรอยด์เวอร์ชัน 5.0 หรือ API Level 21
- หน่วยประมวลผลกลาง Mediatek MT6753 Octa-core ความเร็ว 1.3 กิกะเฮิร์ตซ์ (Gigahertz, GHz)
- หน่วยประมวลผลกราฟฟิกลำโพงน้อย Mali-T720MP3
- หน่วยความจำหลักอย่างน้อย 2 กิกะไบต์ (Gigabyte, GB)
- หน่วยความจำสำรองอย่างน้อย 16 กิกะไบต์ (Gigabyte, GB)
- หน้าจอแสดงผลความละเอียดอย่างน้อย 1080 x 1920 พิกเซล (Pixel)
- หน้าจอแสดงผลขนาดอย่างน้อย 5 นิ้ว
- กล้องถ่ายภาพความละเอียดอย่างน้อย 13 เมกะพิกเซล (Megapixel)

2. เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal computer)

- ทำงานบนระบบปฏิบัติการวินโดวส์ 10 (Windows operating system 10)
- หน่วยประมวลผลกลาง Intel(R) Core(TM) i7-7500U CPU ความเร็ว 2.70 กิกะเฮิร์ตซ์ (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)

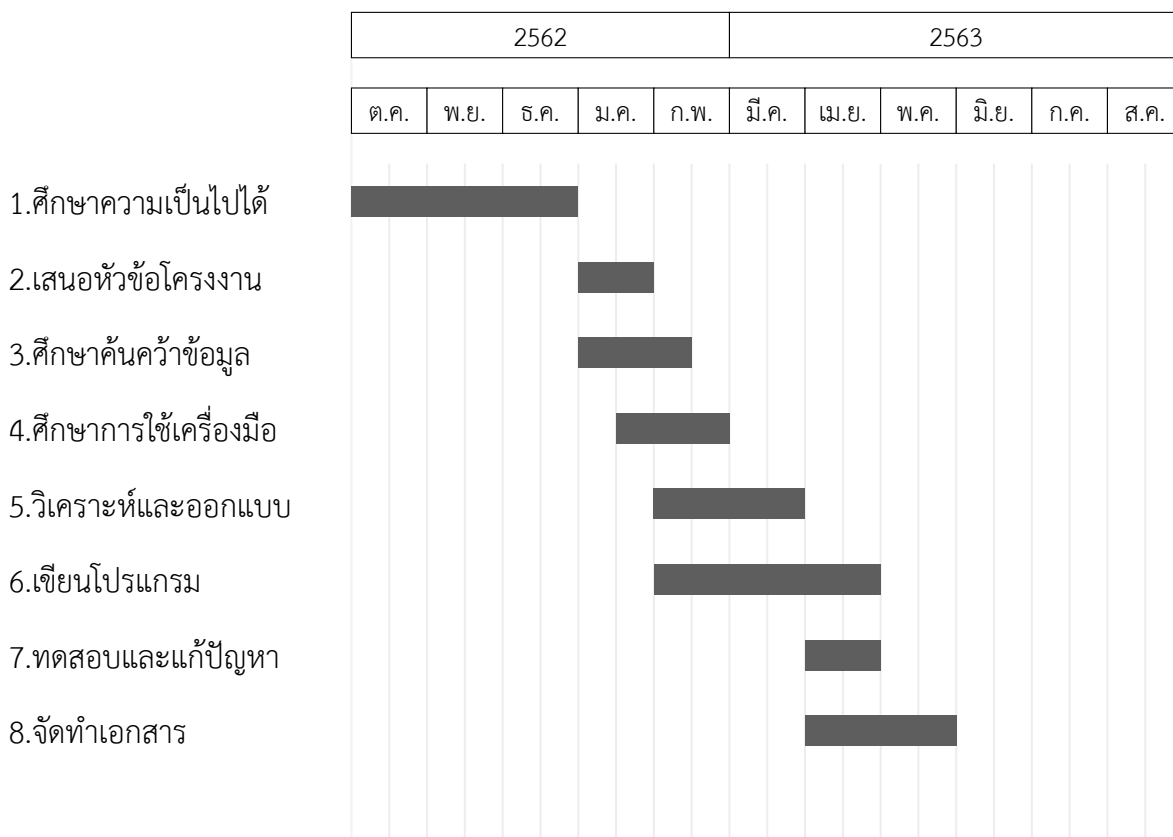
1. Angular เวอร์ชัน 8 เป็นเฟรมเวิร์ค (Framework) สำหรับทำหน้าเว็บแสดงผล โดยใช้ JavaScript ในการพัฒนา
2. MongoDB คือฐานข้อมูลแบบไม่มีความสัมพันธ์ของตารางซึ่งมีโครงสร้างแบบ NoSQL (Not Only Structured Query Language) มีรูปแบบการจัดเก็บข้อมูลแบบ JSON (JavaScript Object Notation)

3. Echo Golang เวอร์ชัน 1.13 ใช้สำหรับทำเว็บเซิร์ฟเวอร์ (Web Server) พัฒนาด้วยภาษา Golang
4. Postman คือเครื่องมือสำหรับช่วยในพัฒนา API (Applocation Programming Interface) ใช้ในการทดสอบการทำงานของ Service รวมถึงจำลอง Service
5. Visual Studio Code คือเครื่องมือแก้ไขและปรับแต่งโค้ดสำหรับพัฒนาเว็บแอปพลิเคชัน
6. Google Chrome คือโปรแกรมเว็บเบราว์เซอร์ (Web Browser) ใช้สำหรับเปิดเว็บไซต์และเป็นเครื่องมือที่ใช้สำหรับตรวจสอบ หรือหาจุดบกพร่องของเว็บแอปพลิเคชัน

1.5.3 แผนการดำเนินการ

ในการสร้างระบบเช็คชื้อด้วยคิวอาร์โค้ด ผู้พัฒนาได้แบ่งขั้นตอนการดำเนินงานไว้ด้วยกัน 7 ขั้นตอน ดังต่อไปนี้

ตารางที่ 1.1: ขั้นตอนการดำเนินงาน



บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

บทนี้จะเป็นรายละเอียดเกี่ยวกับทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการพัฒนาโปรแกรมในครั้งนี้ โดยที่แต่ละหัวข้อจะมีความสัมพันธ์กันเป็นลำดับ โดยหัวข้อที่หนึ่งจะแนะนำความรู้เรื่อง การพัฒนาเว็บแอปพลิเคชัน เพื่อให้เข้าใจพื้นฐานเบื้องต้นเกี่ยวกับที่มาของโครงงาน หัวข้อที่สองที่สาม จะช่วยเตรียมให้ผู้อ่านเข้าใจเทคโนโลยีที่ใช้ในการออกแบบและพัฒนา ส่วนของระบบเช็คชื่อด้วยคิวอาร์โค้ด

2.1 ความรู้พื้นฐานของเว็บแอปพลิเคชัน

เว็บแอปพลิเคชัน [1] คือ โปรแกรมประยุกต์ที่เข้าถึงด้วยโปรแกรมค้นดูเว็บผ่านเครือข่ายคอมพิวเตอร์อย่างอินเทอร์เน็ต โดยระบบเว็บแอปพลิเคชันสามารถแบ่งได้ 2 รูปแบบคือแบบ Online ซึ่งเป็นแบบ Local ใช้ภายในวง LAN โดยไม่จำเป็นต้องใช้อินเทอร์เน็ต และแบบ Global ซึ่งต้องใช้งานผ่านเครือข่ายอินเทอร์เน็ต ทำให้เหมาะสำหรับงานที่ต้องการข้อมูลแบบ Real Time

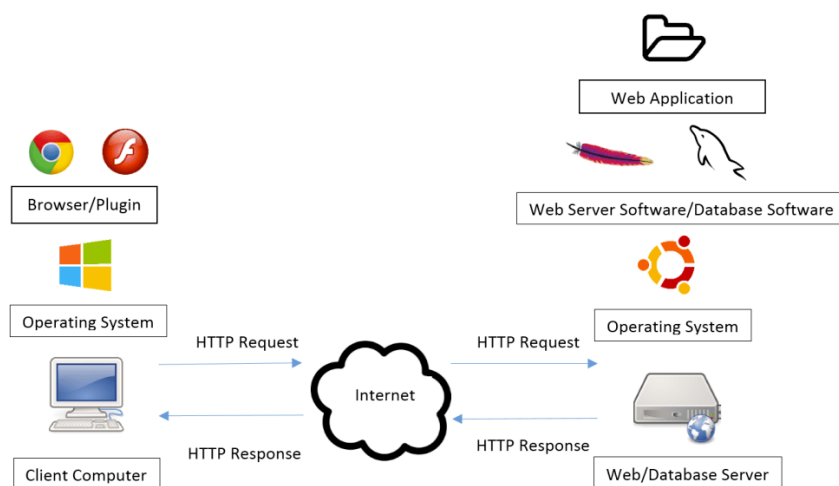
2.1.1 ความสำคัญของเว็บแอปพลิเคชัน

ในอดีตที่ยังไม่มีระบบเว็บแอปพลิเคชันการทำงานของคอมพิวเตอร์ส่วนบุคคลจะเป็น ลักษณะที่เรียกว่า Standalone ยกตัวอย่างเช่น เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรมเช่น Microsoft office เป็นต้น โปรแกรมเหล่านี้จะเรียกว่าเป็น Desktop application ซึ่งติดตั้งที่เครื่องคอมพิวเตอร์แต่ละบุคคล โดยโปรแกรมหากล่าวบางหน่วยงานติดตั้งที่เครื่องคอมพิวเตอร์เป็นลักษณะไคลแอนซ์เซิร์ฟเวอร์ (Client-Server application) เก็บฐานข้อมูลไว้ที่เซิร์ฟเวอร์ (Server) และติดตั้งตัวโปรแกรมที่เครื่องใช้งาน (Client) ซึ่งตอบสนองความต้องการเพิ่มขึ้นในด้าน Multi-User หรือใช้งานพร้อมกันได้หลายคน โดยใช้ฐานข้อมูลเดียวกันเก็บเป็นฐานข้อมูลส่วนกลาง ซึ่งเทคโนโลยี Desktop Application ไม่สามารถตอบสนองความต้องการในการบริหารจัดการข้อมูลได้ โดยเฉพาะการทำธุรกิจที่ต้องการข้อมูลที่เปลี่ยนแปลงตลอดเวลา ระบบ Client-Server Application ตัวโปรแกรมมีความซับซ้อน การแก้ไข การ Upgrade ทำได้ยุ่งยาก ปัญหาเหล่านี้ ถูกจัดการด้วยเทคโนโลยีเว็บแอปพลิเคชัน เพราะสามารถตอบสนองปัญหาข้างต้นได้เป็นอย่างดี และสามารถแทนที่ Desktop Application ที่เป็น Client-Server Application ได้เป็นอย่างดีตัวโปรแกรมของ

เว็บแอปพลิเคชัน จะถูกติดตั้งไว้ที่ Server คอยให้บริการกับ Client และที่ Client ก็ไม่ต้องติดตั้งโปรแกรมเพิ่มเติม สามารถใช้โปรแกรมประเภท Browser ที่ติดมากับ OS ใช้งานได้ทันทีอย่าง Internet Explorer หรือ GoogleChorme

2.1.2 ส่วนประกอบของเว็บแอปพลิเคชัน

การทำความเข้าใจส่วนประกอบของเว็บแอปพลิเคชัน ถือว่าเป็นสิ่งสำคัญเพราะถ้านักพัฒนาโปรแกรม สามารถมองภาพโดยรวมของระบบได้ทั้งหมด จะสามารถเข้าใจถึงกระบวนการทำงาน ได้ดียิ่งขึ้น และสามารถนำไปช่วยในการออกแบบโปรแกรมที่ต้องการพัฒนาเพื่อให้เกิดประสิทธิภาพในการทำงาน



รูปที่ 2.1: โครงสร้างของระบบปฏิบัติการแอนดรอยด์

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

จากโครงสร้างของระบบปฏิบัติการแอนดรอยด์ในรูปที่ 2.1 จะสังเกตได้ว่า มีการแบ่งออกเป็น ส่วน ๆ โดยจะแบ่งเป็น 2 ส่วนใหญ่ๆ คือ ส่วนประกอบฝั่งผู้ใช้งาน (Client-side Technology) และส่วนประกอบฝั่งเซิร์ฟเวอร์ (Server-side Technology)

1. ส่วนประกอบฝั่งผู้ใช้งาน

จากรูป 2.1 ด้านซ้ายจะเป็นเทคโนโลยีฝั่งผู้ใช้งาน ซึ่งประกอบด้วย 3 ส่วนหลักๆ ได้แก่

- 1.1 เว็บเบราว์เซอร์ (Web Browser) เป็นซอฟต์แวร์ที่ผู้ใช้งานใช้เข้าถึงเว็บแอปพลิเคชัน โดยเริ่มต้นให้ผู้ใช้งานใส่ URL หรือว่าชื่อของเว็บไซต์ที่ต้องการเข้าใช้งาน เช่น

www.google.com เมื่อเบราว์เซอร์ได้รับชื่อเว็บไซต์ก็จะทำการแปลงชื่อเว็บไซต์เป็น IP address ผ่าน DNS หลังจากนั้นเบราว์เซอร์จะทำการสร้าง HTTP request เพื่อส่งคำร้องไปยังเว็บเซิร์ฟเวอร์ผ่านทางเครือข่ายอินเทอร์เน็ต 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 เป็นโปรแกรมที่ใช้ในการเพิ่มความสามารถให้กับเบราว์เซอร์ เช่น ส่วนเพิ่มความสามารถที่ช่วยในการจัดการไฟล์ดาวโหลด ส่วนเพิ่มความสามารถที่ช่วยในการดาวโหลดไฟล์วิดีโอ เป็นต้น ซึ่งส่วนเพิ่มความสามารถเบราว์เซอร์เหล่านี้จะเน้นเพิ่มความสามารถให้กับเบราว์เซอร์ มากกว่าการประมวลผลเนื้อหาเว็บไซต์

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

2. ส่วนประกอบฝั่งเซิร์ฟเวอร์

เว็บเซิร์ฟเวอร์ที่ทำหน้าที่เป็นผู้ให้บริการแก่ผู้ใช้งานเว็บไซต์ประกอบไปด้วยเทคโนโลยีและซอฟต์แวร์หลายส่วนทำงานร่วมกัน โดยซอฟต์แวร์หลักที่ใช้ในการให้บริการของเว็บเซิร์ฟเวอร์ประกอบไปด้วย 4 ส่วนประกอบหลัก

1.1 เว็บแอปพลิเคชัน (Web Application) คือส่วนสำคัญของเว็บไซต์เนื่องจากทำหน้าที่ติดต่อกับผู้ใช้งาน รับและแสดงข้อมูล ประมวลผลข้อมูล จัดการข้อมูลในฐานข้อมูล

และอื่น ๆ เรียกได้ว่าเว็บแอปพลิเคชันเป็นซอฟต์แวร์ที่ให้บริการผู้ใช้งานทั่วโลกผ่านอินเทอร์เน็ต หากนักพัฒนาได้เขียนเว็บแอปพลิเคชันตาม Model-View-Controller (MVC) แล้วก็จะสามารถแบ่งเว็บแอปพลิเคชันออกได้เป็นสามส่วนหลัก ๆ คือ

- ส่วนที่ติดต่อกับผู้ใช้งานเพื่อรับข้อมูลและแสดงผล (View)
- ส่วนที่ประมวลผลการทำงาน (Controller)
- ส่วนที่ใช้ในการติดต่อและจัดการกับข้อมูลและฐานข้อมูล (Model)

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 มีการสนับสนุน

การคอมไพล์โปรแกรมเพื่อนำไปใช้บนแพลตฟอร์มอื่นๆ (cross-compiler) และมีการสนับสนุนการทำงานแบบภาวะพร้อมกัน ที่ถูกออกแบบขึ้นตามแนวคิดของ Hoare's Communicating Sequential Processes นอกจากนี้การเขียนโปรแกรมด้วยภาษาโกนั้นง่ายเพราะบางส่วนของภาษาโกมีลักษณะคล้ายคลึงกับภาษาสคริปต์

2.2.1 ชุดคำสั่งในภาษาโก

คำสั่งในภาษาโก มีคำสั่งจำนวนมากให้เลือกใช้งานตามความเหมาะสมแต่ในระบบนี้ขอแนะนำ 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 จากภาพจะเห็นว่าฟังก์ชันชื่อว่า A จะสามารถรับค่าอยู่ในรูปแบบของ ชื่อว่าตัวแปร param1 ตามด้วย ptype1 คือชนิดของตัวแปร param1 และสามารถรับได้หลายๆค่าโดยคั่นด้วยเครื่องหมายจุลภาค (,) และสามารถคืนค่าได้โดยตัวแปร 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
3                                     else
                                           }
```

รูปที่ 2.3: โครงสร้างคำสั่ง func

2.2.2.3 คำสั่ง go เป็นคำสั่งที่กำหนดให้โปรแกรมทำงานแบบภาวะพร้อมกันอย่างอัตโนมัติ ตัวอย่างรูปแบบของคำสั่งแสดงได้ในภาพที่ 2.4 การทำงานของคำสั่งโก โปรแกรมจะถูกแบ่งออกเป็น โพรเซสย่อยๆหลายอันทำงานแบบภาวะพร้อมกัน ซึ่งภาษา go เรียกโพรเซสย่อยแต่ละอันว่า 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 จากภาพเราสามารถประกาศ ตัวแปรได้ 2 รูปแบบ คือการประกาศแบบใช้คำสั่ง var และการประกาศแบบย่อโดยใช้ เครื่องหมาย := ในการประกาศตัวแปร ทั้งสองแบบมีความแตกต่างกันที่การประกาศในรูปแบบ var สามารถประกาศโดยระบุชนิดของข้อมูลหรือหากไม่ระบุชนิดข้อมูลภาษาโกจะทำการกำหนดชนิด ข้อมูลให้ตามค่าที่อยู่ทางขวามือของเครื่องหมายเท่ากับโดยอัตโนมัติและสามารถประกาศได้ทั้งภายใน และภายนอกฟังก์ชัน ส่วนการประกาศในรูปแบบเครื่องหมาย := นั้นสามารถประกาศได้เฉพาะภายในฟังก์ชัน เท่านั้นและไม่ต้องระบุชนิดของข้อมูลโดย ภาษาโกจะกำหนดชนิดข้อมูลให้อัตโนมัติเช่นกัน โครงสร้าง การประกาศตัวแปร แสดงดังภาพที่ 2.5

```

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

```

รูปที่ 2.5: โครงสร้างการประกาศตัวแปร

2.2.2.5 คำสั่ง for เป็นคำสั่งที่ทำงานแบบวนซ้ำ โดยจะทำการวนซ้ำถ้าหากเงื่อนไขเป็นจริง รูปแบบของคำสั่งแสดงดังภาพที่ 2.6 จากภาพถ้า condition เป็นจริงกลุ่มคำสั่งที่อยู่ภายใต้ for จะทำงานวนไปเรื่อยๆจนกว่าเงื่อนไขจะเป็นเท็จ 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 JavaScript

JavaScript [3] คือ ภาษาคอมพิวเตอร์สำหรับการเขียนโปรแกรมบนระบบอินเทอร์เน็ตที่กำลังได้รับความนิยมอย่างสูง Java JavaScript เป็นภาษาสคริปต์เชิงวัตถุที่เรียกว่า “สคริปต์” (Script) ซึ่งในการสร้างและพัฒนาเว็บไซต์ (ใช้ร่วมกับ HTML) เพื่อให้เว็บไซต์ของเรามีการเคลื่อนไหวสามารถตอบสนองผู้ใช้งานได้มากขึ้น ซึ่งมีวิธีการทำงานในลักษณะ “แปลความและดำเนินงานไปทีละคำสั่ง” (Interpret) หรือเรียกว่า อ็อบเจ็กต์โอเรียนเตด (Object Oriented Programming) ที่มีเป้าหมายในการออกแบบและพัฒนาโปรแกรมในระบบอินเทอร์เน็ตสำหรับผู้เขียนด้วยภาษา HTML สามารถทำงานข้ามแพลตฟอร์มได้โดยทำงานร่วมกับภาษา HTML และภาษา JavaScript ทำงานได้ทั้งทางฝั่งไคลเอนต์ (Client) และทางฝั่งเซิร์ฟเวอร์ (Server)

JavaScript ถูกพัฒนาขึ้นโดย เน็ตสเคปคอมมิวนิเคชันส์ (Netscape Communications Corporation) โดยใช้ชื่อว่า Live Script ออกมาพร้อมกับ Netscape Navigator 2.0 เพื่อใช้สร้างเว็บเพจโดยติดต่อกับเซิร์ฟเวอร์แบบ Live Wire ต่อมาเน็ตสเคปจึงได้ร่วมมือกับ บริษัทซันไมโครซิสเต็มส์ปรับปรุงระบบของบราวเซอร์เพื่อให้สามารถติดต่อกับภาษาจาวาได้ และได้ปรับปรุง LiveScript ใหม่เมื่อ ปี 2538 แล้วตั้งชื่อใหม่ว่า JavaScript

JavaScript สามารถทำให้ การสร้างเว็บเพจ มีลูกเล่น ต่าง ๆ มากมายและสามารถโต้ตอบกับผู้ใช้อย่างทันที 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 อย่างไรก็ตามสิ่งที่ต้องระวังคือ JavaScript มีการพัฒนาเป็นเวอร์ชันใหม่ ๆ ออกมาด้วยดังนั้น ถ้านำโค้ดของเวอร์ชันใหม่ไปรันบนบราวเซอร์รุ่นเก่าที่ยังไม่สนับสนุนอาจจะทำให้เกิดข้อผิดพลาด (Error) ได้

2.3.1 ลักษณะการทำงานของ JavaScript

JavaScript เป็นภาษาสคริปต์เชิงวัตถุหรือเรียกว่าอ็อบเจ็กต์โอเรียนเต้ด (Object Oriented Programming) ที่มีเป้าหมายในการออกแบบและพัฒนาโปรแกรมในระบบอินเทอร์เน็ตสำหรับผู้เขียนเอกสารด้วยภาษา HTML สามารถทำงานข้ามแพลตฟอร์มได้ทำงานร่วมกับภาษา HTML และภาษาจาวาได้ทั้งทางฝั่งไคลเอนต์ (Client) และ ทางฝั่งเซิร์ฟเวอร์ (Server) โดยมีลักษณะการทำงานดังนี้

```

1 // import ฟังก์ชันสำคัญของangular/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 (หมายถึงฝั่งเครื่องคอมพิวเตอร์ของผู้ให้บริการเว็บโดยอาจจะเป็นเครื่องของชั้นซิลิคอนกราฟิกส์หรืออื่น ๆ) สามารถใช้ได้เฉพาะกับ LiveWire ของเน็ตสเคป โดยตรง

2.3.2 JavaScript กับ HTML

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

- เขียนด้วยชุดคำสั่งและฟังก์ชันของ JavaScript เอง
- เขียนตามเหตุการณ์ที่เกิดขึ้นตามการใช้งานจากชุดคำสั่งของ HTML

เมื่อเริ่มใช้งานโปรแกรมบราวเซอร์จะอ่านข้อมูลจากส่วนบนของเพจ HTML และทำงานไปตามลำดับจาก บนลงล่าง (top-down) โดยเริ่มที่ส่วน < HEAD >...< /HEAD > ก่อนจากนั้นจึงทำงานในส่วน < BODY >...< /BODY > เป็นลำดับ ซึ่งการทำงานของ JavaScript ดูไม่แตกต่างไปจาก HTML เท่าใดนัก แต่ HTML จะวางเลย์เอาต์โครงสร้างของอ็อบเจกต์ภายใน 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
 - เริ่มต้นด้วยตัวอักษรในภาษาอังกฤษ ตามด้วยตัวอักษรหรือตัวเลขใด ๆ ก็ได้
 - ห้ามเว้นช่องว่าง

- ห้ามใช้สัญลักษณ์พิเศษ ยกเว้นขีดล่าง (_) และดอลลาร์ (\$)
 - สำหรับความยาวของชื่อใน JavaScript จะมีความยาวเท่าใดก็ได้ แต่ที่นิยมใช้ ไม่เกิน 20 ตัวอักษร
 - การตั้งชื่อมีข้อพึงระวังว่า จะต้องไม่ซ้ำกับคำสงวน (Reserve word) และตัวอักษรของชื่อจะจำแนกแตกต่างกันระหว่างอักษรตัวพิมพ์เล็กกับอักษรตัวพิมพ์ใหญ่
 - ควรจะตั้งชื่อโดยให้ชื่อนั้นมีสื่อความหมายให้เข้ากับข้อมูล สามารถอ่านและเข้าใจได้ ตัวอย่างเช่น ชื่อที่ถูกตั้ง 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) และ เลขจำนวนจริง (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 (,) คั่นระหว่างชื่อตัวแปรและปิดท้ายด้วยเครื่องหมายเซมิโคลอน (;) 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 หากกำหนดชื่อตัวแปรไว้ที่โปรแกรมหลักโดยไม่ได้ระบุอยู่ในขอบเขตฟังก์ชันใด ๆ เรียกว่าเป็นตัวแปรแบบโกลบอล (Global) ตัวแปรจำพวกนี้จะมีค่าคงอยู่ในหน่วยความจำตลอดการทำงานของโปรแกรม ทำให้สามารถเรียกใช้ได้จากทุก ๆ ส่วนของโปรแกรม รวมถึงภายในฟังก์ชันต่าง ๆ ด้วย แต่ถ้ากำหนดตัวแปรไว้ภายในขอบเขตฟังก์ชันใด ๆ จะเรียกว่าเป็นตัวแปรแบบ โลคัล (Local) เพราะจะเป็นตัวแปรที่มีค่าคงอยู่ และสามารถเรียกใช้ได้เฉพาะภายในขอบเขตของฟังก์ชันนั้น ๆ เท่านั้น

6. ตัวแปรแบบอาร์เรย์ (Array) หมายถึงตัวแปรซึ่งมีค่าได้หลายค่าโดยใช้ชื่ออ้างอิงเพียงชื่อเดียว ด้วยการให้หมายเลขลำดับเป็นตัวจำแนกความแตกต่างของค่าตัวแปรแต่ละตัว ถ้าจะเรียกตัวแปรชนิดนี้ว่า “ตัวแปรชุด” ก็เห็นจะไม่ผิดนัก ตัวแปรชนิดนี้มีประโยชน์มาก ลองคิดถึงค่าข้อมูลจำนวน 100 ค่า ที่ต้องการเก็บไว้ในตัวแปรจำนวน 100 ตัว อาจทำให้ต้องกำหนดตัวแปรที่แตกต่างกันมากถึง 100 ชื่อ กรณีอย่างนี้ควรจะทำอย่างไรดี แต่ด้วยการใช้สมบัติอาร์เรย์ สามารถนำตัวแปรหลาย ๆ ตัวมาอยู่รวมเป็นชุดเดียวกันได้และสามารถเรียกใช้ตัวแปรทั้งหมดโดยระบุผ่านชื่อเพียงชื่อเดียวเท่านั้น ด้วยการระบุหมายเลขลำดับ หรือดัชนี (index) กำกับตามหลังชื่อตัวแปร ตัวแปรเพียงชื่อเดียวจึงมีความสามารถเทียบได้กับตัวแปรนับร้อยตัว พันตัว (ตัวที่ 1) ในตัวแปรแบบอาร์เรย์มีดัชนีเป็น 0 ส่วนตัวแปรต่อ ๆ ไปก็จะมีดัชนีเป็น 1,2,3,... ไปตามลำดับ เมื่อต้องการระบุชื่อตัวแปรแบบอาร์เรย์แต่ละตัว ก็จะใช้รูปแบบ `name[0]`, `name[1]`,... เรียงต่อกันไปเรื่อย ๆ สามารถสร้างตัวแปรอาร์เรย์ใหม่ด้วย `myArray = new Array()` ดังรูป 2.9

```

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

```

รูปที่ 2.9: ตัวแปรอาร์เรย์

7. ค่าคงที่ (Literal หรือ Constant) หมายถึง ค่าของข้อมูล que เมื่อกำหนดแล้วจะทำการเปลี่ยนแปลงค่าเป็นอย่างอื่นไม่ได้ ชนิดข้อมูลของค่าคงที่ ได้แก่

- 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) เป็นข้อความที่ใช้กำหนดค่าของข้อมูล 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 ดังรูป 2.10

```

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

```

รูปที่ 2.10: การเปรียบเทียบว่าจริงหรือเท็จ

ผลลัพธ์ที่ได้คือ c จะมีค่าเป็นจริง (True)

- นิพจน์ข้อความ (String) เป็นนิพจน์เกี่ยวกับการกำหนดข้อความ การเชื่อมโยงข้อความ ใช้ประมวลผลข้อความในลักษณะต่าง ผลลัพธ์ที่ได้จึงมีค่าเป็นตัวอักษรหรือข้อความเสมอ เช่น ให้ตัวแปร name เก็บชื่อ Adisak จะเขียนได้ดังนี้ name = "Adisak";

10. ตัวดำเนินการ (Operator) หมายถึง เครื่องหมายกำหนดกรรมวิธีทางคณิตศาสตร์ พีชคณิต บูลีน การเปรียบเทียบ ระหว่างข้อมูล 2 ตัว ซึ่งเรียกว่า Operand โดยอาจมีค่าเป็นตัวเลข ข้อความ ค่าคงที่ หรือตัวแปรต่าง ๆ

11. ชนิดของตัวดำเนินการ

- ตัวดำเนินการคณิตศาสตร์ (Arithmetic operator) หมายถึง ใช้สำหรับคำนวณโอเปรานด์ที่เป็นค่าคงที่หรือตัวแปรก็ได้ โดยให้ค่าผลลัพธ์เป็นตัวเลขค่าเดียว
- ตัวดำเนินการเชิงเปรียบเทียบ (Comparison operator) หมายถึง เครื่องหมายในการเปรียบเทียบข้อมูล ผลลัพธ์ที่ได้จะมีค่าตรรกะบูลีนเป็น จริง (True) และ เท็จ (False)
- ตัวดำเนินการเชิงตรรกะ (Logical operator) เป็นเครื่องหมายที่ให้ค่าจริง (True) และ เท็จ (False) ในการเปรียบเทียบ
- ตัวดำเนินการเชิงข้อความ (String operator) เป็นการเชื่อมโยงข้อความเข้าด้วยกัน (concatenation) โดยใช้เครื่องหมายบวก (+) เป็นตัวกระทำ ดังรูป 2.11

```

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 โดยวัตถุประสงค์หลังของการทำงานสำหรับสร้างแอปพลิเคชันในฝั่งไคลเอนต์ในรูปแบบของ HTML, CSS และ JavaScript/TypeScript ซึ่ง TypeScript จะถูก compile ไปเป็น JavaScript

แองกูลาร์เป็นเฟรมเวิร์คช่วยจัดการและทำให้การพัฒนาเว็บง่ายขึ้น เน้นพัฒนาส่วนติดต่อผู้ใช้ของเว็บ เช่น components, declarative UI, hot-reloading, time-travel debugging และอื่น ๆ ให้สะดวกต่อการใช้งาน พยายามลดสิ่งที่ไม่จำเป็นต่อการพัฒนาเว็บออกทำให้มีขนาดเล็กและง่ายต่อการนำไปใช้ของนักพัฒนา

2.4.1 การติดตั้ง

การติดตั้งเพื่อใช้งานแองกูลาร์จำเป็นต้องติดตั้ง Nodejs เวอร์ชัน 6.9 ขึ้นไป และติดตั้ง npm เวอร์ชัน 3 ขึ้นไป เมื่อตรวจสอบการติดตั้งว่าพร้อมแล้ว ก็สามารถติดตั้งและรันแองกูลาร์ได้ดังวิธีต่อไปนี้

- 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 คุณสมบัติของแองกูลาร์

ในหัวข้อนี้จะขอกล่าวถึงคุณลักษณะของแองกูลาร์เวอร์ชัน 8 เพราะเป็นเวอร์ชันที่ใช้พัฒนาระบบนี้

- 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 ดังรูป 2.12

```
1 ng new project --enable-ivy
```

รูปที่ 2.12: สร้างโปรเจคใหม่แบบ Ivy

2.4.3 ข้อดีของ Angular

- ช่วยแยก Logic การตัดสินใจออกจากโค้ด (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 และโดยเฉพาะในปัจจุบันระบบงานมีรูปแบบข้อมูลหลากหลายมาก มีปริมาณสูง ทำให้ระบบจัดเก็บข้อมูลต้องพร้อมรองรับได้ โดยสาเหตุหลักที่สมควรเลือกใช้ฐานข้อมูล MongoDB คือ เพื่อปรับปรุงการพัฒนาระบบฐานข้อมูล และปรับปรุงประสิทธิภาพในการเข้าถึงข้อมูล เมื่อข้อมูลมีปริมาณมาก เพื่อลดเวลาในการรับส่งข้อมูล และเพิ่มจำนวน transaction/request ที่ถูกสร้างขึ้น

2.5.2 Schema-less คืออะไร

Schema-less คือ การไม่กำหนดโครงสร้างของฐานข้อมูล NoSQL ซึ่งมีความแตกต่างจาก SQL ปกติ เช่นแสดงในรูป 2.13 Collection มีแค่ name ต่อมาเราสามารถเพิ่มการเก็บ position ก็สามารเพิ่มได้เลย

```

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

รูปที่ 2.13: ตัวอย่างข้อมูลใน Collection

2.6 เอกสารและงานวิจัยที่เกี่ยวข้อง

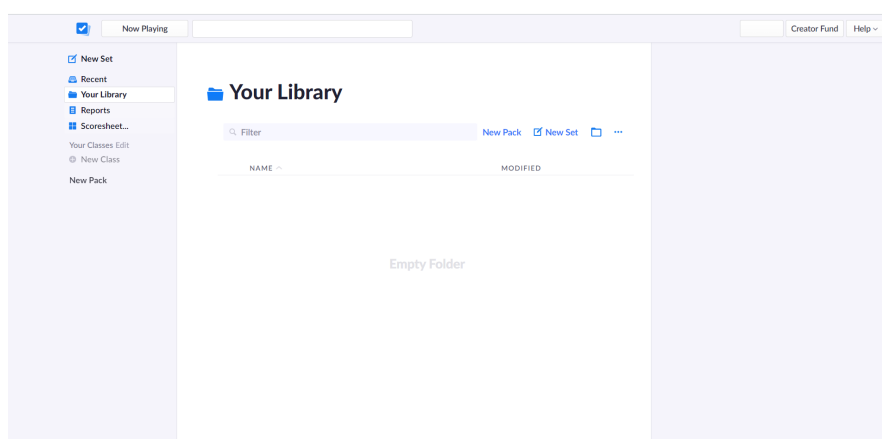
กล่าวถึงเอกสาร งานวิจัย หรือระบบงานที่คล้ายกัน โดยแต่ละหัวข้อให้อธิบายความสำคัญ ฟังก์ชันการทำงาน ข้อดี และข้อเสีย ดังต่อไปนี้

2.6.1 เว็บไซต์ Plickers

Plickers [6] เป็นเว็บไซต์ให้บริการอาจารย์สำหรับการจัดการเรียนการสอน ซึ่งภายในเว็บจะสามารถสร้างชั้นเรียนได้ โดยอาจารย์เป็นคนเพิ่มนักศึกษาภายในชั้นเรียนเข้าระบบ และสามารถเช็คชื่อนักศึกษาได้โดยใช้คิวอาร์โค้ดที่ระบบสร้างมาให้นักศึกษาแต่ละคน อีกทั้งยังสามารถสร้างแบบฝึกหัดภายในชั้นเรียนได้ รูป 2.14 แสดงหน้าหลักในการใช้งานระบบ

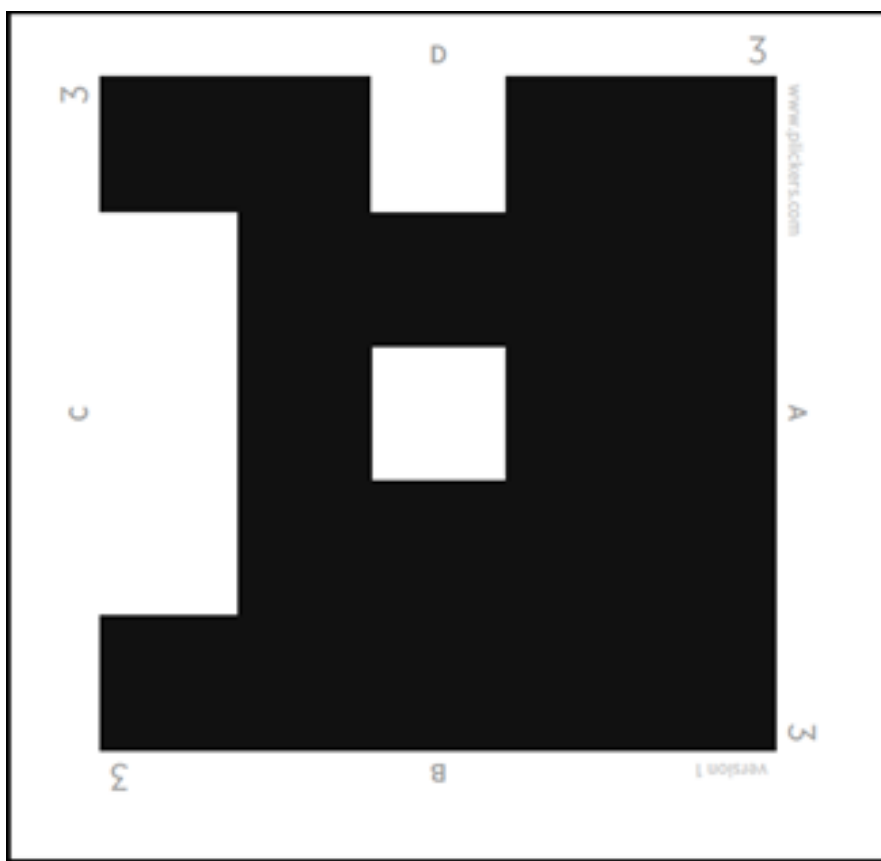
โดยคิวอาร์โค้ดที่ระบบสร้างมานั้นจะมีด้านที่แตกต่างกัน 4 ด้าน โดยแต่ละด้านมีตัวเลขเฉพาะของนักศึกษาแต่ละคนอยู่ที่มุมกระดาดังรูป 2.15 โดยการเช็คชื่ออาจารย์เป็นผู้สแกนคิวอาร์โค้ดของนักศึกษา

เว็บไซต์ plickers สามารถเข้าไปใช้งานได้ที่ <https://get.plickers.com> ซึ่งเป็นระบบเปิดให้ใช้งานได้ฟรี



รูปที่ 2.14: หน้าหลักในการใช้งานระบบ plickers

ที่มา : <https://get.plickers.com>



รูปที่ 2.15: คิวอาร์โค้ดของ plickers

ที่มา : <https://get.plickers.com>

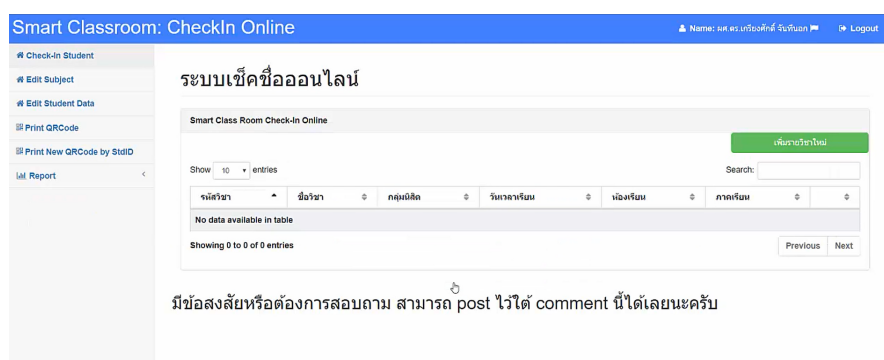
- ข้อดี
 - ฟังก์ชันการใช้งานหลากหลาย เช่น สามารถทำแบบฝึกหัดออนไลน์ได้ และสามารถใช้ในการเช็คชื่อเข้าเรียนได้ เป็นต้น
 - สามารถใช้ได้ทั้งคอมพิวเตอร์ และโทรศัพท์มือถือ โดยเข้าได้ทั้งในเว็บเบราว์เซอร์และแอปพลิเคชันบนโทรศัพท์มือถือ
 - ใช้งานได้ฟรี
- ข้อเสีย
 - ในการเช็คชื่อหรือการทำแบบฝึกหัดยังจำเป็นต้องพิมพ์คิวอาร์โค้ดให้นักเรียน นักศึกษาถือซึ่งถ้าหากคิวอาร์โค้ดชำรุดหรือขาด ก็จะไม่สามารถใช้งานคิวอาร์โค้ดได้

2.6.2 เว็บไซต์ smart classroom QR-Code Check-in

เว็บ smart classroom QR-Code Check-in [7] เป็นเว็บที่ให้บริการอาจารย์ในการเช็คชื่อนักศึกษา และเก็บข้อมูลนักศึกษาที่มาเรียนในรูปแบบออนไลน์บนเว็บเบราว์เซอร์

การทำงานของเว็บ smart classroom คือ อาจารย์สามารถสร้างชั้นเรียนในระบบได้ และจัดการข้อมูลต่าง ๆ ในชั้นเรียนได้ ซึ่งการเช็คชื่อจะทำได้ โดยอาจารย์เป็นผู้พิมพ์คิวอาร์โค้ดให้นักศึกษาทุกคนในชั้นเรียนถือ แล้วอาจารย์เป็นผู้สแกน

โดยเว็บ smart classroom QR-Code Check-in สามารถใช้งานได้ฟรีที่ <http://www.mse-exam.net> ซึ่งระบบเปิดให้ใช้งานได้ฟรี



รูปที่ 2.16: หน้าแรกของเว็บไซต์ smart classroom QR-Code Check-in

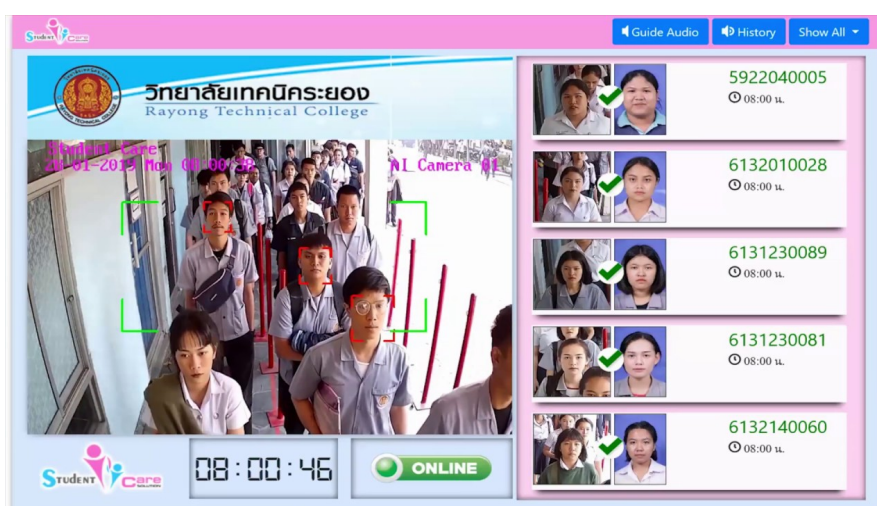
ที่มา : <http://www.mse-exam.net/qr>

- ข้อดี
 - ใช้งานง่าย และฟรี
 - สามารถใช้ได้ทั้งคอมพิวเตอร์ และโทรศัพท์มือถือ โดยเข้าได้ทั้งในเว็บเบราว์เซอร์และแอปพลิเคชันบนโทรศัพท์มือถือ
- ข้อเสีย
 - ในการเช็คชื่อหรือการทำแบบฝึกหัดยังจำเป็นต้องพิมพ์คิวอาร์โค้ดให้นักเรียน นักศึกษาถือซึ่งถ้าหากคิวอาร์โค้ดชำรุดหรือขาด ก็จะไม่สามารถใช้งานคิวอาร์โค้ดได้

2.6.3 โปรแกรม Student Care

โปรแกรม Student Care [8] เป็นโปรแกรมที่ให้บริการเกี่ยวกับการเช็คชื่อนักเรียนในการมาโรงเรียน ซึ่งโปรแกรมจะเป็นระบบสแกนใบหน้าผ่านกล้อง AI โดยระบบจะตรวจจับใบหน้านักเรียน แล้วตรวจสอบข้อมูลในฐานข้อมูล ซึ่งอำนวยความสะดวกให้กับคุณครูและนักเรียนในความรวดเร็วของการเช็คชื่อ ดังแสดงในรูป 2.17 นอกจากนี้ยังมีระบบแจ้งเตือนให้ผู้ปกครองว่านักเรียนมาเรียนหรือไม่ โดยระบบจะแจ้งเตือนผ่านทาง Line Application

โดยโปรแกรม Student Care สามารถติดต่อสอบถามได้ที่ <http://www.student.co.th> ซึ่งเว็บไซต์สำหรับติดต่อเจ้าหน้าที่ที่ติดตั้งโปรแกรม Student Care



รูปที่ 2.17: โปรแกรม Student Care

ที่มา : <http://www.student.co.th/>

- ข้อดี
 - มีเทคโนโลยีที่ทันสมัยและใช้งานง่าย
 - มีการแจ้งเตือนถึงผู้ปกครองของนักเรียน
- ข้อเสีย
 - การจะใช้งานระบบ Student Care จำเป็นต้องเสียค่าบริการรายเดือน
 - ระบบสแกนใบหน้ายังไม่สามารถแยกภาพถ่ายกับคนจริงได้

บทที่ 3

การวิเคราะห์และออกแบบระบบ

การวิเคราะห์และออกแบบระบบก่อนดำเนินการจริงเป็นอีกหนึ่งขั้นตอนที่มีความสำคัญมาก เพราะการวิเคราะห์และออกแบบระบบนั้นเป็นการกระทำที่ทำให้ผู้พัฒนาเห็นรายละเอียดส่วนย่อยของงานทั้งหมด เพิ่มประสิทธิภาพในการวางแผน การทำงาน และยังช่วยลดปัญหาที่อาจเกิดขึ้นในระหว่างพัฒนา เพื่อให้ระบบมีความสมบูรณ์มากยิ่งขึ้น เนื่องจากการวิเคราะห์และออกแบบระบบนั้นจะช่วยให้ให้บริการ จัดการทรัพยากรได้อย่างคุ้มค่าและตรงตามความต้องการของระบบ

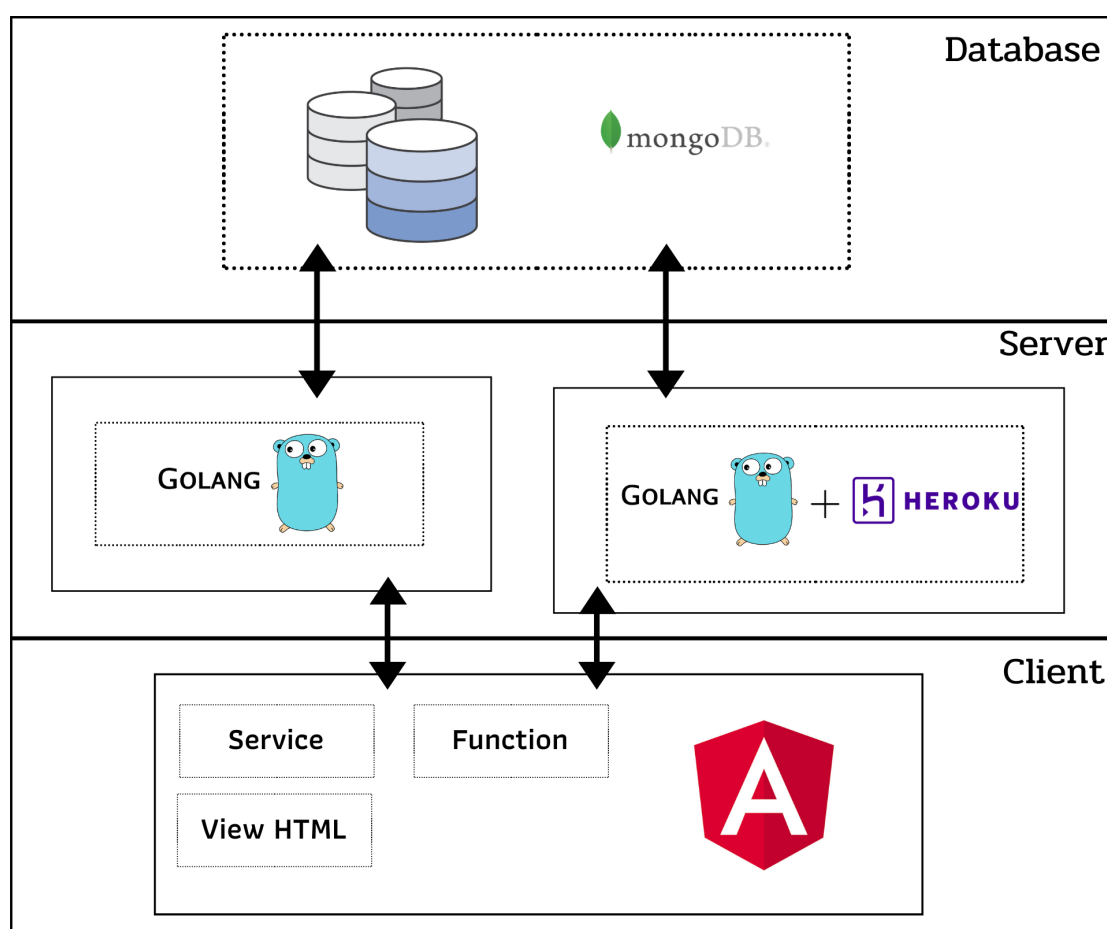
การวิเคราะห์และออกแบบระบบใช้เครื่องมือด้วยคิวอาร์โค้ด ในบทนี้จะมีขั้นตอนเพื่อให้เห็นการดำเนินงานอย่างมีระบบ ในหัวข้อแรกจะนำเสนอภาพรวมของระบบ ก่อนจะนำเสนอเอกสารแสดงความต้องการของระบบซึ่งจะทำให้เห็นที่มาของเพจต่าง ๆ ในขั้นตอนของการออกแบบในหัวข้อที่สาม ส่วนหัวข้อที่เหลือจะแสดงแผนภาพการทำงานของระบบโดยใช้ 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 [9] หมายถึง กรอบโครงสร้างของระบบที่อธิบายความสัมพันธ์ขององค์ประกอบต่าง ๆ ไปจนถึงขั้นการเชื่อมต่อกันของระบบย่อยต่าง ๆ โดยจัดกลุ่มองค์ประกอบไว้ในหลาย ๆ ลักษณะเพื่อให้ผู้เกี่ยวข้อง Stakeholder จากพื้นฐานสาขาอาชีพที่แตกต่าง กันสามารถทำความเข้าใจได้ง่าย เช่น การจัดแบ่งองค์ประกอบตามลักษณะการทำงานของระบบ (functional components) เป็นต้น

การออกแบบ System architecture แสดงภาพรวมและเทคโนโลยีของระบบใช้คีย์ด้วยคิวอาร์โค้ด มีรายละเอียดดังรูปที่ 3.1



รูปที่ 3.1: System architecture ระบบใช้คีย์ด้วยคิวอาร์โค้ด

จากรูปที่ 3.1 สามารถอธิบายโครงสร้างและเทคโนโลยีของระบบโดยแบ่งเป็น 3 ส่วนหลัก ดังนี้

1. Database ระบบให้บริการฐานข้อมูลแบบ NoSQL ของ MongoDB
2. Server กระบวนการทำงานในส่วนของเซิร์ฟเวอร์ (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
 - สามารถสร้าง และเพิ่มชั้นเรียนได้
 - สามารถตรวจสอบรายชื่อของนักศึกษาได้
 - สามารถจัดการคะแนนของนักศึกษาได้
 - สามารถสร้างคิวอาร์โค้ดให้นักศึกษาสแกนได้
 - สามารถกำหนดเวลาของคิวอาร์โค้ดได้
 - สามารถเพิ่ม ลบ และแก้ไข ชื่อกับรหัสนักศึกษาได้
 - สามารถดาวน์โหลดไฟล์รายชื่อของนักศึกษาได้
 - สามารถรวมไฟล์คะแนน สำหรับวิชาเดียวกันได้
2. นักศึกษา
 - สามารถสแกนคิวอาร์โค้ดของอาจารย์เพื่อยืนยันการมาเรียนได้

3.2.2 Non-functional Requirements

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

ความสัมพันธ์ระหว่างคอมพิวเตอร์ของผู้ใช้กับเว็บไซต์

- 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 3.3 แสดงหน้าจอสร้างชั้นเรียนสำหรับอาจารย์ใช้ทำการ เพิ่มชั้นเรียนใหม่ โดยการเพิ่มไฟล์รายชื่อนักศึกษาและใส่ข้อมูลให้ครบ

3.3.1.3 การออกแบบหน้าจัดการคะแนนนักศึกษา

รูปที่ 3.4: หน้าจอจัดการคะแนนนักศึกษา

จากภาพที่ 3.4 แสดงหน้าจอจัดการคะแนนนักศึกษาเพื่อตรวจสอบคะแนนของนักศึกษาและจัดการคะแนนของนักศึกษา

3.3.1.4 การออกแบบหน้าประวัติการเข้าเรียน

MENU

เพิ่มชั้นเรียน

CHECK NAME

ClassRoom Record

รหัสวิชา :
ชื่อวิชา :
วันเวลาเรียน :

ดาวน์โหลดไฟล์ รวมคะแนน

| เลขที่ | รหัสนักศึกษา | ชื่อ | ลำดับ |
|--------|--------------|------|-------|
| | | | |

Prev 1 2 Next

รูปที่ 3.5: หน้าจอประวัติการเข้าเรียน

จากภาพที่ 3.5 แสดงหน้าจอประวัติการเข้าเรียนเพื่อดูข้อมูลการเข้าเรียนของนักศึกษา และในหน้านี้ยังใช้ในการเข้าถึงการดาวน์โหลดไฟล์ หรือเข้าถึงหน้ารวมไฟล์ได้

3.3.1.5 การออกแบบหน้ารวมไฟล์

MENU

เพิ่มชั้นเรียน

CHECK NAME

เลือกไฟล์

| เลขที่ | รหัสนักศึกษา | ชื่อ | ลำดับ |
|--------|--------------|------|-------|
| | | | |

Prev 1 2 Next

บันทึก

Prev 1 2 Next

รูปที่ 3.6: หน้ารวมไฟล์

จากภาพที่ 3.6 แสดงหน้าจอหน้ารวมไฟล์เป็น Pop-up เพื่อจัดการรวมไฟล์โดยการเลือกไฟล์ที่จะทำการรวม ซึ่งไฟล์ที่จะสามารถทำการรวมไฟล์ได้นั้นต้องเป็นวิชาเดียวกันเท่านั้น

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 หน้านี้จะแสดงก็ต่อเมื่ออาจารย์ตั้งค่าและกดสร้างคิวอาร์โค้ดเท่านั้น โดยจะแสดงคิวอาร์โค้ดที่สร้างรวมถึงเวลาที่เหลือในการสแกน

3.3.1.8 การออกแบบหน้าจัดการนักศึกษา

MENU

เพิ่มนักเรียน

CHECK NAME

Update Student

เพิ่ม นักศึกษา

ค้นหา : รายชื่อนักศึกษา

| เลขที่ | รายนามนักศึกษา | ชื่อ | Update |
|--------|----------------|------|----------|
| | | | แก้ไข ลบ |

Prev 1 2 Next



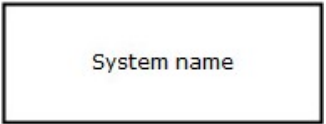
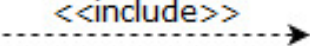
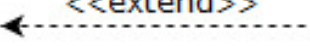
รูปที่ 3.9: หน้าจัดการนักศึกษา

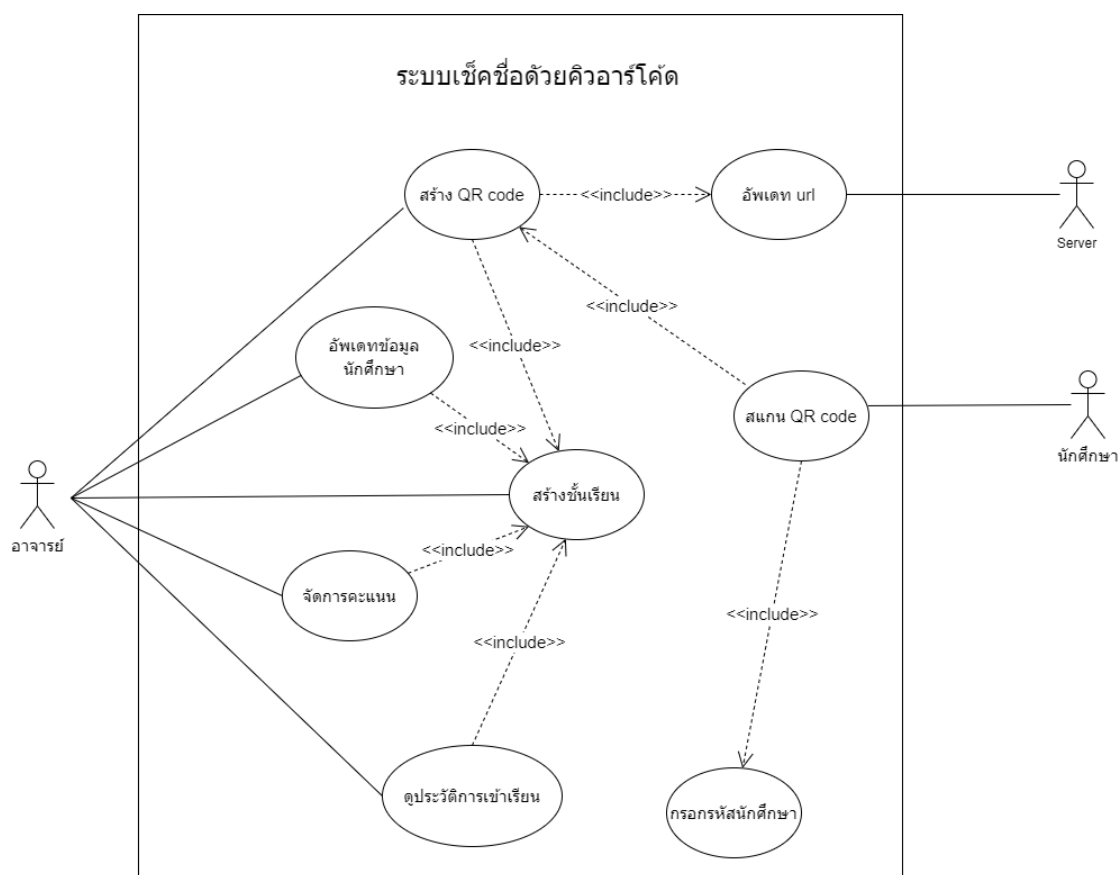
จากภาพที่ 3.9 แสดงหน้าจอจัดการนักศึกษาเพื่อจัดการข้อมูลของนักศึกษา เช่น แก้ไข ลบ และเพิ่มนักศึกษา เป็นต้น

3.4 Use Case Diagram

Use Case Diagram เป็นแผนผังเพื่อแสดงฟังก์ชันการทำงานของระบบโดยรวม แสดงส่วนประกอบในระบบและกิจกรรมที่เกิดขึ้นในระบบซึ่งในระบบใช้คี่ด้วยคิวอาร์โค้ด 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 |
|  | กรอบสี่เหลี่ยม แสดงถึงขอบเขตของระบบ โดยแสดงชื่อระบบภายในหรือด้านบนกรอบสี่เหลี่ยม Use case อยู่ภายในกรอบสี่เหลี่ยม และ actor อยู่ภายนอกกรอบสี่เหลี่ยม |
|  | ความสัมพันธ์แบบ «includes» แสดงว่า Use case หนึ่งดำเนินการตามขั้นตอนของ Use case อื่น โดยแทนด้วยสัญลักษณ์ลูกศรเส้นประ 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 โดยการที่จะสร้างชั้นเรียนใหม่จำเป็นจะต้องอัปโหลดไฟล์ที่ดาวน์โหลดมาจากระบบของมหาวิทยาลัยเท่านั้น ซึ่งไฟล์จะต้องเป็นไฟล์ Excel และไม่สามารถสร้างชั้นเรียนซ้ำได้ |
| อัปเดตข้อมูล นักศึกษา | อาจารย์สามารถจัดการรายชื่อนักศึกษาได้โดยการ เพิ่ม ลบ และแก้ไข ข้อมูลชื่อและรหัสนักศึกษาได้ |
| สร้าง QR code | อาจารย์สามารถสร้าง QR code รวมถึงการตั้งค่า QR code เช่น เวลาในการสแกน QR code เป็นต้น |
| จัดการคะแนน | หน้า แสดง ราย ชื่อ และ คะแนน แต่ละ สัปดาห์ ของ นักศึกษา โดยอาจารย์สามารถจัดการกับ คะแนน ในแต่ละ สัปดาห์ ของ นักศึกษาแต่ละคน |
| ดูประวัติการเข้าเรียน | อาจารย์สามารถตรวจสอบการเข้าเรียนของนักศึกษาทั้งหมดในชั้นเรียนนั้น โดยจะแสดงเป็นรายสัปดาห์ |
| สแกน QR code | นักศึกษาสามารถสแกน QR code ของอาจารย์เพื่อเข้าสู่หน้ากรอกรหัสนักศึกษาเพื่อเป็นการยืนยันตัวตนในการมาเรียน โดยการจะสแกนได้นั้นขึ้นอยู่กับเวลาของ QR code ของอาจารย์ด้วย ซึ่งถ้าหากเวลาของ QR code หมด นักศึกษาก็ไม่สามารถสแกน QR code ได้ |
| อัปเดต url | อัปเดต 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 : หากผู้ใช้ไม่มีไฟล์รายชื่อนักศึกษาที่ดาวน์โหลดมาจากระบบของมหาวิทยาลัย จะไม่สามารถสร้างหรือเพิ่มชั้นเรียนใหม่ได้ | |

ตารางที่ 3.4: Use Case อัปเดตข้อมูลนักศึกษา

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

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 จะอาจารย์ไม่สามารถสร้าง QR code ได้ | |
| Exceptional Flow ที่ 2 : หากผู้ใช้ไม่เชื่อมต่ออินเทอร์เน็ต จะอาจารย์ไม่สามารถสร้าง 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 จะไม่สามารถจัดการคะแนนของนักศึกษาได้ | |

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

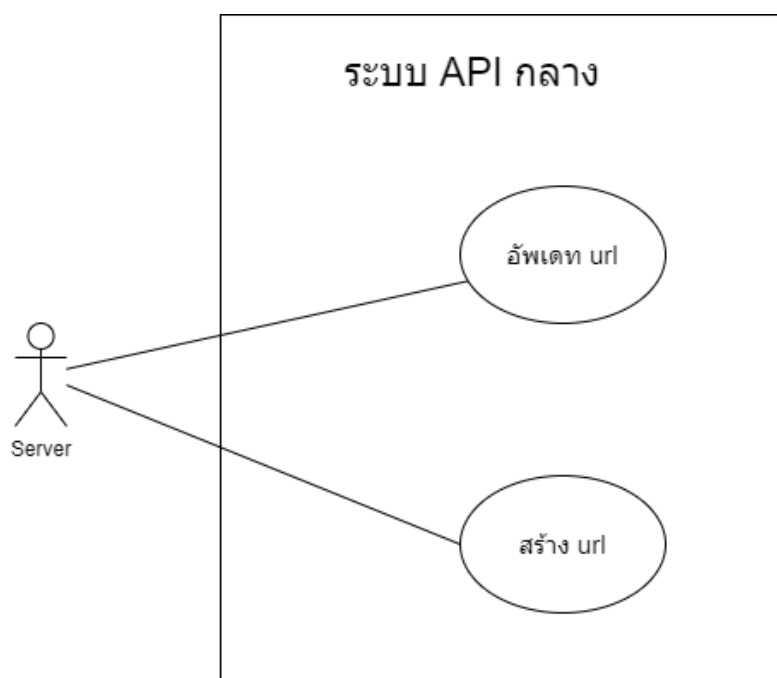
| | |
|--|-----------------|
| Use Case Title : สแกน QR code | Use case Id : 6 |
| Primary Actor : นักศึกษา | |
| Stakeholder Actor : อาจารย์ | |
| Main Flow : สแกน QR code ที่อาจารย์สร้างขึ้น | |
| Exceptional Flow ที่ 1 : หากผู้ใช้ไม่เชื่อมต่อ Wifi เดียวกันกับอาจารย์ จะไม่สามารถสแกน QR code | |

ตารางที่ 3.9: Use Case อัปเดต URL

| | |
|---|-----------------|
| Use Case Title : อาจารย์ | Use case Id : 7 |
| Primary Actor : อาจารย์ | |
| Stakeholder Actor : - | |
| Main Flow : อัปเดต url ของคิวอาร์โค้ด | |
| Exceptional Flow ที่ 1 : ผู้ใช้ต้องทำการสร้างคิวอาร์โค้ดก่อน จึงจะสามารถอัปเดต URL ได้ | |
| Exceptional Flow ที่ 2 : ผู้ใช้จำเป็นต้องเชื่อมต่ออินเทอร์เน็ต จึงจะทำการอัปเดต 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 จะไม่สามารถมาที่หน้ากรอกรหัสนักศึกษาได้ | |
| Exceptional Flow ที่ 1 : หากผู้ใช้ไม่เชื่อมต่อ Wifi เดียวกันกับอาจารย์ จะไม่สามารถมาที่หน้ากรอกรหัสนักศึกษาได้ | |



รูปที่ 3.11: Use Case Diagram ของระบบ API กลาง

ตารางที่ 3.11: อธิบาย Use Case หน้าที่ของระบบ ในภาพที่ 3.11

| Use Case | คำอธิบาย |
|------------|--|
| อัปเดต url | เมื่ออาจารย์สร้างคิวอาร์โค้ดระบบจะส่ง URL ไปเก็บไว้ในฐานข้อมูลเพื่อให้นักศึกษาสแกน |
| สร้าง url | เมื่ออาจารย์สร้างคิวอาร์โค้ดครั้งแรกระบบจะส่ง URL ไปเก็บไว้ในฐานข้อมูลเพื่อให้นักศึกษาสแกน |

ตารางที่ 3.12: Use Case อัปเดต URL

| | |
|---|-----------------|
| Use Case Title : อัปเดต url | Use case Id : 9 |
| Primary Actor : อาจารย์ | |
| Stakeholder Actor : - | |
| Main Flow : อัปเดต url ของคิวอาร์โค้ด | |
| Exceptional Flow ที่ 1 : ผู้ใช้ต้องทำการสร้างคิวอาร์โค้ดก่อน จึงจะสามารถอัปเดต URL ได้ | |
| Exceptional Flow ที่ 2 : ผู้ใช้จำเป็นต้องเชื่อมต่ออินเทอร์เน็ต จึงจะทำการอัปเดต URL ได้ | |



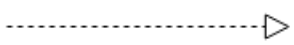
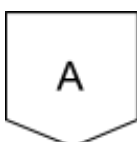
ตารางที่ 3.13: Use Case สร้าง URL

| | |
|--|------------------|
| Use Case Title : สร้าง url | Use case Id : 10 |
| Primary Actor : อาจารย์ | |
| Stakeholder Actor : - | |
| Main Flow : อัปเดต url ของคิวอาร์โค้ด | |
| Exceptional Flow ที่ 1 : ผู้ใช้ต้องทำการสร้างคิวอาร์โค้ดก่อน จึงจะสามารถสร้าง URL ได้ | |
| Exceptional Flow ที่ 2 : ผู้ใช้จำเป็นต้องเชื่อมต่ออินเทอร์เน็ต จึงจะทำการสร้าง URL ได้ | |

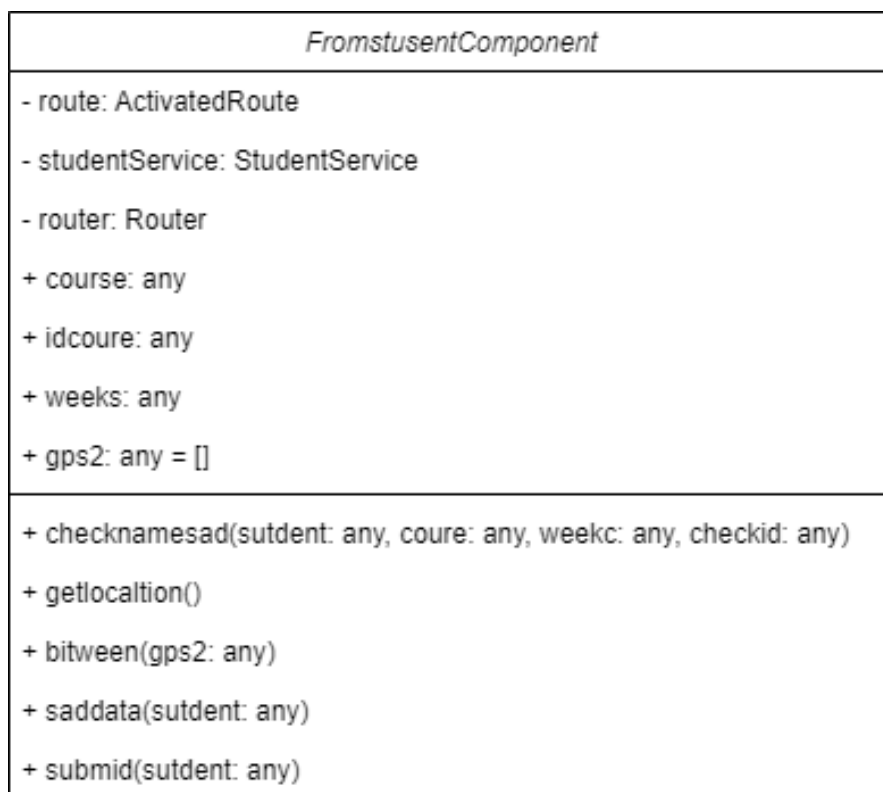
3.5 Class Diagram

Class Diagram คือแผนภาพที่ใช้แสดงคลาสและความสัมพันธ์ในแบบต่างๆ ระหว่างคลาส สัญลักษณ์ที่ใช้ในการเขียน Class Diagram แสดงในตารางที่ 3.14

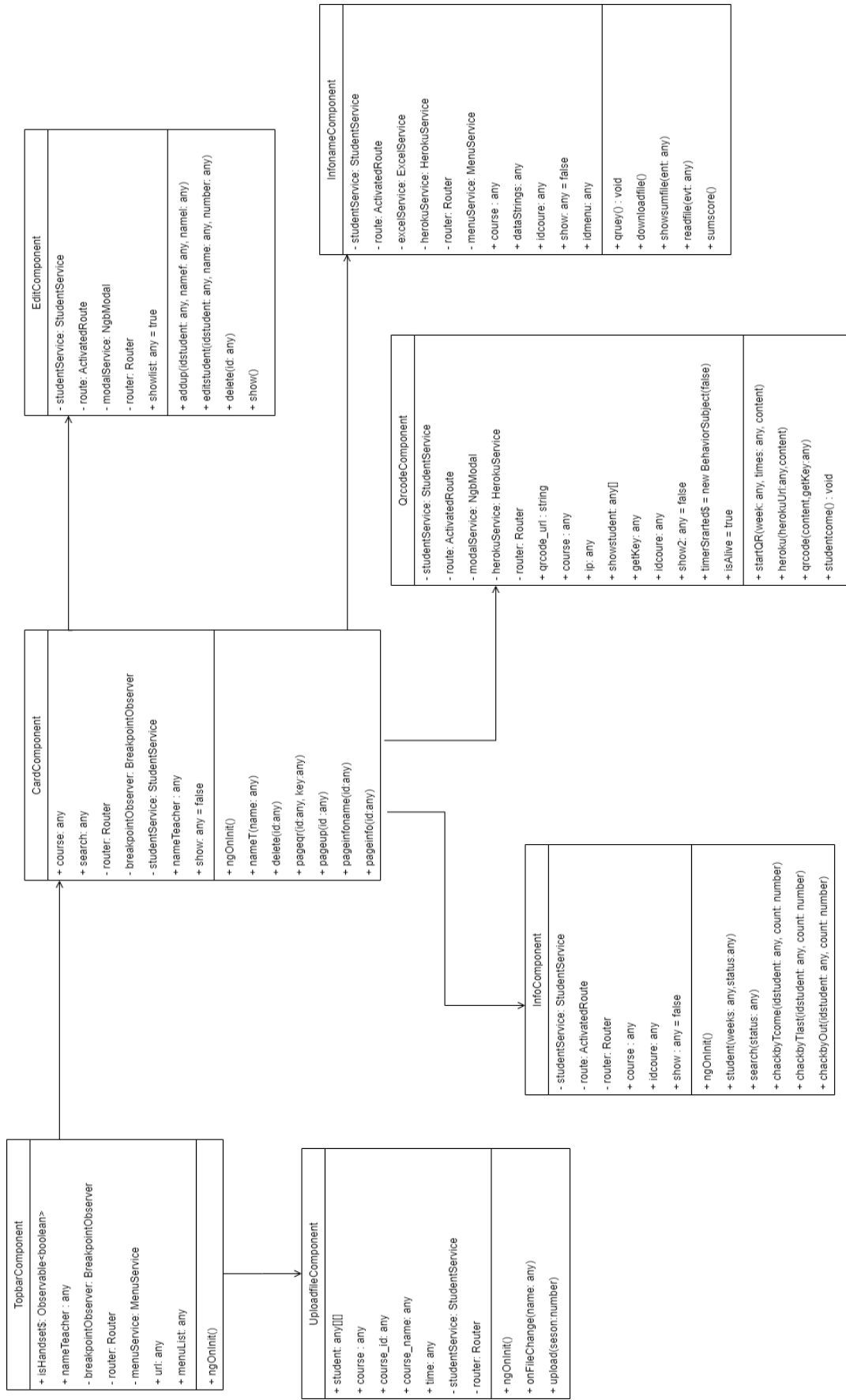
ตารางที่ 3.14: สัญลักษณ์ของ Class Diagram

| สัญลักษณ์ | การใช้งาน |
|---|--|
| <div style="border: 1px solid black; padding: 5px; margin: 10px auto; width: fit-content;"> <div style="border-bottom: 1px solid black; padding-bottom: 5px; text-align: center;">Class Name</div> <div style="border-bottom: 1px solid black; padding: 5px 0 5px 20px; text-align: center;">Attribute Name</div> <div style="padding: 5px 0 5px 20px; text-align: center;">Operation Name()</div> </div> | <p>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</p> <ol style="list-style-type: none"> 1. Public แทนสัญลักษณ์ด้วยเครื่องหมายบวก (+) 2. Private แทนสัญลักษณ์ด้วยเครื่องหมายลบ (-) 3. Protected แทนสัญลักษณ์ด้วยเครื่องหมายชาร์ป () |
|  | Dependency Relationship หมายความว่า คลาสที่อยู่ฝั่งต้นลูกศรสามารถเรียกใช้คลาสที่อยู่ฝั่งหัวลูกศร |
|  | Composition Relationship เป็นความสัมพันธ์ระหว่างออบเจกต์หรือคลาสแบบขึ้นต่อกันและมีความเกี่ยวข้องกันเสมอ |
|  | Realization Relationship เป็นความสัมพันธ์ระหว่าง Object หรือ Class ในลักษณะของการสืบทอดคุณสมบัติจาก Class หนึ่ง (Super class) ไปยังอีก Class หนึ่ง (Subclass) |
|  | Connector เป็นสัญลักษณ์แทนด้วยรูปห้าเหลี่ยมและมีชื่ออยู่ตรงกลาง จะสร้างสัญลักษณ์นี้ไว้เมื่อต้องการเชื่อมต่อคลาสที่อยู่คนละหน้า |

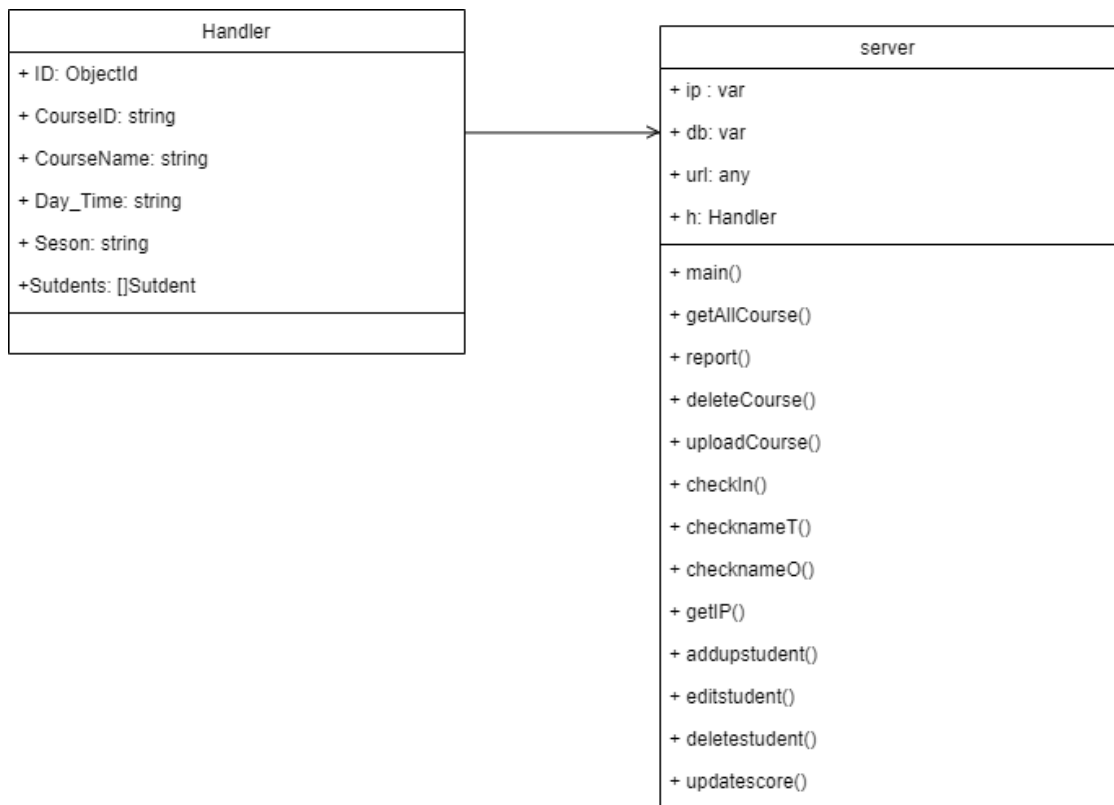
Class Diagram แสดงความสัมพันธ์ในรูปแบบต่างๆ ระหว่างคลาสของเว็บแอปพลิเคชันระบบเช็คชื่อด้วยคิวอาร์โค้ด อธิบายได้ตามภาพที่ 3.12 ดังต่อไปนี้



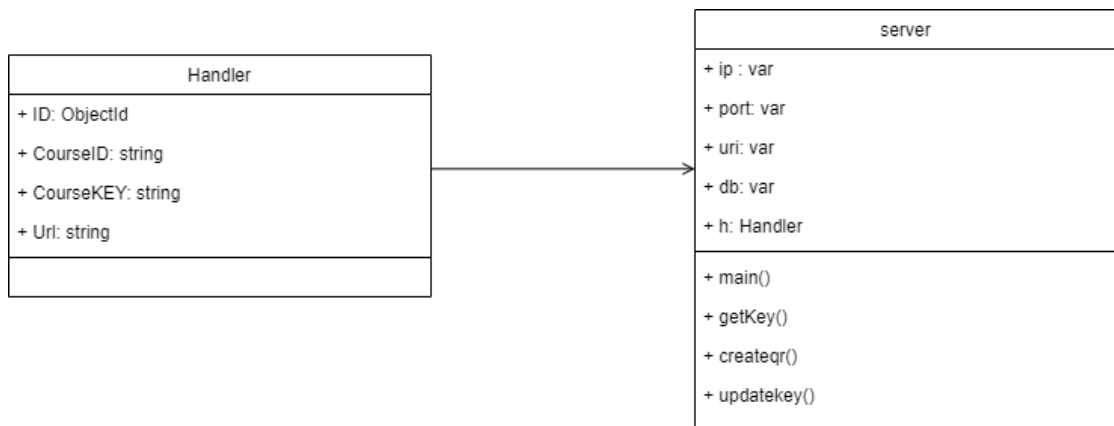
รูปที่ 3.12: Class Diagram ของแอปพลิเคชันระบบ เช็คชื่อด้วยคิวอาร์โค้ด



รูปที่ 3.13: Class Diagram ของแอปพลิเคชันระบบ เชื้อด้วยคิวอาร์โค้ด



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 |
|----------------------|---|
| FromstudentComponent | คลาส FromstudentComponent จะถูกเรียกใช้งานทุกครั้งเมื่อนักศึกษาสามารถสแกนคิวอาร์โค้ดของอาจารย์ได้ โดยวัตถุประสงค์การทำงานของคลาสนี้คือ เพื่อให้นักศึกษายืนยันตัวตนเพื่อรับคะแนนในการเข้าเรียน |

จากรูปภาพที่ 3.13 สามารถอธิบายแผนภาพ Class Diagram ได้ดังนี้

ตารางที่ 3.16: อธิบาย Class Diagram ของคลาสส่วนของอาจารย์

| Class Diagram | คำอธิบาย |
|---------------------|--|
| TopbarComponent | คลาส TopbarComponent จะถูกเรียกใช้งานทุกครั้งเมื่ออาจารย์เข้าสู่เว็บแอปพลิเคชัน โดยวัตถุประสงค์ของการทำงานของคลาสนี้คือ จัดการเกี่ยวกับการแสดงแถบเมนูส่วนบนของเว็บแอปพลิเคชัน ซึ่งสามารถเรียกใช้งานคลาส CardComponent และ UploadfileComponent ได้ |
| UploadfileComponent | คลาส UploadfileComponent เป็นคลาสที่มีหน้าที่ในการทำงานเกี่ยวกับการเพิ่มไฟล์รายชื่อนักศึกษา โดยวัตถุประสงค์ของคลาสนี้คือ ใช้ในการเพิ่มชั้นเรียนใหม่ ซึ่งการเพิ่มชั้นเรียนต้องเพิ่มไฟล์ที่ได้จากระบบของมหาวิทยาลัยเท่านั้น นอกจากนี้ยังแสดงรายชื่อทั้งหมดในไฟล์ที่นำเข้าอีกด้วย |
| CardComponent | คลาส CardComponent จะถูกเรียกใช้งานทุกครั้งเมื่ออาจารย์เข้าสู่เว็บแอปพลิเคชัน โดยวัตถุประสงค์ของการทำงานของคลาสนี้คือ แสดง ชั้นเรียนทั้งหมดที่มี และแสดงเมนูสำหรับการเข้าถึง คลาส ต่างๆ 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 โดยวัตถุประสงค์ของคลาสนี้คือ การจัดการคะแนนของนักศึกษาทั้งหมดในรายวิชานั้นๆ |

จากรูปภาพที่ 3.14 สามารถอธิบายแผนภาพ Class Diagram ได้ดังนี้

ตารางที่ 3.18: อธิบาย Class Diagram ของคลาสส่วนของอาจารย์

| Class Diagram | คำอธิบาย |
|---------------|---|
| server | คลาส server จะถูกเรียกใช้งานทุกครั้งเมื่ออาจารย์เข้าสู่เว็บแอปพลิเคชัน โดยวัตถุประสงค์ของการทำงานของคลาสนี้คือให้บริการ API และตรวจสอบข้อมูลต่างๆ |
| Handler | คลาส Handler เป็นคลาสที่มีหน้าที่ในการเป็นโมเดลของฐานข้อมูล |

จากรูปภาพที่ 3.15 สามารถอธิบายแผนภาพ Class Diagram ได้ดังนี้

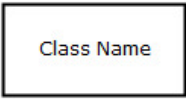


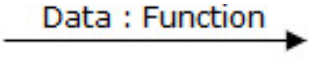
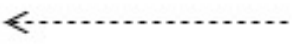


ตารางที่ 3.19: อธิบาย Class Diagram ของคลาสส่วนของอาจารย์

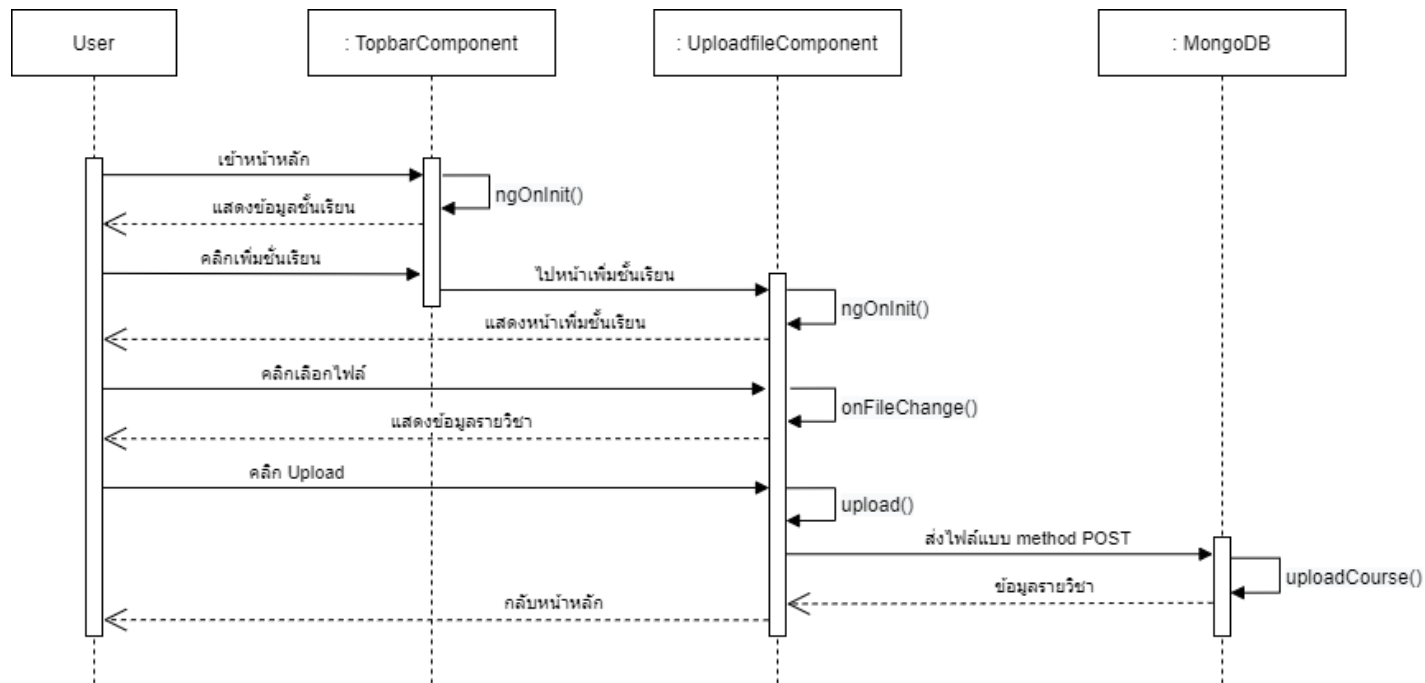
| Class Diagram | คำอธิบาย |
|---------------|---|
| server | คลาส server จะถูกเรียกใช้งานทุกครั้งเมื่ออาจารย์เข้าสู่เว็บแอปพลิเคชัน โดยวัตถุประสงค์ของการทำงานของคลาสนี้คือให้บริการ API และตรวจสอบข้อมูลต่างๆ |
| Handler | คลาส Handler เป็นคลาสที่มีหน้าที่ในการเป็นโมเดลของฐานข้อมูล |

3.6 Sequence Diagram

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

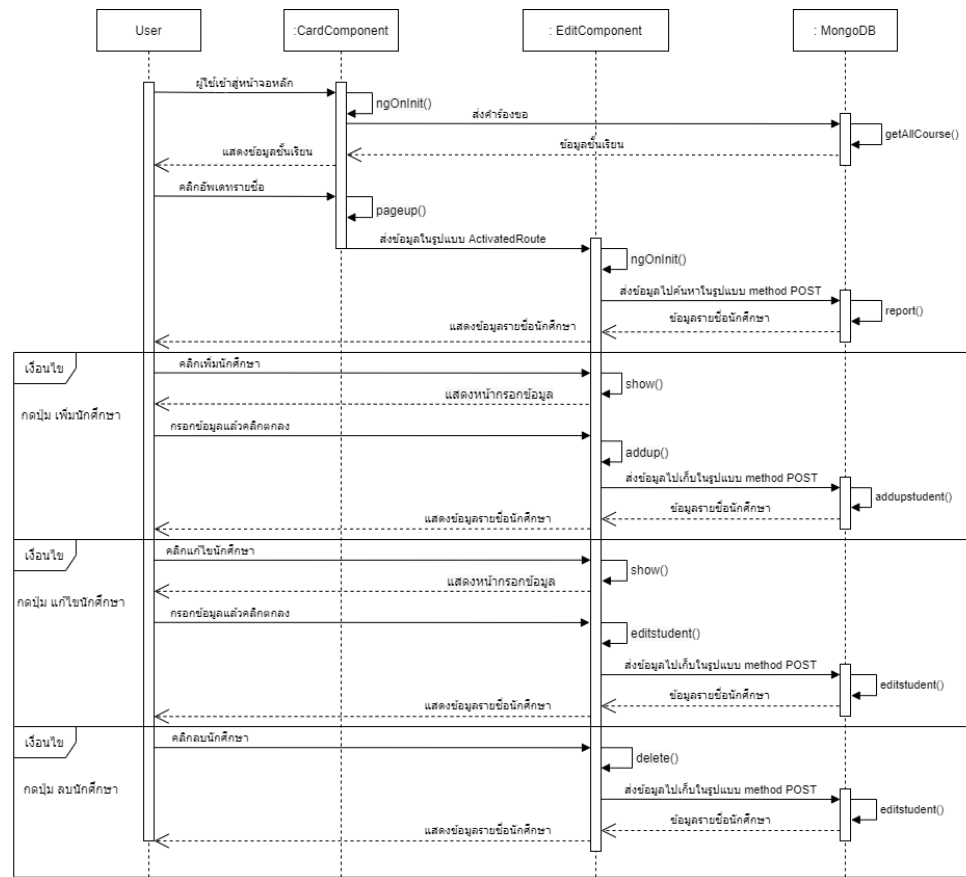
ตารางที่ 3.20: สัญลักษณ์ของ Sequence Diagram

| 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 เป็นการเรียกฟังก์ชันการทำงานภายในตัวเอง |
|  | สร้างกรอบการทำงานของโปรแกรม เพื่อให้รู้ขอบเขตของการทำงานเช่น ลูป(loop) |



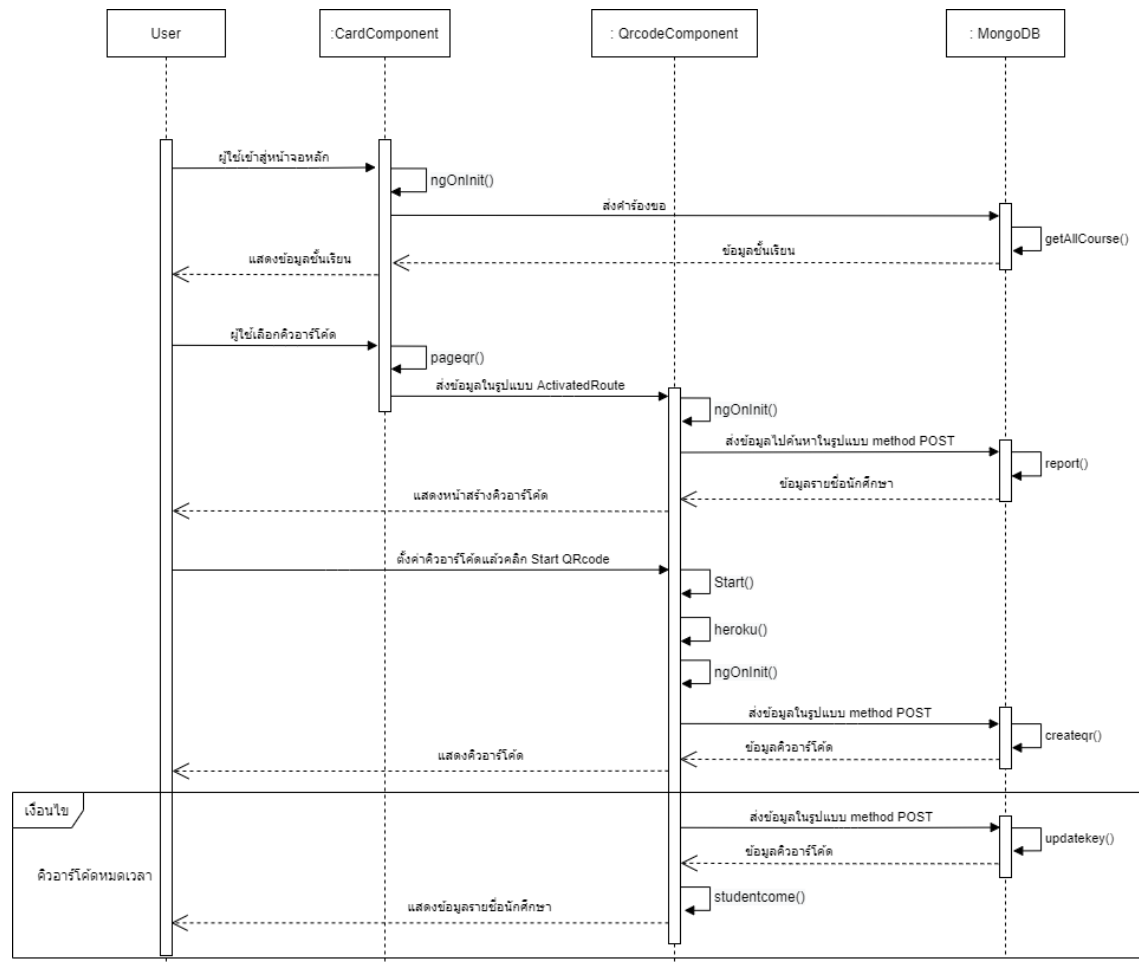
รูปที่ 3.16: Sequence Diagram สร้างชั้นเรียน

จากภาพที่ 3.16 สามารถอธิบายแผนภาพ Sequence Diagram สร้างชั้นเรียน ได้ดังนี้ เมื่อผู้ใช้เปิดโปรแกรมระบบจะเรียกใช้เมธอด `ngOnInit()` ที่คลาส `TopbarComponent` ระบบจะคืนค่าข้อมูลชั้นเรียนโดยจะแสดงเป็นการ์ด เมื่อผู้ใช้คลิกปุ่มเพิ่มชั้นเรียนระบบจะทำการไปที่หน้าเพิ่มชั้นเรียนจากนั้นจะเรียกใช้เมธอด `ngOnInit()` เพิ่มแสดงหน้าเพิ่มชั้นเรียน ผู้ใช้คลิกเลือกไฟล์ระบบจะเรียกใช้เมธอด `onFileChange()` ในคลาส `UploadfileComponent` โดยเมธอดนี้จะทำหน้าที่อ่านไฟล์ที่เลือกเข้ามาและเตรียมข้อมูลสำหรับอัปโหลด และแสดงข้อมูล เมื่อผู้ใช้คลิกปุ่ม Upload ระบบจะเรียกใช้เมธอด `Upload()` ที่คลาส `Upload fileComponent` ระบบจะส่งข้อมูลที่เตรียมไว้ส่งข้อมูลในรูปแบบ POST ไปที่ฐานข้อมูล ฐานข้อมูลจะเรียกใช้เมธอด `uploadCourse()` ซึ่งเมธอดนี้จะทำหน้าที่บันทึกข้อมูลลงฐานข้อมูล เมื่อบันทึกเสร็จจะคืนค่าข้อมูลชั้นเรียนจากนั้นระบบจะกลับสู่หน้าแรก



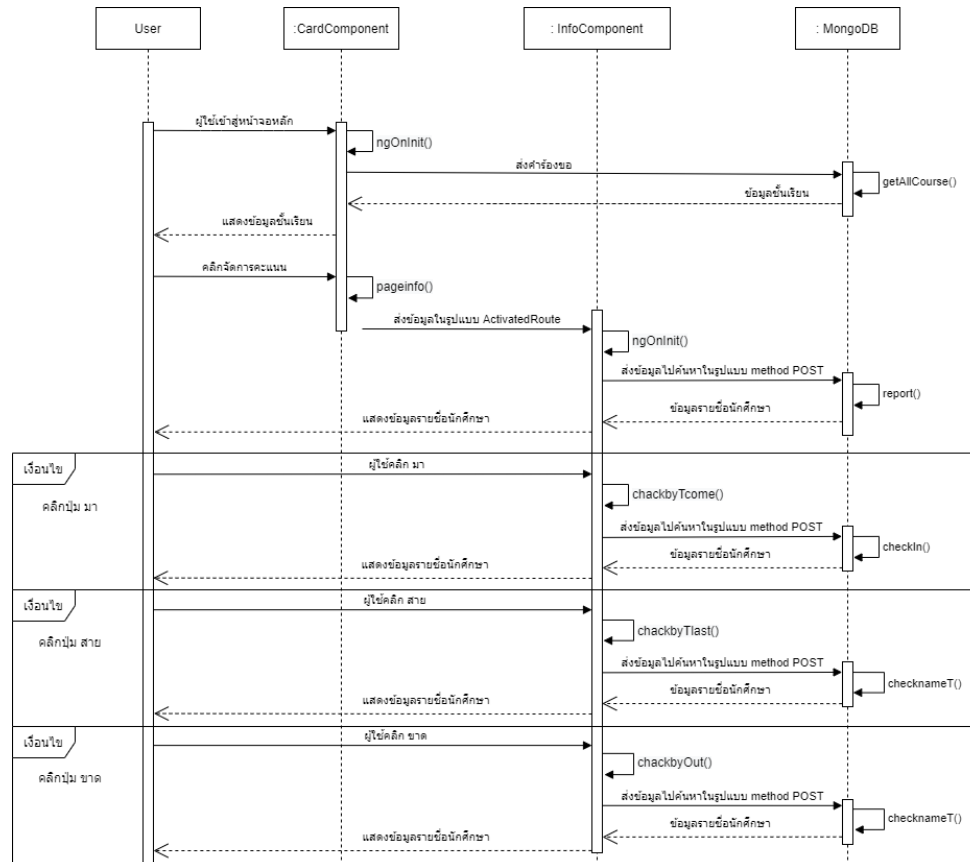
รูปที่ 3.17: Sequence Diagram อัปเดตข้อมูลนักศึกษา

จากภาพที่ 3.17 สามารถอธิบายแผนภาพ Sequence Diagram อัพเดทข้อมูลนักศึกษา ได้ดังนี้ เมื่อ ผู้ใช้เปิดโปรแกรมระบบจะเรียกใช้เมธอด `ngOnInit()` ที่คลาส `CardComponent` ระบบจะส่งคำร้องขอไปที่ฐานข้อมูล ซึ่งฐานข้อมูลจะเรียกใช้เมธอด `getAllCourse()` โดยจะทำหน้าที่ค้นหาและคืนค่าข้อมูลชั้นเรียนที่มีในฐานข้อมูล จากนั้นระบบจะรับค่าที่คืนมาจากฐานข้อมูลโดยจะแสดงเป็นการ์ด เมื่อผู้ใช้เลือกคิวอาร์โค้ดระบบจะเรียกใช้งานเมธอด `pageup()` ระบบจะไปที่หน้าสร้างคิวอาร์โค้ดโดยส่งข้อมูล `id` ของชั้นเรียนไปในรูปแบบ `ActivatedRoute` เมื่อมาที่หน้าสร้างคิวอาร์โค้ดแล้วระบบจะเรียกใช้เมธอด `ngOnInit()` ที่คลาส `EditComponent` ระบบจะส่งข้อมูล `id` ไปที่ฐานข้อมูลไปค้นหาในรูปแบบ `POST` ในเมธอด `report()` โดยเมธอดจะค้นหาชั้นเรียนตาม `id` ที่ได้มาและคืนค่าข้อมูลชั้นเรียนที่หาพบและระบบจะแสดงหน้าอัพเดทข้อมูลนักศึกษา ผู้ใช้งานคลิกเพิ่มนักศึกษาระบบจะเรียกใช้เมธอด `show()` ในคลาส `EditComponent` โดยเมธอดนี้จะทำการแสดงหน้ากรอกข้อมูลนักศึกษา เมื่อกรอกข้อมูลนักศึกษาแล้วกดปุ่มตกลงระบบจะเรียกใช้เมธอด `addup()` ในคลาส `EditComponent` โดยเมธอดนี้จะทำหน้าที่ส่งข้อมูลที่กรอกเข้ามาในหน้ากรอกข้อมูลนักศึกษาส่งไปในรูปแบบ `POST` เพื่อเก็บในฐานข้อมูล ซึ่งจะเรียกใช้เมธอด `addupstudent()` ในการบันทึกลงในฐานข้อมูลและจะคืนค่าข้อมูลนักศึกษาที่เพิ่มใหม่มาที่คลาส `EditComponent` จากนั้นจะแสดงข้อมูลรายชื่อนักศึกษา ต่อมาผู้ใช้งานคลิกปุ่มแก้ไขนักศึกษา ระบบจะเรียกใช้เมธอด `show()` ในคลาส `EditComponent` โดยเมธอดนี้จะทำการแสดงหน้ากรอกข้อมูลนักศึกษา เมื่อกรอกข้อมูลนักศึกษาแล้วกดปุ่มตกลงระบบจะเรียกใช้เมธอด `editstudent()` ในคลาส `EditComponent` โดยเมธอดนี้จะทำหน้าที่ส่งข้อมูลที่กรอกเข้ามาในหน้ากรอกข้อมูลนักศึกษาส่งไปในรูปแบบ `POST` เพื่อเก็บในฐานข้อมูล ซึ่งจะเรียกใช้เมธอด `editstudent()` ในการบันทึกลงในฐานข้อมูลและจะคืนค่าข้อมูลนักศึกษามาที่คลาส `EditComponent` จากนั้นจะแสดงข้อมูลรายชื่อนักศึกษา สุดท้ายผู้ใช้งานคลิกลบนักศึกษาระบบจะเรียกใช้เมธอด `delete()` ในคลาส `EditComponent` โดยเมธอดนี้จะทำหน้าที่ส่งข้อมูล `id` ชั้นเรียนและรหัสนักศึกษาไปในรูปแบบ `POST` เพื่อค้นหาและทำการลบข้อมูลในฐานข้อมูล ซึ่งจะเรียกใช้เมธอด `deletestudent()` ในการลบข้อมูลในฐานข้อมูลและจะคืนค่าข้อมูลนักศึกษามาที่คลาส `EditComponent` จากนั้นจะแสดงข้อมูลรายชื่อนักศึกษา



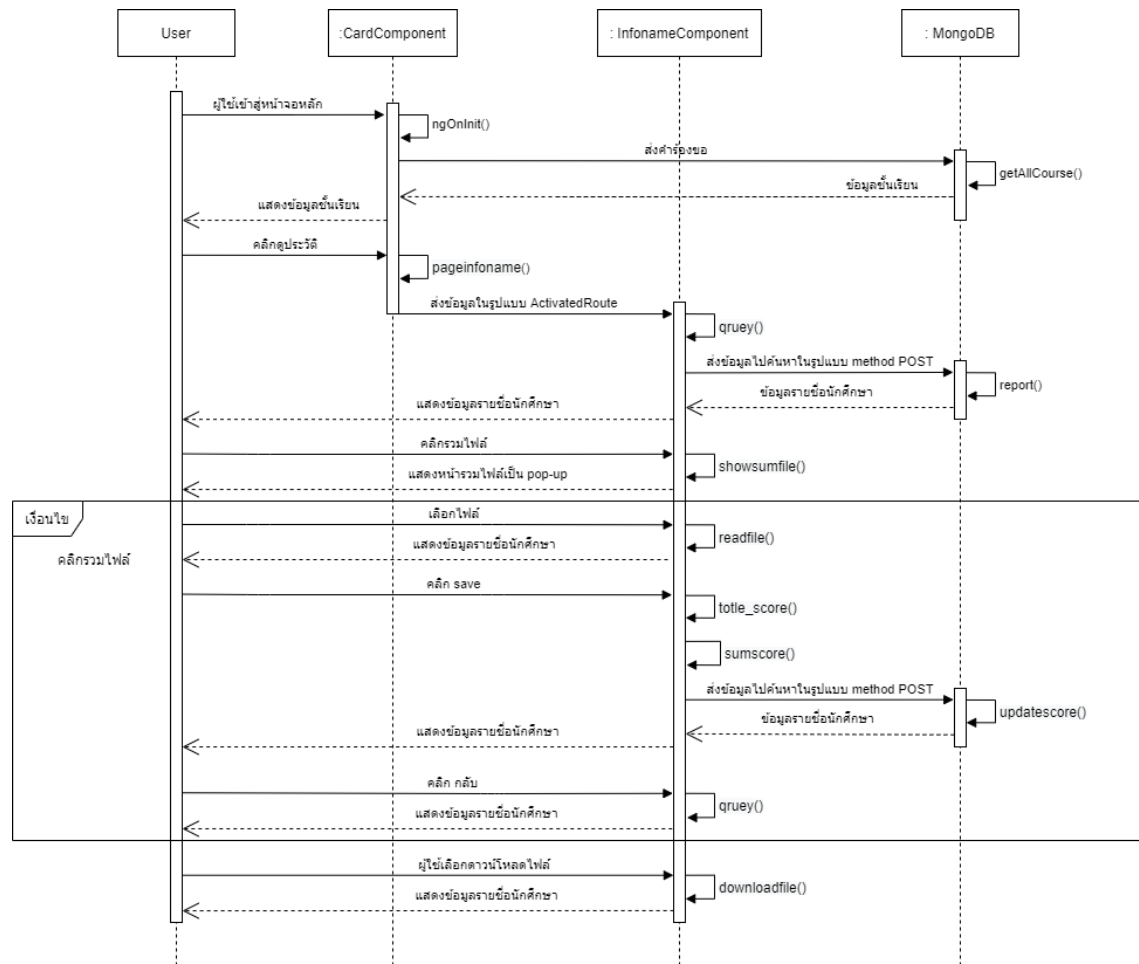
รูปที่ 3.18: Sequence Diagram สร้างคิวอาร์โค้ด

จากภาพที่ 3.18 สามารถอธิบายแผนภาพ Sequence Diagram สร้างคิวอาร์โค้ด ได้ดังนี้ เมื่อ ผู้ใช้เปิดโปรแกรมระบบจะเรียกใช้เมธอด `ngOnInit()` ที่คลาส `CardComponent` ระบบจะส่งคำร้องขอไปที่ฐานข้อมูล ซึ่งฐานข้อมูลจะเรียกใช้เมธอด `getAllCourse()` โดยจะทำหน้าที่ค้นหาและคืนค่าข้อมูลชั้นเรียนที่มีในฐานข้อมูล จากนั้นระบบจะรับค่าที่คืนมาจากฐานข้อมูล โดยจะแสดงเป็นการ์ด เมื่อผู้ใช้เลือกคิวอาร์โค้ดระบบจะเรียกใช้งานเมธอด `pageqr()` ที่คลาส `CardComponent` ระบบจะไปทำหน้าที่สร้างคิวอาร์โค้ดโดยส่งข้อมูล `id` ของชั้นเรียนไปในรูปแบบ `ActivatedRoute` เมื่อมาที่หน้าสร้างคิวอาร์โค้ดแล้วระบบจะเรียกใช้เมธอด `ngOnInit()` ที่คลาส `UploadfileComponent` ระบบจะส่งข้อมูล `id` ไปที่ฐานข้อมูลไปค้นหาในรูปแบบ `POST` ในเมธอด `report()` โดยเมธอดจะค้นหาชั้นเรียนตาม `id` ที่ได้มาและคืนค่าข้อมูลชั้นเรียนที่หาพบและระบบจะแสดงหน้าสร้างคิวอาร์โค้ด จากนั้นเมื่อผู้ใช้งานตั้งค่าคิวอาร์โค้ดแล้วกด `Start QRcode` ระบบจะเรียกใช้เมธอด `Start()` ซึ่งเมธอดนี้จะทำหน้าที่ในการสร้าง `URL` ของคิวอาร์โค้ดและเรียกใช้เมธอด `heroku()` โดยเมธอดนี้จะทำหน้าที่อัปเดต `URL` ที่เก็บไว้ในฐานข้อมูล จากนั้นระบบจะเรียกใช้งานเมธอด `ngOnInit()` ซึ่งจะหน้าที่ในการนับเวลาของคิวอาร์โค้ด เมื่อเวลาเริ่มนับถอยหลัง ระบบจะส่งข้อมูล `URL` ในรูปแบบ `POST` ไปเก็บไว้ในฐานข้อมูลและคืนค่า `URL` ที่บันทึกในฐานข้อมูล จากนั้นระบบจะแสดงคิวอาร์โค้ด เมื่อเวลานับถอยหลังหมดลงระบบจะส่ง `URL` ไปอัปเดตในฐานข้อมูลและคืนค่า `URL` ที่บันทึกในฐานข้อมูล จากนั้นจะแสดงรายชื่อนักศึกษาที่เข้าเรียน



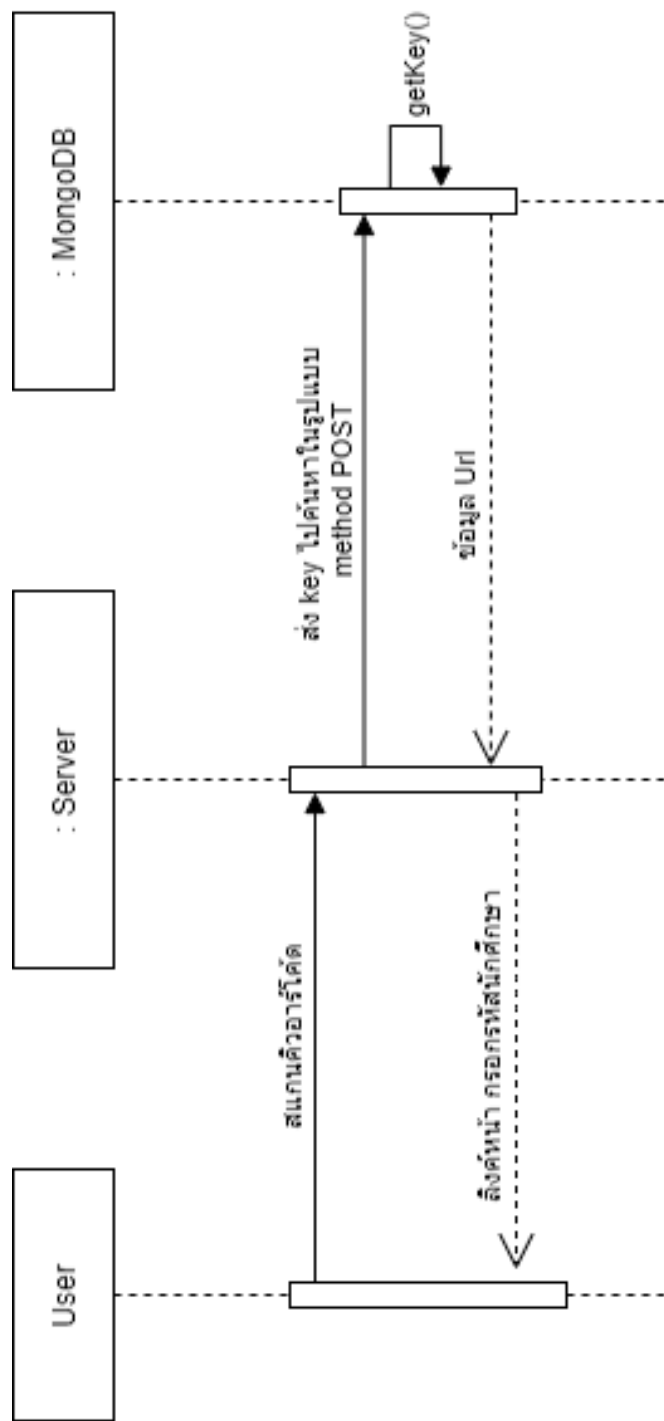
รูปที่ 3.19: Sequence Diagram จัดการคะแนน

จากภาพที่ 3.19 สามารถอธิบายแผนภาพ Sequence Diagram จัดการคะแนน ได้ดังนี้ เมื่อ ผู้ใช้เปิดโปรแกรมระบบจะเรียกใช้เมธอด `ngOnInit()` ที่คลาส `CardComponent` ระบบจะส่งคำร้องขอไปที่ฐานข้อมูล ซึ่งฐานข้อมูลจะเรียกใช้เมธอด `getAllCourse()` โดยจะทำหน้าที่ค้นหาและคืนค่าข้อมูลชั้นเรียนที่มีในฐานข้อมูล จากนั้นระบบจะรับค่าที่คืนมาจากฐานข้อมูลโดยจะแสดงเป็นการ์ด เมื่อผู้ใช้คลิกจัดการคะแนนระบบจะเรียกใช้งานเมธอด `pageinfo()` ที่คลาส `CardComponent` ระบบจะไปทำหน้าที่สร้างคิวอาร์โค้ดโดยส่งข้อมูล `id` ของชั้นเรียนไปในรูปแบบ `ActivatedRoute` เมื่อมาที่หน้าจัดการคะแนนแล้วระบบจะเรียกใช้เมธอด `ngOnInit()` ที่คลาส `InfoComponent` ระบบจะส่งข้อมูล `id` ไปที่ฐานข้อมูลไปค้นหาในรูปแบบ `POST` ในเมธอด `report()` โดยเมธอดจะค้นหาชั้นเรียนตาม `id` ที่ได้มาและคืนค่าข้อมูลชั้นเรียนที่หาพบและระบบจะแสดงหน้าจัดการคะแนน โดยหน้าจัดการคะแนนจะโชว์รายชื่อและข้อมูลคะแนนของนักศึกษา เมื่อผู้ใช้งานคลิกปุ่มมาระบบจะเรียกใช้เมธอด `checkbyTcome()` ในคลาส `InfoComponent` ซึ่งเมธอดนี้จะทำการส่งข้อมูล `id` ชั้นเรียนและรหัสนักศึกษาไปที่ฐานข้อมูลในรูปแบบ `POST` โดยฐานข้อมูลจะเรียกใช้เมธอด `checkIn()` เพื่อให้คะแนนและจะคืนค่าที่อัปเดตคะแนนกลับมาที่คลาส `InfoComponent` จากนั้นก็จะแสดงรายชื่อพร้อมคะแนนที่อัปเดต ต่อมาเมื่อผู้ใช้งานคลิกปุ่มสาย ระบบจะเรียกใช้เมธอด `checkbyTlast()` ในคลาส `InfoComponent` ซึ่งเมธอดนี้จะทำการส่งข้อมูล `id` ชั้นเรียนและรหัสนักศึกษาไปที่ฐานข้อมูลในรูปแบบ `POST` โดยฐานข้อมูลจะเรียกใช้เมธอด `checknameT()` เพื่อให้คะแนนและจะคืนค่าที่อัปเดตคะแนนกลับมาที่คลาส `InfoComponent` จากนั้นก็จะแสดงรายชื่อพร้อมคะแนนที่อัปเดต สุดท้ายเมื่อผู้ใช้งานคลิกปุ่มมาระบบจะเรียกใช้เมธอด `checkbyOut()` ในคลาส `InfoComponent` ซึ่งเมธอดนี้จะทำการส่งข้อมูล `id` ชั้นเรียนและรหัสนักศึกษาไปที่ฐานข้อมูลในรูปแบบ `POST` โดยฐานข้อมูลจะเรียกใช้เมธอด `checknameO()` เพื่อให้คะแนนและจะคืนค่าที่อัปเดตคะแนนกลับมาที่คลาส `InfoComponent` จากนั้นก็จะแสดงรายชื่อพร้อมคะแนนที่อัปเดต



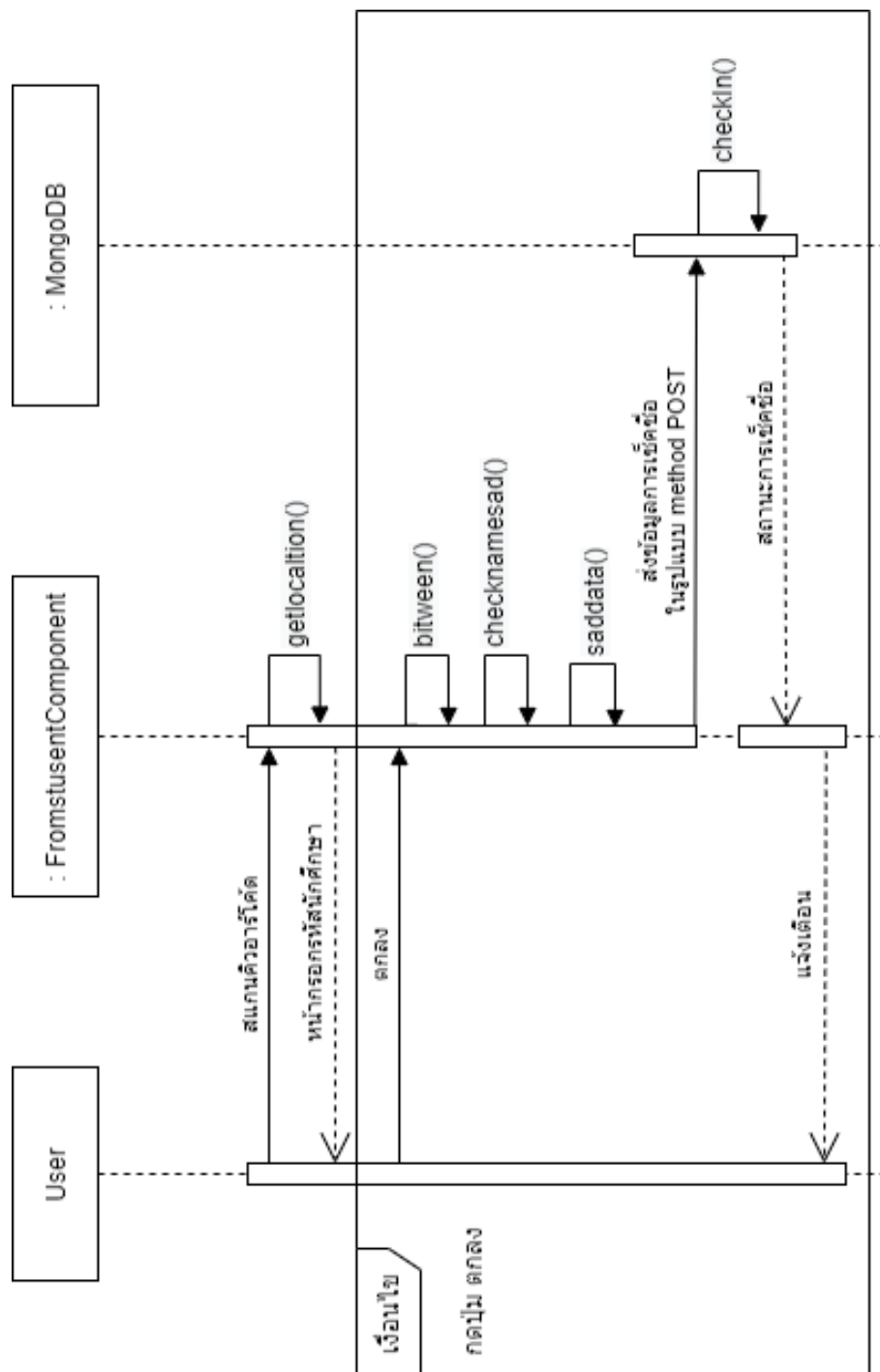
รูปที่ 3.20: Sequence Diagram ดูประวัติ

จากภาพที่ 3.20 สามารถอธิบายแผนภาพ Sequence Diagram ดังต่อไปนี้ เมื่อ ผู้ใช้ เปิดโปรแกรมระบบจะเรียกใช้เมธอด `ngOnInit()` ที่คลาส `CardComponent` ระบบจะส่งคำร้องขอไปที่ฐานข้อมูล ซึ่งฐานข้อมูลจะเรียกใช้เมธอด `getAllCourse()` โดยจะทำหน้าที่ค้นหาและคืนค่าข้อมูลชั้นเรียนที่มีในฐานข้อมูล จากนั้นระบบจะรับค่าที่คืนมาจากฐานข้อมูลโดยจะแสดงเป็นการ์ด เมื่อผู้ใช้คลิกดูประวัติระบบจะเรียกใช้งานเมธอด `pageinfoName()` ที่คลาส `CardComponent` ระบบจะไปทำหน้าที่สร้างคิวอาร์โค้ดโดยส่งข้อมูล `id` ของชั้นเรียนไปในรูปแบบ `ActivatedRoute` เมื่อมาที่หน้าจัดการคะแนนแล้วระบบจะเรียกใช้เมธอด `ngOnInit()` ที่คลาส `InfonameComponent` ระบบจะส่งข้อมูล `id` ไปที่ฐานข้อมูลไปค้นหาในรูปแบบ `POST` ในเมธอด `report()` โดยเมธอดจะค้นหาชั้นเรียนตาม `id` ที่ได้มาและคืนค่าข้อมูลชั้นเรียนที่หาพบและระบบจะแสดงหน้าประวัติ โดยหน้าประวัติจะโชว์ข้อมูลรายชื่อและคะแนนแต่ละสัปดาห์ของนักศึกษา เมื่อผู้ใช้คลิกปุ่มรวมไฟล์ระบบจะเรียกใช้งานเมธอด `showsumfile()` ในคลาส `InfonameComponent` ซึ่งเมธอดนี้ทำหน้าที่แสดงหน้ารวมไฟล์ เมื่อผู้ใช้เลือกไฟล์ระบบจะเรียกใช้งานเมธอด `readfile()` ในคลาส `InfonameComponent` ซึ่งเมธอดนี้จะทำหน้าที่อ่านไฟล์และตรวจสอบ ถ้าผู้ใช้กด `save` ระบบจะเรียกใช้งานเมธอด `totlescore()` ในคลาส `InfonameComponent` ซึ่งเมธอดนี้จะทำการรวมคะแนนจากไฟล์ที่นำเข้าและคะแนนเดิม จากนั้นจะเรียกใช้เมธอด `sumscore()` โดยเมธอดนี้จะทำการส่งข้อมูลไปที่ฐานข้อมูลที่อัปเดตทไปเก็บในฐานข้อมูล ซึ่งฐานข้อมูลจะเรียกใช้เมธอด `updatescore()` เพื่อบันทึกข้อมูล จากนั้นจะส่งค่าข้อมูลที่อัปเดตกลับมาที่คลาส `InfonameComponent` แล้วจึงแสดงข้อมูลที่อัปเดต 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 ที่เก็บไว้ในคิวอาร์โค้ดไปเพื่อนำไปค้นหาชั้นเรียน จากนั้น Server จะเรียกใช้เมธอด getKey() ซึ่งเมธอดนี้จะทำหน้าที่คืนค่า url สำหรับไปที่หน้ากรอกรหัสนักศึกษา





รูปที่ 3.22: Sequence Diagram กรอกรหัสนักศึกษา

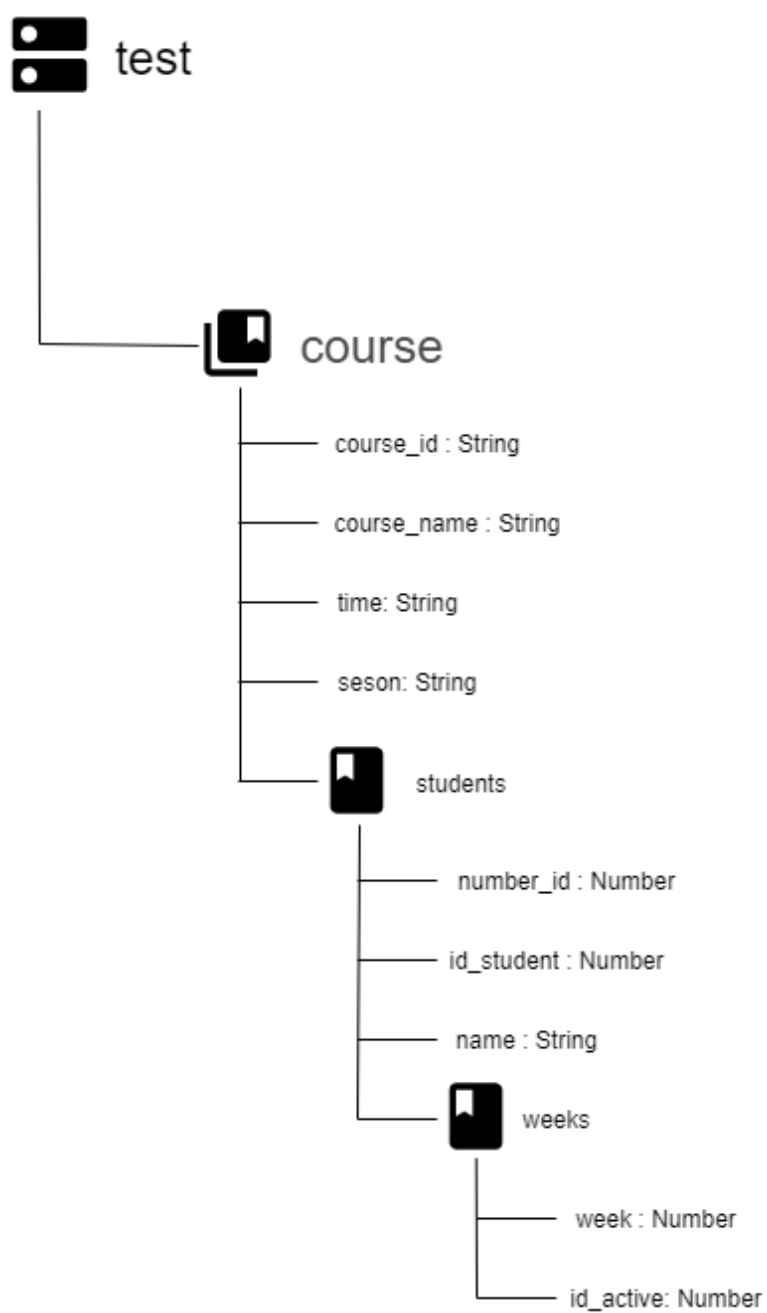
จากภาพที่ 3.22 สามารถอธิบายแผนภาพ Sequence Diagram กรอกรหัสนักศึกษา ได้ดังนี้ เมื่อนักศึกษาสแกนคิวอาร์โค้ดผ่านจะเข้ามาที่หน้ากรอกรหัสนักศึกษาโดยระบบจะให้ กรอกรหัสนักศึกษาเพื่อเป็นการยืนยันตัวตนจากนั้นระบบจะเรียกใช้งานเมธอด `getlocaltion()` ในคลาส `FromstudentComponent` เพื่อระบุตำแหน่งของนักศึกษา เมื่อกดตกลงระบบจะเรียกใช้เมธอด `bitween()` เพื่อคำนวณหาระยะห่างระหว่างอาจารย์กับนักศึกษา จากนั้นจะเรียกใช้เมธอด `checknamesad()` เพื่อตรวจสอบว่านักศึกษาเช็คชื่อไปแล้วหรือยัง แล้วจึงเรียกใช้เมธอด `saddata()` เพื่อส่งข้อมูลไปพื้นฐานข้อมูลในรูปแบบ POST โดยฐานข้อมูลจะเรียกใช้เมธอด `checkIn()` เพื่อจัดการคะแนนของนักศึกษา จากนั้นจะคืนค่าสถานะกลับมาที่คลาส `FromstudentComponent` ซึ่งคลาสจะทำการแจ้งเตือนการเช็คชื่อของนักศึกษา

3.7 โครงสร้างฐานข้อมูลไฟร์เบส (MongoDB Database Structure)

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

ตารางที่ 3.21: สัญลักษณ์ของโครงสร้างฐานข้อมูลแบบ MongoDB

| สัญลักษณ์ | คำอธิบาย |
|---|---|
|  | Database เป็นการเรียกชื่อแทนโหนด (Node) บนสุดที่ใช้ในการเก็บข้อมูล |
|  | Collection เป็นการเรียกชื่อแทนของการเก็บหลาย ๆ เอกสารไว้ด้วยกัน |
|  | Document เป็นการเรียกชื่อแทนหน่วยการเก็บของข้อมูลใน 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 |



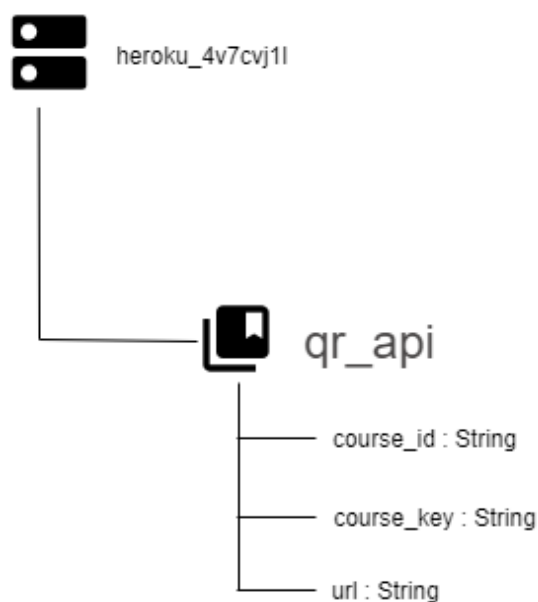
รูปที่ 3.23: โครงสร้างฐานข้อมูลแบบ MongoDB

จากรูที่ 3.23 สามารถอธิบายโครงสร้างของข้อมูลได้ดังนี้

test คือชื่อฐานข้อมูลของระบบเช็คชื่อด้วยคิวอาร์โค้ด ซึ่งจะเก็บข้อมูลต่าง ๆ เช่น ข้อมูลรายวิชา รายชื่อนักศึกษา และคะแนนในการเข้าเรียน โดยระบบจะเก็บข้อมูลเหล่านี้ไว้ในฐานข้อมูลบนคอมพิวเตอร์ของอาจารย์ ซึ่งข้อมูลจะเก็บไว้ใน Collection ชื่อ course โดยใน course จะเก็บข้อมูลชั้นเรียนทั้งหมดที่ถูกเพิ่มเข้ามา ดังตารางที่ 3.22

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 | โหนดสำหรับเก็บข้อมูลคะแนนแต่ละอาทิตย์ |
| week | สำหรับเก็บคะแนนศึกษา |
| id_active | สำหรับเก็บสถานะ |



รูปที่ 3.24: โครงสร้างฐานข้อมูลแบบ MongoDB

จากรูที่ 3.24 สามารถอธิบายโครงสร้างของข้อมูลได้ดังนี้

heroku_4v7v1l คือชื่อฐานข้อมูลของคิวอาร์โค้ด ฐานข้อมูลส่วนนี้จะเก็บข้อมูลต่าง ๆ ของคิวอาร์โค้ดไว้ใน Collection ชื่อ qr_api ซึ่งข้อมูลใน Collection นี้จะเก็บในรูปแบบ Object ดังตารางที่ 3.23

ตารางที่ 3.23: อธิบายส่วนที่ใช้เก็บข้อมูลใน Heroku Server

| Key | คำอธิบาย |
|------------|---|
| qr_api | โหนดสำหรับเก็บข้อมูลของคิวอาร์โค้ดทั้งหมด |
| course_id | สำหรับเก็บไอดีของชั้นเรียน |
| course_key | สำหรับเก็บคีย์ของคิวอาร์โค้ด |
| url | สำหรับเก็บ URL ของคิวอาร์โค้ด |

บทที่ 4

การพัฒนาระบบ

หลังจากที่ได้มีการเตรียมความพร้อมสำหรับการพัฒนาในด้านต่างๆ ไม่ว่าจะเป็นที่มาและความสำคัญของปัญหา เทคโนโลยีที่มีความเหมาะสมกับระบบ และการออกแบบระบบการทำงาน รวมไปถึงโครงสร้างของข้อมูล ในบทนี้จะเป็นการพูดถึงการสร้างระบบที่ได้มีการออกแบบไว้ในบทที่แล้วจะถูกนำเสนอในบทนี้ โดยการพัฒนาระบบแบ่งได้เป็นส่วนต่าง ๆ ดังนี้

4.1 การพัฒนาเว็บเซิร์ฟเวอร์

4.2 การพัฒนาเว็บแอปพลิเคชัน

4.1 การพัฒนาเว็บเซิร์ฟเวอร์

การพัฒนาระบบเซิร์ฟเวอร์ด้วยคิวอาร์โค้ด ฝั่งเว็บเซิร์ฟเวอร์วัตถุประสงค์หลักของคือให้บริการฐานข้อมูลและการเชื่อมต่อข้อมูลต่าง ๆ ของระบบเซิร์ฟเวอร์ด้วยคิวอาร์โค้ด โดยตัวเว็บเซิร์ฟเวอร์นี้ถูกพัฒนาขึ้นด้วย echo golang มีรายละเอียดการทำงานดังนี้

4.1.1 โครงสร้างและโค้ดของการให้บริการ API ของระบบเซิร์ฟเวอร์ด้วยคิวอาร์โค้ด

การพัฒนาเว็บเซิร์ฟเวอร์ระบบเซิร์ฟเวอร์ด้วยคิวอาร์โค้ด มีวัตถุประสงค์หลักเพื่อเก็บข้อมูลและให้บริการต่างๆ เกี่ยวกับการเชื่อมต่อกับข้อมูลที่เก็บไว้

```
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 สามารถอธิบายการทำงานโค้ดส่วนการเชื่อมต่อกับ mongodb โดยไฟล์ server.go สามารถอธิบายได้ดังนี้

- บรรทัดที่ 1-3 เป็นการนำเข้าไลบรารีของ mongodb และ echo framework
- บรรทัดที่ 7 เป็นการเชื่อมต่อกับฐานข้อมูล mongodb โดยที่อยู่ของฐานข้อมูลคือ localhost
- บรรทัดที่ 8-9 เป็นการตรวจสอบการเชื่อมต่อกับฐานข้อมูลว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 11 เป็นการประกาศตัวแปร 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 สามารถอธิบายการทำงานโค้ดส่วนที่สร้างตัวแปรเพื่อรับค่าจากฐานข้อมูล โดยไฟล์ handler.go ได้ดังนี้

- บรรทัดที่ 1-3 เป็นการนำเข้าไลบรารีของ mongodb
- บรรทัดที่ 5-8 เป็นการสร้างตัวแปรเพื่อรับค่าจากฐานข้อมูล 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     })

```

จากภาพที่ 4.3 สามารถอธิบายการทำงานโค้ดส่วนที่สร้างโมเดลของฐานข้อมูล โดยไฟล์ model-Course.go ได้ดังนี้

- บรรทัดที่ 1-3 เป็นการนำเข้าไลบรารีของ mongodb
- บรรทัดที่ 5-11 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 สามารถอธิบายการทำงานโค้ดส่วนที่นำเข้าไลบรารีต่างๆ ในไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 2 เป็นการนำเข้าไลบรารี bytes สำหรับเป็น type
- บรรทัดที่ 3 เป็นการนำเข้าไลบรารี io/ioutil สำหรับอ่านตัวแปรประเภท bytes
- บรรทัดที่ 4 เป็นการนำเข้าไลบรารี net สำหรับเรียกใช้งานฟังก์ชันต่างๆ ของ net
- บรรทัดที่ 5 เป็นการนำเข้าไลบรารี net/http สำหรับเรียกใช้งานฟังก์ชันต่างๆ ของ http
- บรรทัดที่ 6 เป็นการนำเข้าไลบรารี os สำหรับเรียกใช้งานฟังก์ชันต่างๆ ของ os
- บรรทัดที่ 8 เป็นการนำเข้าไลบรารี github.com/labstack/echo สำหรับเรียกใช้งานฟังก์ชันต่างๆ ของ echo
- บรรทัดที่ 9 เป็นการนำเข้าไลบรารี gopkg.in/mgo.v2 สำหรับเรียกใช้งานฟังก์ชันต่างๆ ของ mongodb
- บรรทัดที่ 10 เป็นการนำเข้าไลบรารี gopkg.in/mgo.v2/bson สำหรับเรียกใช้งานฟังก์ชันต่างๆ ของ 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 สามารถอธิบายการทำงานโค้ดส่วนฟังก์ชันที่ดึงข้อมูลของชั้นเรียนทั้งหมดในไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ getAllCourse
- บรรทัดที่ 2 เป็นการประกาศตัวแปรชื่อ users เพื่อรับค่าจากโมเดล Course
- บรรทัดที่ 3 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปรชื่อ 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 สามารถอธิบายการทำงานโค้ดส่วนฟังก์ชันสร้างชั้นเรียน โดยไฟล์ student.go ได้ดังนี้

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

เชื่อมต่อว่ามีข้อผิดพลาดหรือไม่

- บรรทัดที่ 15-17 เป็นการตรวจสอบการเชื่อมต่อกับฐานข้อมูล ถ้าการเชื่อมต่อผิดพลาดจะแสดงข้อผิดพลาด
- บรรทัดที่ 21 คือค่าที่ได้จากการเพิ่มขึ้นเรียน

```

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 สามารถอธิบายการทำงานโค้ดส่วนฟังก์ชันกลับชั้นเรียน โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ deleteCourse
- บรรทัดที่ 2 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปรชื่อ db
- บรรทัดที่ 3 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 5 เป็นกรประกาศตัวแปรเพื่อรับข้อมูล 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 สามารถอธิบายการทำงานโค้ดส่วนฟังก์ชันที่ดึงข้อมูลของชั้นเรียนในไฟล์ โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ report
- บรรทัดที่ 2 เป็นการสร้างตัวแปรประเภทชื่อ user เพื่อจำลองโมเดล Course
- บรรทัดที่ 3 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปรชื่อ db
- บรรทัดที่ 4 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 5 เป็นการประกาศตัวแปรเพื่อรับข้อมูล 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 สามารถอธิบายการทำงานโค้ดส่วนฟังก์ชันเพิ่มนักศึกษาในชั้นเรียน โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ addupstudent
- บรรทัดที่ 2 เป็นการสร้างตัวแปรประเภทชื่อ u เพื่อจำลองโมเดล Sutdent
- บรรทัดที่ 3 เป็นการประกาศตัวแปรเพื่อรับข้อมูล id ของนักศึกษาที่จะค้นหา
- บรรทัดที่ 5-6 เป็นการตรวจสอบการจำลองโมเดลว่าจำลองได้หรือไม่ ถ้าไม่ได้จะแสดงข้อผิดพลาด
- บรรทัดที่ 8 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปรชื่อ 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 สามารถอธิบายการทำงานโค้ดส่วนฟังก์ชันแก้ไขข้อมูลนักศึกษาในชั้นเรียน โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ editstudent
- บรรทัดที่ 2 เป็นการสร้างตัวแปรประเภทชื่อ u เพื่อจำลองโมเดล Sutdent
- บรรทัดที่ 3 เป็นการประกาศตัวแปรเพื่อรับข้อมูล id ของนักศึกษาที่จะค้นหา
- บรรทัดที่ 5-6 เป็นการตรวจสอบการจำลองโมเดลว่าจำลองได้หรือไม่ ถ้าไม่ได้จะแสดงข้อผิดพลาด
- บรรทัดที่ 8 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปรชื่อ db
- บรรทัดที่ 9 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb

- บรรทัดที่ 11 เป็นการสร้างตัวแปร query เพื่อใช้ในฟังก์ชัน Update โดยตัวแปรนี้จะเก็บข้อมูลที่จะค้นหาเรียนที่จะทำการแก้ไข
- บรรทัดที่ 12-14 เป็นการสร้างตัวแปร update เพื่อใช้ในฟังก์ชัน Update โดยตัวแปรนี้จะเก็บข้อมูลที่จะอัปเดตข้อมูลของนักศึกษา
- บรรทัดที่ 17-19 เชื่อมต่อกับฐานข้อมูลและอัปเดตข้อมูลของนักศึกษาพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 17 คือค่าข้อมูลนักศึกษาที่อัปเดต

```

1 func (h *Handler) deletestudent(c echo.Context)
   error {
2   u := new(Sudent)
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 สามารถอธิบายการทำงานโค้ดส่วนฟังก์ชันลบนักศึกษาในชั้นเรียน โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ deletestudent

- บรรทัดที่ 2 เป็นการสร้างตัวแปรประเภทชื่อ u เพื่อจำลองโมเดล Student
- บรรทัดที่ 3 เป็นการประกาศตัวแปรเพื่อรับข้อมูล id ของนักศึกษาที่จะค้นหา
- บรรทัดที่ 5-6 เป็นการตรวจสอบการจำลองโมเดลว่าจำลองได้หรือไม่ ถ้าไม่ได้จะแสดงข้อผิดพลาด
- บรรทัดที่ 8 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปรชื่อ db
- บรรทัดที่ 9 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 12 เป็นการสร้างตัวแปร query เพื่อใช้ในฟังก์ชัน Update โดยตัวแปรนี้จะเก็บข้อมูลที่จะค้นหานักเรียนที่จะทำการลบ
- บรรทัดที่ 13 เป็นการสร้างตัวแปร update เพื่อใช้ในฟังก์ชัน Update โดยตัวแปรนี้จะเก็บข้อมูลที่จะลบข้อมูลของนักศึกษา
- บรรทัดที่ 15-16 เชื่อมต่อกับฐานข้อมูลและอัปเดตข้อมูลของนักศึกษาพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 19 คือค่าข้อมูลนักศึกษาที่อัปเดต

```

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 สามารถอธิบายการทำงานโค้ดส่วนฟังก์ชันที่รวมคะแนนของชั้นเรียน โดยไฟล์ student.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ updatescore
- บรรทัดที่ 2 เป็นการสร้างตัวแปรประเภทชื่อ user เพื่อจำลองโมเดล Course
- บรรทัดที่ 3 เป็นการประกาศตัวแปรเพื่อรับข้อมูล id ของนักศึกษาที่จะค้นหา
- บรรทัดที่ 5-7 เป็นการตรวจสอบการจำลองโมเดลว่าจำลองได้หรือไม่ ถ้าไม่ได้จะแสดงข้อผิดพลาด
- บรรทัดที่ 10 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปรชื่อ db
- บรรทัดที่ 11 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 13 เป็นการสร้างตัวแปร query เพื่อใช้ในฟังก์ชัน Update โดยตัวแปรนี้จะเก็บ

ข้อมูลที่จะค้นหานักเรียนที่จะทำการค้นหาเพื่อจะอัปเดต

- บรรทัดที่ 14 เป็นการสร้างตัวแปร update เพื่อใช้ในฟังก์ชัน Update โดยตัวแปรนี้จะเก็บข้อมูลที่จะอัปเดตข้อมูลของนักศึกษา
- บรรทัดที่ 16-18 เชื่อมต่อกับฐานข้อมูลและอัปเดตข้อมูลของนักศึกษาพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 20 คือค่าข้อมูลนักศึกษาที่อัปเดต

```

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 สามารถอธิบายการทำงานโค้ดส่วนที่นำเข้าไลบรารีต่าง ๆ ในไฟล์ โดยไฟล์ checkname.go ได้ดังนี้

- บรรทัดที่ 2 เป็นการนำเข้าไลบรารี net/http เพื่อใช้งานฟังก์ชัน http
- บรรทัดที่ 4 เป็นการนำเข้าไลบรารี github.com/labstack/echo สำหรับเรียกใช้งานฟังก์ชันต่างๆของ echo
- บรรทัดที่ 5 เป็นการนำเข้าไลบรารี gopkg.in/mgo.v2/bson สำหรับเรียกใช้งานฟังก์ชันต่างๆของ 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 สามารถอธิบายการทำงานโค้ดส่วนฟังก์ชันเช็คชื่อ โดยไฟล์ checkname.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ checkIn
- บรรทัดที่ 2 เป็นการสร้างตัวแปรประเภทชื่อ u เพื่อจำลองโมเดล Course
- บรรทัดที่ 3 เป็นการประกาศตัวแปรชื่อ id เพื่อรับข้อมูลของรายวิชาที่จะค้นหา
- บรรทัดที่ 4 เป็นการประกาศตัวแปรชื่อ stdid เพื่อรับข้อมูลรหัสนักศึกษาที่จะค้นหา
- บรรทัดที่ 5 เป็นการประกาศตัวแปรชื่อ week เพื่อรับข้อมูลสัปดาห์ของรายวิชาที่จะค้นหา
- บรรทัดที่ 6 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปรชื่อ db
- บรรทัดที่ 7 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 8 เป็นการสร้างตัวแปร query เพื่อใช้ในฟังก์ชัน Update โดยตัวแปรนี้จะเก็บข้อมูลที่จะค้นหานักเรียนที่จะทำการค้นหาเพื่อจะอัปเดต
- บรรทัดที่ 9 เป็นการสร้างตัวแปร update เพื่อใช้ในฟังก์ชัน Update โดยตัวแปรนี้จะเก็บข้อมูลที่จะอัปเดตข้อมูลของนักศึกษา
- บรรทัดที่ 10-44 เป็นการสร้างเงื่อนไขตรวจสอบการเช็คชื่อ จะนำข้อมูลสัปดาห์เป็นตัวตรวจสอบ ถ้าข้อมูลตรงกันจะให้คะแนน แต่ถ้าข้อมูลไม่ตรงกันจะคืนค่าสตริง week invalid
- บรรทัดที่ 45-47 เชื่อมต่อกับฐานข้อมูลและอัปเดตข้อมูลของนักศึกษาพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 49 คือค่าข้อมูลนักศึกษาที่อัปเดต

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 4.15 สามารถอธิบายการทำงานโค้ดส่วนที่นำเข้าไลบรารีต่างๆ ในไฟล์ โดยไฟล์ server.go ได้ดังนี้

- บรรทัดที่ 2 เป็นการนำเข้าไลบรารี net/http เพื่อใช้งานฟังก์ชัน http
- บรรทัดที่ 3 เป็นการนำเข้าไลบรารี os เพื่อใช้งานฟังก์ชันของ os
- บรรทัดที่ 4 เป็นการนำเข้าไลบรารี github.com/labstack/echo/v4 สำหรับเรียกใช้งานฟังก์ชันต่างๆของ echo v4
- บรรทัดที่ 5 เป็นการนำเข้าไลบรารี github.com/labstack/echo/v4/middleware สำหรับเรียกใช้งานฟังก์ชันต่างๆของ middleware ใน echo v4
- บรรทัดที่ 6 เป็นการนำเข้าไลบรารี gopkg.in/mgo.v2 สำหรับเรียกใช้งานฟังก์ชันต่างๆของ 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.
11            MethodPut, http.MethodPost, http.MethodDelete},
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 4.16 สามารถอธิบายการทำงานโค้ดส่วนที่เชื่อมต่อกับฐานข้อมูลจากฐานข้อมูล

โดยไฟล์ server.go ได้ดังนี้

- บรรทัดที่ 1 สร้างฟังก์ชันชื่อว่า main
- บรรทัดที่ 2 สร้างตัวแปร e เพื่อเรียกใช้ฟังก์ชัน New() โดยฟังก์ชันนี้จะสามารถใช้งานฟังก์ชันต่างๆ ใน echo v4 ได้
- บรรทัดที่ 3 สร้างตัวแปร port เพื่อเก็บค่าของการสุ่ม PORT
- บรรทัดที่ 4 สร้างตัวแปร uri เพื่อเก็บประเภทของฐานข้อมูลที่จะเชื่อมต่อ
- บรรทัดที่ 6-10 เป็นการอนุญาตให้สามารถเข้าถึงฐานข้อมูลได้ โดยสิ่งที่ทำได้คือ get put post และ delete
- บรรทัดที่ 12 เป็นการสร้างตัวแปร db เพื่อเชื่อมต่อกับฐานข้อมูล
- บรรทัดที่ 13-15 เป็นการตรวจสอบว่าเชื่อมต่อกับฐานข้อมูลได้หรือไม่ ถ้าไม่ก็จะคืนค่า error
- บรรทัดที่ 18 เป็นการประกาศตัวแปร 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 4.17 สามารถอธิบายการทำงานโค้ดส่วนที่นำเข้าไลบรารีต่างๆ ในไฟล์ โดยไฟล์ handler.go ได้ดังนี้

- บรรทัดที่ 2 เป็นการนำเข้าไลบรารีของ mongodb
- บรรทัดที่ 6-9 เป็นการสร้างตัวแปรเพื่อรับค่าจากฐานข้อมูล 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,
9             omitempty"`
10        CourseID string `json:"course_id" bson:"
11            course_id"`
12        CourseKEY string `json:"course_key" bson
13            : "course_key"`
14        Url string `json:"url" bson:"url"`
15    }
16 )

```

รูปที่ 4.18: ไฟล์ module.go

จากภาพที่ 4.18 สามารถอธิบายการทำงานโค้ดส่วนที่สร้างโมเดลของฐานข้อมูล 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 สามารถอธิบายการทำงานโค้ดส่วนที่นำเข้าไลบรารีต่าง ๆ ในไฟล์ โดยไฟล์ checkstatus.go ได้ดังนี้

- บรรทัดที่ 2 เป็นการนำเข้าไลบรารี net/http สำหรับเรียกใช้งานฟังก์ชันต่างๆของ http

- บรรทัดที่ 3 เป็นการนำเข้าไลบรารี `github.com/labstack/echo/v4` สำหรับเรียกใช้งานฟังก์ชันต่างๆของ `echo v4`
- บรรทัดที่ 4 เป็นการนำเข้าไลบรารี `gopkg.in/mgo.v2/bson` สำหรับเรียกใช้งานฟังก์ชันต่างๆของ `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 สามารถอธิบายการทำงานโค้ดส่วนที่สร้างข้อมูลของคิวอาร์โค้ด โดยไฟล์ `checkstatus.go` ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ `createqr`
- บรรทัดที่ 3 เป็นการประกาศตัวแปรชื่อ `users` เพื่อรับค่าจากโมเดล `DataQR` ซึ่งจะทำการเซตค่า `ID` ตั้งแต่ต้น
- บรรทัดที่ 4-6 เป็นการตรวจสอบการรับค่าโมเดลว่าถูกต้องหรือไม่ ถ้าผิดพลาดจะแสดงข้อผิดพลาด
- บรรทัดที่ 9 เป็นการ clone ฐานข้อมูลจาก `mongodb` โดยเก็บไว้ในตัวแปรชื่อ `db`
- บรรทัดที่ 10 เป็นการปิดการ clone ฐานข้อมูลจาก `mongodb`

- บรรทัดที่ 12-13 เชื่อมต่อกับฐานข้อมูลและเพิ่มข้อมูลของชั้นเรียนพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 15 คำนวณค่าที่ได้จากการเพิ่มข้อมูล

```

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 สามารถอธิบายการทำงานโค้ดส่วนที่อัปเดตข้อมูลของคิวอาร์โค้ด โดยไฟล์ checkstatus.go ได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ createqr
- บรรทัดที่ 2 เป็นการประกาศตัวแปรชื่อ users เพื่อรับค่าจากโมเดล DataQR

- บรรทัดที่ 3-4 เป็นการตรวจสอบการรับค่าโมเดลว่าถูกต้องหรือไม่ ถ้าผิดพลาดจะแสดงข้อผิดพลาด
- บรรทัดที่ 6 เป็นการประกาศตัวแปรชื่อ courseid เพื่อรับค่า id สำหรับค้นหาข้อมูลใน DataQR
- บรรทัดที่ 7 เป็นการประกาศตัวแปรชื่อ coursekey เพื่อรับค่า Key สำหรับค้นหาข้อมูลที่จะอัปเดต
- บรรทัดที่ 8 เป็นการประกาศตัวแปรชื่อ urls เพื่อรับค่า Url สำหรับอัปเดต url
- บรรทัดที่ 10 เป็นการ clone ฐานข้อมูลจาก mongodb โดยเก็บไว้ในตัวแปรชื่อ db
- บรรทัดที่ 11 เป็นการปิดการ clone ฐานข้อมูลจาก mongodb
- บรรทัดที่ 13 เป็นการสร้างตัวแปร query เพื่อใช้ในฟังก์ชัน Update โดยตัวแปรนี้จะเก็บข้อมูลที่จะคั่นนักเรียนที่จะทำการค้นหาเพื่อจะอัปเดต
- บรรทัดที่ 14 เป็นการสร้างตัวแปร update เพื่อใช้ในฟังก์ชัน Update โดยตัวแปรนี้จะเก็บข้อมูลที่จะอัปเดตข้อมูลของนักศึกษา
- บรรทัดที่ 16-17 เชื่อมต่อกับฐานข้อมูลและอัปเดตข้อมูลของชั้นเรียนพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 19-20 เชื่อมต่อกับฐานข้อมูลและค้นหาข้อมูลของชั้นเรียนพร้อมทั้งตรวจสอบการเชื่อมต่อว่ามีข้อผิดพลาดหรือไม่
- บรรทัดที่ 23 คั่นค่าที่ Key ได้จากการอัปเดตข้อมูล


```

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 สามารถอธิบายการทำงานโค้ดส่วนที่ดึงข้อมูลคิวอาร์โค้ดไปใช้ โดยไฟล์ checkstatus.go ได้ดังนี้

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

4.2 การพัฒนาเว็บแอปพลิเคชัน

การพัฒนาระบบเช็คชื่อด้วยคิวอาร์โค้ดสำหรับเว็บแอปพลิเคชันนั้นวัตถุประสงค์หลักเพื่อสร้างความสะดวกต่อการใช้งานระบบเช็คชื่อด้วยคิวอาร์โค้ด โดยตัวเว็บแอปพลิเคชันนี้ถูกพัฒนาขึ้นด้วย Angular มีรายละเอียดการทำงานดังนี้

4.2.1 โครงสร้างและโค้ด Service เพื่อใช้บริการ API

ในการเชื่อมต่อเว็บแอปพลิเคชันกับ API จำเป็นต้องสร้างโฟลเดอร์ Service เพื่อให้บริการต่างๆ 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 เป็นการประกาศตัวแปรชื่อ apiURL เพื่อเก็บโฮสต์ของ 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 เป็นการสร้างฟังก์ชันชื่อ getStudent
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาร์ท 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 เป็นการสร้างฟังก์ชันชื่อ getIP
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาร์ท 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 (`${this.apiUrl}/
   chakname/${id}`, Data);
6  }

```

รูปที่ 4.26: ไฟล์ student.service.ts

จากภาพที่ 4.26 โครงสร้างของไฟล์ student.service.ts สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ chakname โดยการจะเรียกใช้ฟังก์ชันนี้ได้ต้องส่งค่า id stdids และ weeks มา

- บรรทัดที่ 2 เป็นการสร้างตัวแปรชื่อ Data เพื่อเซตข้อมูลที่จะส่งไปหา API เป็นรูปแบบ FormData
- บรรทัดที่ 3 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า stdid
- บรรทัดที่ 4 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า week
- บรรทัดที่ 5 เป็นการนำเรียกใช้ API โดยใช้พาหุ 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 เป็นการสร้างฟังก์ชันชื่อ chacknameby-t โดยการจะเรียกใช้ฟังก์ชันนี้ได้ต้องส่งค่า id stdids และ weeks มา
- บรรทัดที่ 2 เป็นการสร้างตัวแปรชื่อ Data เพื่อเซตข้อมูลที่จะส่งไปหา API เป็นรูปแบบ FormData
- บรรทัดที่ 3 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า stdid
- บรรทัดที่ 4 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า week
- บรรทัดที่ 5 เป็นการนำเรียกใช้ API โดยใช้พาหุ 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 เป็นการสร้างฟังก์ชันชื่อ checknameOut_t โดยการจะเรียกใช้ฟังก์ชันนี้ได้ ต้องส่งค่า id stids และ weeks มา
- บรรทัดที่ 2 เป็นการสร้างตัวแปรชื่อ Data เพื่อเซตข้อมูลที่จะส่งไปหา API เป็นรูปแบบ FormData
- บรรทัดที่ 3 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า stdid
- บรรทัดที่ 4 เป็นการเพิ่มข้อมูลใน FormData โดยให้ชื่อว่า week
- บรรทัดที่ 5 เป็นการนำเรียกใช้ API โดยใช้พารามิเตอร์ 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 เป็นการสร้างฟังก์ชันชื่อ uploadStudent โดยการจะเรียกใช้ฟังก์ชันนี้ได้ต้อง ส่งค่า courseid coursenames times sesons และ student มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พารามิเตอร์ upload
- บรรทัดที่ 3-7 เป็นการเซตข้อมูลที่จะส่งไปให้กับ 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 เป็นการสร้างฟังก์ชันชื่อ updateStudent โดยการจะเรียกใช้ฟังก์ชันนี้ได้ต้องส่งค่า course-id และ dataupdate มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาร์ท updatescore
- บรรทัดที่ 3 เป็นการเซตข้อมูลที่จะส่งไปให้กับ 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 เป็นการสร้างฟังก์ชันชื่อ addstudent โดยการจะเรียกใช้ฟังก์ชันนี้ได้ต้องส่งค่า idcoure idstudent names และ number มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาร์ท addupstudent
- บรรทัดที่ 3-5 เป็นการเซตข้อมูลที่จะส่งไปให้กับ 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 เป็นการสร้างฟังก์ชันชื่อ editstudent โดยการจะเรียกใช้ฟังก์ชันนี้ได้ต้องส่งค่า idcoure idstudent names และ number มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาร์ท editstudent
- บรรทัดที่ 3-5 เป็นการเชื่อมต่อข้อมูลที่จะส่งไปให้กับ 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 เป็นการสร้างฟังก์ชันชื่อ deletestudent โดยการจะเรียกใช้ฟังก์ชันนี้ได้ต้องส่งค่า idcoure และ idstudent มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาร์ท deletestudent
- บรรทัดที่ 3-5 เป็นการเชื่อมต่อข้อมูลที่จะส่งไปให้กับ 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 เป็นการสร้างฟังก์ชันชื่อ deleteCourse โดยการจะเรียกใช้ฟังก์ชันนี้ได้ต้องส่งค่า id มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาร์ท 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 เป็นการสร้างฟังก์ชันชื่อ reportCourse โดยการจะเรียกใช้ฟังก์ชันนี้ได้ต้องส่งค่า id มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาร์ท 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 เป็นการสร้างตัวแปรชื่อ apiURL เพื่อเก็บโฮตของ 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 เป็นการสร้างฟังก์ชันชื่อ getKey โดยการจะเรียกใช้ฟังก์ชันนี้ได้ต้องส่งค่า key มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาร์ท 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 เป็นการสร้างฟังก์ชันชื่อ craeteQR โดยการจะเรียกใช้ฟังก์ชันนี้ได้ต้องส่งค่า idcours keycours และ herokuUrl มา
- บรรทัดที่ 2 เป็นการนำเรียกใช้ API โดยใช้พาร์ท createqr
- บรรทัดที่ 3-5 เป็นการเซตข้อมูลที่จะส่งไปให้กับ 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 เป็นการสร้างฟังก์ชันชื่อ updateQR โดยการจะเรียกใช้ฟังก์ชันนี้จะต้องส่งค่า idcours keycours และ herokuUrl มา
- บรรทัดที่ 2 เป็นการสร้างตัวแปรชื่อ Data เพื่อเซตข้อมูลที่จะส่งไปหา 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 โดยใช้พาร์ท updateqr โดยส่ง Data ไปด้วย

4.2.2 โครงสร้างของโค้ดใน Angular หน้าสร้างชั้นเรียน

```

1  <input class="btn btn-info " type="file" (change)="
    onFileChange($event)" multiple="false" />
2
3  <div class="row">
4
5  <div class="col-md-6"></div>
6  <div class="input-group col-md-4">
7  </div>
8  <div class="col-md-2 container">
9  <select class="form-control" #seson name="seson" id
    ="">
10 <option selected>เลือกเพิ่ม</option>
11 <option value="1">1</option>
12 <option value="2">2</option>
13 <option value="3">3</option>
14 </select>
15 </div>
16 </div>
17 <pรหัสวิชา> :{{course_id}}</p>
18 <pชื่อวิชา> : {{course_name}}</p>
19 <pวันเวลาเรียน> : {{time}}</p>
20 <div class="text-right">
21 <button class="btn btn-success " (click)="upload(seson.value)">Upload
    !</button>
22 </div>
23 <hr>
24 <table class="table">
25 <thead>
26 <tr>
27 <th scope="col" >เลขที่</th>
28 <th scope="col" >รหัสนักศึกษา</th>
29 <th scope="col" >ชื่อนานสกุล</th>
30 </tr>
31 </thead>
32 <tbody *ngFor="let s of student | paginate: { itemsPerPage: 20,
    currentPage: p }">
33 <tr>
34 <th>{{s.number}}</th>
35 <td>{{s.id_student}}</td>
36 <td>{{s.name}}</td>
37 </tr>
38 </tbody>
39 </table>

```

รูปที่ 4.40: โค้ดในไฟล์ formfile.component.html

จากภาพที่ 4.40 โค้ดในไฟล์ `formfile.component.html` ของหน้าสร้างชั้นเรียน สามารถอธิบายการทำงานได้ดังนี้

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

```

1 onFileChange(evt: any) {
2   try {
3     /* wire up file reader */
4     const target: DataTransfer = <DataTransfer>(evt.
       target);
5     if (target.files.length !== 1) throw new Error('
       Cannot use multiple files');
6     const reader: FileReader = new FileReader();
7     reader.onload = (e: any) => {
8       /* read workbook */
9       const bstr: string = e.target.result;
10      const wb: XLSX.WorkBook = XLSX.read(bstr, { type: '
        binary' });
11
12      /* grab first sheet */
13      const wsname: string = wb.SheetNames[0];
14      const ws: XLSX.WorkSheet = wb.Sheets[wsname];
15      this.wss = [ws.B4.w, ws.B6.w, ws.M4.w]
16
17      /* save data */
18      this.data = <any>(XLSX.utils.sheet_to_json(ws, {
        header: 1 }));
19      this.course = ws.B4.w;
20      this.time = ws.B6.w;
21      for (let i = 9; i <= this.data.length - 3; i++) {
22        this.student[i - 9] = this.data[i];
23        for (let j = 1; j <= 3; j++) {
24          this.student[i - 9]['number'] = this.data[i][1];
25          this.student[i - 9]['id_student'] = this.data[i][2];
26          this.student[i - 9]['name'] = this.data[i][3];
27        }
28        this.student[i - 9].splice(0)
29      }
30      this.course_id = this.course.substr(10,7);
31      this.course_name=this.course.substr(19);
32      this.time=this.time.substr(13);
33    };
34    return reader.readAsBinaryString(target.files[0]);
35  } catch (error) {
36    alert('fire ไม่ถูกต้อง')
37  }
38 }

```

รูปที่ 4.41: โค้ดในไฟล์ formfile.component.ts

จากภาพที่ 4.42 โค้ดในไฟล์ `formfile.component.ts` ของหน้าสร้างชั้นเรียน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ `onFileChange` โดยจับรับไฟล์ข้อมูลมา
- บรรทัดที่ 4-34 เป็นการอ่านไฟล์ที่รับเข้ามา และเก็บข้อมูลที่ได้จากการอ่านไว้ใน `student` ซึ่งโค้ดการอ่านไฟล์นี้จะถูกครอบด้วย `Try catch`
- บรรทัดที่ 35-37 เป็นการเช็คว่าการอ่านข้อมูลถูกต้องหรือไม่ถ้าไม่จะแสดงข้อความ `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: โค้ดในไฟล์ formfile.component.ts

จากภาพที่ 4.42 โค้ดในไฟล์ formfile.component.ts ของหน้าสร้างชั้นเรียน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ upload โดยจับรับข้อมูล seson มา
- บรรทัดที่ 2-9 เป็นการสร้างตัวแปรชื่อ Data เพื่อเตรียมข้อมูลสำหรับอัปโหลด
- บรรทัดที่ 11-12 เป็นการเช็คว่ามี student มีข้อมูลหรือไม่

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

4.2.3 โครงสร้างของโค้ดใน Angular หน้าอัปเดตข้อมูลนักศึกษา

```

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       ">
14       <td class="text-center">{{s.number_id}}</td>
15       <td>{{s.id_student}}</td>
16       <td>{{s.name}}</td>
17       <td><button class="btn-lg btn-block btn-success" (
18         click)="showadd(content)แก้ไข"></button></td>
19       <td><button type="submit" class="btn-lg btn-block
20         btn-danger" (click)="delete(s.id_student)ลบ"></
        button></td>
  </tr>
  </tbody>
</table>

```

รูปที่ 4.43: โค้ดในไฟล์ formup.component.html

จากภาพที่ 4.43 โค้ดในไฟล์ formup.component.html ของหน้าอัปเดตข้อมูลนักศึกษาสามารถอธิบายการทำงานได้ดังนี้

- 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,
5     idstudent, name, this.number).subscribe(
6     () => {
7       alert("status OK")
8       this.studentService.reportCourse(this.idcoure).
9         subscribe(
10          (data) => {
11            this.course = data[0];
12          }
13        )
14      document.getElementById('close1').click()
15    }
16  }

```

รูปที่ 4.44: โค้ดในไฟล์ formup.component.ts

จากภาพที่ 4.44 โค้ดในไฟล์ formup.component.ts ของหน้าอ็อปเตทข้อมูลนักศึกษา สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ 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 เป็นการเรียนรู้ฟังก์ชัน addstudent ในไฟล์ student.service.ts ซึ่งเป็นฟังก์ชันสำหรับเพิ่มนักศึกษา

```

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

```

รูปที่ 4.45: โค้ดในไฟล์ formup.component.ts

จากภาพที่ 4.45 โค้ดในไฟล์ formup.component.ts ของหน้าอัปเดตข้อมูลนักศึกษา สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ editstudent โดยจับรับข้อมูล idstudent name และ number มา
- บรรทัดที่ 2-12 เป็นการเรียนรู้ฟังก์ชัน editstudent ในไฟล์ student.service.ts ซึ่งเป็นฟังก์ชันสำหรับแก้ไขข้อมูลนักศึกษา

```

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: โค้ดในไฟล์ formup.component.ts

จากภาพที่ 4.46 โค้ดในไฟล์ formup.component.ts ของหน้าอัปเดตข้อมูลนักศึกษา สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ delete โดยจับรับข้อมูล id มา
- บรรทัดที่ 2-10 เป็นการเรียกใช้ฟังก์ชัน deletestudent ในไฟล์ student.service.ts ซึ่งเป็นฟังก์ชันสำหรับลบข้อมูลนักศึกษา

4.2.4 โครงสร้างของโค้ดใน Angular หน้าสร้างคิวอาร์โค้ด

```

1 <button type="button" class="btn btn-info btn-lg
  text-center"
2 (click)="student(weeks.value,times.value,content);
  timerSrarted$.next(true);">Start QR
3 code</button>
4 <ng-template #content let-modal>
5 <div class="modal-header">
6 <h4 class="modal-title text-centerสัปดาห์
  ที่">{{weeks.value}}</h4>
7 <a type="button" class="close" aria-label="Close" id="close"
8 (click)="timerSrarted$.next(false); modal.dismiss('Cross click')">
9 <span aria-hidden="true">&times;</span>
10 </a>
11 </div>
12 <div class="modal-body">
13 <div class="text-center">
14 <!-- <p>week at {{weeks.value}}</p> -->
15 <h1>
16 <countdown #countdown [config]="{leftTime: time, demand: ti}">$!m!:$!s
  !</countdown>
17 </h1>
18 </div>
19 <div class="container col-md-8">
20 <ngx-qrcode
21 [qrc-value]="qrcode_url"
22 qrc-element-type="url"
23 qrc-class="aclass"
24 qrc-errorCorrectionLevel="L"
25 qrc-scanQrcode="timeqrcode"
26 >
27 </ngx-qrcode>
28 </div>
29 </div>
30 </ng-template>

```

รูปที่ 4.47: โค้ดในไฟล์ qrcode.component.html

จากภาพที่ 4.47 โค้ดในไฟล์ qrcode.component.html ของหน้าสร้างคิวอาร์โค้ด 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.idcours + ', ' + this.weeks;
6
7   this.heroku(herokuUrl, content)
8   this.query()
9
10 }
```

รูปที่ 4.48: โค้ดในไฟล์ qrcode.component.ts

จากภาพที่ 4.48 โค้ดในไฟล์ qrcode.component.ts ของหน้าสร้างคิวอาร์โค้ด สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ Start โดยจํารับข้อมูล week times และ content มา
- บรรทัดที่ 2 เป็นการเก็บข้อมูล week ไว้ใน weeks
- บรรทัดที่ 3 เป็นการเก็บข้อมูล times ไว้ใน time
- บรรทัดที่ 5 เป็นการประกาศตัวแปรชื่อ herokuUrl เพื่อเก็บข้อมูลพารามิเตอร์ที่จะไปยังหน้ากรอกรหัสนักศึกษาเพื่อเช็คชื่อ
- บรรทัดที่ 7 เป็นการเรียกใช้ฟังก์ชัน heroku โดยฟังก์ชันนี้จะทำหน้าที่ส่งค่า url ไปเก็บไว้ใน heroku server
- บรรทัดที่ 8 เป็นการเรียกใช้ฟังก์ชัน query โดยฟังก์ชันนี้จะทำหน้าที่สร้างคิวอาร์โค้ด

4.2.5 โครงสร้างของโค้ดใน Angular หน้าจัดการคะแนน

```

1 <select class="form-control" #status name="status"
   style="width: 20%;">
2 <option selectedทั้งหมด></option>
3 <option valueมา="1"></option>
4 <option valueสาย="0.5"></option>
5 <option valueขาด="0"></option>
6 </select>
7 <table class="table table-bordered">
8 <thead>
9 <tr class="text-center">
10 <th scope="colเลขที่"></th>
11 <th scope="colรหัสนักศึกษา"></th>
12 <th scope="colชื่อ"></th>
13 <th colspanสี่ปดาห์ที่="4"> {{twee}}</th>
14 <th scope="colรวม"></th>
15 </tr>
16
17 </thead>
18 <tbody>
19 <tr *ngFor="let s of course.students | studentFilter:[{week:twee},{status:
   statusShow}] | paginate: { itemsPerPage: 20, currentPage: p }">
20 <td class="text-center">{{s.number_id}}</td>
21 <td>{{s.id_student}}</td>
22 <td>{{s.name}}</td>
23 <td class="text-center" style="font-size: 1.2rem;">
24 <span >{{retest(s.weeks)}}</span>
25 </td>
26 <td class="text-center"><button class="btn-lg btn-block btn-info" id
   ="click1" (click)="chackbyTcome(s.id_student,1)มา"></button></td>
27 <td class="text-center"><button class="btn-lg btn-block btn-warning
   " (click)="chackbyTlast(s.id_student,0.5)สาย"></button></td>
28 <td class="text-center"><button class="btn-lg btn-block btn-danger
   " id="click1" (click)="chackbyOut(s.id_student,1)ขาด"></button></td>
29 <td class="text-center" style="font-size: 1.2rem;">{{getTotal(s.weeks)
   }}</td>
30 </tr>
31 </tbody>
32 </table>

```

รูปที่ 4.49: โค้ดในไฟล์ info.component.html

จากภาพที่ 4.49 โค้ดในไฟล์ info.component.html ของหน้าจัดการคะแนน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-6 เป็นการสร้างตัวเลือกสำหรับค้นหาคนที่ มา สาย และขาด
- บรรทัดที่ 7-32 เป็นการโชว์ข้อมูลคะแนนของนักศึกษาแต่ละอาทิตย์ 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: โค้ดในไฟล์ info.component.ts

จากภาพที่ 4.50 โค้ดในไฟล์ info.component.ts ของหน้าจัดการคะแนน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ chackbyTcome โดยจะรับข้อมูล idstudent และ count มา
- บรรทัดที่ 2-10 เป็นการเรียนรู้ฟังก์ชัน chakname ในไฟล์ student.service.ts ซึ่งเป็นฟังก์ชันสำหรับให้คะแนนนักศึกษา

```

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: โค้ดในไฟล์ info.component.ts

จากภาพที่ 4.51 โค้ดในไฟล์ info.component.ts ของหน้าจัดการคะแนน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ chackbyTlast โดยจะรับข้อมูล idstudent และ count มา
- บรรทัดที่ 2-10 เป็นการเรียกใช้ฟังก์ชัน chacknamebyt ในไฟล์ student.service.ts ซึ่งเป็นฟังก์ชันสำหรับให้คะแนนนักศึกษา


```
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: โค้ดในไฟล์ info.component.ts

จากภาพที่ 4.52 โค้ดในไฟล์ info.component.ts ของหน้าจัดการคะแนน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ checkbyOut โดยจะรับข้อมูล idstudent และ count มา
- บรรทัดที่ 2-10 เป็นการเรียกใช้ฟังก์ชัน checknameOut ในไฟล์ student.service.ts ซึ่งเป็นฟังก์ชันสำหรับให้คะแนนนักศึกษา

4.2.6 โครงสร้างของโค้ดใน Angular หน้าดูประวัติการเข้าเรียน

```

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: โค้ดในไฟล์ infoname.component.html

จากภาพที่ 4.53 โค้ดในไฟล์ `infoname.component.html` ของหน้าจอประวัติการเข้าเรียน สามารถอธิบายการทำงานได้ดังนี้

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

```

1  query() {
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: โค้ดในไฟล์ `infoname.component.ts`

จากภาพที่ 4.54 โค้ดในไฟล์ `infoname.component.ts` ของหน้าจอประวัติการเข้าเรียน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ `query`
- บรรทัดที่ 3 เป็นการประกาศตัวแปรชื่อ `id` สำหรับเก็บข้อมูล `id` ของรายวิชาซึ่งดึงมาจากพารามิเตอร์
- บรรทัดที่ 4 เป็นการเรียกใช้ฟังก์ชัน `reportCourse` ในไฟล์ `student.service.ts` โดยฟังก์ชันนี้จะทำหน้าที่ดึงข้อมูลชั้นเรียนที่มี `id` ตรงกับ `id` ที่ส่งไป
- บรรทัดที่ 6 เป็นการเก็บข้อมูลที่ได้จากการเรียกใช้ฟังก์ชัน `reportCourse` ไว้ในตัวแปรชื่อ `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: โค้ดในไฟล์ infoname.component.ts

จากภาพที่ 4.55 โค้ดในไฟล์ infoname.component.ts ของหน้าจอดูประวัติการเข้าเรียน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ downloadfile
- บรรทัดที่ 2 เป็นการประกาศตัวแปรชื่อ excelfile สำหรับเก็บข้อมูล
- บรรทัดที่ 3-11 เป็นการเตรียมข้อมูลสำหรับการดาวน์โหลด
- บรรทัดที่ 12 เป็นการประกาศตัวแปร nameCoure เพื่อเก็บชื่อของไฟล์ที่จะดาวน์โหลด
- บรรทัดที่ 13 เป็นการเรียกใช้ฟังก์ชัน exportAsExcelFile ในไฟล์ excel.service.ts โดยฟังก์ชันนี้จะทำหน้าที่แปลงข้อมูลอาเรย์ให้เป็นไฟล์ excel แล้วดาวน์โหลดไฟล์
- บรรทัดที่ 14 เป็นการเรียกใช้ฟังก์ชัน 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 ไม่ร่งกัน')
7  }
8  }

```

รูปที่ 4.56: โค้ดในไฟล์ infoname.component.ts

จากภาพที่ 4.56 โค้ดในไฟล์ infoname.component.ts ของหน้าจอดูประวัติการเข้าเรียน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ totlescore
- บรรทัดที่ 2 เป็นการประกาศตัวแปรชื่อ a สำหรับเก็บข้อมูลที่คืนค่ามาจากฟังก์ชัน sumscore โดยฟังก์ชันจะคืนค่าข้อมูลการรวมคะแนนเป็นอาเรย์
- บรรทัดที่ 3-7 สร้างเงื่อนไขตรวจสอบว่า a มีข้อมูลหรือไม่ถ้ามีจะเรียกใช้ฟังก์ชัน updateStudent โดยฟังก์ชันนี้จะอัปเดตคะแนนของนักศึกษา ถ้าไม่มีข้อมูลจะแสดงข้อความ fire ไม่ร่งกัน

4.2.7 โครงสร้างของโค้ดใน Angular หน้ากรอกรหัสนักศึกษา

```

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: โค้ดในไฟล์ fromstudent.component.html

จากภาพที่ 4.57 โค้ดในไฟล์ fromstudent.component.html ของหน้ากรอกรหัสนักศึกษา 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: โค้ดในไฟล์ fromstudent.component.ts

จากภาพที่ 4.58 โค้ดในไฟล์ fromstudent.component.ts ของหน้ากรอกรหัสนักศึกษาสามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันชื่อ submid โดยฟังก์ชันนี้จะรับข้อมูลรหัสนักศึกษา
- บรรทัดที่ 2 เป็นการประกาศตัวแปรชื่อ coure สำหรับเก็บข้อมูลที่ดึงจาก localStorage
- บรรทัดที่ 3 เป็นการประกาศตัวแปรชื่อ weekc สำหรับเก็บข้อมูลที่ดึงจาก localStorage
- บรรทัดที่ 4 เป็นการประกาศตัวแปรชื่อ checkid สำหรับเก็บข้อมูลที่ดึงจาก localStorage
- บรรทัดที่ 6 เป็นการประกาศตัวแปรชื่อ between สำหรับเก็บข้อมูลที่คืนค่าจากฟังก์ชัน getlocaltion โดยฟังก์ชันนี้จะคำนวณระยะห่างระหว่างอุปกรณ์ของนักศึกษาและอุปกรณ์ของอาจารย์
- บรรทัดที่ 7 เป็นการสร้างเงื่อนไขตรวจสอบระยะห่างของอุปกรณ์ระหว่างอาจารย์และนักศึกษา โดยห่างไม่เกิน 150 เมตร ถ้าไม่เกินจะเรียกใช้ฟังก์ชัน checknamesad ซึ่งฟังก์ชันนี้เป็นฟังก์ชันสำหรับเช็คชื่อและตรวจสอบข้อมูลว่านักศึกษาเช็คชื่อไปแล้วหรือไม่ ถ้าเกินระยะจะแสดงข้อความ ไม่สามารถเช็คชื่อได้

บทที่ 5

การทดสอบระบบ

การทดสอบการทำงานของเว็บแอปพลิเคชันระบบเช็คซื้อด้วยคิวอาร์โค้ด โดยทำการทดสอบในลักษณะ Black-box Testing [10] หรือ Data-Driven testing ซึ่งเป็นการทดสอบที่ไม่สนใจโปรเซส (Process) การทำงานภายในของโปรแกรมว่าทำงานอย่างไร แต่จะเน้นไปที่ Input และ Result ที่ได้มากกว่าว่าการทำงานต่าง ๆ ถูกต้องตามความต้องการ (Requirement) หรือไม่ ซึ่งการทดสอบการใช้งานเว็บแอปพลิเคชัน ได้ผลดังนี้

5.1 การทดสอบการใช้งานเว็บแอปพลิเคชัน

5.1.1 การทดสอบหน้าสร้างชั้นเรียน

การทดสอบหน้าสร้างชั้นเรียนของเว็บแอปพลิเคชันในการสร้างชั้นเรียนของอาจารย์ ซึ่งการทำงานประกอบด้วย เมนูเลือกไฟล์ เมนูอัปโหลดไฟล์ และรายละเอียดของชั้นเรียนที่อัปโหลด ผลทดสอบแสดงดังตารางที่ 5.1

ตารางที่ 5.1: ผลการทดสอบการสร้างชั้นเรียน

| การทำงาน | เงื่อนไขการทดสอบ | ผลการทดสอบ |
|----------------|---------------------------------|--|
| สร้างชั้นเรียน | กดปุ่มเมนูเลือกไฟล์ | ระบบ แสดง ผล หน้า จอ เลือกไฟล์ที่จะอัปโหลด |
| | เลือกไฟล์ excel ที่ถูกเติม-เพลด | ระบบ แสดง ราย ละเอียด ข้อมูลของชั้นเรียน ถ้าไฟล์ไม่ถูกเติมเพลดระบบแสดงข้อความ “ไฟล์ไม่ถูกต้อง” |
| | กดปุ่มเมนูเลือกเทอม | ระบบแสดงตัวเลือก 1 2 และ 3 |
| | กดปุ่มเมนูอัปโหลดไฟล์ | ระบบ จะ ทำการ อัปโหลดไฟล์ ไปที่ฐานข้อมูล และกลับไป หน้าหลัก |

5.1.2 การทดสอบหน้าอัปเดตข้อมูลนักศึกษา

ในการแสดงผลหน้าอัปเดตข้อมูลนักศึกษานั้นจะประกอบไปด้วย รายละเอียดรายชื่อนักศึกษา เมนูเพิ่มนักศึกษา เมนูแก้ไขและเมนูลบ ผลการทดสอบดังตารางที่ 5.2

ตารางที่ 5.2: ผลการทดสอบหน้าอัปเดตข้อมูลนักศึกษา

| การทำงาน | เงื่อนไขการทดสอบ | ผลการทดสอบ |
|--------------------------|---------------------|--|
| หน้าอัปเดตข้อมูลนักศึกษา | กดปุ่มเพิ่มนักศึกษา | ระบบ แสดง ผล หน้า สำหรับกรอกข้อมูลนักศึกษา |
| | กดปุ่มแก้ไข | ระบบ แสดง ผล หน้า สำหรับกรอกข้อมูลนักศึกษา |
| | กดปุ่มลบ | ระบบลบข้อมูลนักศึกษา |
| | กดปุ่มกลับ | ระบบแสดงผลหน้าจอหลัก |

5.1.3 การทดสอบหน้าสร้างคิวอาร์โค้ด

ในการแสดงผลหน้าสร้างคิวอาร์โค้ดนั้นจะประกอบไปด้วย เมนูเลือกสัปดาห์ เมนูเลือกเวลา และเมนูสร้างคิวอาร์โค้ด ผลการทดสอบดังตารางที่ 5.3

ตารางที่ 5.3: ผลการทดสอบหน้าสร้างคิวอาร์โค้ด

| 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 เมนูสาย และเมนูขาด ผลการทดสอบดังตารางที่ 5.4

ตารางที่ 5.4: ผลการการทดสอบหน้าจัดการคะแนน

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

5.1.5 การทดสอบหน้าดูประวัติการเข้าเรียน

ในการแสดงผลหน้าดูประวัติการเข้าเรียนนั้นจะประกอบไปด้วย รายชื่อและคะแนนนักศึกษาทั้งหมด เมนูดาวน์โหลดไฟล์และเมนูรวมไฟล์ ผลการทดสอบดังตารางที่ 5.5

ตารางที่ 5.5: ผลการการทดสอบหน้าดูประวัติการเข้าเรียน

| การทำงาน | เงื่อนไขการทดสอบ | ผลการทดสอบ |
|-----------------------|----------------------------|---|
| ดูประวัติการเข้าเรียน | กดปุ่มดูประวัติจากหน้าหลัก | ระบบ แสดงผล หน้าจอ ตาราง ราย ชื่อ และ คะแนน ของ นัก-ศึกษาทั้งหมดในชั้นเรียน |
| | กดปุ่มดาวน์โหลดไฟล์ | ระบบ ทำการ ดาวน์โหลด ราย ชื่อ และ คะแนน ของ นักศึกษาทั้งหมดในชั้นเรียน โดยดาวน์โหลดเป็นไฟล์ excel |
| | กดปุ่มรวมไฟล์ | ระบบ แสดง ผล หน้า จอ เลือก ไฟล์ ที่ จะ อัปโหลด โดย ไฟล์ ต้อง เป็น ไฟล์ ที่ ดาวน์โหลด จาก ระบบ ชี ค ชื่อ ด้วย คิวอาร์โค้ด ถ้า ไม่ใช่ ระบบ จะ แสดง ข้อความ “ไฟล์ ไม่ ถูก ต้อง” เมื่อ เลือก ไฟล์ แล้ว ระบบ จะ แสดง รายละเอียด ข้อมูล ของ ชั้นเรียน |
| | กดปุ่ม save | ระบบ ทำการ อัปเดต ข้อมูล คะแนน แต่ละ สัปดาห์ และ บันทึกลงในฐานข้อมูล |
| | กดปุ่มกลับ | ระบบแสดงผลหน้าจอหลัก |

5.1.6 การทดสอบหน้ากรอกรหัสนักศึกษา

ในการแสดงผลหน้ากรอกรหัสศึกษานั้นจะประกอบไปด้วย ช่องกรอกข้อมูล เมนูตกลง ผลการทดสอบดังตารางที่ 5.6

ตารางที่ 5.6: ผลการการทดสอบหน้ากรอกรหัสนักศึกษา

| การทำงาน | เงื่อนไขการทดสอบ | ผลการทดสอบ |
|------------------|---------------------------------------|---|
| กรอกรหัสนักศึกษา | สแกนคิวอาร์โค้ดของอาจารย์ ก่อนหมดเวลา | ระบบ แสดง หน้า กรอก ข้อมูล สำหรับรหัสนักศึกษา |
| | สแกนคิวอาร์โค้ดของอาจารย์ หลังหมดเวลา | ระบบ แสดง หน้า หมด เวลา สแกน |
| | กดปุ่มเมนูตกลง | ระบบ ทำการ ส่ง ข้อมูล รหัส นักศึกษา ไป ตรวจสอบ และ บันทึกคะแนนลงในฐานข้อมูล |

บทที่ 6

สรุปและข้อเสนอแนะ

การดำเนินโครงการเพื่อพัฒนาระบบเช็คชื่อด้วยคิวอาร์โค้ดนี้ พบว่าระบบสามารถทำงานได้ตามที่วิเคราะห์และออกแบบไว้ แต่ก็พบปัญหาและอุปสรรคระหว่างการพัฒนา ในบทนี้ผู้พัฒนาจึงขอสรุปความสามารถของระบบ ชี้แจงปัญหาและอุปสรรค พร้อมเสนอแนวทางในการพัฒนาระบบเช็คชื่อด้วยคิวอาร์โค้ดต่อตามลำดับ

6.1 สรุปความสามารถของระบบ

ระบบเช็คชื่อด้วยคิวอาร์โค้ด เว็บแอปพลิเคชันสามารถสรุปความสามารถที่ระบบทำได้ดังนี้

6.1.1 เว็บแอปพลิเคชัน

ความสามารถหลักของเว็บแอปพลิเคชันนั้นเน้นสร้างความสะดวกต่อการจัดการเอกสารเรื่องข้อมูลต่างๆ ที่เกี่ยวข้องกับระบบเช็คชื่อด้วยคิวอาร์โค้ด โดยแบ่งความสามารถของระบบตามประเภทของผู้ใช้งานดังนี้

1. อาจารย์

- ตรวจสอบรายชื่อนักศึกษาในคลาสเรียน
- เพิ่ม และลบ คลาสเรียน
- จัดการคะแนนของนักศึกษา
- เพิ่ม ลบ และแก้ไข ชื่อนักศึกษาในคลาสเรียนได้
- รวมไฟล์คะแนน ของคลาสเรียนเดียวกัน
- สร้างคิวอาร์โค้ด สำหรับให้นักศึกษาสแกนเพื่อเช็คชื่อได้

2. นักศึกษา

- สแกนคิวอาร์โค้ดของผู้สอนเพื่อเป็นการเช็คชื่อเข้าเรียน

6.2 ปัญหาและอุปสรรคในการพัฒนา

1. เนื่องจากตัวระบบเป็นเว็บแอปพลิเคชันจึงไม่สามารถตั้งค่า IP ของเครื่องคอมพิวเตอร์ของอาจารย์ได้จึงจำเป็นต้องตั้งค่าเลข IP ก่อนจะใช้งานระบบ
แนวทางการแก้ไข : ทำเป็นโปรแกรมเดสก์ทอปที่สามารถติดตั้งบนเครื่องคอมพิวเตอร์ ซึ่งสามารถตั้งค่า IP ของเครื่องคอมพิวเตอร์มาใช้งานได้

6.3 แนวทางการพัฒนาต่อ

1. พัฒนาระบบ Login สำหรับอาจารย์เพื่อใช้ฐานข้อมูลเดียวกัน ซึ่งจะแก้ปัญหาเรื่องการแก้ไขจำนวนนักศึกษาของอาจารย์แต่ละคน และทำไม่สามารถรวมไฟล์คะแนนได้เนื่องจากจำนวนนักศึกษาไม่ตรงกันจากการแก้ปัญหานี้
2. สร้าง Web server ของระบบซึ่งเป็นโปรแกรมที่มีหน้าที่ให้บริการด้านการจัดการเว็บไซต์ และ Database server ซึ่งเป็นโปรแกรมที่ทำหน้าที่ให้บริการด้านการจัดการดูแลข้อมูลต่าง ๆ ภายในเว็บไซต์
3. พัฒนาระบบให้เป็นโปรแกรมบนเดสก์ทอป โดยสามารถเปิดใช้งานผ่านเครื่องคอมพิวเตอร์ได้เลยไม่จำเป็นต้องเข้าผ่านเว็บเบราว์เซอร์

บรรณานุกรม

- [1] KimJaeHa (2549). เว็บแอปพลิเคชัน [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] ผู้ช่วยศาสตราจารย์ ดร.เกรียงศักดิ์ จันทน์นอก (2560). smartclassroom qr-code check-in [ออนไลน์]. สืบค้นเมื่อ 20 ธันวาคม 2562. จาก <http://www.mse-exam.net/q> .
- [8] Student Care Co., Ltd.. (2556). ระบบ student care [ออนไลน์]. สืบค้นเมื่อ 20 ธันวาคม 2562. จาก <http://www.student.co.th/> .
- [9] Kunchit Phiu-Nual. (2557). ความหมายและความสำคัญของ system architecture [ออนไลน์]. สืบค้นเมื่อ 12 พฤษภาคม 2562. จาก <https://goo.gl/6ZhGQo> .
- [10] 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

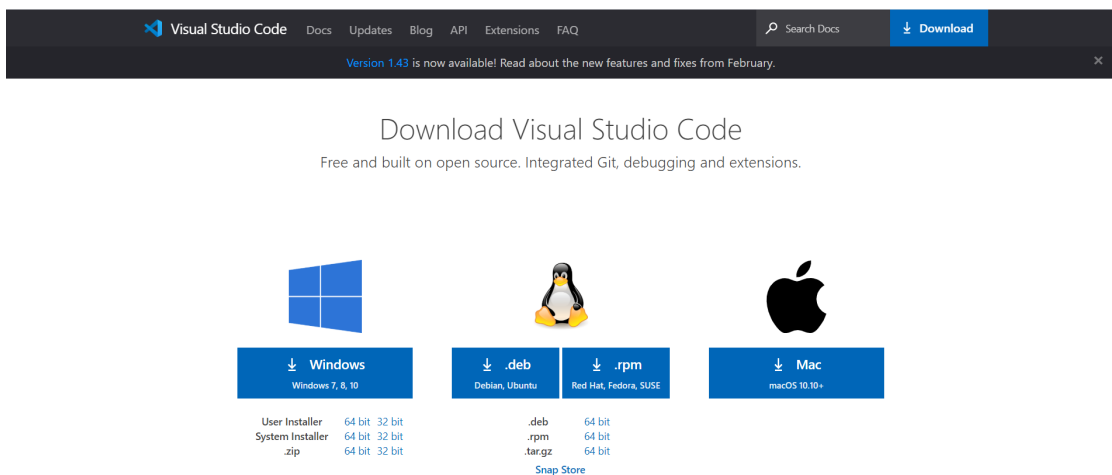
การติดตั้งเครื่องมือที่ใช้พัฒนาโปรแกรม

การติดตั้งเครื่องมือที่ใช้ในการพัฒนาเว็บแอปพลิเคชันระบบเช็คชื่อกับคิวอาร์โค้ด มีโปรแกรมที่จำเป็นในการพัฒนาระบบดังต่อไปนี้

- การติดตั้ง Visual Studio Code
- การติดตั้ง Go
- การติดตั้ง Angular Framework

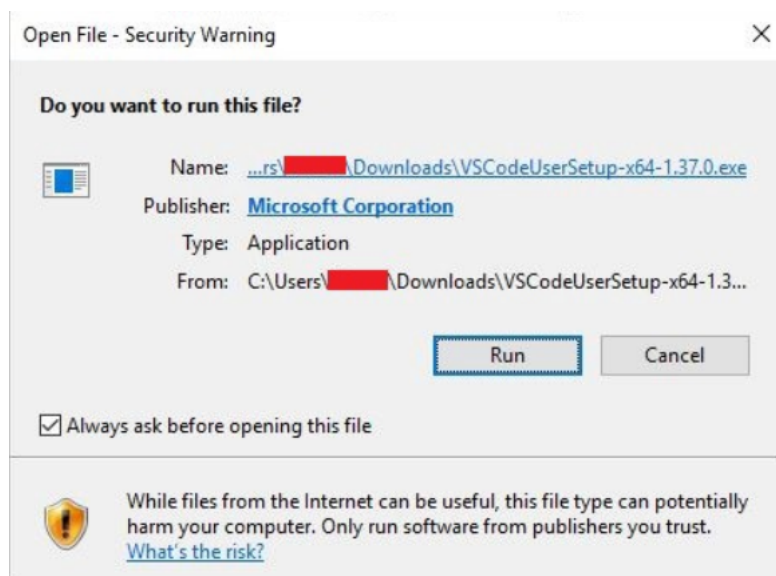
ก.1 การติดตั้ง Visual Studio Code

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



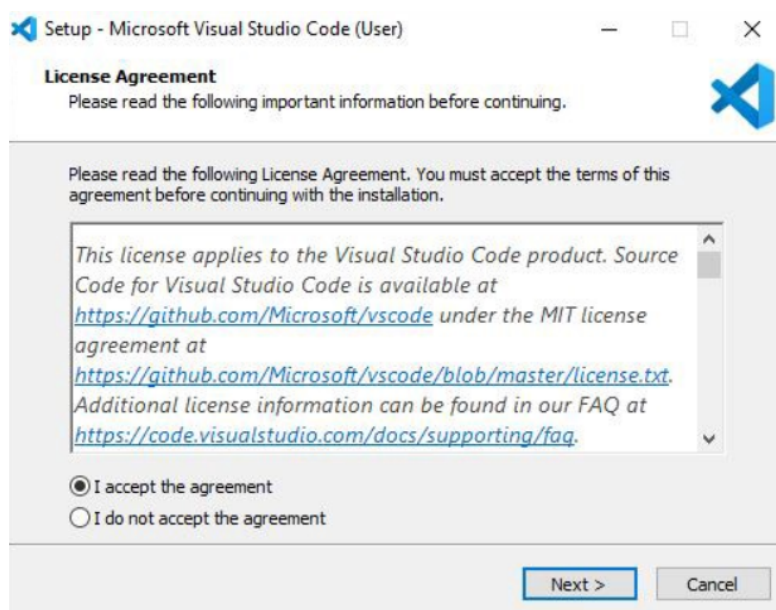
รูปที่ ก.1: หน้าเว็บดาวน์โหลด Visual Studio Code

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



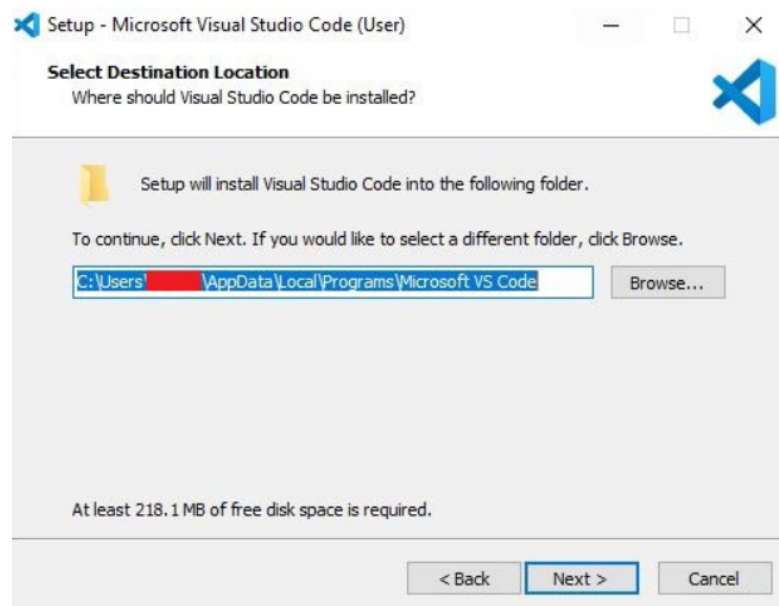
รูปที่ ก.2: หน้าต่างต้อนรับของ Visual Studio Code

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



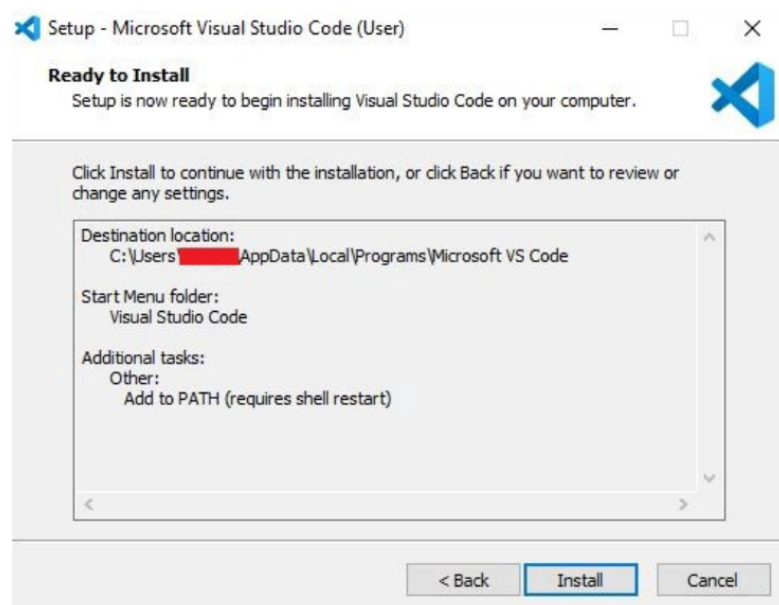
รูปที่ ก.3: หน้าต่างข้อตกลงการใช้งาน Visual Studio Code

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



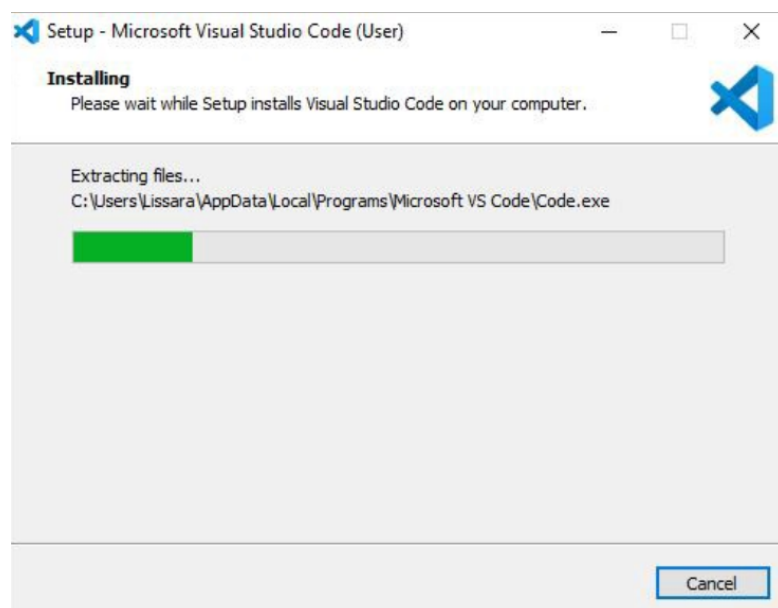
รูปที่ ก.4: หน้าต่างที่จัดเก็บไฟล์ต่างๆ ของ Visual Studio Code

5. หน้าต่างเริ่มทำการติดตั้งทำการกด Install ดังแสดงในรูปที่ ก.5



รูปที่ ก.5: หน้าต่างที่จัดเก็บไฟล์ต่างๆ ของ Visual Studio Code

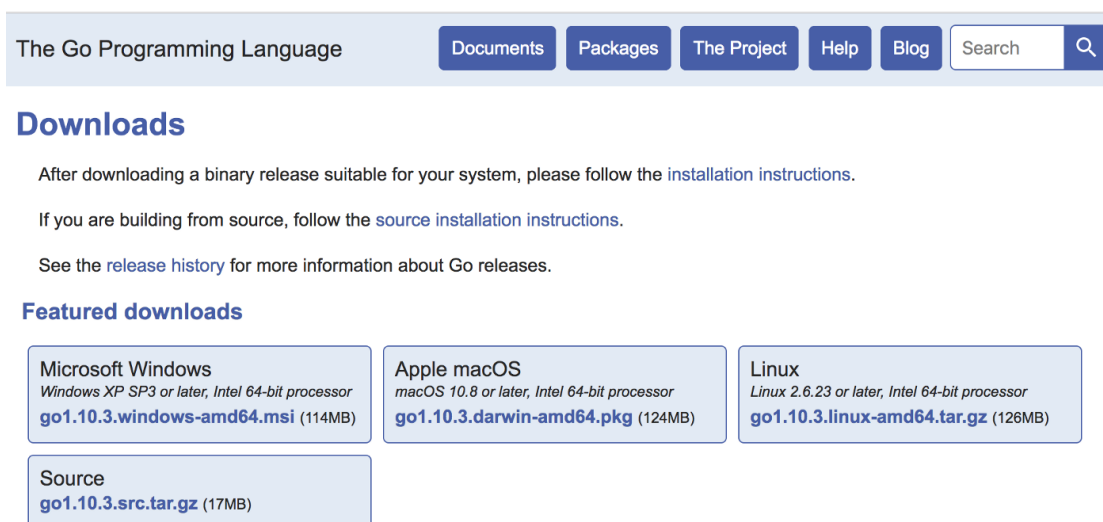
6. หน้าต่างผลการติดตั้ง Visual Studio Code ดังแสดงในรูปที่ ก.6



รูปที่ ก.6: หน้าต่างผลการติดตั้ง Visual Studio Code

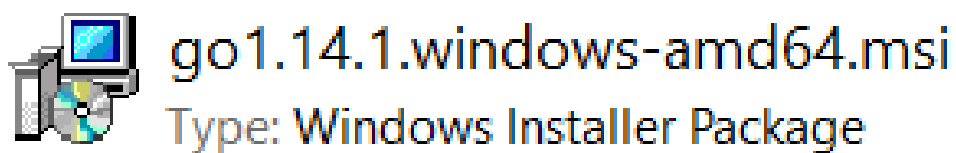
ก.2 การติดตั้ง Go

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



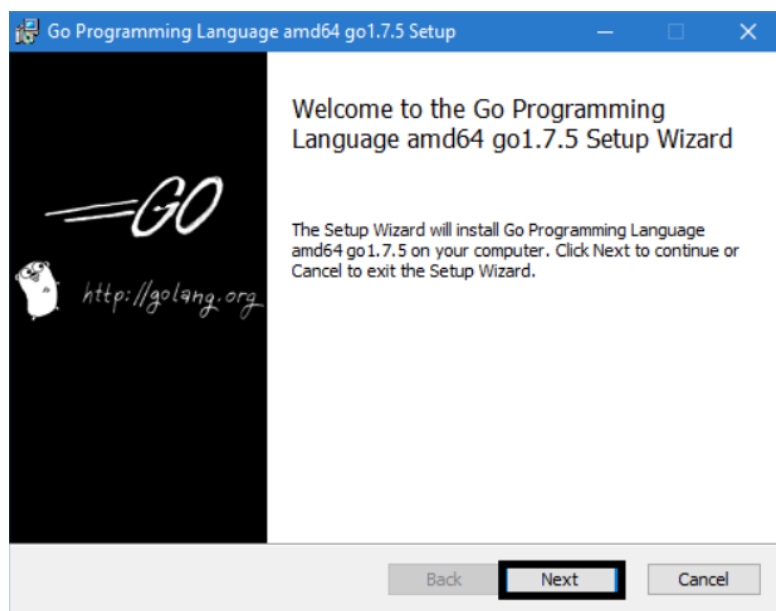
รูปที่ ก.7: หน้าเว็บดาวน์โหลด Go

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



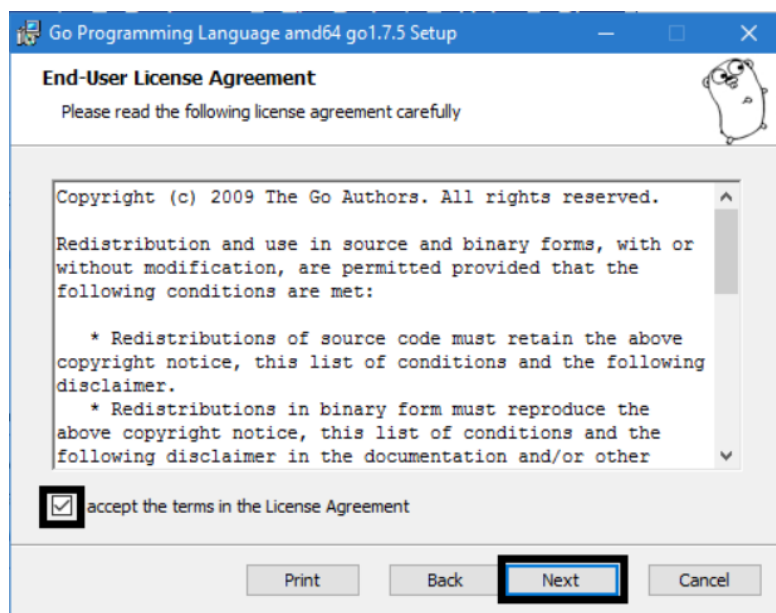
รูปที่ ก.8: ไฟล์ติดตั้งสำหรับติดตั้ง Go

3. แสดงหน้าต่างต้อนรับของ Go ทำการกด Next เพื่อเริ่มกระบวนการติดตั้ง ดังแสดงในรูปที่ ก.9



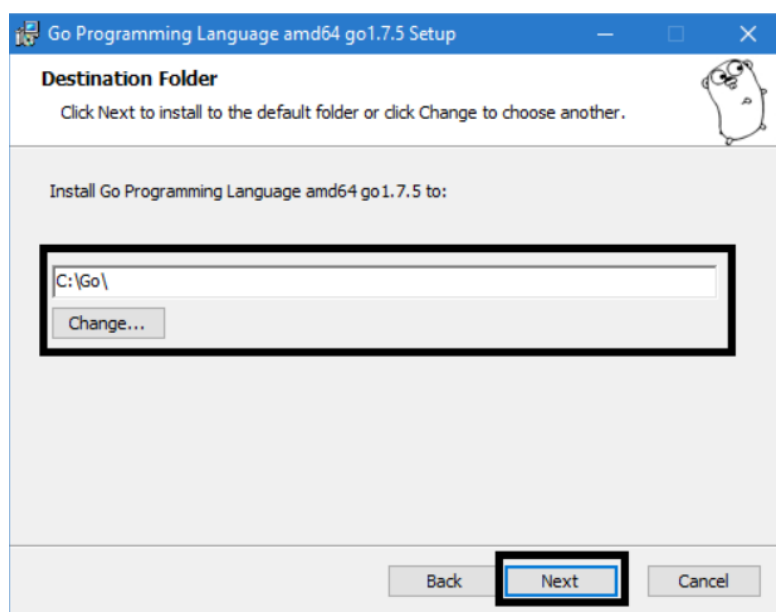
รูปที่ ก.9: หน้าต่างต้อนรับของ Go

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



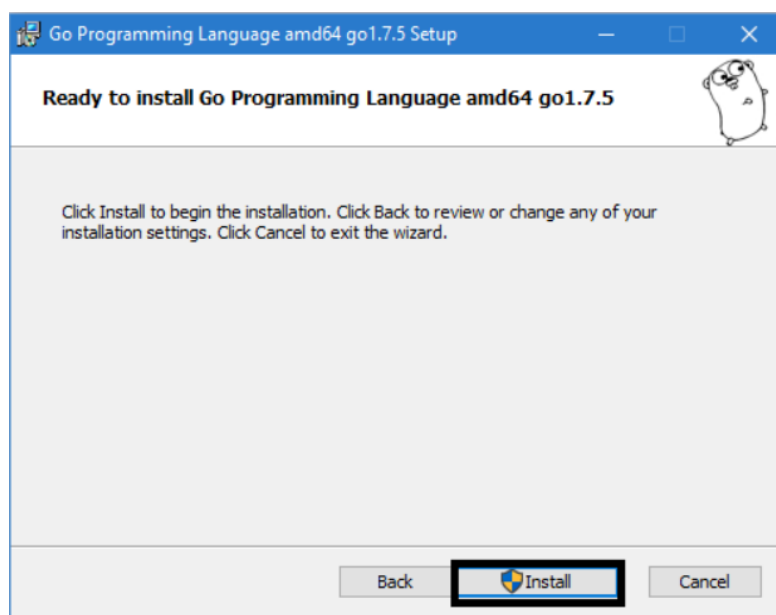
รูปที่ ก.10: หน้าต่างข้อตกลงในการใช้ Go

5. แสดงหน้าต่างเลือกโฟลเดอร์ที่จะทำการติดตั้ง ดังแสดงในรูปที่ ก.11



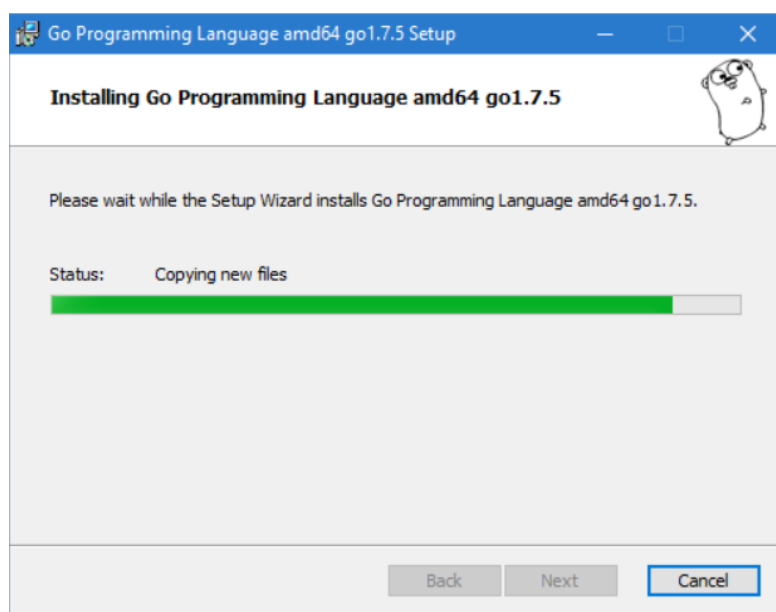
รูปที่ ก.11: หน้าต่างเลือกโฟลเดอร์ที่จะทำการติดตั้ง Go

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 Framework

การติดตั้ง Angular Framework สามารถผ่านคำสั่ง command line ได้ โดยจำเป็นต้องทำการติดตั้ง Node.js ก่อนเพื่อใช้ในกระบวนการติดตั้งนี้ ดังแสดงในรูปที่ ก.14

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

รูปที่ ก.14: คำสั่งสำหรับติดตั้ง Angular Framework

ผลการติดตั้ง Angular Framework 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

เนื่องด้วยระบบเช็คชื่อด้วยคิวอาร์โค้ดเป็นเว็บแอปพลิเคชัน ถูกพัฒนาบนพื้นฐานการทำงานบนเว็บไซต์รองรับการใช้งานทั้งบนคอมพิวเตอร์และโทรศัพท์มือถือ ทำให้ผู้ใช้สามารถเข้าใช้งานผ่านเว็บเบราว์เซอร์ได้โดยไม่ต้องทำการติดตั้งเพื่อใช้งาน

1. หน้าเว็บเบราว์เซอร์สำหรับหน้าจอคอมพิวเตอร์ ดังแสดงในรูปที่ ข.1

Menu

เพิ่ม ชั้นเรียน

CHECK NAME

Classroom All

ผู้สอน : เจดพล ชินพันธ์

ค้นหาวิชา

1144133

:

โครงสร้างข้อมูล Sec 02

เทอม : 1

เวลาเรียนสอน : จ. 13:00-15:50 SC301

QRcode

ลบ

1144133

:

โครงสร้างข้อมูล Sec 02

เทอม : 2

เวลาเรียนสอน : จ. 13:00-15:50 SC301

QRcode

ลบ

1144334

:

การสื่อสารข้อมูลและเครือข่ายอินเทอร์เน็ต Sec 01

เทอม : 1

เวลาเรียนสอน : ศ. 09:00-12:00 SC202

QRcode

ลบ

1144334

:

การสื่อสารข้อมูลและเครือข่ายอินเทอร์เน็ต Sec 01

เทอม : 2

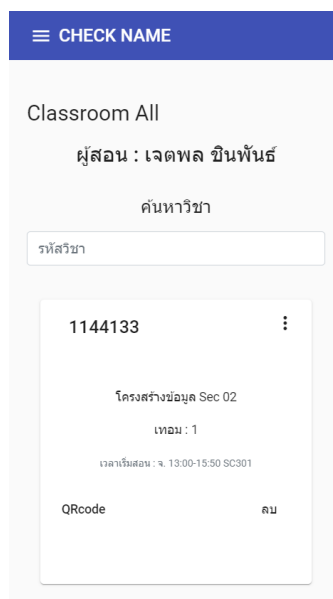
เวลาเรียนสอน : ศ. 09:00-12:00 SC202

QRcode

ลบ

รูปที่ ข.1: หน้าเว็บแอปพลิเคชัน ระบบเช็คชื่อด้วยคิวอาร์โค้ดมุมมองของคอมพิวเตอร์

2. หน้าเว็บเบราว์เซอร์สำหรับหน้าจอโทรศัพท์มือถือ ดังแสดงในรูปที่ ข.2



≡ CHECK NAME

Classroom All

ผู้สอน : เจตพล ชื่นพันธ์

ค้นหาวิชา

รหัสวิชา

1144133

โครงสร้างข้อมูล Sec 02

เทอม : 1

เวลาเรียนสอน : ก. 13:00-15:50 SC301

QRcode ลบ

รูปที่ ข.2: หน้าเว็บแอปพลิเคชัน ระบบเช็คชื่อด้วยคิวอาร์โค้ดมุมมองของโทรศัพท์มือถือ

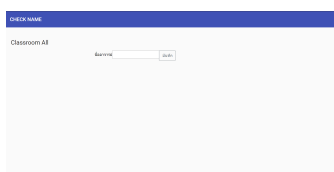
ภาคผนวก ค

คู่มือการใช้งานระบบ

คู่มือการใช้งานทั้งหมดของระบบเช็คชื่อด้วยคิวอาร์โค้ด เบื้องต้น

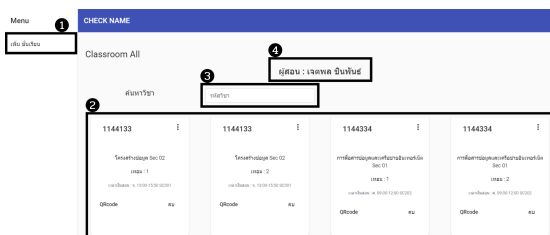
1. ส่วนของหน้าเมนูเว็บแอปพลิเคชันสำหรับอาจารย์

- หน้าจอหลักแสดงผลครั้งแรกเมื่อผู้ใช้ทำการเปิดใช้งานเว็บแอปพลิเคชัน ดังแสดงในรูปที่ ค.1



รูปที่ ค.1: หน้าจอหลักเมื่อเข้าใช้งานครั้งแรก

- เมื่อทำการกรอกชื่อในระบบและกดตกลง ระบบจะทำการบันทึกชื่อลงใน localStorage จากนั้นระบบจะแสดงข้อมูลชั้นเรียนที่มีทั้งหมด ดังแสดงในรูปที่ ค.2

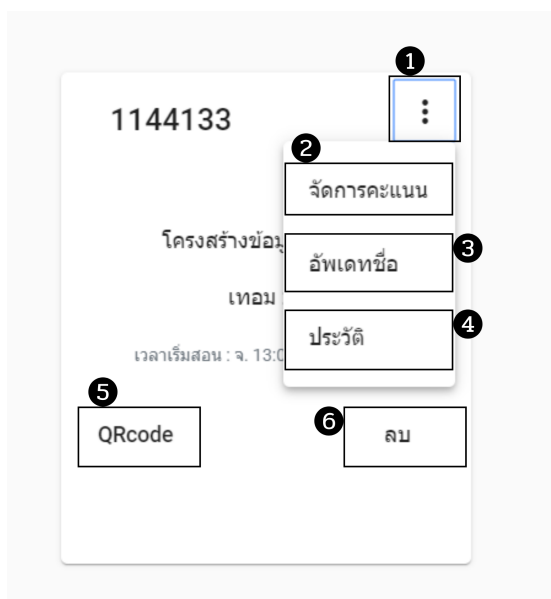


รูปที่ ค.2: หน้าหลัก

จากรูปที่ ค.2 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ ปุ่มสำหรับการไปยังหน้าเพิ่มชั้นเรียน
- หมายเลข 2 คือ แสดงชั้นเรียนทั้งหมดที่มีในฐานข้อมูล
- 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 ปุ่มสำหรับการไปยังหน้าอัปเดตชื่อ
- หมายเลข 4 คือ ปุ่มสำหรับการไปยังหน้าประวัติ
- หมายเลข 5 คือ ปุ่มสำหรับการไปยังหน้าสร้างคิวอาร์โค้ด
- หมายเลข 6 คือ ปุ่มสำหรับลบชั้นเรียน

- เมื่อผู้ใช้กดปุ่ม เพิ่มชั้นเรียน ระบบจะแสดงหน้าเพิ่มชั้นเรียนซึ่งเป็นหน้าสำหรับเพิ่มชั้นเรียนใหม่ ดังแสดงในรูปที่ ค.4

Menu

เช็คชั้นเรียน

CHECK NAME

New Classroom

1

2

3

4

| ลำดับ | รหัสประจำตัว | ชื่อของนักเรียน |
|-------|--------------|------------------|
| 1 | 0012402769 | นางสาวณิชา พลสุข |
| 2 | 0012403131 | นางสาวณิชา พลสุข |
| 3 | 0018402012 | นางสาวณิชา พลสุข |
| 4 | 0020403027 | นางสาวณิชา พลสุข |

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 ปุ่มสำหรับอัปเดตข้อมูลชั้นเรียน
- เมื่อผู้ใช้กดปุ่ม จัดการคะแนน ระบบจะแสดงหน้าจัดการคะแนนนักศึกษาซึ่งเป็นหน้าสำหรับจัดการคะแนนของนักศึกษาในชั้นเรียน ดังแสดงในรูปที่ ค.5

Menu

เช็คชั้นเรียน

CHECK NAME

Classroom Info

1

2

3

4

| ลำดับ | รหัสประจำตัว | ชื่อ | คะแนน | หมายเหตุ |
|-------|--------------|------------------|-------|----------|
| 1 | 0012401014 | นางสาวณิชา พลสุข | 1 | |
| 2 | 0012403029 | นางสาวณิชา พลสุข | 1 | |
| 3 | 0012404046 | นางสาวณิชา พลสุข | 1 | |
| 4 | 0012404025 | นางสาวณิชา พลสุข | 0.5 | |
| 5 | 0012404004 | นางสาวณิชา พลสุข | 0.5 | |

รูปที่ ค.5: หน้าจัดการคะแนน

จากรูปที่ ค.5 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ พื้นที่แสดงข้อมูลของชั้นเรียน
- หมายเลข 2 คือ ปุ่มตัวเลือกสำหรับเลือกเพื่อค้นหาข้อมูล

- หมายเลข 3 คือ ตารางแสดงรายชื่อทั้งหมดในชั้นเรียน
- หมายเลข 4 คือ แสดงคะแนนในอาทิต้นนั้นๆ
- หมายเลข 5 คือ ปุ่มสำหรับให้คะแนนนักศึกษา
- หมายเลข 6 คือ แสดงคะแนนรวมทั้งหมด
- เมื่อผู้ใช้กดปุ่ม อัปเดตชื่อ ระบบจะแสดงหน้าอัปเดตชื่อซึ่งเป็นหน้าสำหรับ เพิ่ม ลบ และแก้ไข ชื่อและรหัสนักศึกษา ดังแสดงในรูปที่ ค.6

รูปที่ ค.6: หน้าอัปเดตชื่อ

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

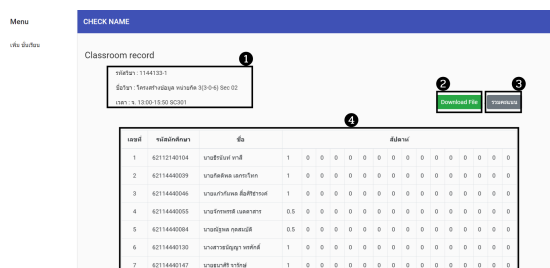
- หมายเลข 2 คือ ปุ่มยกเลิก สำหรับกลับสู่หน้าออฟเดทชื่อ
- เมื่อผู้ใช้กดปุ่ม แก้ไข ระบบจะแสดงหน้าสำหรับกรอกข้อมูลนักศึกษา ดังแสดงในรูปที่ ค.8



รูปที่ ค.8: หน้าจอต้อนรับ

จากรูปที่ ค.8 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ ช่องสำหรับกรอกข้อมูลนักศึกษา
 - หมายเลข 2 คือ ปุ่ม save สำหรับบันทึกข้อมูล
 - หมายเลข 3 คือ ปุ่ม close สำหรับกลับสู่หน้าอ็อปเดทชื่อ
 - หมายเลข 3 คือ ปุ่ม X สำหรับกลับสู่หน้าอ็อปเดทชื่อ
- เมื่อผู้ใช้กดปุ่ม ประวัติ ระบบจะแสดงหน้าประวัติซึ่งเป็นหน้าสำหรับแสดงข้อมูลของชั้นเรียนทั้งหมด ดังแสดงในรูปที่ ค.9



รูปที่ ค.9: หน้าประวัติ

จากรูปที่ ค.9 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ พื้นที่แสดงข้อมูลของชั้นเรียน
- หมายเลข 2 คือ ปุ่มดาวโหลด เป็นปุ่มสำหรับดาวโหลดข้อมูลชั้นเรียน โดยจะดาวโหลดเป็นไฟล์ 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 ช่องสำหรับเลือกไฟล์ข้อมูลชั้นเรียนที่จะทำการรวมคะแนนโดยจะต้องเป็นไฟล์ที่ดาวน์โหลดจากระบบเท่านั้น
- หมายเลข 2 คือ ตารางแสดงรายชื่อทั้งหมดในชั้นเรียน ของไฟล์ที่เลือก
- หมายเลข 3 คือ ปุ่ม save สำหรับบันทึกข้อมูล
- หมายเลข 4 คือ ปุ่ม close สำหรับกลับสู่หน้าจอพีเดทชื่อ
- หมายเลข 4 คือ ปุ่ม X สำหรับกลับสู่หน้าจอพีเดทชื่อ
- เมื่อผู้ใช้กดปุ่ม QRcode ระบบจะแสดงหน้าสร้างคิวอาร์โค้ดซึ่งเป็นหน้าสำหรับสร้างคิวอาร์โค้ดให้นักศึกษาสแกน ดังแสดงในรูปที่ ค.11

รูปที่ ค.11: หน้าสร้างคิวอาร์โค้ด

จากรูปที่ ค.11 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ ปุ่มตัวเลือกสำหรับตั้งค่าสปีดอาร์และเวลาของคิวอาร์โค้ด
- หมายเลข 2 คือ ปุ่ม Start QR code เป็นปุ่มสำหรับสร้างคิวอาร์โค้ด
- เมื่อผู้ใช้กดปุ่ม Start QR code ระบบจะแสดงคิวอาร์โค้ด ดังแสดงในรูปที่ ค.12



รูปที่ ค.12: หน้าแสดงคิวอาร์โค้ด

จากรูปที่ ค.12 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ แสดงสปีดาร์ของคิวอาร์โค้ด
- 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

ประวัติผู้พัฒนา

ชื่อ-สกุล: นายเจตพล ชินพันธ์

รหัสประจำตัวนักศึกษา: 59110440112

วันเกิด: 09 06 2540

ที่อยู่ที่สามารถติดต่อได้: 11 หมู่1 ตำบลหนองแค อำเภอราชไศล จังหวัดศรีสะเกษ 33160

เบอร์โทรศัพท์: (+66) 0660643497

อีเมลล์: jadtaphon.ch.59@ubu.ac.th

ระดับมัธยมต้น: โรงเรียนราชไศล จังหวัดศรีสะเกษ

ระดับมัธยมปลาย: โรงเรียนราชไศล จังหวัดศรีสะเกษ

ระดับอุดมศึกษา: ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ สาขาวิทยาการ คอมพิวเตอร์ คณะ
วิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี