ระบบจองคิวร้านเสริมสวย
Beauty salon queue reservation system

นางสาวปิยพร อาภรศรี

โครงงานนี้เป็นส่วนหนึ่งของการศึกษา
หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ คณะวิทยาศาสตร์
มหาวิทยาลัยอุบลราชธานี
ปีการศึกษา 2562

โครงงาน	:	ระบบจองคิวร้านเสริมสวย Beauty salon queue reservation system
โดย	:	นางสาวปิยพร อาภรศรี
อาจารย์ที่ปรึกษา	:	ดร.ทศพร จูฉิม
ระดับการศึกษา	:	วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ปีการศึกษา	:	2562

ได้รับการพิจารณาให้เป็นส่วนหนึ่งของการศึกษา

ตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์

คณะกรรมการสอบประเมินความรู้โครงงานคอมพิวเตอร์

. . . . . อาจารย์ที่ปรึกษา
(ดร.ทศพร จูฉิม)

. . . . . กรรมการ
(อาจารย์ วาโย ปุยะติ)

. . . . . กรรมการ
(อาจารย์ วาสนา เหง้าเกษ)

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

วันที่ . . . / . . . / . . .

กิตติกรรมประกาศ

การพัฒนาโครงงานระบบจองคิวร้านเสริมสวย สำเร็จลุล่วงได้ด้วยความกรุณาแลความช่วยเหลือจากหลายๆ ท่าน ข้าพเจ้าขอขอพระคุณทุกท่าน ที่มีส่วนร่วมในการพัฒนาโครงงานนี้

ขอขอบพระคุณอาจารย์ ดร ทศพร จูฉิม อาจารย์ที่ปรึกษาโครงงานที่ได้แนะนำทฤษฎีและแนวทางในแก้ปัญหาต่าง ๆ ที่เกิดขึ้นระหว่างการพัฒนาระบบ อีกครั้งยังคอยตรวจสอบความก้าวหน้าของการทำงานเป็นระยะ ๆ รวมทั้งสร้างกำลังใจให้ผู้พัฒนาอยู่เสมอ

ขอบพระคุณอาจารย์ประจำสาขาวิทยาการคอมพิวเตอร์ อาจารย์ประจำภาควิชาคณิตศาสตร์สถิติ และคอมพิวเตอร์ และอาจารย์ในคณะวิทยาศาสตร์ทุก ๆ ท่าน ที่คอยให้คำแนะนำ อบรมสั่งสอน และคอยช่วยเหลือข้าพเจ้าในการศึกษาตลอดมาขอบคุณเจ้าหน้าที่และบุคลากรของคณะวิทยาศาสตร์ ที่ได้อำนวยความสะดวกทางด้านอุปกรณ์และเครื่องมือต่าง ๆ

ขอบพระคุณบิดา มารดา ที่เป็นกำลังใจ คอยให้ความรักและความห่วงใยเสมอมา ตลอดจนคอยช่วยเหลือทุนทรัพย์ทางด้านการศึกษา และอุปกรณ์ในการพัฒนาโครงงาน

ขอบคุณเพื่อน ๆ สาขาวิทยาการคอมพิวเตอร์ชั้นปีที่ 4 ที่ได้คอยช่วยแก้ไขปัญหาและให้คำปรึกษาในการพัฒนาโครงงานครั้งนี้จนเสร็จสิ้น

นางสาวปิยพร อาภรศรี

17 มีนาคม 62

โครงงาน	:	ระบบจองคิวร้านเสริมสวย
โดย	:	นางสาวปิยพร อาภรศรี
อาจารย์ที่ปรึกษา	:	ดร.ทศพร จูฉิม
ระดับการศึกษา	:	วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ปีการศึกษา	:	2562

บทคัดย่อ

การเสริมสวยเป็นที่นิยมและเป็นธุรกิจที่ได้รับความนิยมจากลูกค้าเป็นจำนวนมากในปัจจุบัน โดยปกติเมื่อลูกค้าต้องการใช้บริการจะมาที่ร้านโดยไม่ได้นัดหมาย ซึ่งปัญหาคือร้านมีลูกค้าที่ใช้บริการอยู่ในขณะนั้นทำให้ต้องผู้ที่เข้ามาโดยไม่ได้นัดต้องรอคิวหรือถ้ามีลูกค้ากำลังรอรับบริการอยู่เป็นจำนวนมากอาจทำให้ต้องมาใช้บริการในวันอื่นแทน แม้ในบางกรณีที่ลูกค้าโทรมาสอบถามเพื่อทำการจองคิวล่วงหน้า แต่ช่างไม่สะดวกรับโทรศัพท์เนื่องจากกำลังให้บริการลูกค้าคนอื่นอยู่

ดังนั้นผู้พัฒนาจึงมีแนวคิดสร้างเว็บแอปพลิเคชันระบบการจองคิวร้านเสริมสวยขึ้น เพื่อช่วยจัดการปัญหาดังกล่าว ระบบนี้ถูกพัฒนาด้วย React framework , material ui , nodejs และ MySql โดยระบบสามารถให้ลูกค้าทำการจองคิว ดูคิวว่างของร้านเสริมสวย หาตำแหน่งของร้าน ดูข้อมูลทั่วไปของร้าน และเขียนรีวิวติชมได้ ในส่วนของเจ้าของร้าน สามารถเพิ่มข้อมูลทั่วไปของร้าน และจัดการการจองคิวของร้านเสริมสวยได้ ระบบการจองคิวรองรับการแสดงผลบนอุปกรณ์สมาร์ทโพนและเว็บบราวเซอร์

ระบบที่พัฒนาขึ้นจะช่วยอำนวยความสะดวกในการนัดหมายล่วงหน้า ลดการรอคิวของผู้ใช้บริการ และช่วยให้การจองคิวมีระเบียบมากขึ้น

คำสำคัญ: เว็บแอปพลิเคชัน , ระบบจองคิวร้านเสริมสวย

Topic	:	Beauty salon queue reservation system
Author	:	PIYAPHORN ARPHORNSRI
Advisor	:	TOSSAPORN JOOCHIM, Ph.D..
Degree	:	Bachelor of Science (Computer Science)
Academic Year	:	2019

Abstract

Beauty is popular and is a business that has been popular with many customers today. Usually when customers want to use the service, will come to the shop without an appointment. The problem is that the shop has customers who use the service at that time, causing those who enter without an appointment to wait in the queue. Or if there are many customers waiting to receive the service, may cause to use the service on another day instead Even in some cases where users call to inquire to reserve a queue in advance But the technician was not comfortable to answer the phone because he was serving other customers Therefore, the developer has the idea to create a salon queue reservation system. To help manage the said problem This system was developed with the React framework, material ui, nodejs and MySql. The system allows customers to reserve a queue. See the salon queue. Find a store location See general information of the store And can write a review In the part of the store owner Can add general information of the shop And can manage the beauty salon queue reservations The queue reservation system supports display on Smart Phon devices and web browsers. The developed system will help facilitate advance appointments. Reduce user waiting And helps to make queue reservations more organized.

Keywords: Web Application,Beauty salon queue reservation system

สารบัญ

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

บทที่ 1

บทนำ

1.1 ที่มาและเหตุผล

เนื่องจากปัจจุบัน การดำเนินธุรกิจร้านเสริมสวยที่ให้บริการเสริมความงาม เช่น การทำผม ตัดผม ออกแบบทรงผม อบไอน้ำ เวลาผู้ใช้บริการมาใช้บริการโดยจะมาที่ร้านเลยโดยที่ไม่จองคิว พบว่าร้านที่มาใช้บริการมีลูกค้าเป็นจำนวนมาก อาจจะทำให้ต้องรอคิวนานหรือต้องเสียเวลามาใช้บริการในวันอื่นบางครั้งผู้ใช้บริการมีเบอร์ของร้านเสริมสวยก็จะโทรมา สอบถามคิวและจองคิว แต่ช่างติดทำผมให้ลูกค้าท่านอื่นก็ไม่สามารถรับโทรศัพท์ได้ผู้พัฒนาจึงมีแนวคิดว่าจะทำระบบการจอง คิวร้านเสริมสวยขึ้น เพื่อแก้ปัญหาการรอคิวนานและให้มีความทันสมัยตลอดจนสามารถรองรับการแสดงผลบนอุปกรณ์ สมาร์ทโฟนในปัจจุบัน ทำให้สามารถจองคิวหรือติดต่อสื่อสารในเรื่องของการจองคิวทำผมกับทางร้านได้สะดวกมากยิ่งขึ้น

แนวทางการแก้ปัญหา จัดทำการพัฒนาเป็นเว็บแอปพิเคชัน ระบบจองคิวร้านเสริมสวยที่ถูกพัฒนาขึ้นเป็นเว็บแอปพิเคชัน จะช่วยเพิ่มระเบียบในการจัดการจองคิวให้เป็นระบบ ลดขั้นตอนการดำเนินงานที่ซับซ้อน ลดระยะเวลาในการดำเนินงาน ลดความผิดพลาดที่จะเกิดขึ้นในขั้นตอนการดำเนินงาน และช่วยเพิ่มประสิทธิภาพในการทำงาน

1.2 วัตถุประสงค์

1. เพื่อออกแบบและพัฒนาเว็บแอปพลิเคชั่น จองคิวร้านเสริมสวย
2. เพื่อแก้ปัญหาการรอคิวร้านเสริมสวย

1.3 ขอบเขตของโครงงาน

1.3.1 เจ้าของร้าน

- สามารถลงทะเบียนเข้าสู่ระบบด้วย Email ได้
- สามารถดูคิวที่ผู้ใช้บริการได้ทำการจองคิวไว้
- สามารถ post ภาพผลงานทั้งหมดของร้านได้

- สามารถเพิ่ม แก้ไข และลบรายการให้บริการประจำร้านได้
- สามารถเพิ่ม แก้ไข ข้อมูลร้านได้
- สามารถเพิ่ม แก้ไข ข้อมูลตำแหน่งร้านได้

1.3.2 ช่างประจำร้าน

- ลงทะเบียนใช้ web ด้วย Email ได้
- สามารถดูตารางการทำงานของตนเองได้
- สามารถ post ภาพผลงานของตัวเองได้
- สามารถแก้ไขข้อมูลส่วนตัวได้

1.3.3 ผู้ใช้บริการ

- สามารถลงทะเบียนเข้าสู่ระบบด้วย Email ได้
- สามารถค้นหาร้านเสริมสวยได้
- สามารถจองคิวของร้านเสริมสวยได้
- สามารถดูคิวว่างของร้านเสริมสวยได้
- สามารถดูข้อมูลต่างๆของร้านเสริมสวยได้
- สามารถดูตำแหน่งของทางร้านได้
- สามารถดูผลงานของร้านได้
- สามารถเขียนรีวิว ติชม ได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ช่วยลดขั้นตอนการจองคิวร้านเสริมสวย
2. ช่วยลดปัญหาในการใช้บริการที่เกิดจากการรอคิวนาน
3. ช่วยให้การจองคิวมีระเบียบมากขึ้น

1.5 เครื่องมือที่ใช้ในการพัฒนา (Development tools)

1.5.1 ฮาร์ดแวร์

1. เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal computer) เพื่อใช้ในการพัฒนาโมบายแอปพลิ-เคชัน โดยมีคุณสมบัติอย่างน้อยดังนี้
    - ทำงานบนระบบปฏิบัติการ Elementary OS พื้นฐานการทำงานบน Windows 10

- หน่วยประมวลผลกลาง AMD Rezen(™) 5 3500U
- หน่วยประมวลผลกราฟฟิก AMD Radeon(™) Vega(8) Mobile Graphics
- หน่วยความจำหลักอย่างน้อย 8 กิกะไบต์ (Gigabyte, GB)
- หน่วยความจำสำรองอย่างน้อย 256 กิกะไบต์ (Gigabyte, GB)

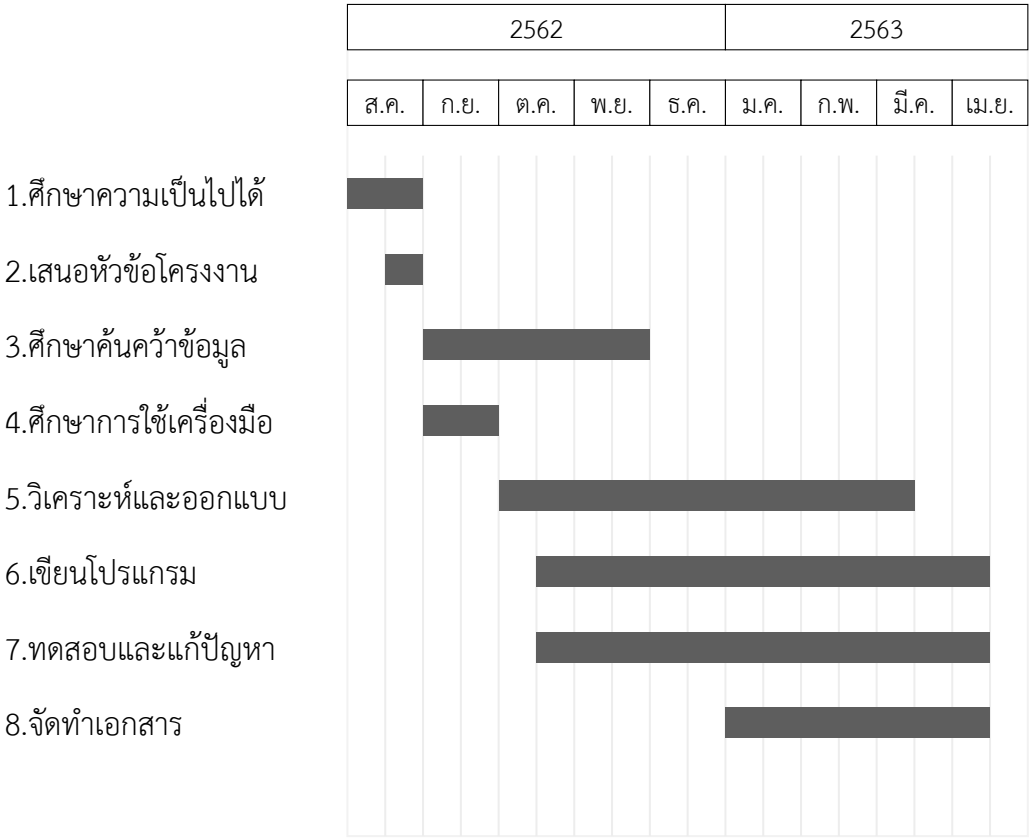
1.5.2 ซอฟต์แวร์ (Software)

1. ReactJS เป็น JavaScript Framework โดยมีชุดคำสั่งและไลบารี่ (Library) ให้ใช้งานมากมาย
2. Node.js คือ Cross Platform Runtime Environment หรือเรียกอีกอย่างว่า Backend Framework ใช้สำหรับเป็นเว็บเซิฟเวอร์ (Web Server) ซึ่งเขียนด้วยภาษา JavaScript
3. JavaScript เป็น ภาษาที่ใช้ในการพัฒนาเว็บ Environment)
4. Xampp เป็นโปรแกรม Apache web server ไว้จำลอง web server เพื่อทดสอบระบบระหว่างพัฒนา
5. MySQL เป็น โปรแกรมระบบจัดการฐานข้อมูล
6. Google Map API เทคโนโลยีที่ใช้ในการใช้งานแผนที่
7. Visual Studio Code เครื่องมือสำหรับพัฒนาเว็บแอปพลิเคชัน

1.5.3 แผนการดำเนินการ

ในการสร้างระบบแนะนำสถานที่ท่องเที่ยวในจังหวัดอุบลราชธานี ผู้พัฒนาได้แบ่งขั้นตอนการดำเนินงานไว้ด้วยกัน 8 ขั้นตอน ดังตารางที่ 1.1

ตารางที่ 1.1: ขั้นตอนการดำเนินงาน



บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงรายละเอียดเกี่ยวกับทฤษฎีและงานวิจัยที่เกี่ยวของกับการพัฒนาระบบจองคิวร้านเสริมสวย โดยแบ่งเนื้อหาออกเป็น 2 ส่วน ได้แก่ ส่วนที่หนึ่งเป็นเนื้อหาพื้นฐานเกี่ยวกับทฤษฎีการเขียนโปรแกรมและเทคโนโลยีที่นำมาใช้ในการพัฒนาในหัวข้อที่ 2.1 - 2.6 ได้แก่ ความรู้พื้นฐานเกี่ยวกับ React Node.js JavaScript MySQL Visual studio code Google maps API และในส่วนที่สองเป็นเนื้อหาเกี่ยวกับเว็ปแอพพลิเคชันที่เกี่ยวข้องกับโครงงานนี้เว็บแอพพลิเคชัน Gowabi

2.1 ความรู้พื้นฐานเกี่ยวกับ React

React [1] เป็น JavaScript Library ที่ถูกสร้างโดย Facebook ซึ่ง React ทำหน้าที่เป็นเพียง User Interface (UI) ที่สร้างมาจากพื้นฐานแนวความคิดแบบ Model View Controller (MVC)

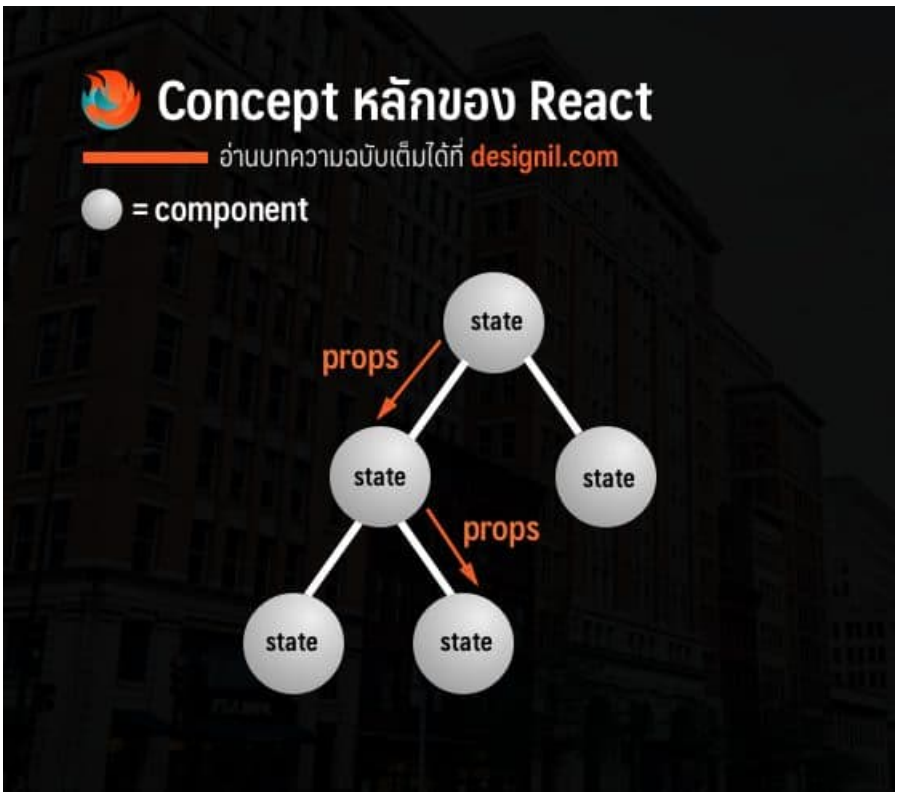
React ทำหน้าที่เฉพาะส่วน View (จาก Model View Controller) เหมาะกับงาน Web Front-End ที่สามารถแบ่งออกเป็น Web Component ย่อยๆ โดยหลักการวิเคราะห์ควรแยกให้ย่อยที่สุดเท่าที่จะทำได้ ซึ่งสามารถแบ่ง Component ออกเป็น 2 รูปแบบ คือ

- Container สำหรับบรรจุ Component หลักย่อยอื่นๆ ซึ่งไม่ควรมีการเก็บค่าใดๆ (สามารถทำหน้าที่เป็นตัวกลางในการส่งผ่านค่าได้) เน้นไปที่การจัด Layout
- Web Component คือ ส่วนที่ต้อง interact กับผู้ใช้จริงๆ เช่น ช่องกรอกข้อมูล ลิสต์แสดงข้อมูล ลาเบล (Label) และ ปุ่ม เป็นต้น ซึ่งอาจมีการเก็บค่าบางค่าเอาไว้ที่ สเตท (State) เพื่อนำมาแสดงผล

การเขียน React จำเป็นต้องมีความรู้ใน 3 ประเด็น ได้แก่

- Component – ส่วนประกอบต่างๆ ในเว็บ จะถูกมองเป็น Component
- State – ข้อมูลที่อยู่ใน Component แต่ละชิ้น เรียกว่า State
- Props – ข้อมูลที่ถูกส่งต่อจาก Component ชั้นบนลงไปชั้นล่าง ซึ่งเรียกว่า Props (Properties)

อธิบายประเด็นหลักทั้ง 3 ประเด็น ดังรูปที่ 2.1



รูปที่ 2.1: concept หลักของ React

ที่มา: https://www.designil.com/wp-content/uploads/2017/07/react-concept-designil.jpg

2.1.1 React Life Cycle

การเขียน render() ฟังก์ชันใน component [2] นั้น ควรจะเขียนในแบบ pure function ซึ่งจะไม่มีการเปลี่ยน state และสร้าง side effect ต่อภายนอกทั้งสิ้น อย่างเช่น การ call extenal service แบบ Ajax request, Firebase calling เป็นต้น เพราะหน้าที่ของ render() มีแค่การ render UI เท่านั้น หากไม่สามารถทำสิ่งดังกล่าวภายใน render() แล้ว กิจกรรมเหล่านั้นจึงสามารถทำได้ที่ life cycle ของ React

ตลอดช่วงวงจรชีวิต [3] สามารถควบคุมเหตุการณ์ต่างๆ ที่เกิดขึ้นในการแสดงผล UI การอัพเดทข้อมูล และการ re-rendering จนกระทั้งข้อมูลนั้นหายไป โดยที่ React ได้มีการเตรียมฟังก์ชันต่างๆ ไว้ สามารถอธิบายการทำงานของฟังก์ชันได้ ดังนี้

- componentWillMount() : คุณสมบัติของ componentWillMount ไม่มีอะไรเกี่ยวกับการใช้งาน component เพราะยังไม่มีการ mount อะไรขึ้นมา โดยมีหน้าที่ คือ การกำหนดค่าเริ่มต้นสำหรับการใช้งาน
- componentDidMount() : เกิดขึ้นเมื่อทำการ Mount เรียบร้อย พร้อมที่จะใช้งาน โดยปกติจะใช้ในการกำหนดค่าทุกอย่างที่ต้องใช้ DOM และรับข้อมูลที่ต้องการมาแสดงผล
- componentWillReceiveProps(nextProps) : เมื่อ Component ทำงาน จนกระทั้งมี pro-ps ใหม่เข้ามา เพื่อทำการเปลี่ยนแปลงข้อมูล componentWillReceive Props จะถูกเรียก โดยมี nextProps เป็นตัวแปรที่ถูกส่งเข้ามา
- shouldComponentUpdate(nextProps, nextState) : ถูกเรียกเมื่อ component มีการเปลี่ยนแปลงด้วย nextProps กับ nextState
- componentWillUpdate(nextProps, nextState) : ถูกเรียกก่อนที่จะ render หลังจากได้รับค่าใหม่ของ props หรือ state คุณสมบัติของคล้ายกับ componentWillReveiveProps
- componentDidUpdate(prevProps, prevState) : ถูกเรียกทันที่หลังจากเกิดการเปลี่ยน-แปลงของ component แต่จะไม่ถูกเรียกตอนครั้งแรกที่ render โดยที่ componentDidUpd-ate สามารถใช้งานได้เหมือน componentDidMount
- componentWillUnmount() : ถูกเรียกก่อนที่ component ทำการ unmount และ destroy โดยปกติแล้วจะใช้เพื่อทำการรีเซ็ต (reset) ค่าต่างๆ

2.2 ความรู้พื้นฐานเกี่ยวกับ Node.js

Node.js [4] เป็นภาษาที่ทำงานอยู่ในฝั่งเซิร์ฟเวอร์ (server) ซึ่ง syntax ที่ใช้ในการเขียนคือ JavaScript และเป็นภาษาที่ออกแบบมาให้ทำงานแบบ Event-Driven หรือทำงานเมื่อเกิดเหตุการณ์ตามที่กำหนดไว้ และการทำงานแบบ Asynchronous ซึ่งสามารถทำงานในลำดับต่อไปโดยที่ไม่ต้องรอให้งานก่อนหน้าเสร็จก่อนแล้วจึงทำงานขั้นต่อไป แต่ก็สามารถกำหนดให้ทำงานแบบ Synchronous ได้เช่นกัน โดยการกำหนด Callback เมื่องานแรกทำงานเสร็จแล้ว นอกจากนี้ Node.js นั้นจะใช้ Compiler จาก Google JavaScript Engine V8

ส่วนใหญ่จะนิยมใช้ node.js ในงานที่ทำเป็นเบื้องหลัง คือ งานที่ประมวลผลสั่งเซิร์ฟเวอร์ ซึ่งเป็นงานที่อาจจะต้อง interface กับผู้ใช้ หรือไม่ต้อง interface กับผู้ใช้ ตัวอย่างงานที่ต้อง interface กับผู้ใช้ เช่น การทำตัวเองเป็น http server ในการดึงหน้าเว็บมาแสดงผลให้กับ user

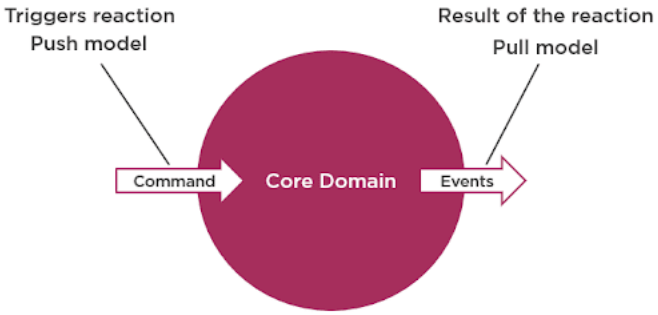
หรือว่า การเปิด socket เพื่อรับส่งข้อมูลกันระหว่างเซิร์ฟเวอร์กับผู้ใช้งาน เช่น ทำเป็นห้อง chat ทำเกม ทำระบบที่ป้อนข้อมูลเพื่อคำนวณผลลัพธ์ เป็นต้น ตัวอย่างงานที่ไม่ต้อง interface กับผู้ใช้ เช่น ทำ spider crawler เว็บ คือ การเปิดเว็บแล้วเก็บข้อมูลไปเรื่อยๆ หรือ โปรแกรมที่ รอรับค่าจาก streaming ต่างๆ เพื่อนำมาบันทึกไว้ ซึ่งการทำงานเหล่านี้ไม่จำเป็นต้อง interface กับผู้ใช้

node.js มีส่วนเสริมที่ชื่อว่า node package management (npm) ซึ่งเปรียบเหมือน google play ใน android หรือ app store ใน iOS ที่สามารถเอา package ที่คนอื่นเขียนเอาไว้แล้ว เพื่อแจกฟรี (free) มาต่อยอดเพื่อใช้ในงานของตนได้ โดยตัวอย่างที่ได้รับความนิยมจะเป็น underscore, async, request และ express เป็นต้น สำหรับการติดตั้ง ใช้คำสั่ง npm install ตามด้วยชื่อ package ที่ต้องการติดตั้ง [5]

node.js มีการทำงานเป็น Asynchronous คือ การทำงานบางอย่างไม่ต้องรอให้บรรทัดนั้นทำงานเสร็จ เช่น ส่งคำสั่งไป query ข้อมูลจากฐานข้อมูล แล้วสามารถข้ามไปทำงานบรรทัดต่อไปโดยไม่ต้องรอผลจากฐานข้อมูล เมื่อการทำงานนั้นทำงานเสร็จจึงค่อยรอผลลัพธ์กลับมา ดังนั้นปัญหาจะเกิดทันที ถ้าการทำงานต่อไปนำผลลัพธ์จากคำสั่งก่อนหน้านั้นมาใช้ต่อ ซึ่งส่งผลให้เกิดการทำงานผิดพลาด เพราะผลลัพธ์ยังไม่ได้รับกลับมา

2.2.1 node.js ทำงานแบบ event driven

การทำงานของ node [6] เรียกว่าเป็นการขับเคลื่อนด้วย event ต่างๆ ที่เกิดขึ้น ทำให้สามารถข้ามจาก event หนึ่งที่เสร็จแล้วไปยัง event อื่นได้ด้วยการสั่งงานต่อเนื่องกันไป หรือการสั่งให้ event หลาย event เริ่มทำงานในเวลาใกล้เคียงกัน ประโยชน์ที่ได้จาก event driven คือ การสั่งให้รอรับ event นั้นไปตลอดการณ์ โดยไม่เปลืองทรัพยากร เช่น การเชื่อมต่อไปยัง streaming channel ที่หนึ่ง ซึ่งอาจเป็น text หรือข้อมูลบางอย่าง เช่น ปริมาณน้ำฝน เอาไว้ หากต้นทางของ streaming ยังไม่มีข้อมูลส่งมา จะไม่เกิด event ใดๆ และ node.js จะรออยู่ แต่หากต้นทาง streaming มีข้อมูลมา node.js จะทำงานเพื่อตอบสนองต่อ event ที่เกิดขึ้นนั้นทันที สามารถแสดงการทำงานดังกล่าวได้ ดังรูปที่ 2.2



รูปที่ 2.2: การทำงานแบบ event driven

ที่มา: http://meewebfree.com/u/i/nodejs/node_js_stage.png

จากรูปที่ 2.2 สามารถอธิบายการทำงานได้ดังนี้ การทำงานของ stage1 เมื่อเรียกใช้การทำงานของ stage2 แล้ว ไม่จำเป็นต้องรอให้ stage2 ทำงานเสร็จก่อน ซึ่ง stage1 สามารถเรียกการทำงานของ stage3 ได้เลยโดยไม่จำเป็นต้องรอการทำงานของ stage2

2.2.2 ข้อดีของ Node.js

- มีการทำงานแบบ Event-Driven และ Asynchronous
- เหมาะกับการทำ Web แบบ Real time
- ประหยัดทรัพยากร ในการทำงาน
- มีการประมวลผลที่รวดเร็ว

2.3 ความรู้พื้นฐานเกี่ยวกับ JavaScript

JavaScript [7] คือ ภาษาคอมพิวเตอร์สำหรับการเขียนโปรแกรมบนระบบอินเทอร์เน็ต (Internet) ที่กำลังได้รับความนิยมอย่างสูง JavaScript เป็น ภาษาสคริปต์ (script) เชิงวัตถุ ที่เรียกกันว่า ”สคริปต์” ซึ่งการใช้ JavaScript ในการสร้างและพัฒนาเว็บไซต์ (ใช้ร่วมกับ HTML) จะช่วยให้เว็บ-ไซต์ดูมีการเคลื่อนไหว สามารถตอบสนองผู้ใช้งานได้มากขึ้น โดยมีวิธีการทำงานในลักษณะ ”แปลความและดำเนินงานไปทีละคำสั่ง” (interpret) หรือเรียกว่า โปรแกรมเชิงวัตถุ (Object Oriented Programming) ที่มีเป้าหมายในการ ออกแบบและพัฒนาโปรแกรมในระบบอินเทอร์เน็ต สำหรับผู้เขียนด้วยภาษา HTML สามารถทำงานข้ามแพลตฟอร์มได้ โดยทำงานร่วมกับ ภาษา HTML และภาษา Java ได้ทั้งทางฝั่งไคลเอนต์ (client) และ ทางฝั่งเซิร์ฟเวอร์

JavaScript ถูกพัฒนาขึ้นโดย บริษัท เน็ตสเคปคอมมิวนิเคชันส์ (Netscape Communications Corporation) โดยใช้ชื่อว่า Live Script ออกมาพร้อมกับ Netscape Navigator 2.0 เพื่อใช้สร้างเว็บเพจ (Web page) โดยติดต่อกับเซิร์ฟเวอร์แบบ Live Wire ต่อมาเน็ตสเคปได้ร่วมมือกับ บริษัทซันไมโครซิสเต็มส์ (Sun Microsystems, Inc) ปรับปรุงระบบของเบราว์เซอร์ (Browser) เพื่อให้สามารถติดต่อใช้งานกับภาษาจาวา (Java) ได้ และได้ปรับปรุง LiveScript ใหม่เมื่อ ปี พ.ศ. 2538 แล้วตั้งชื่อใหม่ว่า JavaScript ซึ่ง JavaScript ทำให้การสร้างเว็บเพจมีลูกเล่นต่างๆ มากมาย และยังสามารถโต้ตอบกับผู้ใช้ได้อย่างทันที เช่น การใช้เมาส์คลิก หรือ การกรอกข้อความในฟอร์ม เป็นต้น

เนื่องจาก JavaScript ช่วยให้ผู้พัฒนา สามารถสร้างเว็บเพจได้ตรงกับความต้องการ และมีความน่าสนใจมากขึ้น ประกอบกับเป็นภาษาเปิด ที่ทุกคนสามารถนำไปใช้ได้ ดังนั้นจึงได้รับความนิยมเป็นอย่างสูง มีการใช้งานอย่างกว้างขวาง รวมทั้งได้ถูกกำหนดให้เป็นมาตรฐานโดย European Computer Manufacturer’s Association (ECMA) การทำงานของ JavaScript จะต้องมีการแปลความคำสั่ง ซึ่งขั้นตอนนี้จะถูกจัดการโดยเบราว์เซอร์ (เรียกว่าเป็น client-side script) ดังนั้น JavaScript จึงสามารถทำงานได้เฉพาะบนเบราว์เซอร์ที่สนับสนุน ซึ่งปัจจุบันเบราว์เซอร์เกือบทั้งหมดสามารถสนับสนุน JavaScript แล้ว อย่างไรก็ดี สิ่งที่ต้องระวังคือ JavaScript มีการพัฒนาเป็นเวอร์ชันใหม่ๆ ออกมาด้วย ดังนั้น ถ้านำโค้ด (code) ของเวอร์ชันใหม่ ไปรันบนเบราว์เซอร์รุ่นเก่าที่ยังไม่สนับสนุน อาจจะทำให้เกิด error ได้

2.3.1 ประโยชน์ของ JavaScript

- JavaScript ทำให้สามารถใช้เขียนโปรแกรมแบบง่ายได้ โดยไม่ต้องพึ่งภาษาอื่น
- JavaScript มีคำสั่งที่ตอบสนองกับผู้ใช้งาน เช่น เมื่อผู้ใช้คลิกที่ปุ่ม หรือ Checkbox สามารถสั่งให้เปิดหน้าใหม่ได้ ทำให้เว็บไซต์มีปฏิสัมพันธ์กับผู้ใช้งานมากขึ้น
- JavaScript สามารถเขียนหรือเปลี่ยนแปลง HTML Element ได้ นั่นคือสามารถเปลี่ยนแปลงรูปแบบการแสดงผลของเว็บไซต์ได้ หรือหน้าแสดงเนื้อหาสามารถซ่อนหรือแสดงเนื้อหาได้โดยง่าย
- JavaScript สามารถใช้ตรวจสอบข้อมูลได้ เช่น เมื่อผู้ใช้งานกรอกข้อมูลบางเว็บไซต์ เช่น Email เมื่อบันทึกข้อมูลผิดจะมีหน้าต่างฟ้องขึ้นมาว่ากรอกผิด หรือลืมบันทึกอะไรบางอย่าง เป็นต้น

- JavaScript สามารถใช้ในการตรวจสอบผู้ใช้ได้เช่น ตรวจสอบว่าผู้ใช้ ใช้ web browser อะไร
- JavaScript สร้าง Cookies (เก็บข้อมูลของผู้ใช้ในคอมพิวเตอร์ของผู้ใช้เอง) ได้

2.3.2 ข้อดีและข้อเสียของ Java JavaScript

JavaScript [8] ทำงานบนเว็บบราวเซอร์ (client-side script) จึงไม่มีข้อจำกัดว่าจะใช้เซิร์ฟเวอร์แบบไหนก็ตาม เพราะ JavaScript ทำงานเฉพาะในเครื่องของผู้ใช้งานเท่านั้น ซึ่งต่างกับภาษาสคริปต์อื่น เช่น PHP , ASP, JSP หรือ Perl ซึ่งต้องประมวลผลและทำงานที่เครื่องเซิร์ฟเวอร์ (server-side script) จึงจำเป็นต้องใช้บนเซิร์ฟเวอร์ ที่สนับสนุนภาษาเหล่านี้เท่านั้นจึงจะสามารถใช้งาน server-side script ได้ แต่อย่างไรก็ตาม จากลักษณะการทำงานที่กล่าวมาก็ทำให้ JavaScript มีข้อจำกัด กล่าวคือคือไม่สามารถรับและส่งข้อมูลต่างๆ กับเซิร์ฟเวอร์โดยตรง เช่น การอ่านไฟล์จากเซิร์ฟเวอร์ เพื่อนำมาแสดงบนเว็บเพจ หรือรับข้อมูลจากผู้ชม เพื่อนำไปเก็บบนเซิร์ฟเวอร์ เป็นต้น ดังนั้นงานลักษณะนี้ จึงยังคงต้องอาศัยภาษา server-side script อยู่ (ความจริงมี JavaScript ที่ทำงานบนเซิร์ฟเวอร์เช่นกัน ซึ่งต้องอาศัยเซิร์ฟเวอร์ที่สนับสนุนโดยเฉพาะเช่นกัน แต่ไม่เป็นที่นิยมนัก)

นักพัฒนาเว็บส่วนใหญ่จึงนิยมใช้ JavaScript ร่วมกับ ภาษา Server Script เพื่อทำการส่งข้อมูลระหว่าง เซิร์ฟเวอร์กับเครื่องของผู้ใช้งาน ซึ่งทำให้การแสดงผลของหน้าเว็บมีความสวยงามและราบรื่นมากยิ่งขึ้น

2.4 ความรู้พื้นฐานเกี่ยวกับ MySQL

MySQL [9] เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System : RDBMS) ตัวหนึ่ง ซึ่งเป็นที่นิยมกันมากในปัจจุบัน โดยเฉพาะอย่างยิ่งในโลกของอินเตอร์เน็ต สาเหตุเพราะว่า MySQL เป็นฟรีแวร์ ทางด้านของฐานข้อมูลที่มีประสิทธิภาพสูง เป็นทางเลือกใหม่จากผลิตภัณฑ์ระบบจัดการฐานข้อมูลในปัจจุบัน ที่มักจะเป็นการผูกขาดของผลิตภัณฑ์เพียงไม่กี่ตัว นักพัฒนาระบบฐานขอมูลที่เคยใช้ MySQL ต่างยอมรับในความสามารถความรวดเร็ว การรองรับจำนวนผู้ใช้ และขนาดของข้อมูลจำนวนมหาศาล ทั้งยังสนับสนุนการใช้งานบนระบบปฏิบัติการมากมาย ไม่ว่าจะเป็น Unix , OS/2 , Mac , OS หรือ Windows ก็ตาม นอกจากนี้ MySQL ยังสามารถใช้ งานร่วมกับ Web Development Platform ทั้งหลาย ไม่ว่าจะเป็น C,

C++, Java, Perl, PHP, Python, Tcl หรือ ASP ดังนั้น MySQL จึงได้รับความนิยมอย่างมากในปัจจุบัน และมีแนวโน้มสูงยิ่งขึ้นต่อไปในอนาคต

2.4.1 โครงสร้างของ MySQL

โครงสร้างภายในของ MySQL [10] คือ การออกแบบการทำงานในลักษณะของ Client Server นั่นเอง ซึ่งประกอบด้วย 2 ส่วน คือ ส่วนของผู้ให้บริการ (Server) และ ส่วนของผู้ใช้บริการ (Client) โดยในแต่ละส่วนจะมีโปรแกรมสำหรับการทำงานตามหน้าที่ของโปรแกรมนั้น ส่วนของผู้ให้บริการ (Server) จะเป็นส่วนที่ทำหน้าที่บริหารจัดการระบบฐานข้อมูล (MySQL Server) และเป็นที่จัดเก็บข้อมูลทั้งหมด ข้อมูลที่เก็บไว้นี้มีข้อมูลที่จำเป็นสำหรับการทำงานกับระบบฐานข้อมูล และข้อมูลที่เกิดจากการที่ผู้ใช้แต่ละคนสร้างขึ้นมา ส่วนของผู้ใช้บริการ (Client) คือ ส่วนที่ผู้ใช้ใช้งาน โดยโปรแกรมสำหรับใช้งานในส่วนนี้ได้แก่ MySQL , Client , Access . Web Development เป็นต้น

2.4.2 หลักการทำงานในลักษณะ Client Server

เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็น Client หรือ Server อาจจะอยู่บนเครื่องเดียวกันหรือแยกเครื่องกันก็ได้ทั้งนี้ขึ้นอยู่กับลักษณะการทำงาน หรือการกำหนดของผู้บริหารระบบตามปกติ ถ้าเป็นการทำงานลักษณะ Web-based มีการใช้ฐานข้อมูลขนาดไม่ใหญนัก ตัว MySQL Server และ Client มักจะมีอยู่บนเครื่องเดียวกัน โดยเครื่องคอมพิวเตอร์ดังกล่าวจะต้องมีทรัพยากรเพื่อการทำงาน เช่น เนื้อที่ฮาร์ดดิสก์ RAM มากพอสมควร แต่สำหรับการทำงานจริง (Real-world Application) มักจะแยก Client และ Server ออกเป็นคนละเครื่องกัน และสามารถรองรับงานได้ดีมากกว่า ดังนั้น ผู้บริหารระบบหรือผู้กำหนดนโยบายสำหรับการทำงานเครือข่ายจะต้อง คำนึงถึงเรื่องที่เกี่ยวข้องเหล่านี้ให้ดี เพื่อที่จะทำให้ระบบมีการทำงานรับการให้บริการแก่ผู้ใช้อย่างมีประสิทธิภาพ และข้อมูลมีความปลอดภัยมากที่สุด หลักการทำงานในลักษณะ Client Server มีดังนี้

- ที่ฝั่งของ Server จะมีโปรแกรมหรือระบบสำหรับจัดการฐานข้อมูลทำงานรออยู่ เพื่อเตรียมหรือรอ คอยการร้องขอการใช้บริการจาก Client
- เมื่อมีการรองขอการใช้บริการเข้ามา Server จะทำการตรวจสอบ เช่น อาจจะมีการให้ผู้ใช้บริการระบุชื่อและรหัสผ่าน และสำหรับ MySQL สามารถกำหนดได้ว่าจะอนุญาตหรือ

ปฏิเสธ Client ใดๆ ในระบบที่จะเข้าใช้บริการอีกด้วย

- ถ้าผ่านการตรวจสอบ Server ก็จะอนุมัติการให้บริการแก่ Client ที่รองขอการใช้ บริการนั้นๆ ต่อไปและถ้าในกรณีที่ไม่ได้ รับการอนุมัติServer ก็จะส่งข่าวสารความผิดพลาดแจ้งกลับไปที่ Client ที่รองขอการใช้ บริการนั้น

2.4.3 การทำงานของโปรแกรมของ MySQL

MySQL ถือเป็นระบบจัดการฐานข้อมูล (Database Management System : DBMS) ฐานข้อมูลมีลักษณะเป็นโครงสร้างของการเก็บรวบรวมข้อมูล การที่จะเพิ่มเติม เข้าถึงหรือประมวลผลข้อมูลที่เก็บในฐานข้อมูลจำเป็นจะต้องอาศัยระบบจัดการ ฐานข้อมูล ซึ่งจะทำหน้าที่เป็นตัวกลางในการจัดการกับข้อมูลในฐานข้อมูลทั้งสำหรับการ ใช้งานเฉพาะ และรองรับการทำงานของแอพพลิเคชั่น (Application) ที่ต้องการใช้งานข้อมูลในฐานข้อมูล เพื่อให้ได้รับความสะดวกในการจัดการกับข้อมูลจำนวนมาก MySQL ทำหน้าที่เป็นทั้งตัวฐานข้อมูลและระบบจัดการฐานข้อมูล ดังนี้

- MySQL เป็นระบบจัดการฐานข้อมูลแบบ relational ฐานข้อมูลแบบ relational จะทำการเก็บข้อมูลทั้งหมดในรูปแบบของตารางแทนการเก็บข้อมูลทั้งหมดลงในไฟล์ เพียงไฟล์เดียว ทำให้ทำงานได้รวดเร็วและมีความยืดหยุ่น นอกจากนั้น แต่ละตารางที่เก็บข้อมูลสามารถเชื่อมโยงเข้าหากันทำให้สามารถรวมหรือจัด กลุ่มข้อมูลได้ตามต้องการ โดยอาศัยภาษา SQL ที่เป็นส่วนหนึ่งของโปรแกรม MySQL ซึ่งเป็นภาษามาตรฐานในการเข้าถึงฐานข้อมูล
- MySQL แจกจ่ายให้ใช้งานแบบ Open Source นั่นคือ ผู้ใช้งาน MySQL ทุกคนสามารถใช้งานและปรับแต่งการทำงานได้ตามต้องการ สามารถดาวน์โหลดโปรแกรม MySQL ได้จากอินเทอร์เน็ตและนำมาใช้งานโดยไม่มีค่าใช้จ่าย

2.4.4 คุณสมบัติของ MySQL

1. สนับสนุน Cross-platform support
2. รองรับ Stored procedures
3. รองรับ Triggers และ Cursors
4. สนับสนุน Information schema
5. สนับสนุน SSL
6. รองรับการทำ Query caching

7. รองรับการทำ Sub-SELECTs
8. รอบรับการทำ Replication ทั้งแบบ Master-Master Replication และ Master-Slave Replication
9. รองรับ Unicode

2.5 ความรู้เกี่ยวกับ Visual Studio Code

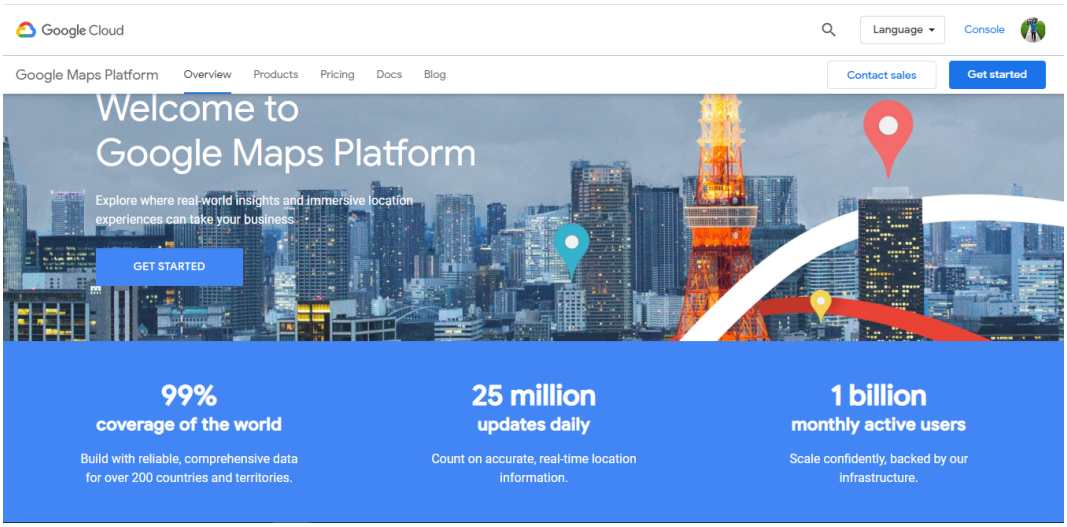
วิชวล สตูดิโอโค้ด (Visual Studio Code หรือ VSCode) [11] เป็นโปรแกรม Code Editor ที่ใช้ในการแก้ไขและปรับแต่งโค้ดถูกพัฒนาโดยค่ายไมโครซอฟท์(Microsoft) มีการพัฒนาออกมาในรูปแบบของ OpenSource จึงสามารถนำมาใช้งานได้แบบไม่ต้องเสียค่าใช้จ่าย Visual Studio Code เหมาะสำหรับนักพัฒนาโปรแกรมที่ต้องการใช้งานข้ามแพลตฟอร์ม (Cross-platform) โดยจะรองรับการใช้งานทั้งบนระบบปฏิบัติการ Windows,macOS และ Linux ซึ่งภาษาที่ Visual Studio Code รองรับการทำงานซึ่งมีมากกว่า 30 ภาษาโปรแกรม เช่น C++, C, CSS, Dockerfile, HTML, JavaScript, JSON, Less, Markdown, PHP, Python, Sass, TypeScript, Node.js และ Java เป็นต้น

2.6 ความรู้พื้นฐานเกี่ยวกับ Google Maps API

Google Maps API [12] เป็นชุด API ของ Google สำหรับพัฒนา web application และ mobile application (Android,ios) มีไว้สำหรับเรียกใช้แผนที่และชุด service ต่างๆ มากมายให้เรียกใช้ เช่น

- ชุดควบคุมแผนที่ (Map Control)
- การปรับแต่งแผนที่ (Styled Map)
- การนำทางจากจุดหนึ่งไปยังอีกจุดหนึ่ง (Directions Service)
- Street view
- การดึงข้อมูล POI (Point of Interest) คือข้อมูลสถานที่ต่างๆ ที่ Google รวบรวมไว้ เช่น โรงแรม ห้างสรรพสินค้า โรงเรียน สถานที่ราชการ เป็นต้น เพื่อให้ผู้ใช้งานนำมาใช้งานได้
- การแปลงที่อยู่เป็นพิกัด Latitude และ Longitude (Geocoding Service)
- แผนที่รองรับระบบ 3 มิติ ผู้ใช้งาน สามารถปรับมุมมองเป็นลักษณะ 3 มิติได้ ซึ่งระบบนี้มีให้บริการบางพื้นที่

การใช้งาน Google Maps API สามารถขอใช้งานโดยใช้บัญชี Gmail และเข้าไปที่เว็บ https://cloud.google.com/maps-platform/ ดังรูปที่ 2.3



รูปที่ 2.3: หน้าเว็บบริการของ Google Maps

ที่มา : https://cloud.google.com/maps-platform/

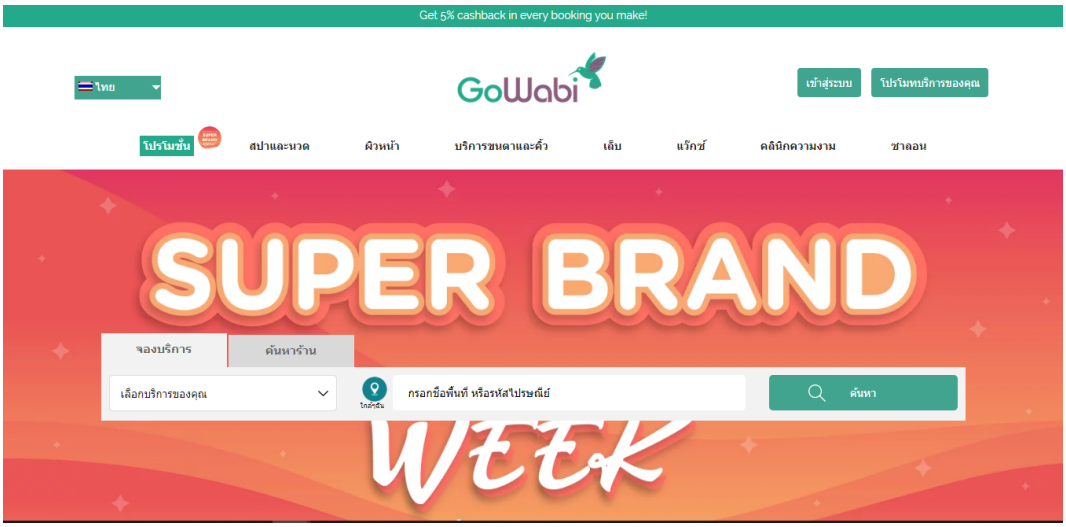
2.7 เอกสารและงานวิจัยที่เกี่ยวข้อง

2.7.1 เว็บแอพพลิเคชัน Gowabi

Gowabi[13] เป็นเว็บไซต์และแอปพลิเคชันที่ให้บริการเกี่ยวกับการค้นหาและจองคิวร้านเสริมสวย มีฟังก์ชันการทำงานพื้นฐานอันได้แก่ การค้นหา การจองคิว ดูข้อมูล เป็นต้น

2.7.2 ข้อแตกต่างระหว่างเว็บแอปพลิเคชั่น Gowabi กับเว็บของโครงงาน

เว็บแอปพลิเคชั่นของ Gowabi ยังไม่มีฟังก์ชันเลือกรายการ และช่างในการจองคิว ผู้พัฒนาจึงได้ทำฟังก์ชันเลือกรายหารและะช่างที่เหมาะสมตามรสนิยมของผู้ใช้เพิ่มลงในเว็บของโครงงาน



รูปที่ 2.4: หน้าแรกของเว็บไซต์ Gowabi

ที่มา :https://www.gowabi.com

บทที่ 3

การวิเคราะห์และออกแบบระบบ

การวิเคราะห์และออกแบบระบบก่อนดำเนินการจริงเป็นอีกหนึ่งขั้นตอนที่มีความสำคัญมาก เพราะการวิเคราะห์และออกแบบระบบนั้นเป็นการกระทำที่ทำให้ผู้พัฒนาเห็นรายละเอียดส่วนย่อยของงานทั้งหมด เพิ่มประสิทธิภาพในการวางแผน การทำงาน และยังช่วยลดปัญหาที่อาจจะเกิดขึ้นในระหว่างพัฒนา เพื่อให้ระบบมีความสมบูรณ์มากยิ่งขึ้น เนื่องจากการวิเคราะห์และออกแบบระบบนั้นจะช่วยให้ให้บริการ จัดการทรัพยากรได้อย่างคุ้มค่าและตรงตามความต้องการของระบบ

การวิเคราะห์และออกแบบระบบ การจองคิวร้านเสริมสวย ในบทนี้จะแบ่งออกเป็น 6 ขั้นตอนเพื่อให้เห็นการดำเนินงานอย่างมีระบบ ในหัวข้อแรกจะนำเสนอภาพรวมของระบบ ก่อนจะนำเสนอเอกสารแสดงความต้องการของระบบซึ่งจะทำให้เห็นที่มาของเพจต่าง ๆ ในขั้นตอนของการออกแบบในหัวข้อที่สาม ส่วนหัวข้อที่เหลือจะแสดงแผนภาพการการทำงานของระบบโดยใช้ UML diagram ซึ่งประกอบไปด้วย Use Case, Class และ Sequence Diagram เพื่อแสดงรายละเอียดของระบบก่อนนำไปเขียนคำสั่งด้วยภาษาโปรแกรมในบทต่อไป

3.1 โครงสร้างภาพรวมของระบบ (System Architecture) เป็นการออกแบบภาพรวมและเทคโนโลยีของระบบ

3.2 System Requirements คือ ความต้องการหรือสิ่งที่ระบบควรจะทำ หรือหน้าที่หลักของระบบที่จะต้องทำ

3.3 User Interface Design เป็นการออกแบบส่วนต่อประสานกับผู้ใช้

3.4 Use Case Diagram เป็นแผนภาพที่ใช้แสดงให้ทราบว่าระบบทำงานหรือมีหน้าที่ใดบ้าง

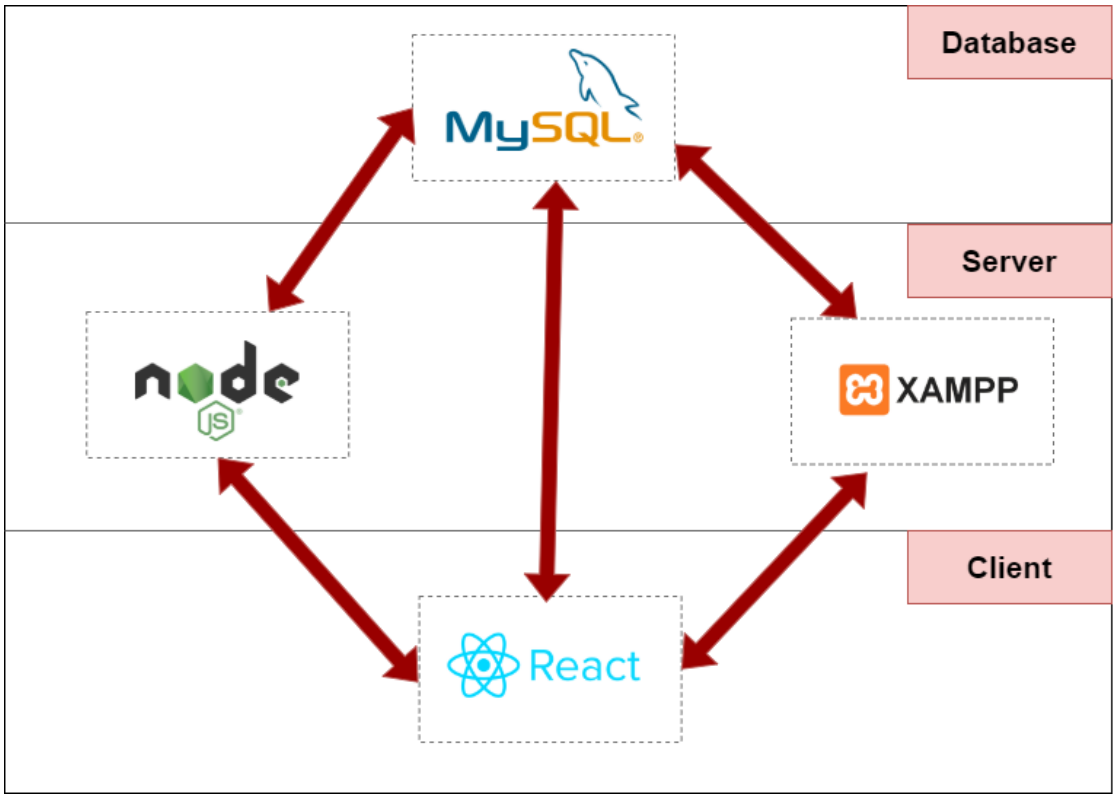
3.5 Class Diagram เป็นแผนภาพที่ใช้แสดง Class และความสัมพันธ์ระหว่าง Class

3.6 Sequence Diagram เป็นแผนภาพที่ใช้แสดงให้เห็นถึงการตอบโต้ข้อมูลระหว่างคลาส เรียงตามลำดับของเวลาที่เกิดเหตุการณ์จากน้อยไปมาก

3.1 โครงสร้างภาพรวมของระบบ

ความหมายของ System Architecture [14] หมายถึง กรอบโครงสร้างของระบบที่อธิบายความสัมพันธ์ขององค์ประกอบต่าง ๆ ไปจนถึงขั้นการเชื่อมต่อกันของระบบย่อยต่าง ๆ โดยจัดกลุ่มองค์ประกอบไว้ในหลาย ๆ ลักษณะเพื่อให้ผู้เกี่ยวข้อง (Stakeholder) จากพื้นฐานสาขาอาชีพที่แตกต่าง กันสามารถทำความเข้าใจได้ง่าย เช่น การจัดแบ่งองค์ประกอบตามลักษณะการทำงานของระบบ (functional components) เป็นต้น

การออกแบบ System architecture แสดงภาพรวมและเทคโนโลยีของระบบกองทุนเงินให้กู้ยืมเพื่อการศึกษา คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี มีรายละเอียดดังรูปที่ 3.1



รูปที่ 3.1: System architecture ระบบการจองคิวร้านเสริมสวย

จากรูปที่ 3.1 สามารถอธิบายโครงสร้างและเทคโนโลยีของระบบโดยแบ่งเป็น 3 ส่วนหลักดังนี้

1. Database ระบบใช้บริการฐานข้อมูลแบบ MySQL ของไฟร์เบสชื่อ Relational Database Management System : RDBMS

2. Server กระบวนการทำงานในส่วนของเซิฟเวอร์ (server) แบ่งเป็น 2 ส่วนได้แก่
    - ชุดเผยแพร่สำหรับการพัฒนาเว็บไซต์ซึ่งในที่นี้ใช้ Node.jsและExpress ในการพัฒนา
    - ชุดบริการ phpmyadmin Api ใช้สำหรับการทำงานกับบริการต่าง ๆ ของ MySLQ
3. Client เว็บแอปพลิเคชันการจองคิวร้านเสริมสวย มีวัตถุประสงค์เพื่อรองรับการทำงานของผู้ใช้บนเว็บบราวเซอร์พัฒนาด้วย React

3.2 System Requirements

3.2.1 Functional Requirements

ระบบการจองคิวร้านเสริมสวย แบ่งความสามารถของระบบตามประเภทของผู้ใช้งานดังนี้

1. เจ้าของร้าน
    - สามารถลงทะเบียนเข้าสู่ระบบด้วย Email ได้
    - สามารถดูคิวที่ผู้ใช้บริการได้ทำการจองคิวไว้
    - สามารถจัดการคิวได้
    - สามารถ post ภาพผลงานทั้งหมดของร้านได้
    - สามารถเพิ่ม แก้ไข และลบรายการให้บริการประจำร้านได้
    - สามารถเพิ่ม แก้ไข ข้อมูลร้านได้
    - สามารถเพิ่ม แก้ไข ข้อมูลตำแหน่งร้านได้
2. ช่างประจำร้าน
    - ลงทะเบียนใช้ web ด้วย Email ได้
    - สามารถดูตารางการทำงานของตนเองได้
    - สามารถ post ภาพผลงานของตัวเองได้
    - สามารถแก้ไขข้อมูลส่วนตัวได้
3. ผู้ใช้บริการ
    - สามารถลงทะเบียนเข้าสู่ระบบด้วย Email ได้
    - สามารถค้นหาร้านเสริมสวยได้
    - สามารถจองคิวของร้านเสริมสวยได้
    - สามารถดูคิวว่างของร้านเสริมสวยได้

- สามารถดูข้อมูลต่างๆของร้านเสริมสวยได้
- สามารถดูตำแหน่งของทางร้านได้
- สามารถดูผลงานของร้านได้
- สามารถเขียนรีวิว ติชม ได้

3.2.2 Non-functional Requirements

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

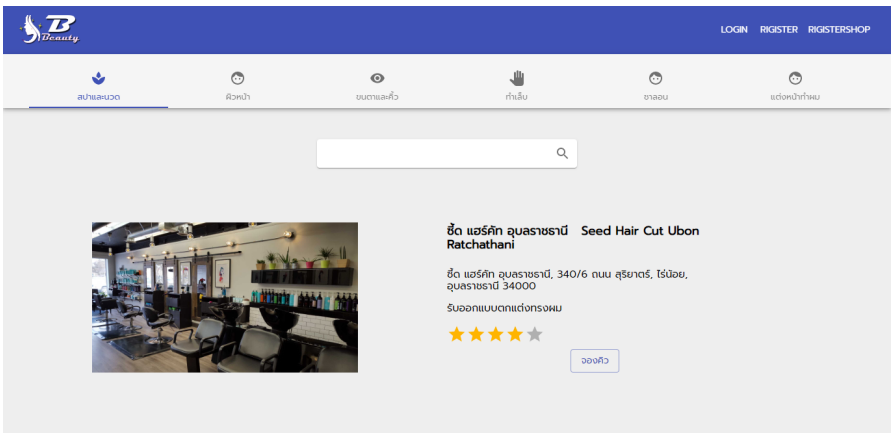
3.3 User Interface Design

User Interface Design ของระบบการจองคิวร้านเสริมสวย แบ่งเป็น 3 ส่วน มีดังนี้

1. ผู้ใช้บริการ

   User Interface ผู้ใช้บริการ จะออกแบบให้สะดวกต่อการใช้งาน

   - การออกแบบหน้าจอหลัก

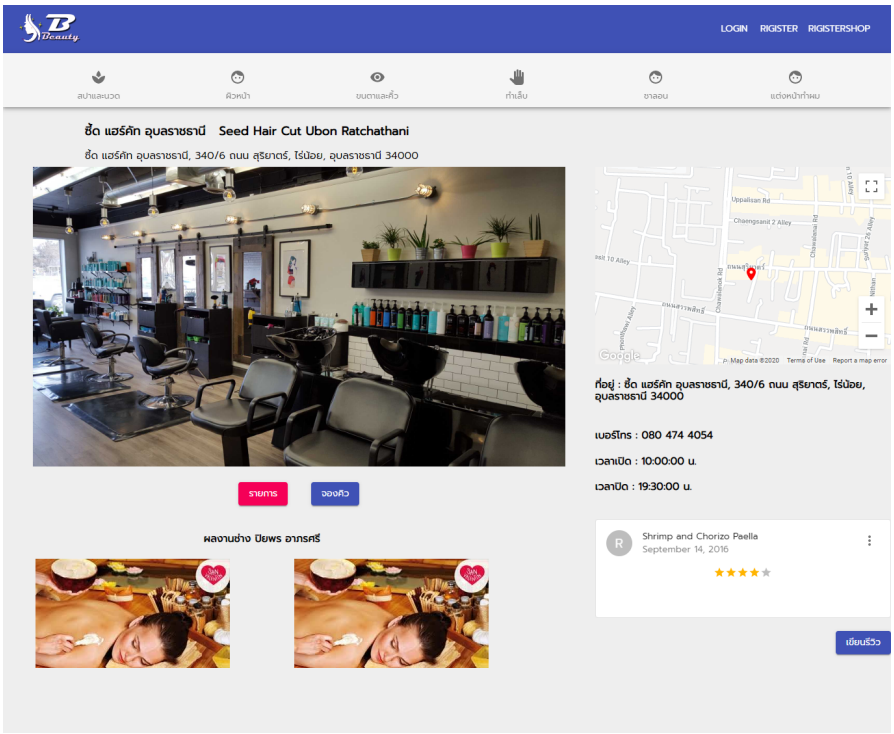


รูปที่ 3.2: หน้าจอหลัก

จากภาพที่ 3.2 แสดงหน้าจอหลักบนเว็บแอปพลิเคชัน เพื่ออำนวยความสะดวกต่อผู้ใช้งาน ในหน้าหลักได้รวบรวมข้อมูลสรุปของร้านและเมนูเข้าถึงด่วนซึ่งแบ่งเป็น 3

ส่วนหลักได้แก่ เมนูจองคิว รายละเอียดร้านและข้อมูลรีวิว

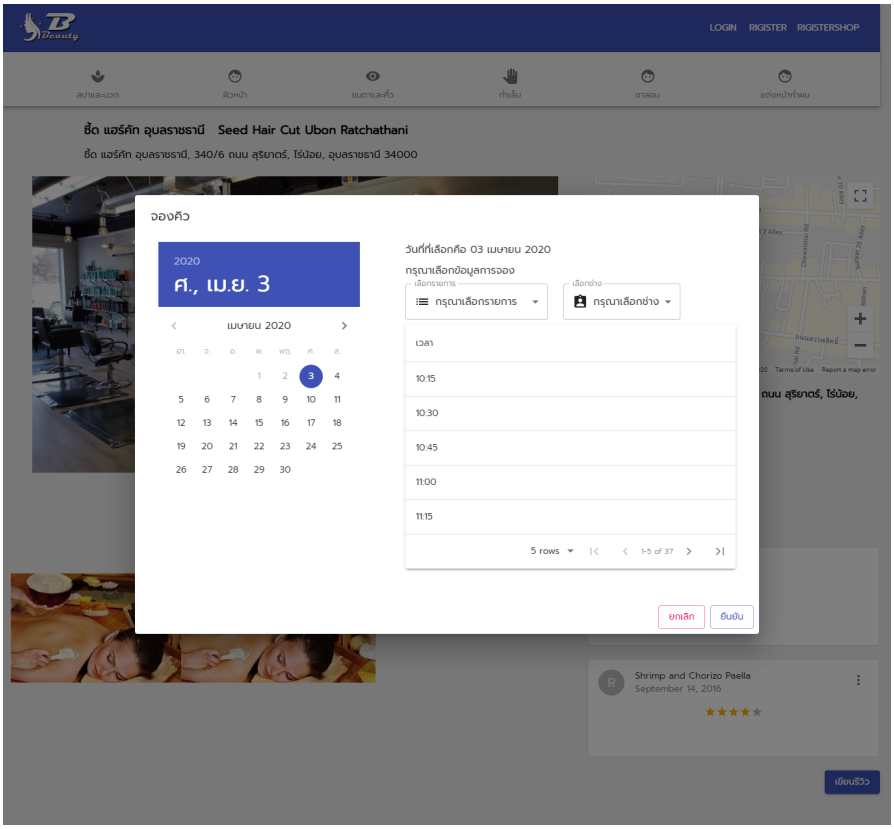
- การออกแบบหน้าจอข้อมูลร้าน



รูปที่ 3.3: หน้าจอข้อมูลร้าน

จากภาพที่ 3.3 แสดงหน้าจอข้อมูลรายละเอียดภายในร้าน

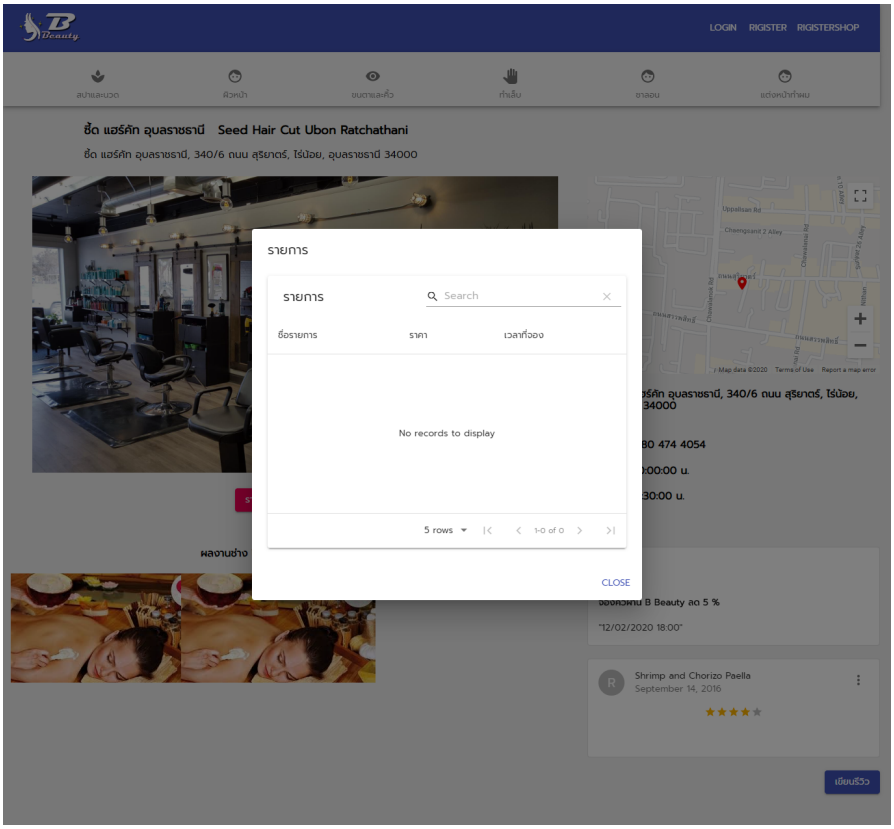
- การออกแบบหน้าจอการจองคิว



รูปที่ 3.4: หน้าจอการจองคิว

จากภาพที่ 3.4 แสดงหน้าจอการจองคิวร้านเสริมสวย ทั้งนี้ผู้ที่มีสิทธิ์ในการจองคิวมีเพียงผู้ใช้งานเท่านั้น

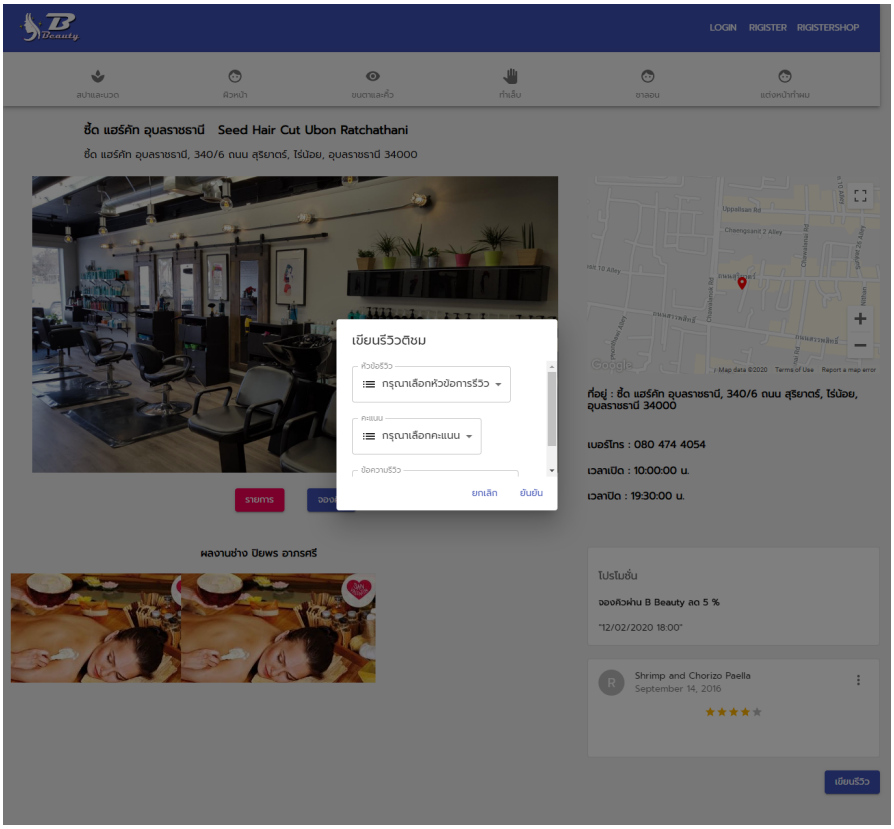
- การออกแบบหน้าจอดูรายการ



รูปที่ 3.5: หน้าจอดูรายการ

จากภาพที่ 3.5 แสดงหน้าจอรายการที่ให้บริการประจำร้าน

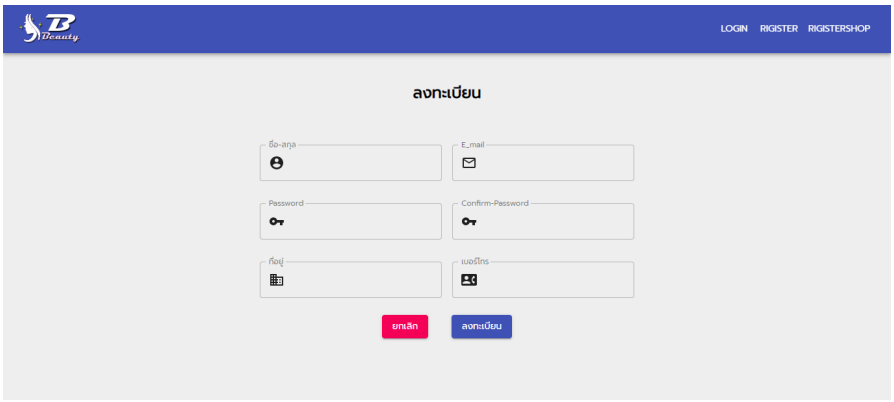
- การออกแบบหน้าจอเขียนรีวิว



รูปที่ 3.6: หน้าจอเขียนรีวิว

จากภาพที่ 3.6 แสดงหน้าจอการเขียนรีวิว ทั้งนี้ผู้ที่มีสิทธิ์ในการอัพโหลดเอกสารมีเพียงผู้ใช้งานเท่านั้น

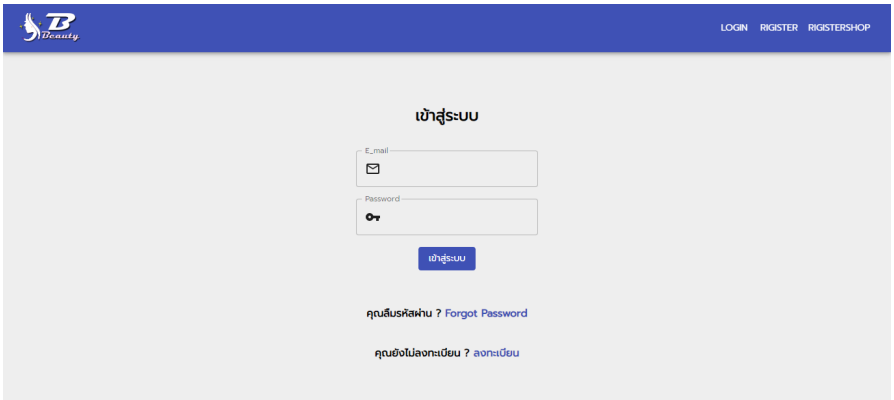
- การออกแบบหน้าจอลงทะเบียน



รูปที่ 3.7: หน้าจอลงทะเบียน

จากภาพที่ 3.7 แสดงหน้าจอการลงทะเบียนของผู้ใช้บริการ เพื่อใช้ในการเข้าสู่ระบบ

- การออกแบบหน้าจอเข้าสู่ระบบ



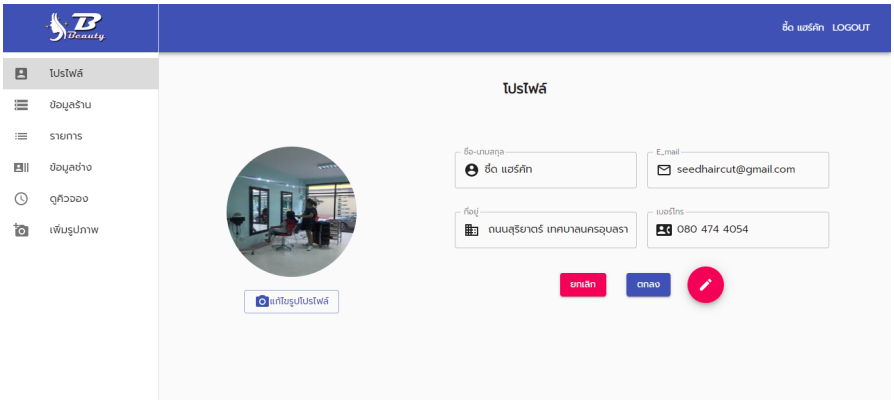
รูปที่ 3.8: หน้าจอเข้าสู่ระบบ

จากภาพที่ 3.8 แสดงหน้าจอการเข้าสู่ระบบของผู้ใช้โดยผู้ใช้จำเป็นต้องกรอกข้อมูลอีเมลและรหัสผ่านเพื่อเข้าใช้งานระบบ

2. เจ้าของร้าน

User Interface เจ้าของร้าน จะออกแบบให้สะดวกต่อการใช้งาน

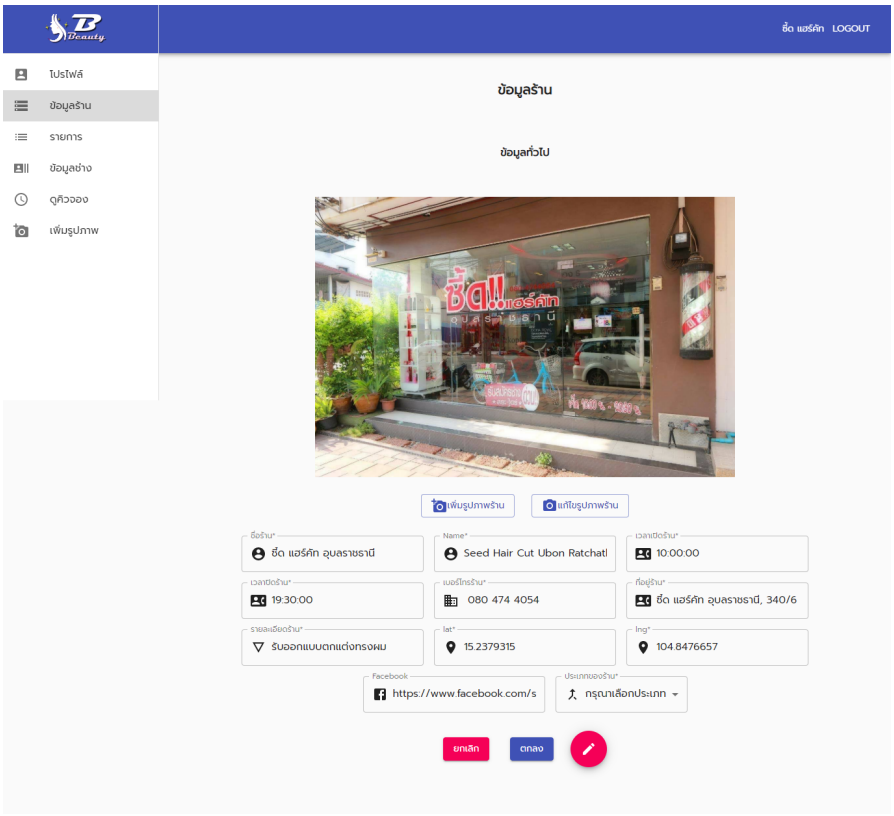
- การออกแบบหน้าจอโปรไฟล์ร้าน



รูปที่ 3.9: หน้าจอโปรไฟล์ร้าน

จากภาพที่ 3.9 แสดงหน้าจอโปรไฟล์บนเว็บแอปพลิเคชัน เพื่อดูข้อมูลของเจ้าของร้าน

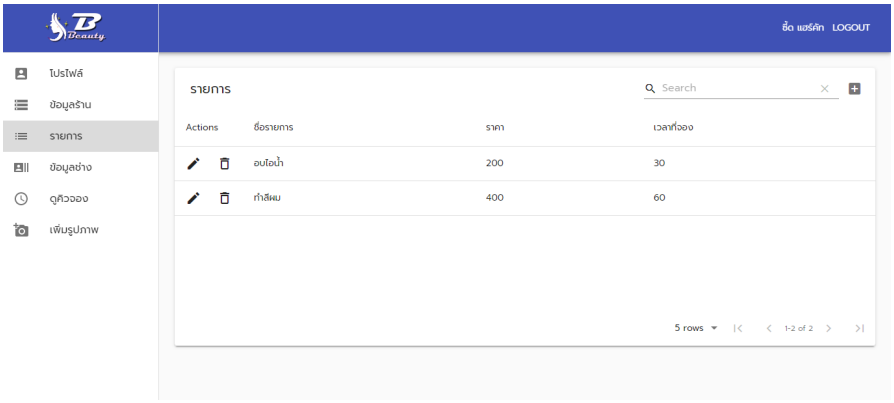
- การออกแบบหน้าจอเพิ่มข้อมูลร้าน



รูปที่ 3.10: หน้าจอเพิ่มข้อมูลร้าน

จากภาพที่ 3.10 แสดงหน้าจอเพิ่มข้อมูลร้าน เพื่อให้เจ้าของร้านเพิ่มข้อมูลร้าน

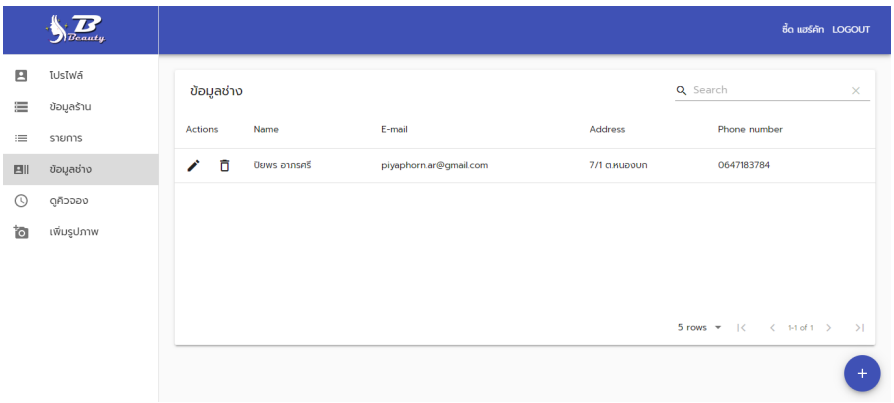
- การออกแบบหน้าจอเพิ่มรายการ



รูปที่ 3.11: หน้าจอเพิ่มรายการ

จากภาพที่ 3.11 แสดงหน้าจอเพิ่มข้อมูลรายการของร้าน เพื่อให้เจ้าของร้านเพิ่มข้อมูลรายการที่ให้บริการประจำร้าน

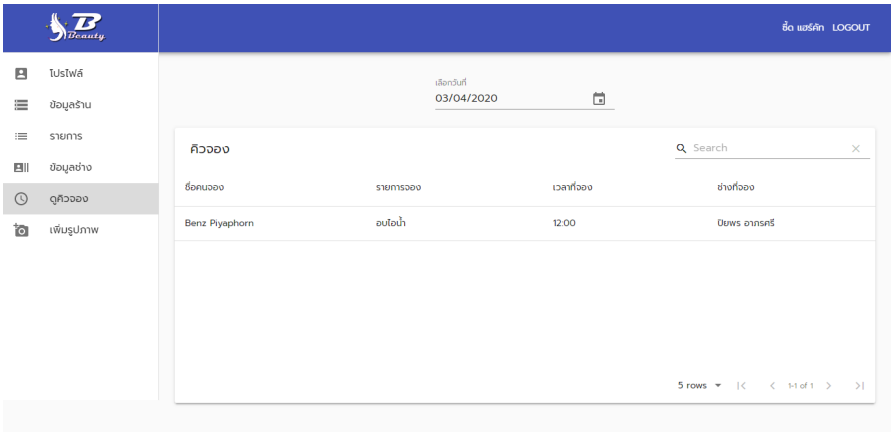
- การออกแบบหน้าจอเพิ่มข้อมูลช่าง



รูปที่ 3.12: หน้าจอเพิ่มข้อมูลช่าง

จากภาพที่ 3.12 แสดงหน้าจอข้อมูลช่าง เพื่อเพิ่มและอสดงข้อมูลช่างประจำร้าน

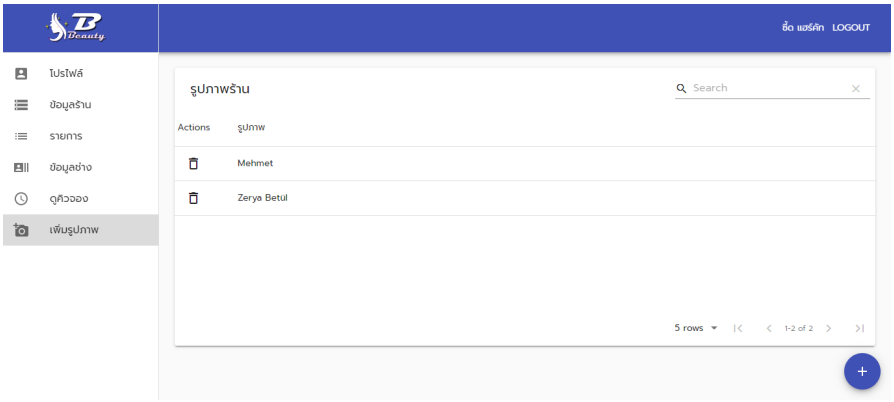
- การออกแบบหน้าจอดูการจองคิว



รูปที่ 3.13: หน้าจอดูการจองคิว

จากภาพที่ 3.13 แสดงหน้าจอการจองคิว เพื่อให้เจ้าของร้านดูข้อมูลการจองคิวจากผู้ใช้งาน

- การออกแบบหน้าจอเพิ่มรูปภาพ



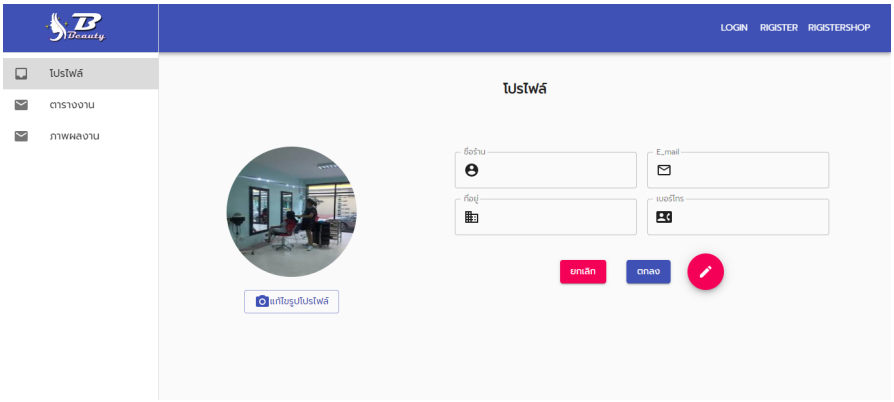
รูปที่ 3.14: หน้าจอเพิ่มรูปภาพ

จากภาพที่ 3.14 แสดงหน้าจอเพิ่มรูปภาพ เพื่ออัปโหลดรูปภาพเฉพาะเจ้าของร้าน

3. เจ้าของร้าน

User Interface เจ้าของร้าน จะออกแบบให้สะดวกต่อการใช้งาน

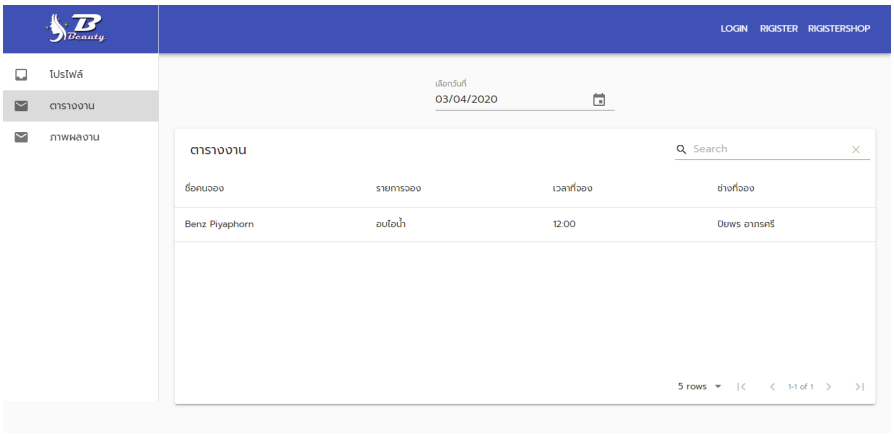
- การออกแบบหน้าจอโปรไฟล์ช่าง



รูปที่ 3.15: หน้าจอโปรไฟล์ช่าง

จากภาพที่ 3.15 แสดงหน้าจอโปรไฟล์บนเว็บแอปพลิเคชัน เพื่อดูข้อมูลของช่าง

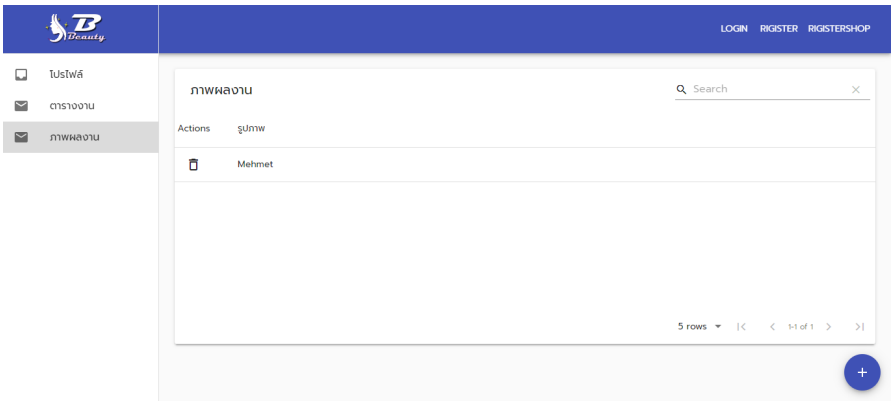
- การออกแบบหน้าจอดูตารางงาน



รูปที่ 3.16: หน้าจอดดูตารางงาน

จากภาพที่ 3.16 แสดงหน้าจอตารางงาน เฉพาะช่าง

- การออกแบบหน้าจอเพิ่มรูปภาพผลงานช่าง



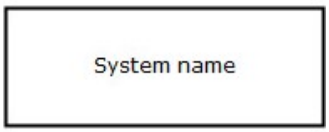
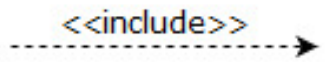
รูปที่ 3.17: หน้าจอเพิ่มรูปภาพผลงานช่าง

จากภาพที่ 3.17 แสดงหน้าจอเพิ่มรูปภาพ เพื่ออัพโหลดรูปภาพผลงานเฉพาะช่าง

3.4 Use Case Diagram

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

ตารางที่ 3.1: สัญลักษณ์ของ Use case Diagram

สัญลักษณ์	การใช้งาน
Use case	Use case คือส่วนย่อยของระบบงาน แทนด้วยวงรีและชื่อของ Use case ภายในวงรี
	Actor คือบุคคลหรือระบบงานอื่นที่ใช้งานระบบหรือได้รับประโยชน์จากระบบซึ่งอยู่ภายนอกระบบ แทนด้วยรูปคนและมีชื่อบทบาทการใช้งานระบบ
	เส้นตรงที่แสดงถึงการใช้งาน Use case ของผู้กระทำ
	กรอบ สี่เหลี่ยม แสดง ถึง ขอบเขต ของ ระบบ โดย แสดง ชื่อ ระบบ ภายในหรือด้านบนกรอกสี่เหลี่ยม Use case อยู่ภายในกรอบสี่เหลี่ยม และ actor อยู่ภายนอกกรอบสี่เหลี่ยม
	ความสัมพันธ์แบบ «includes» แสดงว่า Use case หนึ่งดำเนินการตามขั้นตอนของ Use case อื่น 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 3.18: Use Case Diagram ของระบบ จองคิวร้านเสริมสวย

ตารางที่ 3.2: อธิบาย Use Case หน้าที่ของระบบ ในภาพที่ 3.18

Use Case	คำอธิบาย
ดูข้อมูลต่างๆของร้านเสริมสวย	ผู้ใช้บริการสามารถดูข้อมูลได้โดยไม่จำเป็นต้องทำการเข้าสู่ระบบก่อน
ดูตำแหน่งของร้านเสริมสวย	ผู้ใช้บริการสามารถดูตำแหน่งร้านได้ โดยไม่จำเป็นต้องทำการเข้าสู่ระบบก่อน
ค้นหาร้านเสริมสวย	ผู้ใช้งานทั่วไปสามารถค้นหาร้านได้โดยไม่จำเป็นต้องทำการเข้าสู่ระบบก่อน
ดูผลงานของร้านเสริมสวย	ผู้ใช้บริการสามารถดูผลงานของร้านเสริมสวยได้โดยไม่ต้องทำการเข้าสู่ระบบก่อน
ดูคิวว่างของร้านเสริมสวย	ผู้ใช้บริการสามารถดูคิวว่างของร้านเสริมสวยได้โดยไม่ต้องทำการเข้าสู่ระบบก่อน
ดูรีวิว	ผู้ใช้บริการสามารถดูรีวิวได้โดยไม่จำเป็นต้องเข้าสู่ระบบก่อน
เขียนรีวิวติชม	ผู้ใช้บริการสามารถเขียนรีวิวติชมได้โดยจำเป็นต้องเข้าสู่ระบบก่อน
จองคิวร้านเสริมสวย	ผุ้ใช้บริการสามารถจองคิวร้านเสริมสวยได้โดยจำเป็นต้องเข้าสู่ระบบก่อน และจำเป็นต้อง เลือก วันที่ รายการ ช่าง เวลา
เพิ่ม แก้ไข และลบรายการ	ใช้เจ้าของร้านเพื่อ เพิ่ม แก้ไขหรือลบข้อมูลรายการ
post ภาพผลงานทั้งหมดของร้านเสริมสวย	ใช้สำหรับเจ้าของร้านเพื่อ เพิ่ม post รูปภาพผลงานของร้าน
เพิ่มแก้ไขและลบข้อมูลร้าน	ใช้สำหรับเจ้าของร้านเพื่อ เพิ่ม แก้ไขหรือลบข้อมูลร้าน

ตารางที่ 3.3: อธิบาย Use Case หน้าที่ของระบบ(ต่อ) ในภาพที่ 3.18

Use Case	คำอธิบาย
ดูการจองคิว	ใช้สำหรับเจ้าของร้านเพื่อ ดูคิวจองที่ผู้ใช้บริการได้ทำการจองคิว
เพิ่มข้อมูลช่าง	ใช้สำหรับเจ้าของร้านเพื่อ เพิ่ม ข้อมูลช่าง
แก้ไขข้อมูล	ใช้สำหรับช่างเพื่อ แก้ไขข้อมูลส่วนตัว
ดูตารางงาน	ใช้สำหรับช่างเพื่อ ดูตารางงาน
post ภาพผลงานของตัวเอง	ใช้สำหรับช่างเพื่อ post ภาพผลงาน

ตารางที่ 3.4: Use Case ดูข้อมูลต่างๆของร้านเสริมสวย

Use Case Title : ดูข้อมูลต่างๆของร้านเสริมสวย	Use case Id : 1
Primary Actor : ผู้ใช้บริการ	
Stakeholder Actor : เจ้าของร้าน	
Main Flow : ผู้ใช้บริการดูข้อมูลต่างๆของร้านเสริมสวยโดยไม่จำเป็นต้องเข้าสู่ระบบ	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่เชื่อมต่ออินเทอร์เน็ต จะไม่สามารถดูข้อมูลต่างๆ ของร้านเสริมสวยได้	

ตารางที่ 3.5: Use Case ดูตำแหน่งของร้านเสริมสวย

Use Case Title : ดูตำแหน่งของร้านเสริมสวย	Use case Id : 2
Primary Actor : ผู้ใช้บริการ	
Stakeholder Actor : เจ้าของร้าน	
Main Flow : ผู้ใช้บริการสามารถดูตำแหน่งร้านเสริมสวย ไม่จำเป็นต้องเข้าสู่ระบบ	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่เชื่อมต่ออินเทอร์เน็ต จะไม่สามารถดูตำแหน่งของร้านเสริมสวยได้	

ตารางที่ 3.6: Use Case ค้นหาร้านเสริมสวย

Use Case Title : ค้นหาร้านเสริมสวย	Use case Id : 3
Primary Actor : ผู้ใช้บริการ	
Stakeholder Actor : เจ้าของร้าน	
Main Flow : ผู้ใช้บริการสามารถค้นหาร้านเสริมสวย ไม่จำเป็นต้องเข้าสู่ระบบ	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่เชื่อมต่ออินเทอร์เน็ต จะไม่สามารถค้นหาร้านเสริมสวยได้	

ตารางที่ 3.7: Use Case ดูผลงานของร้านเสริมสวย

Use Case Title : ดูผลงานของร้านเสริมสวย	Use case Id : 4
Primary Actor : ผู้ใช้บริการ	
Stakeholder Actor : เจ้าของร้าน	
Main Flow : ดูผลงานของร้านเสริมสวย โดยไม่จำเป็นต้องเข้าสู่ระบบ	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่เชื่อมต่ออินเทอร์เน็ต จะไม่สามารถดูผลงานของร้านเสริมสวยได้	

ตารางที่ 3.8: Use Case ดูคิวว่างของร้านเสริมสวย

Use Case Title : ดูคิวว่างของร้านเสริมสวย	Use case Id : 5
Primary Actor : ผู้ใช้บริการ	
Stakeholder Actor : เจ้าของร้าน	
Main Flow : ดูคิวว่างของร้านเสริมสวยได้ โดยไม่จำเป็นต้องเข้าสู่ระบบ	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่เชื่อมต่ออินเทอร์เน็ต จะไม่สามารถดูคิวว่างของร้านเสริมสวยได้	

ตารางที่ 3.9: Use Case ดูรีวิว

Use Case Title : ดูรีวิว	Use case Id : 6
Primary Actor : ผู้ใช้บริการ	
Stakeholder Actor : เจ้าของร้าน	
Main Flow : ดูรีวิวได้โดยไม่จำเป็นต้องเข้าสู่ระบบ	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่เชื่อมต่ออินเทอร์เน็ต จะไม่สามารถดูรีวิวได้	

ตารางที่ 3.10: Use Case เขียนรีวิว

Use Case Title : เขียนรีวิว	Use case Id : 7
Primary Actor : ผู้ใช้บริการ	
Stakeholder Actor : -	
Main Flow : เขียนรีวิวติชมได้ต้องเข้าสู่ระบบ	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่เชื่อมต่ออินเทอร์เน็ต จะไม่สามารถเขียนรีวิวได้	

ตารางที่ 3.11: Use Case จองคิวร้านเสริมสวย

Use Case Title : จองคิวร้านเสริมสวย	Use case Id : 10
Primary Actor : ผู้ใช้บริการ	
Stakeholder Actor : เจ้าของร้าน,ช่าง	
Main Flow : ผู้ใช้บริการทำการจองคิวร้านเสริมสวย ได้โดยจำเป็นต้องเข้าสู่ระบบ และจำเป็นต้องเลือก วันที่ รายบการ ช่าง เวลา	
Exceptional Flow ที่ 1 : หากผู้ใช้ไม่เชื่อมต่ออินเทอร์เน็ต จะไม่สามารถ จองคิวร้านเสริมสวยได้	

ตารางที่ 3.13: Use Case post ภาพผลงานทั้งหมดของร้าน

Use Case Title : post ภาพผลงานทั้งหมดของร้าน	Use case Id : 12
Primary Actor : เจ้าของร้าน	
Stakeholder Actor : -	
Main Flow : เจ้าของร้าน post ภาพผลงานทั้งหมดของร้านได้	
Exceptional Flow ที่ 1 : หากเจ้าหน้าที่ไม่เชื่อมต่ออินเทอร์เน็ต จะไม่สามารถ post ภาพผลงานทั้งหมดของร้านได้	

ตารางที่ 3.14: Use Case เพิ่มแก้ไขและลบข้อมูลร้าน

Use Case Title : เพิ่มแก้ไขและลบข้อมูลร้าน	Use case Id : 13
Primary Actor :เจ้าของร้าน	
Stakeholder Actor : -	
Main Flow : เจ้าของร้าน เพิ่ม แก้ไขและลบข้อมูลร้าน	
Exceptional Flow ที่ 1 : หากเจ้าหน้าที่ไม่เชื่อมต่ออินเทอร์เน็ต จะไม่สามารถ เพิ่ม แก้ไขและลบข้อมูลร้านได้	

ตารางที่ 3.15: Use Case ดูการจองคิว

Use Case Title : ดูการจองคิว	Use case Id : 14
Primary Actor :เจ้าของร้าน	
Stakeholder Actor : -	
Main Flow : ดูการจองคิวที่ผู้ใช้บริการได้ทำการจองคิว โดยผู้ใช้บริการ	
Exceptional Flow ที่ 1 : หากเจ้าของร้านที่ไม่เชื่อมต่ออินเทอร์เน็ต ไม่สามารถดูการจองคิวที่ผู้ใช้บริการได้ทำการจองคิว โดยผู้ใช้บริการได้	

ตารางที่ 3.16: Use Case เพิ่มข้อมูลช่าง

Use Case Title : เพิ่มข้อมูลช่าง	Use case Id : 15
Primary Actor :เจ้าของร้าน	
Stakeholder Actor : -	
Main Flow : เจ้าของร้านเพิ่มข้อมูลช่าง	
Exceptional Flow ที่ 1 : หากเจ้าของร้านที่ไม่เชื่อมต่ออินเทอร์เน็ต ไม่สามารถเพิ่มข้อมูลช่างได้	

ตารางที่ 3.17: Use Case แก้ไขข้อมูล

Use Case Title : แก้ไขข้อมูล	Use case Id : 16
Primary Actor :ช่าง	
Stakeholder Actor : -	
Main Flow : ช่างแก้ไขข้อมูลส่วนตัว	
Exceptional Flow ที่ 1 : หากช่างที่ไม่เชื่อมต่ออินเทอร์เน็ต ไม่สามารถแก้ไขข้อมูลส่วนตัวได้	

ตารางที่ 3.18: Use Case ดูตารางงาน

Use Case Title : ดูตารางงาน	Use case Id : 17
Primary Actor :ช่าง	
Stakeholder Actor : -	
Main Flow : ดูตารางงาน ที่ผู้ใช้บริการได้ทำการจองคิว	
Exceptional Flow ที่ 1 : หากช่างที่ไม่เชื่อมต่ออินเทอร์เน็ต ไม่สามารถดตารางงานได้	

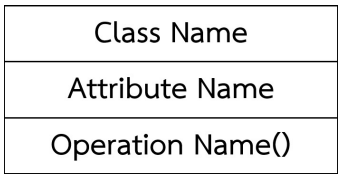
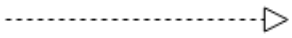
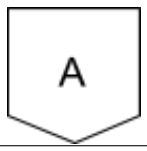
ตารางที่ 3.19: Use Case post ภาพผลงานของตัวเอง

Use Case Title : post ภาพผลงานของตัวเอง	Use case Id : 18
Primary Actor :ช่าง	
Stakeholder Actor : -	
Main Flow : post ภาพผลงานของตัวเองได้	
Exceptional Flow ที่ 1 : หากช่างที่ไม่เชื่อมต่ออินเทอร์เน็ต ไม่สามารถpost ภาพผลงานของตัวเองได้	

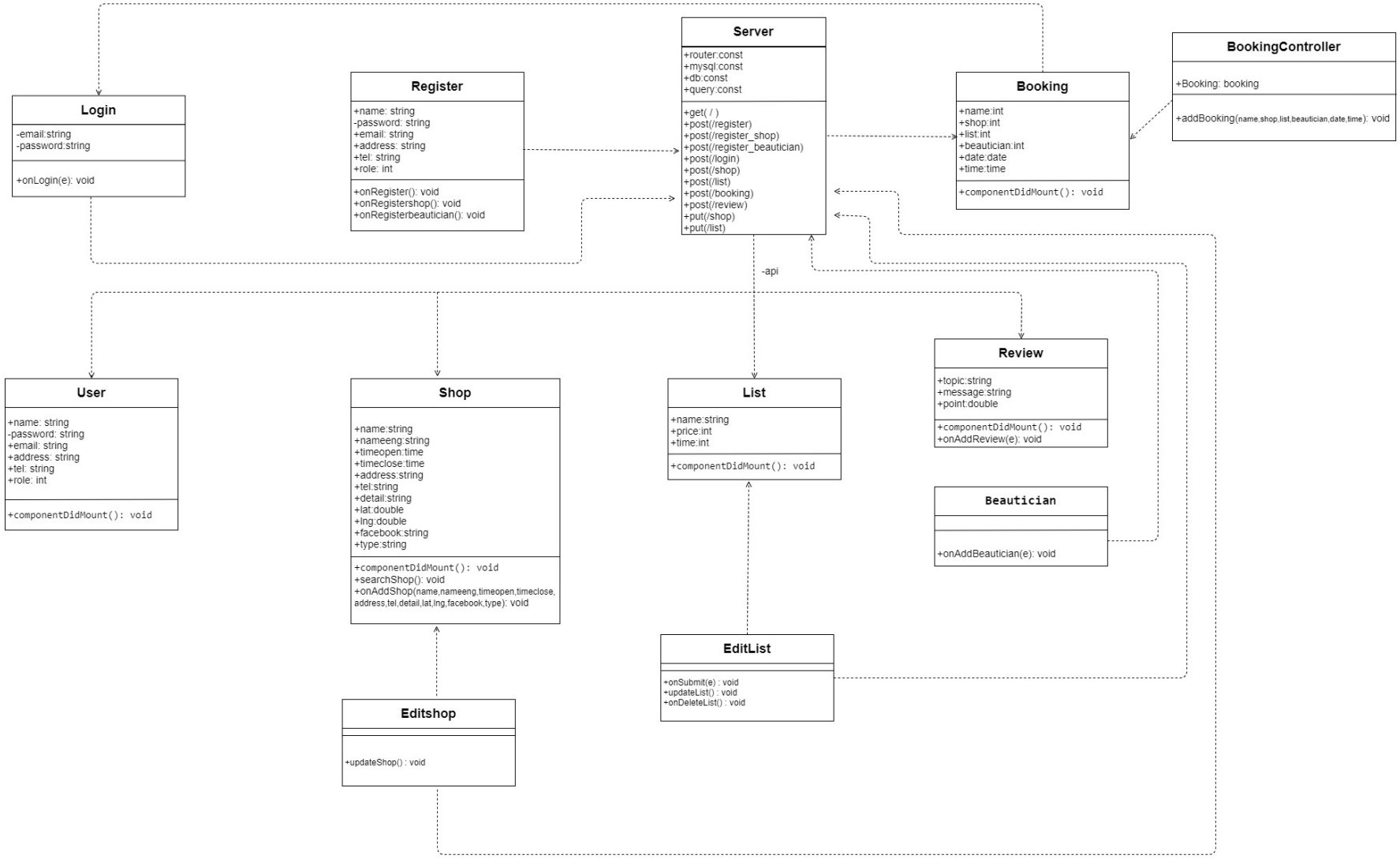
3.5 Class Diagram

Class Diagram คือแผนภาพที่ใช้แสดงคลาสและความสัมพันธ์ในแบบต่างๆ ระหว่างคลาส สัญลักษณ์ที่ใช้ในการเขียน Class Diagram แสดงในตารางที่ 3.20

ตารางที่ 3.20: สัญลักษณ์ของ Class Diagram

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

Class Diagram แสดงความสัมพันธ์ในรูปแบบต่างๆ ระหว่างคลาสของแอปพลิเคชันระบบกองทุนเงินให้กู้ยืมเพื่อการศึกษา คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี อธิบายได้ตามภาพที่ 3.19 ดังต่อไปนี้



รูปที่ 3.19: Class Diagram ของแอปพลิเคชันระบบจองคิวร้านเสริมสวย

จากรูปภาพที่ 3.19 สามารถอธิบายแผนภาพ Class Diagram ได้ดังนี้

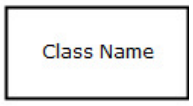
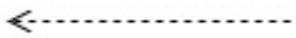
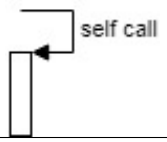
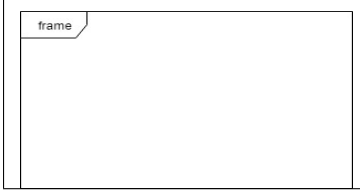
ตารางที่ 3.21: อธิบาย Class Diagram ของคลาสพื้นฐานของระบบ

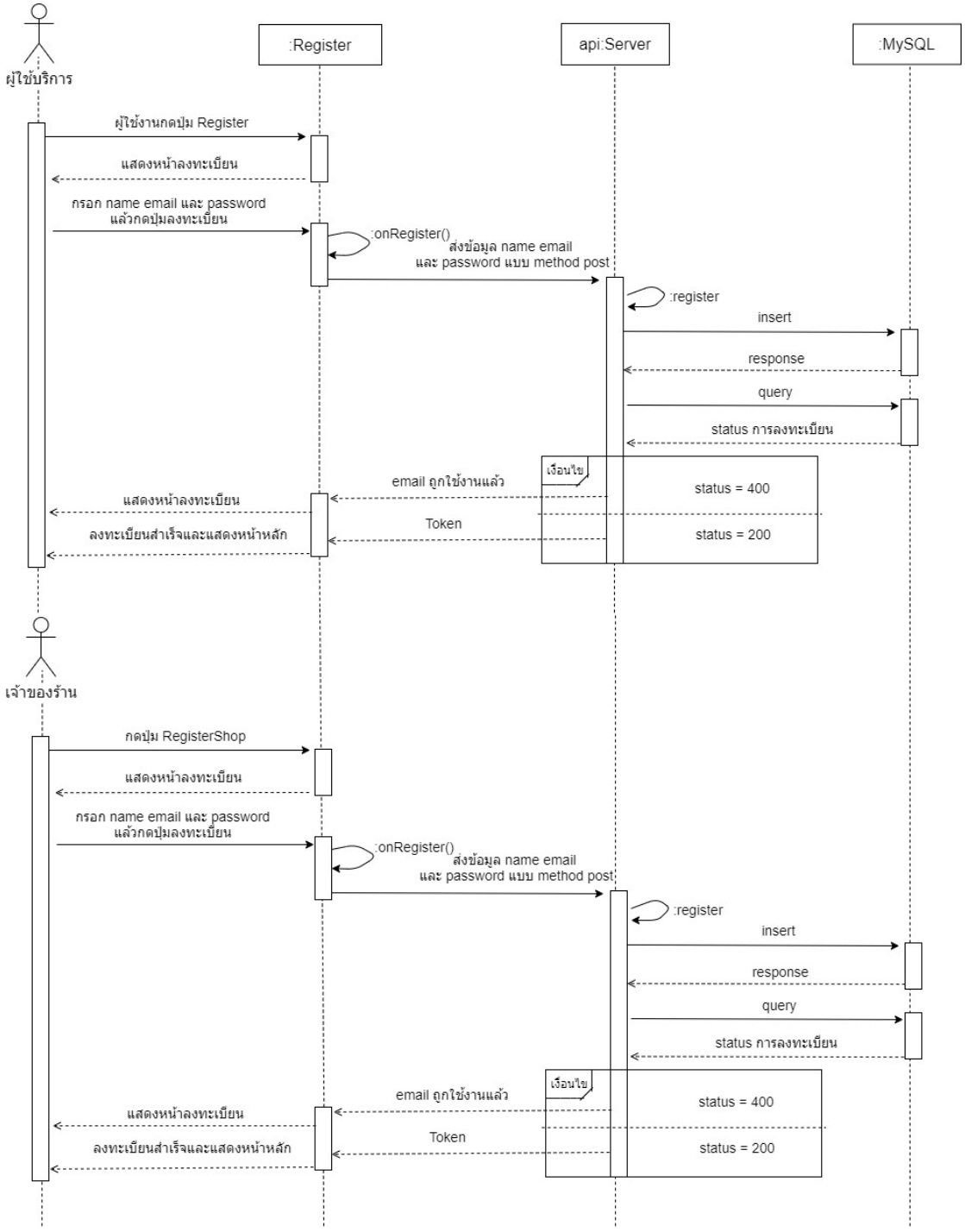
Class Diagram	คำอธิบาย
Server	คลาส Server จะถูกเรียกใช้งานทุกครั้งเมื่อผู้ใช้เปิดเว็บแอปพลิเคชัน โดยวัตถุประสงค์การทำงานของคลาสนี้คือ เพื่อใช้ในการเชื่อมต่อ api ระหว่างคลาสอื่นๆ กับฐานข้อมูล
Register	คลาส Register จะถูกเรียกใช้งานทุกครั้งเมื่อผู้ใช้เปิดเว็บแอปพลิเคชัน โดยวัตถุประสงค์การทำงานของคลาสคือ เพื่อใช้ในการลงทะเบียนขอใช้เว็บแอปพลิเคชัน
Login	คลาส Login เป็นคลาสที่ใช้เพื่อให้ผู้ใช้ที่ได้ลงทะเบียนกับระบบเข้าสู่ระบบเพื่อใช้งานบริการต่าง ๆ จากระบบ
User	คลาส User เป็นคลาสที่ใช้เก็บข้อมูลจากการลงทะเบียน
BookingController	คลาส BookingController เป็นคลาสที่ใช้เพิ่มข้อมูลช่าง
Shop	คลาส Shop เป็นคลาสที่ใช้จัดการการทำงาน Shop
EditShop	คลาส EditShop เป็นคลาสที่ใช้ในการจัดการการ update ข้อมูลร้าน
List	คลาส List เป็นคลาสที่ใช้จัดการการทำงานของรายการ
EditList	คลาส EditList เป็นคลาสที่ใช้จัดการการ update ข้อมูลรายการ
Review	คลาส Review เป็นคลาสที่ใช้จัดการการรีวิวของผู้ใช้งาน
Booking	คลาส Booking เป็นคลาสที่ใช้เก็บข้อมูลการจองคิว
Bookingcontroller	คลาส Bookingcontroller เป็นคลาสที่ใช้จัดการการจองคิว

3.6 Sequence Diagram

Sequence Diagram เป็น Diagram ที่แสดงขั้นตอนการทำงานของแต่ละ Use Case ระหว่าง Object ต่างๆ ที่ส่งข้อความถึงกันและกัน โดย Sequence Diagram จะช่วยให้มองเห็นการทำงานของภาพรวมของระบบ ส่วนประกอบสัญลักษณ์ที่ใช้ในการเขียน Sequence Diagram แสดงดังตารางที่ 3.22

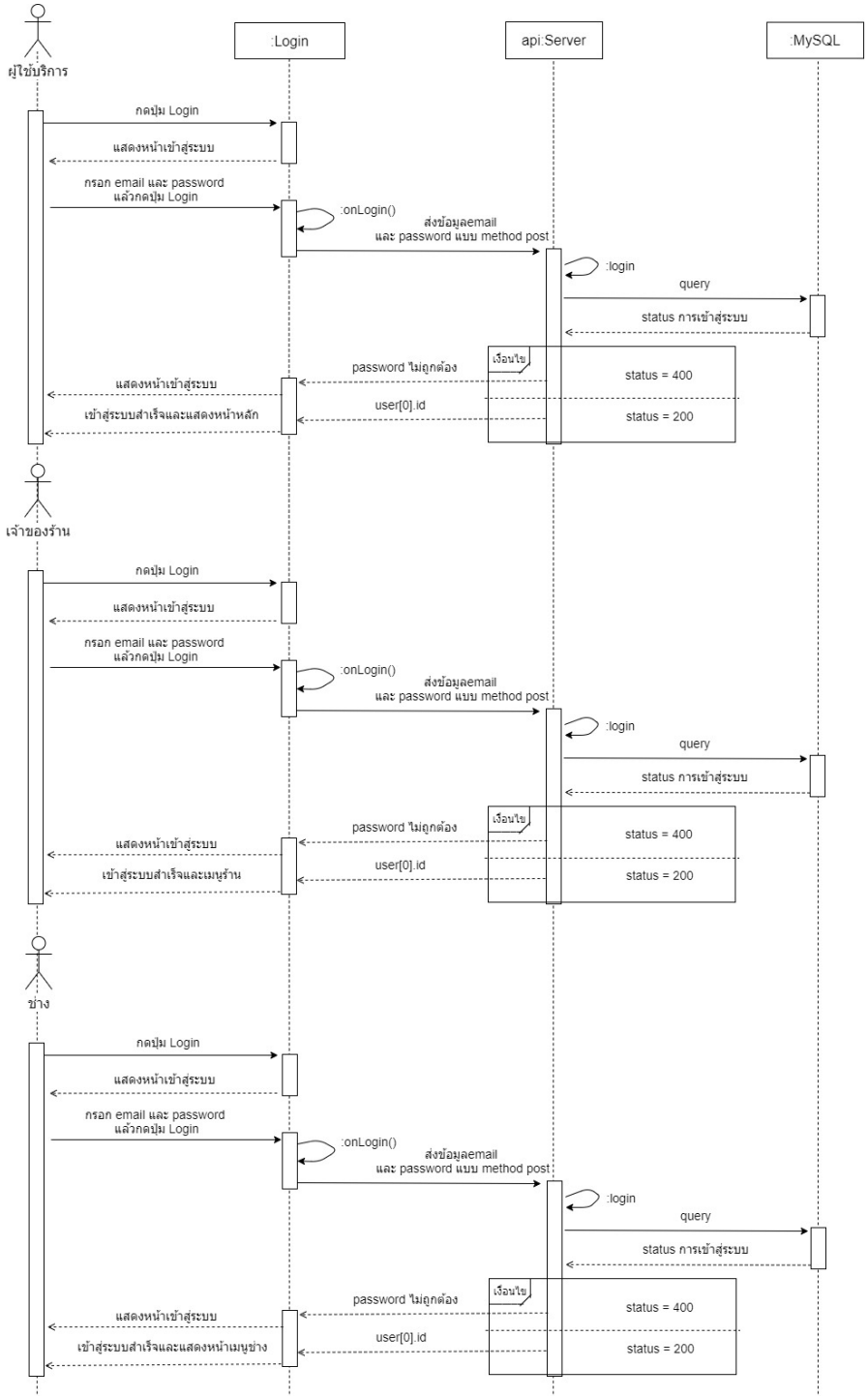
ตารางที่ 3.22: สัญลักษณ์ของ Sequence Diagram

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



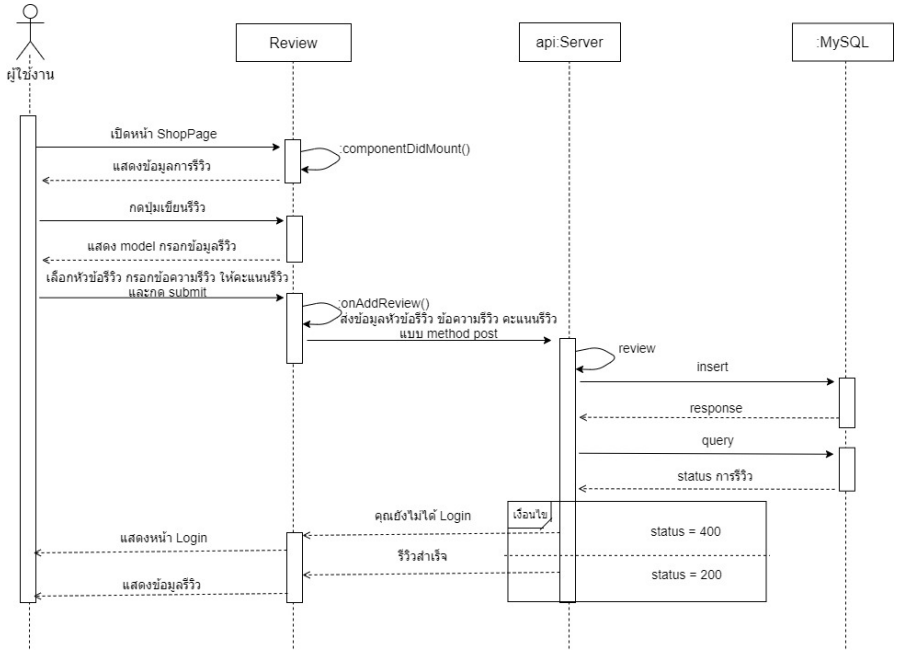
รูปที่ 3.20: Sequence Diagram ลงทะเบียน

จากภาพที่ 3.20 สามารถอธิบายแผนภาพ Sequence Diagram การลงทะเบียน ได้ดังนี้ เมื่อ ผู้ใช้กดปุ่ม register คลาส Register แสดงหน้าลงทะเบียน เมื่อผู้ใช้กรอกข้อมูลระบบจะเรียกใช้เมธอด onRegister() ที่คลาส Register เมื่อ ข้อมูลการลงทะเบียน ถูกติดตั้งบน api:Server เมธอด register จะทำการเพิ่มข้อมูลการลงทะเบียนลงในฐานข้อมูล และนำข้อมูลที่ได้เพิ่มลงมาเก็บในรูปแบบ response ที่คลาส api:Server โดยมีการสืบค้นข้อมูลเพื่อตรวจสอบการลงทะเบียนที่คลาส MySQL และเมื่อข้อมูลที่เพิ่มมีการลงทะเบียนแล้วจะแสดง error email ถูกใช้งานแล้ว แต่ถ้าลงทะเบียนสำเร็จจะส่งข้อมูลมาที่คลาส Register และคลาส Register จะส่งข้อมุลการลงทะเบียนแจ้งผู้ใช้ว่าลงทะเบียนสำเร็จและจะแสดงหน้าหลัก



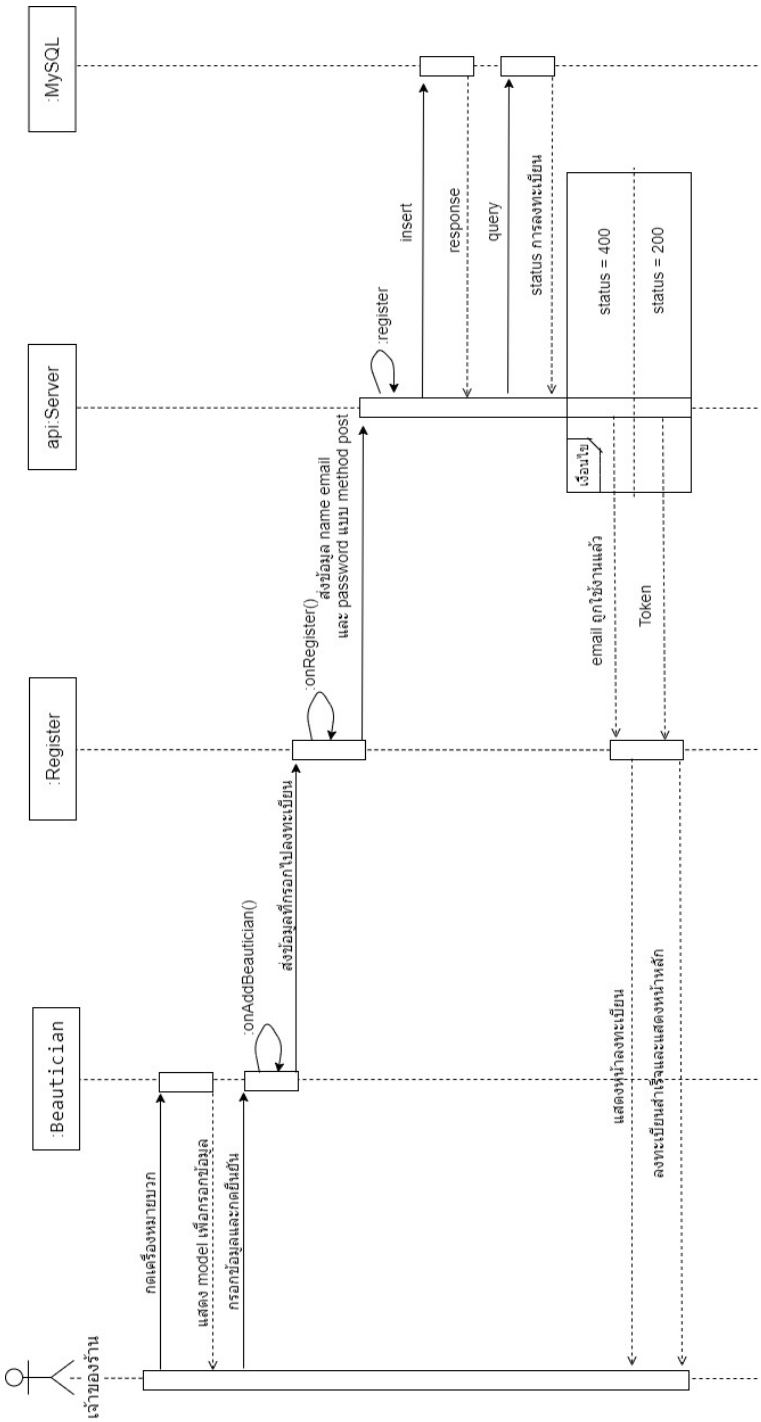
รูปที่ 3.21: Sequence Diagram เข้าสู่ระบบ

จากภาพที่ 3.21 สามารถอธิบายแผนภาพ Sequence Diagram การเข้าสู่ระบบ ได้ดังนี้ เมื่อ เมื่อผู้ใช้กดปุ่ม Login ที่คลาส Login ระบบจะแสดงหน้า login ผู้ใช้ทำการกรอก email และ password ที่คลาส Login เมธอด onLogin() จะทำการเพิ่มข้อมูล Login โดยคลาส api:Server เมธอด login จะทำการสืบค้นข้อมูลจากฐานข้อมูลมาตรวจสอบสถานะการเข้าสู่ระบบ api:Server ทำการส่งข้อมูลการเข้าสู่ระบบไปยังคลาส Login และคลาส Login ส่งข้อมูลการเข้าสู้ระบบไปยังผู้ใช้แจ้งว่าเข้าสู่ระบบสำเร็จ



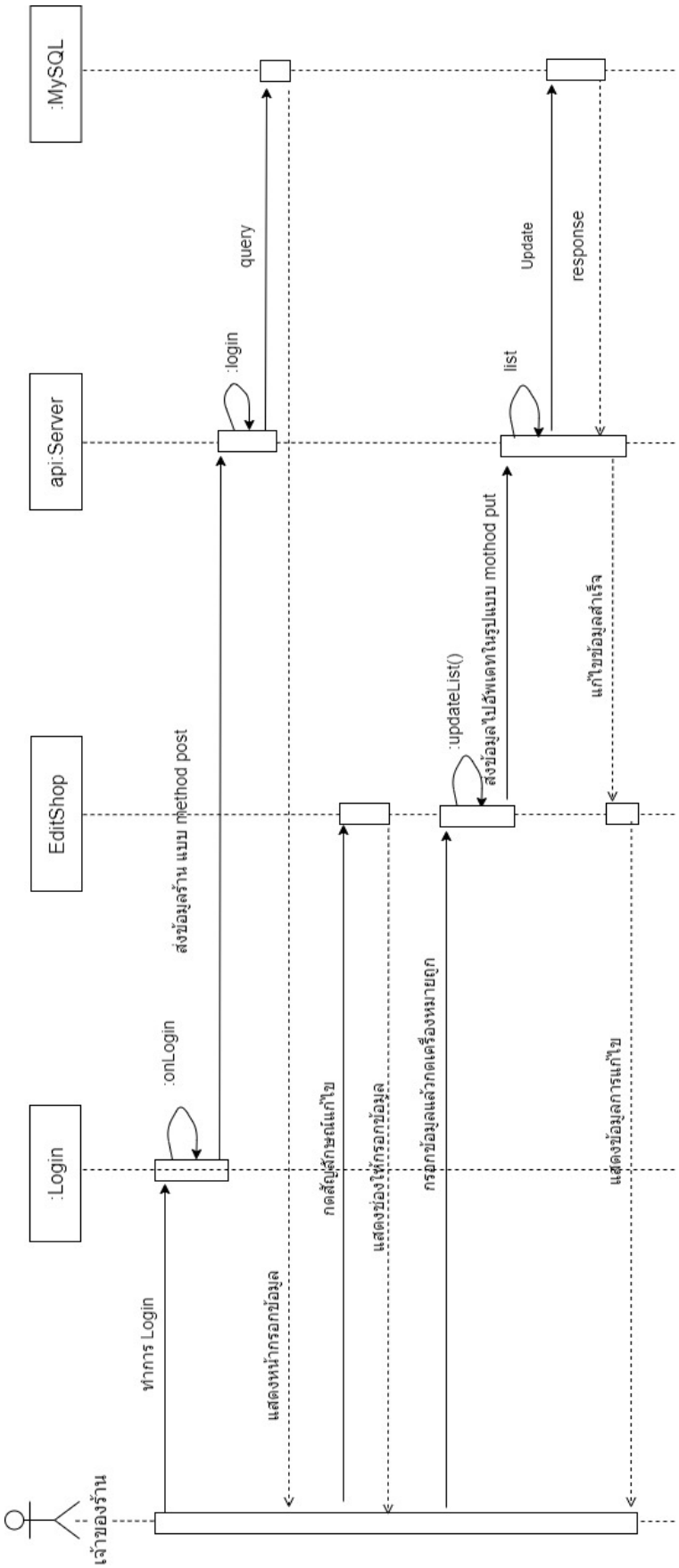
รูปที่ 3.22: Sequence Diagram การรีวิว

จากภาพที่ 3.22 สามารถอธิบายแผนภาพ Sequence Diagram การรีวิว ได้ดังนี้ เมื่อผู้ใช้เปิดหน้า ShopPage ระบบจะเรียกใช้เมธอด componentDidMount() ที่คลาส Review ระบบจะแสดงข้อมูลการรีวิวออกทางหน้าจอ ผู้ใช้กดปุ่มเขียนรีวิวระบบจะแสดง model กรอกข้อมูลรีวิว และเมื่อผู้ใช้งานเลือกหัวข้อรีวิว กรอกข้อความรีวิว ให้คะแนนรีวิว และกด submit ระบบจะเรียกใช้เมธอด onAddReview() ที่คลาส Review โดยระบบจะส่งข้อมูลการรีวิวในรูปแบบ post ไปที่คลาส api:Server เมธอด review ทำการ insert ข้อมูลรีวิวลงฐานข้อมูล และระบบจะส่งสถานะการรีวิวมาที่คลาส api:Server และคลาส api:Server ทำการส่งสถานะการรีวิวสำเร็จมาที่คลาส Review ระบบจะแสดงข้อมูลการรีวิวออกทางหน้าจอ



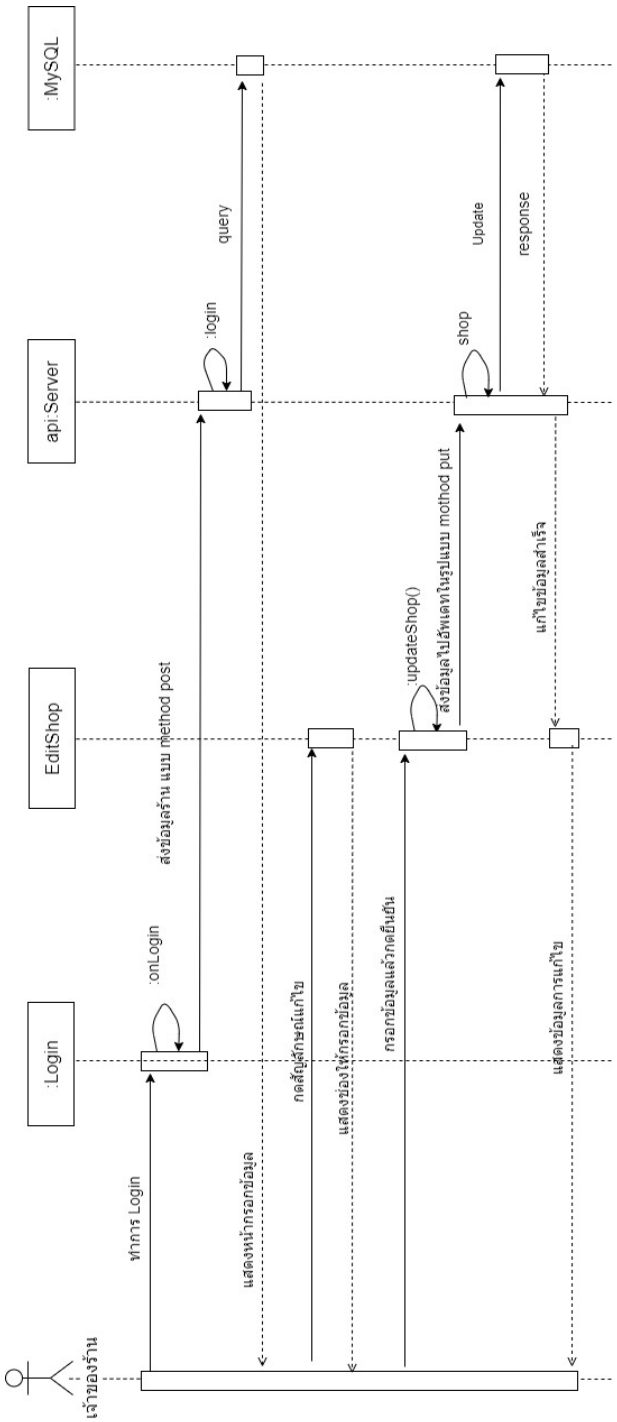
รูปที่ 3.23: Sequence Diagram การเพิ่มข้อมูลช่าง

จากภาพที่ 3.23 สามารถอธิบายแผนภาพ Sequence Diagram การเพิ่มข้อมูลช่าง ได้ดังนี้ เมื่อเจ้าของร้านกดเครื่องหมายบอกระบบจะทำการแสดง model เพื่อกรอกข้อมูลเจ้าของร้านทำการกรอกข้อมูลและกดยืนยัน ระบบจะเรียกใช้เมธอด onAddBeautician ที่คลาส Beautician ระบบจะทำการส่งข้อมูลการลงทะเบียนช่างไปยังคลาส Register โดยใช้เมธอด Onregister() ที่คลาส Register ส่งข้อมูลการลงทะเบียนระบบจะเรียกใช้เมธอด register ที่คลาส api:Server ทำการเชื่อม api insert ข้อมูลลงฐานข้อมูล ฐานข้อมูลจะส่งสถานะการลงทะเบียนมาที่คลาส api:Server ระบบจะแสดงสถานะการลงทะเบียนมายังคลาส Register และจะแสดงออกทางหน้าจอไปยังเจ้าของร้าน



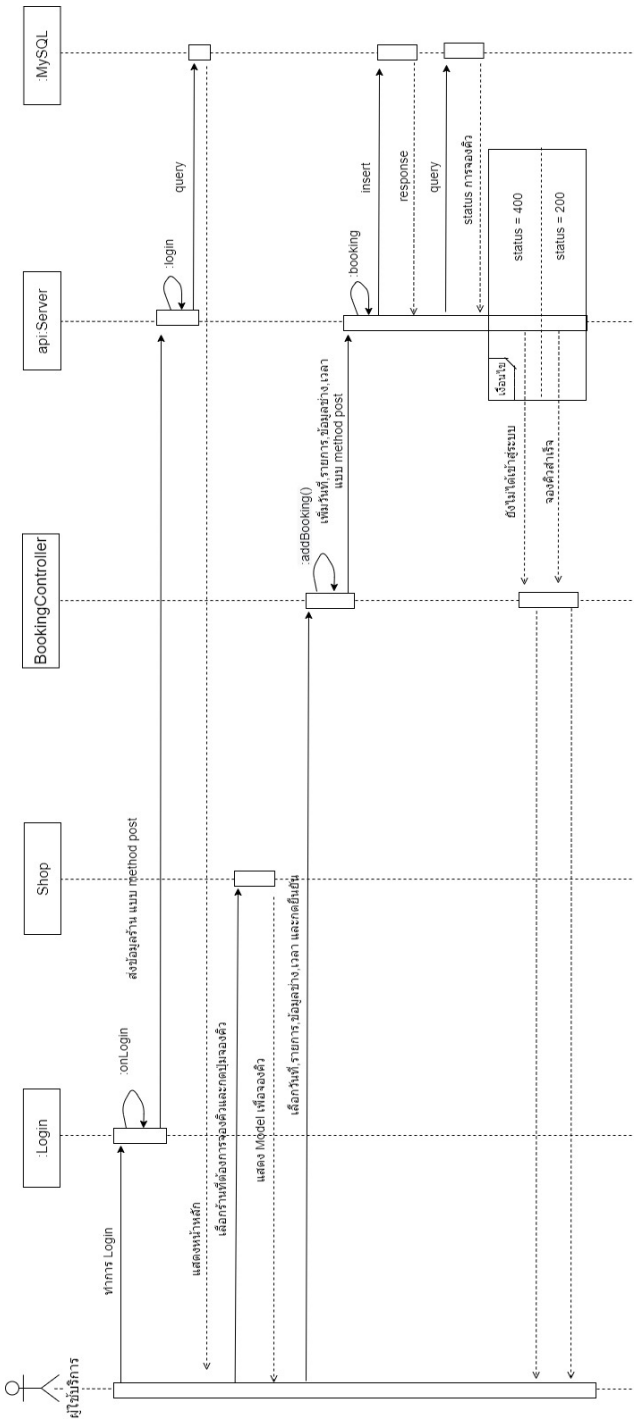
รูปที่ 3.24: Sequence Diagram การแก้ไขรายการ

จากภาพที่ 3.24 สามารถอธิบายแผนภาพ Sequence Diagram การแก้ไขรายการ ได้ดังนี้ เมื่อเจ้าของร้านทำการ login เสร็จ เจ้าของร้านกดที่สัญลักษณ์แก้ไขระบบจะแสดงช่องให้กรอกข้อมูล และเมื่อเจ้าของร้านกรอกข้อมูลที่ต้องการแก้ไขเสร็จแล้วกดเครื่องหมายถูก ระบบจะทำการเรียกใช้เมธอด updateList() ที่คลาส Editlist ระบบจะทำการส่งข้อมูลแบบ put ไปยังคลาส api:Server ด้วยเมธอด list จะทำการ update ข้อมูลลงฐานข้อมูล ระบบจะทำการแจ้งเตือนแก้ไขข้อมูลสำเร็จและแสดงข้อมูลแก้ไขออกทางหน้าจอ



รูปที่ 3.25: Sequence Diagram การแก้ไขข้อมูลร้าน

จากภาพที่ 3.25 สามารถอธิบายแผนภาพ Sequence Diagram การแก้ไขข้อมูลร้าน ได้ดังนี้ แมื่อเจ้าของร้านทำการ login เสร็จ เจ้าของร้านกดที่สัญลักษณ์แก้ไขระบบจะแสดงช่องให้กรอกข้อมูล และเมื่อเจ้าของร้านกรอกข้อมูลที่ต้องการแก้ไขเสร็จแล้วกดยืนยัน ระบบจะทำการเรียกใช้เมธอด updateShop() ที่คลาส EditShop ระบบจะทำการส่งข้อมูลแบบ put ไปยังคลาส api:Server ด้วยเมธอด shop จะทำการ update ข้อมูลลงฐานข้อมูล ระบบจะทำการแจ้งเตือนแก้ไขข้อมูลสำเร็จและแสดงข้อมูลแก้ไขออกทางหน้าจอ



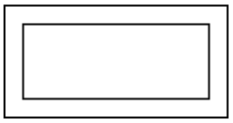
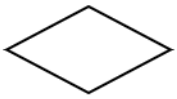
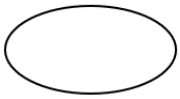
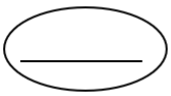
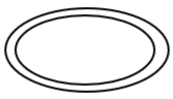
รูปที่ 3.26: Sequence Diagram การจองคิว

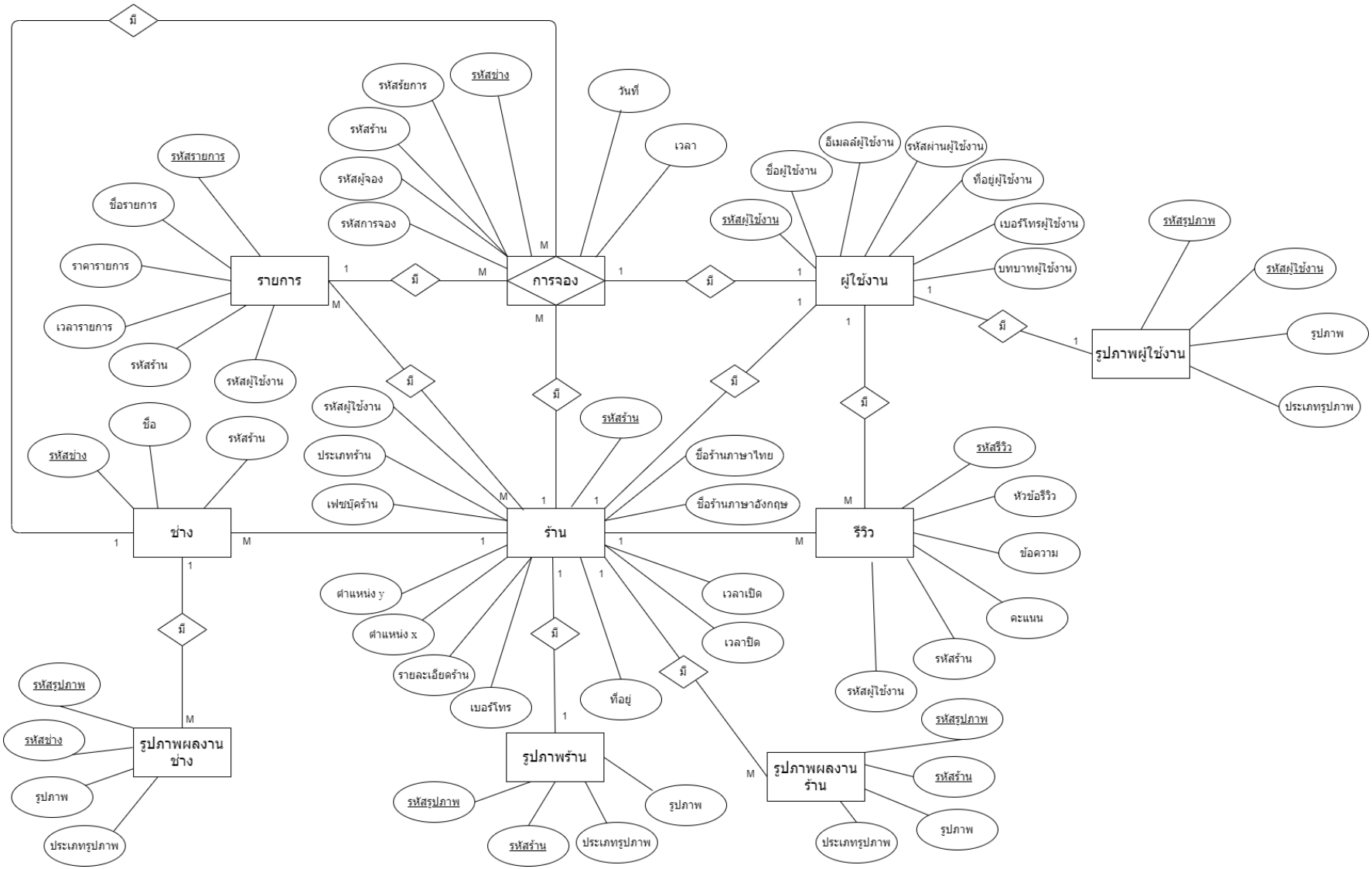
จากภาพที่ 3.26 สามารถอธิบายแผนภาพ Sequence Diagram การจองคิว ได้ดังนี้ เมื่อผู้ใช้ทำการเข้าสู่ระบบสำเร็จ ทำการเลือกร้านที่ต้องการจองคิวที่คลาส Shop ระบบจะทำการแสดง model เพื่อจองคิว ผู้ใช้เรียกวันที่ รายการ ข้อมูลช่าง เวลา และกดยืนยัน ระบบจะเรียกใช้เมธอด addBooking() ที่คลาส BookingController เพิ่มข้อมูลวันที่ รายการ ข้อมูลช่าง เวลา แบบเมธอด post ในคลาส api:Server ด้วย เมธอด booking ทำการ insert ข้อมูลการจองคิวลงฐานข้อมูล ระบบจะทำการเช็คการจองคิวที่คลาส api:Server จะทำการตรวจสอบข้อมูลการจองคิวกับฐานข้อมูลและส่งสถานะกับมาที่คลาส api:Server เพื่อแจ้งเตือนสถานะการจองคิว ถ้าการจองคิวไม่สำเร็จระบบจะแจ้งเตือนว่ายังไม่เข้าสู่ระบบและจะแสดงหน้าเข้าสู่ระบบ และเมื่อจองคิวสำเร็จระบบจะแจ้งเตือนว่าจองคิวสำเร็จระบบจะแสดงข้อมูลการจองคิวออกทางหน้าจอ

3.7 ER-Diagram

แผนภาพแสดงความสัมพันธ์ระหว่างข้อมูล (Entity Relationship Diagram : ERD) คือ แผนภาพที่ใช้เป็นเครื่องมือสำหรับจำลองข้อมูล ซึ่งจะประกอบไปด้วย เอนทิตี (Entity) แอททริบิว (Attribute) และความสัมพันธ์ (Relationship) ซึ่งการออกแบบฐานข้อมูลของเว็บบล็อกสำหรับวิจารณ์ภาพยนตร์ โดยใช้รูปแบบ ERD Chen Model ซึ่งมีสัญลักษณ์ดังตารางที่ 3.23

ตารางที่ 3.23: สัญลักษณ์ของ ER-Diagram

สัญลักษณ์	การใช้งาน
	การใช้งาน ใช้แสดงเอนทิตี้ เอนทิตี้ คือ กลุ่มของข้อมูลที่เป็นเรื่องเดียวกันที่ เกี่ยวข้องกัน
	เอนทิตี้อ่อนแอ (Weak entity) คือ เอนทิตี้ที่มีการคงอยู่เกี่ยวข้องกับเอนทิตี้อื่นในระบบฐานข้อมูล
	เส้นความสัมพันธ์ (Relationship line) คือ เส้นเชื่อมความสัมพันธ์ระหว่าง เอนทิตี้
	ความสัมพันธ์ ใช้แสดงความสัมพันธ์ระหว่าง 2 เอนทิตี้ขึ้นไป
	Generalization แอททริบิว ใช้แสดง แอททริบิว ของ เอนทิตี้
	คีย์หลัก (Key Attribute) ใช้แสดงคีย์หลัก
	Multi-Value Attribute แอตทริ้วที่มีค่าของข้อมูลในแต่ละสมาชิกของเอนทิตี้ได้หลายค่า
1:1	ความสัมพันธ์แบบหนึ่งต่อหนึ่ง
1:M	ความสัมพันธ์แบบหนึ่งต่อกลุ่ม
M:M	ความสัมพันธ์แบบกลุ่มต่อกลุ่ม



รูปที่ 3.27: ER Diagram ระบบจองคิวร้านเสริมสวย

ตารางที่ 3.24: แสดงรายละเอียด Entity ของ ER-Diagram ระบบจองคิวร้านเสริมสวย

ชื่อตาราง(ภาษาอังกฤษ)	ชื่อตาราง(ภาษาไทย)	อธิบาย
Users	ข้อมูลผู้ใช้	ตารางเก็บข้อมูลผู้ใช้งาน
shops	ข้อมูลร้าน	ตารางเก็บข้อมูลร้าน
Lists	ข้อมูลรายการ	ตารางเก็บข้อมูลรายการ
Bookings	ข้อมูลการจอง	ตารางเก็บข้อมูการจองคิว
Beauticians	ข้อมูลผช่าง	ตารางเก็บข้อมูช่าง
Reviews	ข้อมูลรีวิว	ตารางเก็บข้อมูลรีวิว
Userimages	ข้อมูลรูปภาพผู้ใช้	ตารางเก็บข้อมูลรูปภาพผู้ใช้งาน
Shopimages	ข้อมูลรูปภาพร้าน	ตาราง เก็บ ข้อมูล รูปภาพร้าน
Workimages	ข้อมูล รูปภาพ ผลงานร้าน	ตาราง เก็บ ข้อมูล รูปภาพผลงานร้าน
Beauticianimages	ข้อมูล รูปภาพ ผลงานช่าง	ตาราง เก็บ ข้อมูล รูปภาพผลงานช่าง

ตารางที่ 3.25: แสดงรายละเอียด Attribute ของ Entity Users

Field	Type	Description	Key	Reference
id	int(11)	รหัสผู้ใช้งาน	PK	
name	varchar(50)	ชื่อ		
email	varchar(255)	อีเมลล์		
password	varchar(255)	รหัสผ่าน		
address	text	ที่อยู่		
tel	int(11)	เบอร์โทร		
role	int (11)	บทบาท		

ตารางที่ 3.26: แสดงรายละเอียด Attribute ของ Entity Shops

Field	Type	Description	Key	Reference
id	int(11)	รหัสร้าน	PK	
name	varchar(50)	ชื่อ		
nameeng	varchar(50)	ชื่อ ภาษา อังกฤษ		
timeopen	time	เวลาเปิด		
timeclose	time	เวลาปิด		
address	text	ที่อยู่		
tel	varchar(20)	เบอร์โทร		
detail	text	รายละเอียด		
lat	double	ตำแหน่ง x		
lng	double	ตำแหน่ง y		
facebook	text	เฟซบุ๊ค		
userId	int(11)	รหัสผู้ใช้	FK	Users
type	text	บทบาท		

ตารางที่ 3.27: แสดงรายละเอียด Attribute ของ Entity Lists

Field	Type	Description	Key	Reference
id	int(11)	รหัสรายการ	PK	
name	varchar(50)	ชื่อ		
price	int(11)	ราคา		
time	int(11)	เวลา		
shop	int(11)	รหัสร้าน	FK	Shops
userId	int(11)	รหัสผู้ใช้งาน	FK	Users

ตารางที่ 3.28: แสดงรายละเอียด Attribute ของ Entity Bookings

Field	Type	Description	Key	Reference
id	int(11)	รหัสการจอง	PK	
name	varchar(50)	ชื่อ		
shop	int(11)	รหัสร้าน	FK	Shops
list	int(11)	รหัสรายการ	FK	Lists
beautician	int(11)	รหัสช่าง	FK	Beauticians
date	int(11)	วันที่		
time	time	เวลา		

ตารางที่ 3.29: แสดงรายละเอียด Attribute ของ Entity Beauticians

Field	Type	Description	Key	Reference
id	int(11)	รหัสช่าง	PK	
name	varchar(50)	ชื่อ		
userID	int(11)	รหัสผู้ใช้งาน	FK	Users
shopID	int (11)	รหัสร้าน	FK	Shops

ตารางที่ 3.30: แสดงรายละเอียด Attribute ของ Entity Reviews

Field	Type	Description	Key	Reference
id	int(11)	รหัสรีวิว	PK	
topic	text	หัวข้อรีวิว		
message	text	ข้อความรีวิว		
point	double	คะแนน		
shop	int(11)	รหัสร้าน	FK	Shops
userID	int(11)	เบอร์โทร	FK	Users

ตารางที่ 3.31: แสดงรายละเอียด Attribute ของ Entity Userimages

Field	Type	Description	Key	Reference
image$_id$	int(11)	รหัสรูปภาพ	PK	
user$_id$	int(11)	รหัสผู้ใช้งาน	PK	
image	longblob	รูปภาพ ผู้ ใช้งาน		
type	varchar(50)	ประเภทรูปภาพ		

ตารางที่ 3.32: แสดงรายละเอียด Attribute ของ Entity Shopimages

Field	Type	Description	Key	Reference
image$_id$	int(11)	รหัสรูปภาพ	PK	
shop$_id$	int(11)	รหัสร้าน	PK	
image	longblob	รูปภาพร้าน		
type	varchar(50)	ประเภทรูปภาพ		

ตารางที่ 3.33: แสดงรายละเอียด Attribute ของ Entity Workimages

Field	Type	Description	Key	Reference
image$_id$	int(11)	รหัสรูปภาพ	PK	
shop$_id$	int(11)	รหัสร้าน	PK	
image	longblob	รูปภาพ ผลงานร้าน		
type	varchar(50)	ประเภทรูปภาพ		

ตารางที่ 3.34: แสดงรายละเอียด Attribute ของ Entity Beauticianimages

Field	Type	Description	Key	Reference
image$_id$	int(11)	รหัสรูปภาพ	PK	
beautician$_id$	int(11)	รหัสช่าง	PK	
image	longblob	รูปภาพ ผลงานช่าง		
type	varchar(50)	ประเภทรูปภาพ		

บทที่ 4

การพัฒนาระบบ

หลังจากที่ได้มีการเตรียมความพร้อมสำหรับการพัฒนาในด้านต่าง ไม่ว่าจะเป็นที่มาและความสำคัญของปัญหา เทคโนโลยีที่มีความเหมาะสมกับระบบ และการออกแบบระบบการทำงาน รวมไปถึงโครงสร้างของข้อมูล ในบทนี้จะเป็นการพูดถึงการสร้างระบบที่ได้มีการออกแบบไว้ในบทที่แล้วจะถูกนำเสนอในบทนี้ โดยการพัฒนาระบบแบ่งได้เป็นส่วนต่าง ๆ ดังนี้

4.1 การพัฒนาเว็บแอปพลิเคชัน

การพัฒนาระบบจองคิวร้านเสริมสวยสำหรับเว็บแอปพลิเคชันนั้นวัตถุประสงค์หลังเพื่อสร้างความสะดวกต่อการกำงานของผู้ใช้บริการและเจ้าของร้านอันเนื่องมาจากข้อจำกัดบางประการหากใช้ระบบทำงานบนอุปกรณ์สมาร์ทโฟนเพียงอย่างเดียว โดยตัวเว็บแอปพลิเคชันนี้ถูกพัฒนาขึ้นด้วย React.js มีรายละเอียดการทำงานดังนี้

4.1.1 การเชื่อมต่อ Cloud SQLstore

bsection, ในการเชื่อมต่อเว็บแอปพลิเคชันกับ mySQL เพื่อใช้บริการต่างๆ ของ mySQL ทำได้ดังนี้

```
1 module.exports = {
2   hostname: 'localhost',
3   username: 'piyaphorn',
4   password: 'zxcasdqwe',
5   database_name: 'b_beauty_project',
6   JWT_SECRET: 'benz'
7 }
```

รูปที่ 4.1: ไฟล์ config.js

จากภาพที่ 4.1 โครงสร้างของไฟล์ config.js สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการส่งออกโมดูลเพื่อใช้งานในไฟล์อื่น
- บรรทัดที่ 2 - 6 เป็นการตั้งค่าระบุตัวตนเพื่อใช้งาน my SQL

```
1  const Sequelize = require('sequelize');
2  const config = require('./config');
3  const db = {};
4
5  const connect = new Sequelize(config.database_name,
     config.username, config.password, {
6        host: config.hostname,
7        dialect: 'mysql',
8        operatorsAliases: false,
9              pool: {
10                   max: 5,
11                   min: 0,
12                   acquire: 30000,
13                   idle: 10000
14             }
15 })
16 db.sequelize = connect
17 db.Sequelize = Sequelize
18
19 module.exports = db
```

รูปที่ 4.2: ไฟล์ db.js

จากภาพที่ 4.2 โครงสร้างของไฟล์db.js สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการเรียกใช้งาน package
- บรรทัดที่ 2 เป็นการนำเข้าโมดูลเพื่อใช้งาน เป็นการนำเข้าโมดูลตั้งค่าที่ได้จากรูปภาพ 4.1
- บรรทัดที่ 5 - 17 เป็นการตั้งค่า host เพื่อนำไปใช้งาน
- บรรทัดที่ 19 เป็นการส่งออกโมดูลเพื่อใช้งานในไฟล์อื่น

```
1  const express = require("express");
2  const cors = require("cors");
3  const bodyParser = require("body-parser");
4  const router = require("./routers");
5
6  const port = process.env.PORT || 9000;
7  const app = express();
8
9  app.use(cors());
10 app.use(bodyParser.json());
11 app.use(bodyParser.urlencoded({ extended: false }));
12 app.use("/api/auth", router.auth);
13 app.use("/api/list", router.list);
14 app.use("/api/shop", router.shop);
15 app.use("/api/review", router.review);
16 app.use("/api/booking", router.booking);
17 app.use("/api/storage", router.storage);
18 app.use("/api/storageprofile", router.storageprofile);
19 app.use("/api/storageshop", router.storageshop);
20 app.use("/api/storagebeautician", router.
     storagebeautician)
21 app.listen(port, () => {
22 console.log("Express server listening on port " +
     port);
23 });
```

รูปที่ 4.3: ไฟล์ server.js

จากภาพที่ 4.3 โครงสร้างของไฟล์ server.js สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 - 3 เป็นการเรียกใช้งาน package
- บรรทัดที่ 4 เป็นการเรียกใช้งานไฟล์ routers
- บรรทัดที่ 6 เป็นการกำหนด port เพื่อใช้เชื่อ api ระหว่าง backend กับ fontend
- บรรทัดที่ 7 เป็นการประกาศตัวแปล เพื่อเรียกใช้งาน package ตามบรรทัดที่ 1

4.1.2 โครงสร้างของการสร้างหน้าเข้าสู่ระบบ

```
1
2  <div>
3   <TextField className={classes.margin}
4    id="email"
5    name="email"
6    value={email}
7    label="E_mail"
8    variant="outlined"
9    InputProps={{
10   startAdornment: (
11    <InputAdornment position="start">
12     <MailOutlineIcon />
13    </InputAdornment>
14   )
15  }}
16    onChange={this.handleChange}/>
17 </div>
18 <div>
19  <TextFieldclassName={classes.margin}
20   id="password"
21   name="password"
22   value={password}
23   label="Password"
24   type="password"
25   autoComplete="current-password"
26   variant="outlined"
27   InputProps={{
28   startAdornment: (
29   <InputAdornment position="start">
30    <VpnKeyIcon />
31   </InputAdornment>)
32  }}
33   onChange={this.handleChange}/>
34 </div>
35 <div>
36  <Button variant="contained" color="primary"
37   disableElevation>เข้าสู่ระบบ
38
39  </Button>
40 </div>
```

รูปที่ 4.4: การสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าเข้าสู่ระบบ loginPage.js

จากภาพที่ 4.4 โครงสร้างของการสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าเข้าสู่ระบบ สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 2-17 เป็นสร้างช่องกรอกข้อมูลอีเมล (e-mail) จากผู้ใช้
- บรรทัดที่ 18-34 เป็นสร้างช่องกรอกข้อมูลรหัสผ่าน (password) จากผู้ใช้
- บรรทัดที่ 34-40 สร้างปุ่มเข้าสู่ระบบ

```
1
2  axios
3  .post("http://localhost:9000/api/auth/login", {
4  email: email,
5  password: password
6  })
7  .then(async response => {
8  const token = response.data.token;
9  localStorage.setItem("token", token);
10 const user = await axios.get(
11 "http://localhost:9000/api/auth/currentUser",
12 { headers: { token: token } }
13 );
14 if (user) {
15 this.props.setUser(user.data);
16 if (user.data.role === 0) {
17 this.props.history.push("/");
18 } else if (user.data.role === 1) {
19 this.props.history.push("/DataShopPage");
20 } else {
21 this.props.history.push("/BeauticianShopPage");
22 }
23 }
24 })
25 .catch(error => {
26 console.log(error);
27 alert("e-mail หรือรหัสผ่านไม่ถูกต้อง");
28 });
```

รูปที่ 4.5: การสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าเข้าสู่ระบบ LoginPage.js

จากภาพที่ 4.5 โครงสร้างของการสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าเข้าสู่ระบบ สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 2-6 เป็นการเรียกใช้ mysql พร้อมส่งค่า email และ password เพื่อทำการเข้าสู่ระบบ
- บรรทัดที่ 7-13 เป็นการกำหนดกที่ใช้ในการตรวจสอบความถูกต้องของอีเมลและรหัสผ่าน
- บรรทัดที่ 14-24 เป็นการเช็คบทบาทของ user
- บรรทัดที่ 25-28 เป็นการแจ้งเตือน error

```
1  exports.login = async (req, res) => {
2    const loginData = req.body;
3    const user = await User.findOne({
4      where: { email: loginData.email }
5    });
6    if (!user) {
7      return res.status(400).json({ message: "email
           หรือรหัสผ่านไม่ถูก
           ต้อง" });
8    } else {
9    const chkPassword = await bcryptjs.compare(
10     loginData.password,
11     user.dataValues.password
12   );
13   if (chkPassword === false) {
14     return res.status(400).json({ message: รหัสผ่านไม่ถูกต้อง"" });
15   }
16   const token = await createToken(user.dataValues.id);
17     res.status(200).send({token});
18   }
19 };
```

รูปที่ 4.6: การสร้างลอจิก (logic) ของหน้าเข้าสู่ระบบ authcontroller.js

จากภาพที่ 4.6 โครงสร้างลอจิกของหน้าเข้าสู่ระบบ สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เรียกใช้ฟังก์ชันอื่นเพื่อทำการอัพเดทสถานะการเข้าสู่ระบบ
- บรรทัดที่ 2 สร้างตัวแปลเพื่อใช้เก็บข้อมูลเข้าสู่ระบบ
- บรรทัดที่ 3-5 เป็นการเช็คข้อมูลการเข้าสู่ระบบ
- บรรทัดที่ 7-15 เป็นการอัพเดทสถานะเมื่อเข้าสู่ระบบไม่สำเร็จ
- บรรทัดที่ 16-19 เป็นการอัพเดทสถานะเมื่อเข้าสู่ระบบสำเร็จ

4.1.3 โครงสร้างของการสร้างหน้าหลัก

```
componentDidMount = async () => {
 let { pathname } = this.props.location;
 const response = await axios.get("http://localhost
    :9000/api/shop/all");
  for (let i = 0; i < response.data.length; i++) {
    const resposeimgshop = await axios.get(`http://
       localhost:9000/api/storage/file/shop/${response
       .data[i].id}`)
    response.data[i].image = ('data:' +
       resposeimgshop.data.type+';base64,'+
       resposeimgshop.data.url)}
   console.log("response", response.data);
   this.setState({shops: response.data,});

};
```

รูปที่ 4.7: การสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าข่าวสาร HomePage.JS

จากภาพที่ 4.7 โครงสร้างของการสร้างหน้าจอข้อมูลร้าน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 และ 10 เป็นการกำหนดฟังก์ชันให้ข้อมูลแสดงเวลาเปิดระบบ
- บรรทัดที่ 2 เป็นการเก็บตัวใน props
- บรรทัดที่ 3 เป็นการเรียกใช้ mysql พร้อมดึงข้อมูลร้าน
- บรรทัดที่ 4-7 เป็นการเรียกใช้ mysql พร้อมดึงข้อมูลรูปภาพร้าน
- บรรทัดที่ 8 เป็นการ set ค่าใน State

```
1 exports.getShop = async (req, res) => {
2       try {
3             let shop = await Shop.findAll();
4             res.status(200).send(shop);
5       } catch (err) {
6             console.log(err);
7             res.sendStatus(401);
8       }
9 };
```

รูปที่ 4.8: การสร้างลอจิก(logic)ของหน้าข่าวสาร Shopcontroller.js

จากภาพที่ 4.8 โครงสร้างลอจิกของหน้าข่าวสาร สามารถอธิบายการทำงานได้ดังนี้

– บรรทัดที่ 1-9 เป็นการสร้างฟังก์ชันเพื่อดึงข้อมูลร้านทั้งหมดในฐานข้อมูลมาแสดง

4.1.4 โครงสร้างของการสร้างหน้าดูรายละเอียดร้าน

```
componentDidMount = async () => {
      const { location, userInfo } = this.props;
      const id = location.state.id;
      const response1 = await axios.get(
      'http://localhost:9000/api/shop/getShopId/${id
         }'
      );
      const response2 = await axios.get(
      'http://localhost:9000/api/list/getListshop/${
         id}'
      );
      console.log("response", response1.data);
      console.log("response", response2.data);
      let { timeopen, timeclose } = response1.data;
      timeopen = timeopen.split(":");
      timeclose = timeclose.split(":");
      let dateClose = new Date();
      dateClose.setHours(parseInt(timeclose[0]));
      dateClose.setMinutes(parseInt(timeclose[1]));
      const result = this.calculate(moment(dateClose
         ), timeopen);

      this.setState({
            shop: response1.data,
            data: result,
            datalist: response2.data
      });

};
```

รูปที่ 4.9: การสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้ารายละเอียดร้าน ShopPage.js

จากภาพที่ 4.9 โครงสร้างของการสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าดูรายละเอียดข่าวสาร สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 และ 26 เป็นการสร้างฟังก์ชันเพื่อให้ข้อมูลแสดงเวลาเปิดหน้า ShopPage
- บรรทัดที่ 2 เป็นการประกาศตัวแปลใน props
- บรรทัดที่ 3 สร้างตัวแปล id เพื่อมาเก็บข้อมูล id จาก location.props
- บรรทัดที่ 5-6 เป็นการดึงข้อมูลร้านจากฐานข้อมูลโดยดึงตาม id ร้าน
- บรรทัดที่ 7-9 เป็นการดึงข้อมูลรายการจากฐานข้อมูลโดยดึงตาม id ร้าน
- บรรทัดที่ 10-11 เป็นการ log ข้อมูลมาดูว่าแสดงตามที่ต้องการหรือไม่
- บรรทัดที่ 12-19 เป็นการกำหนดช่วงเวลาที่จะให้แสดง
- บรรทัดที่ 20-24 เป็นการ set ค่าใน state

```
1  exports.getShopId = async (req, res) => {
2        try {
3              const shopId = req.params.id;
4              let shop = await Shop.findOne({
5                    where: {
6                    id: shopId,
7                    },
8              });
9        res.status(200).send(shop);
10       } catch (err) {
11             console.log(err);
12             res.sendStatus(401);
13       }
14 };
```

รูปที่ 4.10: การสร้างลอจิกของหน้าดูรายละเอียดของร้าน shopcontroller.js

จากภาพที่ 4.10 โครงสร้างลอจิกของหน้าดูรายละเอียดของร้าน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการส่งออกเพื่อใช้งานในไฟล์อื่น
- บรรทัดที่ 2-9 เป็นการกำหนดค่าข้อมูลเพื่อให้ข้อมูลแสดงตาม id ร้าน
- บรรทัดที่ 10-13 เป็นการกำหนด error

4.1.5 โครงสร้างของการสร้างหน้าโปรไฟล์

```
1  componentDidMount = async () => {
2        const { location, userInfo } = this.props;
3        const response = await axios.get(
4        'http://localhost:9000/api/auth/getUserById/${
           userInfo.id}' );
5        const resposeimguser = await axios.get('http
           ://localhost:9000/api/storageprofile/file/
           user/${userInfo.id}')
6        response.data.image = ('data:' +
           resposeimguser.data.type+';base64,'+
           resposeimguser.data.url)
7        this.setState({
8        user: response.data,
9        });
10 };
```

รูปที่ 4.11: การสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าโปรไฟล์ Profile.js

จากภาพที่ 4.11 โครงสร้างของการสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าโปรไฟล์ สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างฟังก์ชันเพื่อให้แสดงข้อมูลเวลาเปิดหน้าโปรไฟล์
- บรรทัดที่ 2 เป็นการประกาศตัวแปลใน props
- บรรทัดที่ 3-4 เป็นการดึงข้อมูล user จากฐานข้อมูลโดยดึงตาม id user
- บรรทัดที่ 5-6 เป็นการดึงข้อมูล imageuser จากฐานข้อมูลโดยดึงตาม id user
- บรรทัดที่ 7-9 เป็นการ set ค่า ใน state

```
1  exports.getUserById = async (req, res) => {
2        try {
3              console.log("req.param", req.param);
4              const userId = req.params.id;
5              const user = await User.findOne({
6                    where: { id: userId }
7              });
8              res.status(200).send(user);
9        } catch (err) {
10             console.log(err);
11             res.sendStatus(401);
12       }
13 };
```

รูปที่ 4.12: การสร้างลอจิกของหน้าโปรไฟล์ authcontroller.js

จากภาพที่ 4.12 โครงสร้างลอจิกของหน้าโปรไฟล์ สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 การส่งออกข้อมูลเพื่อไปใช้ในไฟล์อื่น
- บรรทัดที่ 2-8 คำสั่งที่ใช้ดึงข้อมูล user จากฐานข้อมูลตาก id user
- บรรทัดที่ 9-12 คำสั่งแสดง error เมื่อไม่พบข้อมูล

4.1.6 โครงสร้างของการสร้างหน้าเพิ่มข้อมูลร้าน

```
<div className="row center">>
 <img src={shop.image} width="60%"/>
 <div className="row center">
  <Button variant="outlined" color="primary">เพิ่มรูปภาพร้าน

  </Button>
</div>
<div className="col s12 m12 l12">
 <div ClassName="row">
  <TextField
   className={classes.margin} id="name" name="name" value={shop.name} labelชื่อร้าน="*" variant="outlined" />
  <TextField
   className={classes.margin} id="nameeng" name="nameeng"
   value={shop.nameeng}label="Name*" variant="outlined" />
<div className="row">
  <Button variant="contained" color="secondary">ยกเลิก

  </Button>{" "}
  <Button variant="contained" color="primary"
   onClick={() => {
    axios
    .post("http://localhost:9000/api/shop/add", {
      ...shop,
    userId: userInfo.id
  })
  .then(response => {
    console.logเพิ่มข้อมูลร้านสำเร็จ("", response);
    alertกรอกข้อมูลไม่ครบ("");
   })
  .catch(error => {
    console.log(error);
  });
  }}
  > ตกลง</Button>
</div>
```

รูปที่ 4.13: การสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าเพิ่มข้อมูลร้าน DataShop.js

จากภาพที่ 4.13 โครงสร้างของการสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าเพิ่มข้อมูลร้าน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-2 แสดงรูปภาพ
- บรรทัดที่ 3-7 ปุ่มเพิ่มรูปภาพ
- บรรทัดที่ 8-15 ฟอร์มเพื่อกรอกข้อมูลร้าน
- บรรทัดที่ 16-18 ปุ่มยกเลิก
- บรรทัดที่ 19-35 ปุ่มตกลง เมื่อกดปุ่มระบบจะบันทึกข้อมูลลงฐานข้อมูล

```
1  exports.addShop = async (req, res) => {
2        try {
3              const addshopData = req.body;
4              console.log(addshopData);
5              await Shop.create(addshopData);
6              res.send(”create”);
7        } catch (err) {
8              console.log(err);
9              res.sendStatus(401);
10       }
11 };
```

รูปที่ 4.14: การสร้างลอจิกของหน้าเพิ่มข้อมูล shopcontroller.js

จากภาพที่ 4.14 โครงสร้างลอจิกของหน้าเพิ่มข้อมูลร้าน สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการส่งออกข้อมูลเพื่อใช้ในไฟล์อื่น
- บรรทัดที่ 2-6 เป็นคำสั่งที่ใช้เพิ่มข้อมูลลงฐานข้อมูล
- บรรทัดที่ 7-10 เป็นการแสดง error เมื่อเพิ่มข้อมูลไม่ได้

4.1.7 โครงสร้างของการสร้างหน้าเพิ่มรายการ

```
<div className="row">
 <MaterialTable titleรายการ="" columns={columns} data={data}
   options={{
    selection: false,
   }}
   editable={{
     onRowAdd: (newData) =>
      new Promise((resolve) => {
           resolve();
           this.setState((prevState) => {
        let data = [...prevState.data];
           data = [newData, ...data];
           return { ...prevState, data };
         })
       }),
      onRowUpdate: (newData, oldData) =>
        new Promise((resolve) => {
          setTimeout(() => {
           resolve();
            if (oldData) {
            this.setState((prevState) => {
            const data = [...prevState.data];
            data[data.indexOf(oldData)] = newData;
             return { ...prevState, data };
        });
        }
      }, 600);}),
    onRowDelete: (oldData) =>
        new Promise((resolve) => {
         setTimeout(() => {
         resolve();
         this.setState((prevState) => {
         const data = [...prevState.data];
         data.splice(data.indexOf(oldData), 1);
           return { ...prevState, data };
         }); }, 600); }),}} />
</div>
```

รูปที่ 4.15: การสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้ารายการ List.js

จากภาพที่ 4.15 โครงสร้างของการสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้ารายการ สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-5 เป็นการแสดงตาราง
- บรรทัดที่ 6-15 เป็นการเพิ่มข้อมูลในตาราง
- บรรทัดที่ 16-27 เป็นการแก้ไขในตาราง
- บรรทัดที่ 28-37 เป็นการลบข้อมูลในตาราง

```
1  exports.addlist = async (req, res) => {
2        try {
3              const addlistData = req.body;
4              await List.create({
5                    name: addlistData.name,
6                    price: addlistData.price,
7                    time: addlistData.time,
8                    shop: addlistData.shop,
9                    userId: addlistData.userId
10             });
11             res.send(”create”);
12       } catch (err) {
13             console.log(err);
14             res.sendStatus(401);
15       }
16 };
```

รูปที่ 4.16: การสร้างลอจิกของหน้ารายการ listcontroller.js

จากภาพที่ 4.16 โครงสร้างลอจิกของหน้ารายการ สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 การส่งออกข้อมูลเพื่อใช้ในไฟล์อื่น
- บรรทัดที่ 2-11 เป็นการเพิ่มข้อมูลลงในฐานข้อมูล
- บรรทัดที่ 12-15 เป็นคำสั่งแสดง error เมื่อไม่สามารถเพิ่มข้อมูลได้

4.1.8 โครงสร้างของการสร้างหน้าการจองคิว

```
1  <div className="row center">
2        <KeyboardDatePicker
3              disableToolbar
4              variant="inline"
5              format="dd/MM/yyyy"
6              margin="normal"
7              id="date-picker-inline"
8              labelเลือกวันที่=""
9              value={selectedDate}
10             onChange={this.handleDateChange}
11             KeyboardButtonProps={{
12             "aria-label": "change date"
13             }}
14       />
15 </div>
16 <div className="row">
17       <MaterialTable
18             titleคิวจอง=""
19             columns={columns}
20             data={data}
21             options={{
22             selection: false
23             }}
24       />
25 </div>
```

รูปที่ 4.17: การสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าดูคิวจอง Manageqeueue.js

จากภาพที่ 4.17 โครงสร้างของการสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าดูคิวจอง สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-15 เป็นการสร้างวันที่แบบคีบอร์ดเพื่อเลือกวันที่ในการแสดงข้อมูล
- บรรทัดที่ 16-25 ตารางแสดงข้อมูลจองคิว

```
1 exports.getBooking = async (req, res) => {
2       try {
3             let booking = await Booking.findAll();
4             res.status(200).send(”success”);
5       } catch (err) {
6             console.log(err);
7             res.sendStatus(401);
8       }
9 };
```

รูปที่ 4.18: การสร้างลอจิกของหน้าดูการจองคิว booking.js

จากภาพที่ 4.18 โครงสร้างลอจิกของหน้าดูการจองคิว สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการส่งออกข้อมูลเพื่อใช้ในไฟล์อื่น
- บรรทัดที่ 2-9 เป็นการดึงข้อมูลจากฐานข้มูล

4.1.9 โครงสร้างของการสร้างหน้าเพิ่มข้อมูลช่าง

```
<TextField
        className={classes.margin}
        id="tel"
        name="tel"
        value={tel}
        labelเบอร์โทร=""
        autoComplete="Phone number"
        variant="outlined"
        onChange={this.handleChange}
/>
<Button
        color="primary"
        autoFocus
        onClick={() => {
        if (password === password2) {
        const register = axios
        .post("http://localhost:9000/api/auth/register", {
                email: email,
                password: password,
                name: name,
                address: address,
                tel: tel,
                role: "2",
        })
        const Beautician = axios
        .post("http://localhost:9000/api/beautician/add", {
                email: email,
                name: name,
                address: address,
                tel: tel,
                shopID: shops.id
        })
        .then(response => {
        console.logสร้างผู้ใช้สำเร็จ("", response);
        this.handleClose();
        })
        .catch(error => {
        console.log(error);
        });
        } else {
                alert("password ไม่ถูกต้อง");
        }
        }}
        > ยืนยัน</Button>
```

รูปที่ 4.19: การสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าเพิ่มข้อมูลช่าง beautician.js

จากภาพที่ 4.19 โครงสร้างของการสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าเพิ่มข้อมูลช่าง สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-10 เป็นตัวอย่างการสร้าง form เพื่อกรอกข้อมูล
- บรรทัดที่ 11-44 เป็นปุ่มยืนยัน และเพิ่มข้อมูลลงฐานข้อมูล

```
1  exports.add = async (req, res) => {
2        try {
3              const addbeauticianData = req.body;
4              console.log(addbeauticianData);
5                    await Beautician.create({
6                    name: addbeauticianData.name,
7                    email: addbeauticianData.email,
8                    address: addbeauticianData.address
                        ,
9                    tel: addbeauticianData.tel,
10                   shopID: addbeauticianData.shopID
11       });
12             res.send(”create”);
13       } catch (err) {
14             console.log(err);
15             res.sendStatus(401);
16       }
17 };
```

รูปที่ 4.20: การสร้างลอจิกของหน้าสร้างกำหนดการเพิ่มข้อมูลช่าง beauticianconrolletr.js

จากภาพที่ 4.20 โครงสร้างลอจิกของหน้าสร้างกำหนดการเพิ่มข้อมูลช่าง สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการส่งออกไฟล์เพื่อใช้ในไฟล์อื่น
- บรรทัดที่ 2-17 เป็นคำสั่งที่ใช้ในการเพิ่มข้อมูลช่างลงฐานข้อมูล

4.1.10 โครงสร้างของการสร้างหน้าอัพโหลดรูปภาพ

```
<Dialog open={open} onClose={this.handleClose}
      aria-labelledby="alert-dialog-title"
      aria-describedby="alert-dialog-description" >
      <DialogTitle id="alert-dialog-titleเพิ่ม
          รูปภาพ">{""}</DialogTitle>
      <DialogContent>
      <TextField className={classes.margin}
      id="outlined-file-input" labelเพิ่มรูปภาพ=""
      type="file" autoComplete="current-password"
      variant="outlined"
      InputProps={{
      startAdornment: (
      <InputAdornment position="start">
      <AddPhotoAlternateIcon />
      </InputAdornment>),}}
      onChange={(e) => {
      this.setState({
      file: e.target.files[0],
      }); }} />
      </DialogContent>
      <DialogActions>
      <Button color="primary" autoFocus
      onClick={() => {
      const { location, userInfo } = this.props;
      const formData = new FormData();
      console.log(file);
      formData.append("images", file);
      formData.append("shop_id", shop.id);
      axios
      .post(
      `http://localhost:9000/api/storageshop/upload/work/${shop.id}`,
          formData)
      .then((response) => {
      console.logเพิ่มรูปภาพสำเร็จ("", response);
      alertเพิ่มรูปภาพสำเร็จ("");
      this.handleClose();
      })
      .catch((error) => {
      console.log(error);
      }); }} > ยืนยัน</Button>
</DialogActions> </Dialog>
```

รูปที่ 4.21: การสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าอัพโหลดรูปภาพ Addimage.js

จากภาพที่ 4.21 โครงสร้างของการสร้างหน้าจอส่วนติดต่อผู้ใช้ของหน้าอัพโหลดรูปภาพ สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-4 สร้าง Dialog เพื่อเพิ่มอัพโหลดรูปภาพ
- บรรทัดที่ 5-19 สร้าง form สำหรับอัพโหลดรูปภาพ
- บรรทัดที่ 20-39 เป็นปุ่มเพื่อเพิ่มข้อมูลลงฐานข้อมูล

```
1  const router = require('express-promise-router')();
2  const storageshop = require('../../multer');
3  const WorkImage = require('../../models/workimage');
4  const Shop = require('../../models/shop')
5  router.post('/upload/work/:shopId',storageshop.
     single('images'), async (req, res) => {
6        const images = req.file;
7        const shop_id = req.body.shopId
8              await WorkImage.create({
9                    shop_id: req.params.shopId,
10                   image: images.buffer,
11                   type: images.mimetype
12             });
13             res.send('success')
14 })
15 module.exports = router;
```

รูปที่ 4.22: การสร้างลอจิกของหน้าสร้างกำหนดการอัพโหลดรูปภาพ storageshop.js

จากภาพที่ 4.22 โครงสร้างลอจิกของหน้าสร้างกำหนดการอัพโหลดรูปภาพ สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการเรียกใช้ package express-promise-router
- บรรทัดที่ 2-4 เป็นการเรียกใช้ข้อมูลจากไฟล์อื่น
- บรรทัดที่ 5-14 เป็นคำสั่งอัพโหลดรูปภาพ
- บรรทัดที่ 15 เป็นการส่งออก module เพื่อใช้ในไฟล์อื่น

บทที่ 5

การทดสอบระบบ

การทดสอบการทำงานของเว็บไซต์ โดยทำการทดสอบในลักษณะ Black-box Testing [?] หรือ Data-Driven testing ซึ่งเป็นการเทสแบบที่ไม่สนใจโปรเซส (Process) การทำงานภายในของโปรแกรมว่าทำงานอย่างไร แต่จะเน้นไปที่ Input และ Result ที่ได้มากกว่าว่าการทำงานต่าง ๆ ถูกต้องตามความต้องการ (Requirement) หรือไม่ ซึ่งการทดสอบการใช้งานเว็บแอปพลิเคชันได้ผลดังนี้

5.1 การทดสอบการใช้งานเว็บแอปพลิเคชัน

- การทดสอบหน้ารายละเอียดร้าน ในการแสดงผลหน้าจอรายละเอียดร้านนั้นจะประกอบไปด้วย ข้อมูลร้าน แผนที่ร้าน รายการร้าน การจองคิว รีวิว ผลการทดสอบดังตารางที่ 5.1

ตารางที่ 5.1: ผลการทดสอบหน้ารายละเอียดร้าน

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้ารายละเอียดร้าน	กดที่รูปภาพร้าน	ระบบ แสดง ผล หน้า จอ ร้าน พร้อม ทั้ง แสดง รายการ ร้าน ทั้งหมด
	กดที่แผ่นที่	ระบบแสดงผลแผ่นที่
	กดปุ่มรายการ	ระบบแสดงผลรายการ
	กดปุ่มจองคิว	ระบบแสดงผลหน้าจอจองคิว
	กดปุ่มจเขียนรีวิว	ระบบแสดงผลหน้าจอเขียนรี-วิว

- การทดสอบหน้าเจ้าของร้าน ในการแสดงผลหน้าจอเจ้าของร้านนั้นจะประกอบไปด้วย โปร-ไฟล์ ข้อมูลร้าน รายการ ข้อมูลช่าง ดูการจองคิว เพิ่มรูปภาพ ผลการทดสอบดังตารางที่ 5.2

ตารางที่ 5.2: ผลการการทดสอบหน้าเจ้าของร้าน

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้าจอเจ้าของร้าน	กดปุ่มเมนูโปรไฟล์	ระบบ แสดง ผล หน้า จอ โปร-ไฟล์ ทั้งแสดงรูปภาพโปรไฟล์
	กดปุ่มเมนูข้อมูลร้าน	ระบบแสดงหน้าจอข้อมูลร้าน ทั้งหมด พร้แม ทั้ง แสดง รูป-ภาพร้าน
	กดปุ่มเมนูรายการ	ระบบแสดงหน้ารายการ
	กดปุ่มเมนูข้อมูลช่าง	ระบบแสดงหน้าข้อมูลช่าง
	กดปุ่มเมนูดูการจองคิว	ระบบ แสดง ผล หน้า จอ การจองคิว
	กดปุ่มเมนูเพิ่มรูปภาพ	ระบบ แสดง ผล หน้า จอ การเพิ่มรูปภาพ

- การทดสอบหน้าช่าง ในการแสดงผลหน้าจอเจ้าของร้านนั้นจะประกอบไปด้วย โปรไฟล์ ตารางงาน เพิ่มรูปภาพ ผลการทดสอบดังตารางที่ 5.3

ตารางที่ 5.3: ผลการการทดสอบหน้าเจ้าของร้าน

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้าจอเจ้าของร้าน	กดปุ่มเมนูโปรไฟล์	ระบบ แสดง ผล หน้า จอ โปร-ไฟล์ ทั้งแสดงรูปภาพโปรไฟล์
	กดปุ่มเมนูตารางงาน	ระบบ แสดง ผล หน้า จอ ตาราง งาน
	กดปุ่มเมนูเพิ่มรูปภาพ	ระบบ แสดง ผล หน้า จอ การ เพิ่มรูปภาพ

บทที่ 6

สรุปและข้อเสนอแนะ

การดำเนินโครงงานเพื่อพัฒนาระบบจองคิวร้านเสริมสวยนี้ พบว่าระบบสามารถทำงานได้ตามที่วิเคราะห์และออกแบบไว้ แต่ก็พบปัญหาและอุปสรรคระหว่างการพัฒนา ในบทนี้ผู้พัฒนาจึงขอสรุปความสามารถของระบบ ชี้แจงปัญหาและอุปสรรค พร้อมเสนอแนวทางในการพัฒนาระบบจองคิวร้านเสริมสวย ต่อ ตามลำดับ

6.1 สรุปความสามารถของระบบ

ระบบจองคิวร้านเสริมสวย เว็บแอปพลิเคชันสามารถสรุปความสามารถที่ระบบทำได้ดังนี้

6.1.1 เว็บแอปพลิเคชัน

ความสามารถหลักของเว็บแอปพลิเคชันนั้นเน้นสร้างความสะดวกต่อการจัดการเอกสารเรื่องข้อมูลต่างๆ ที่เกี่ยวข้องกับ ระบบจองคิวร้านเสริมสวย โดยแบ่งความสามารถของระบบตามประเภทของผู้ใช้งานดังนี้

1. เจ้าของร้าน
    - สามารถลงทะเบียนเข้าสู่ระบบด้วย Email ได้
    - สามารถดูคิวที่ผู้ใช้บริการได้ทำการจองคิวไว้
    - สามารถ post ภาพผลงานทั้งหมดของร้านได้
    - สามารถเพิ่ม แก้ไข และลบรายการให้บริการประจำร้านได้
    - สามารถเพิ่ม แก้ไข ข้อมูลร้านได้
    - สามารถเพิ่ม แก้ไข ข้อมูลตำแหน่งร้านได้
    - สามารถเพิ่ม ลบ ข้อมูลช่างได้
2. ช่างประจำร้าน
    - สามารถดูตารางการทำงานของตนเองได้
    - สามารถ post ภาพผลงานของตัวเองได้
    - สามารถแก้ไขข้อมูลส่วนตัวได้

3. ผู้ใช้บริการ
    - สามารถลงทะเบียนเข้าสู่ระบบด้วย Email ได้
    - สามารถค้นหาร้านเสริมสวยได้
    - สามารถจองคิวของร้านเสริมสวยได้
    - สามารถดูคิวว่างของร้านเสริมสวยได้
    - สามารถดูข้อมูลต่างๆของร้านเสริมสวยได้
    - สามารถดูตำแหน่งของทางร้านได้
    - สามารถดูผลงานของร้านได้
    - สามารถเขียนรีวิว ติชม ได้

6.2 ปัญหาและอุปสรรคในการพัฒนา

1. เนื่องจากครั้งแรกที่ใช้ lalavel framework ไม่ค่อยสอดคล้องกับระบบ
   แนวทางการแก้ไข : ทำการเปลี่ยน จาก lalavel framework เป็น React.js
2. เนื่องจากได้ทำการเปลี่ยน framework จึงทำให้ต้องเสียเวลา
   แนวทางการแก้ไข : เร่งศึกษา และเร่งทำ
3. โรค covit 19 ทำให้การเข้าพบอาจารย์ไม่สะดวก
   แนวทางการแก้ไข : พบทางออนไลน์แทนการเข้าพบแบบตัวต่อตัว

6.3 แนวทางการพัฒนาต่อ

1. การพัฒนาช่องทางการติดต่อ
2. เจ้าของร้านสามารถยืนยันการจองคิวได้
3. การพัฒนาเป็นแอปพลิเคชัน
4. การเพิ่มข้อมูลโปรดมชั่น
5. การค้นหาร้านตามวันและเวลาว่างของร้าน

บรรณานุกรม

[1] Phonbopit, C. (2558). React คืออะไร เริ่มต้นเขียน react [ออนไลน์]. สืบค้นเมื่อ 09 เมษายน 2562. จาก https://devahoy.com/posts/getting-started-with-reactjs .

[2] him aeng (2559). React life cycle [ออนไลน์]. สืบค้นเมื่อ 09 เมษายน 2562. จาก http://www.forminit.com/react-basic-07 .

[3] Ruangvivattanaroj, N. (2560). อธิบาย react lifecycle แต่ละอันมีหน้าที่อย่างไร [ออนไลน์]. สืบค้นเมื่อ 09 เมษายน 2562. จาก https://igokuz.com/อธิบาย-react-lifecycle-แต่ละอันมีหน้าที่อย่างไร-48e65c922af1 .

[4] M.D.Soft (ม.ป.ป.). Node.js คืออะไร [ออนไลน์]. สืบค้นเมื่อ 09 เมษายน 2562. จาก https://www.mdsoft.co.th/ความรู้/140-what-node-js.html .

[5] BeYourCyber (2556). เจาะลึกกับ node.js แบบเริ่มต้นทำความรู้จัก [ออนไลน์]. สืบค้นเมื่อ 09 เมษายน 2562. จาก http://meewebfree.com/site/nodejs/441-learn-about-node-js-with-basic-of-node-js .

[6] คีรีพัฒนานนท์, . (2559). node.js คืออะไร คือ programming language ที่เขียนด้วย javascript [ออนไลน์]. สืบค้นเมื่อ 09 เมษายน 2562. จาก https://beyourcyber.com/2016/node-js-is-programming-language-by-javascript/ .

[7] mindphp (2560). Javascript คืออะไร [ออนไลน์]. สืบค้นเมื่อ 09 เมษายน 2562. จาก http://www.mindphp.com/คู่มือ/73-คืออะไร/2187-java-javascript-คืออะไร.html .

[8] admin (2558). Javascript คืออะไร [ออนไลน์]. สืบค้นเมื่อ 09 เมษายน 2562. จาก https://nokhookstudio.com/javascript-คืออะไร .

[9] itgenius (2556). จุดเด่นและหลักการทำงาน mysql [ออนไลน์]. สืบค้นเมื่อ 08 มกราคม 2562. จาก https://www.itgenius.co.th/article/

[10] easyhostdomain (2557). Mysql มีความสำคัญอย่างไรกับเซิร์ฟเวอร์ [ออนไลน์]. สืบค้นเมื่อ 08 มกราคม 2562. จาก http://th.easyhostdomain.com/dedicated-servers/mysql.html .

[11] รู้จักกับ visual studio code (วิชวล สตูดิโอ โค้ด) โปรแกรมฟรีจากค่ายไมโครซอฟท์ [ออนไลน์]. สืบค้นเมื่อ เมษายน 2562. จาก https://www.mindphp.com/บทความ/microsoft/4829-visual-studio-code.html .

[12] google map api [ออนไลน์]. สืบค้นเมื่อ มีนาคม 2562. จาก https://cloud.google.com/maps-platform/ .

[13] gowabi [ออนไลน์]. สืบค้นเมื่อ เมษายน 2563. จาก https://www.gowabi.com .

[14] Kunchit Phiu-Nual. (2557). ความหมายและความสำคัญของ system architecture [ออนไลน์]. สืบค้นเมื่อ 12 พฤษภาคม 2561. จาก https://goo.gl/6ZhGQo .

ภาคผนวก

ภาคผนวก ก

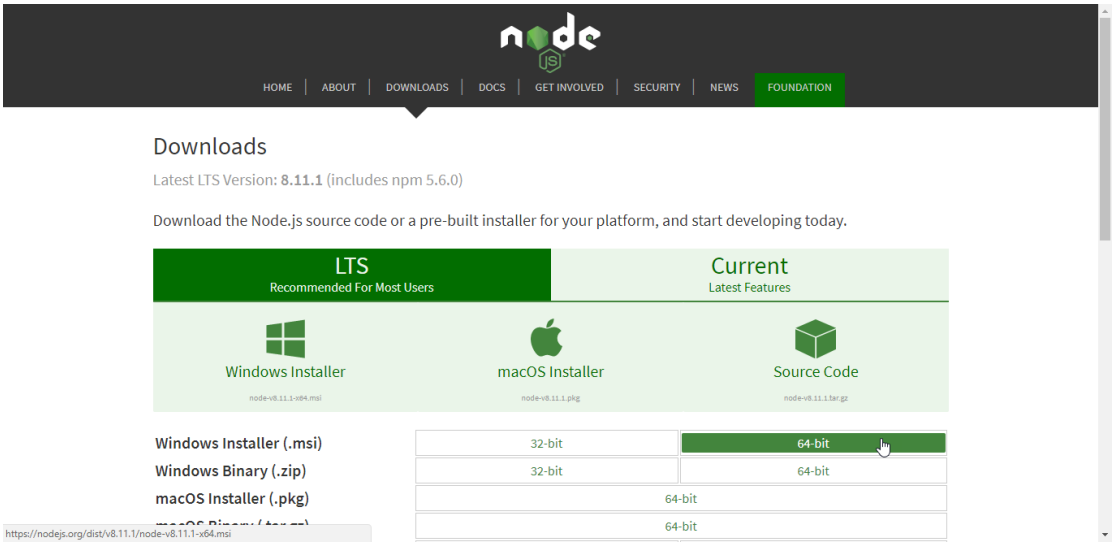
การติดตั้งเครื่องมือที่ใช้พัฒนาโปรแกรม

การติดตั้งเครื่องมือที่ใช้ในการพัฒนาระบบสหกิจศึกษา คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี มีโปรแกรมที่จำเป็นในการพัฒนาระบบดังต่อไปนี้

- การติดตั้ง Node.js
- การติดตั้ง React.js
- การติดตั้ง Visual Studio Code

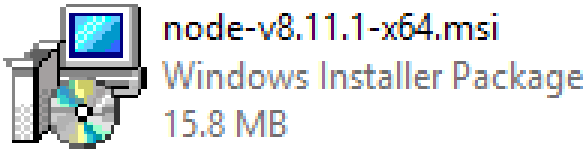
ก.1 การติดตั้ง Node.js

1) สามารถดาวน์โหลด Node.js ได้ที่ https://nodejs.org/en/download/ แสดงดังรูปที่ ก.1



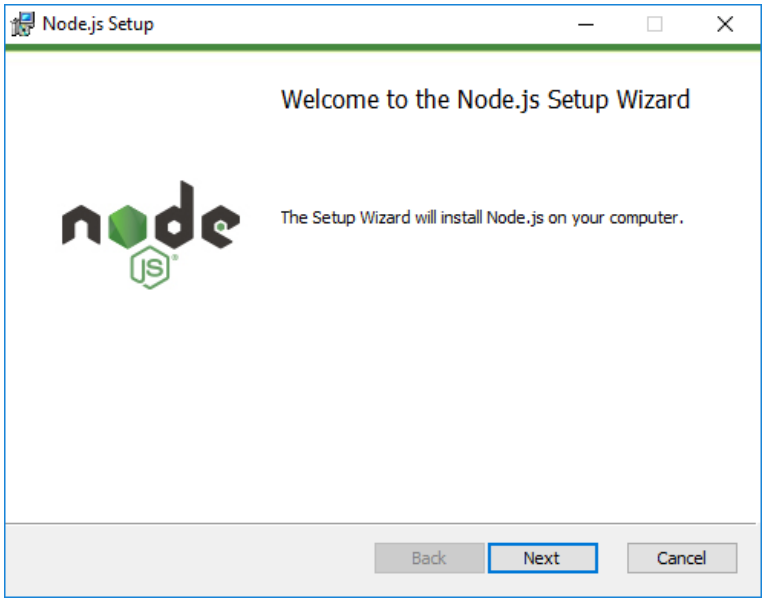
รูปที่ ก.1: หน้าเว็บดาวน์โหลด Node.js

2) เปิดไฟล์ติดตั้ง ชื่อ node-v10.15.3-x64.msi เพื่อติดตั้ง แสดงดังรูปที่ ก.2



รูปที่ ก.2: ไฟล์ติดตั้งสำหรับติดตั้ง Node.js

3) แสดงหน้าต่างตอนรับของ Node.js ให้กด Next แสดงดังรูปที่ ก.3

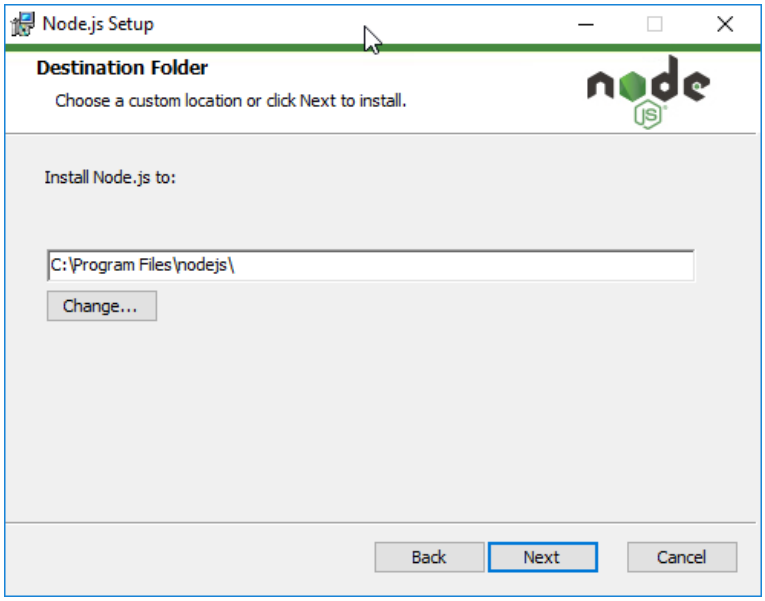


รูปที่ ก.3: หน้าต่างตอนรับของ Node.js

4) แสดงหน้าต่างข้อตกลงในการใช้ Node.js ให้เลือกช่อง I accept the terms in the License Agreement และกด Next แสดงดังรูปที่ ก.4

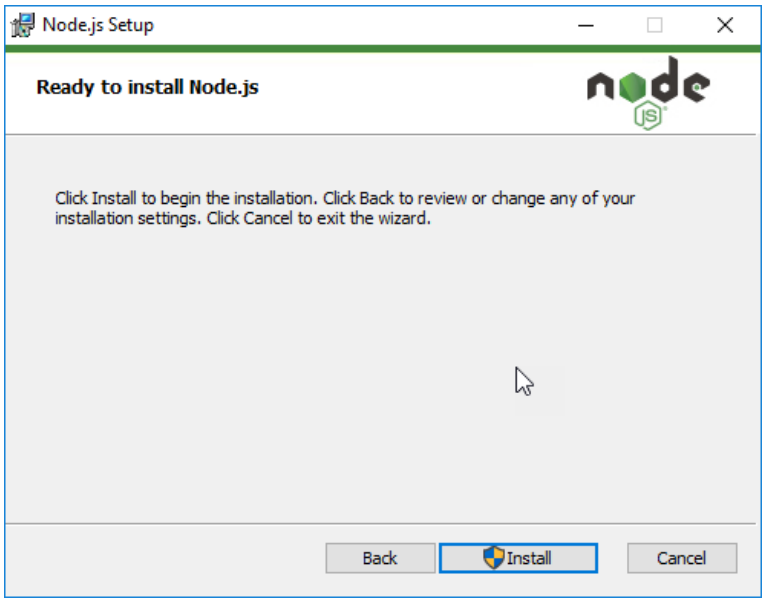
รูปที่ ก.4: หน้าต่างข้อตกลงในการใช้ Node.js

5) แสดงหน้าต่างเลือกโฟลเดอร์ที่จะทำการติดตั้ง แสดงดังรูปที่ ก.5



รูปที่ ก.5: หน้าต่างเลือกโฟลเดอร์ที่จะทำการติดตั้ง Node.js

6) แสดงหน้าต่างสำหรับติดตั้ง Node.js ให้กด Install เพื่อทำงานติดตั้ง แสดงดังรูปที่ ก.6



รูปที่ ก.6: หน้าต่างติดตั้ง Node.js

ก.2 การติดตั้ง React.js

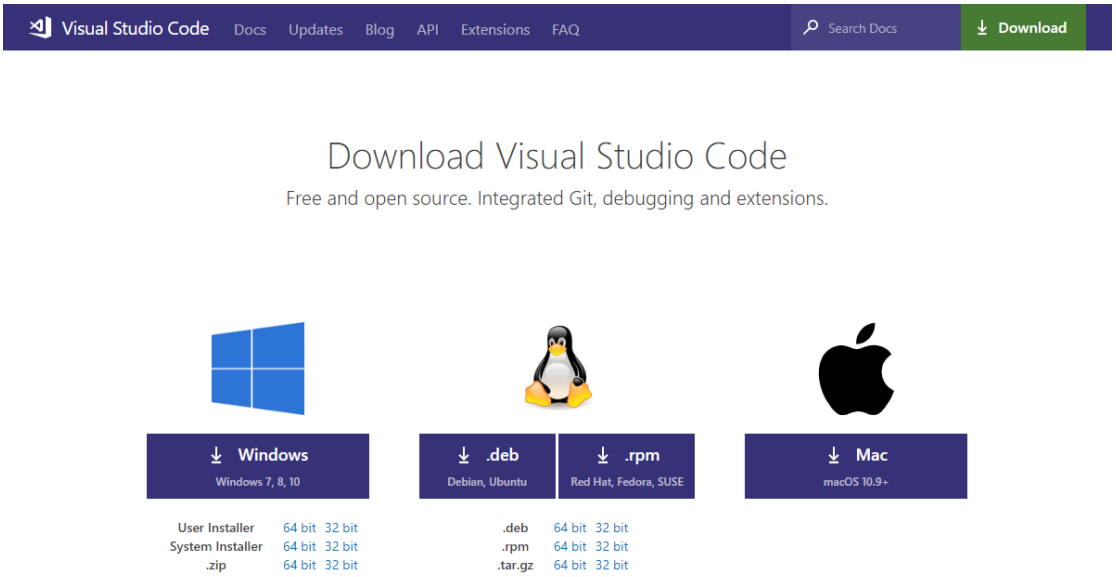
การติดตั้ง React.js สามารถทำผ่านคำสั่ง command line ได้ แสดงดังรูปที่ ก.7

```
npm install create-react-app --save
```

รูปที่ ก.7: คำสั่งสำหรับติดตั้ง React.js

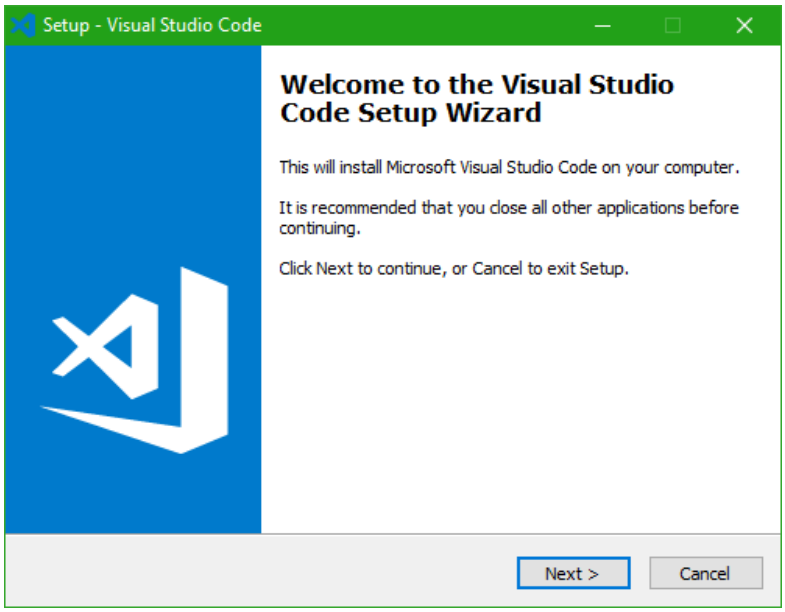
ก.3 การติดตั้ง Visual Studio Code

1. สามารถดาวน์โหลด Visual Studio Code ได้ที่ https://code.visualstudio.com/download ดังแสดงในรูปที่ ก.8



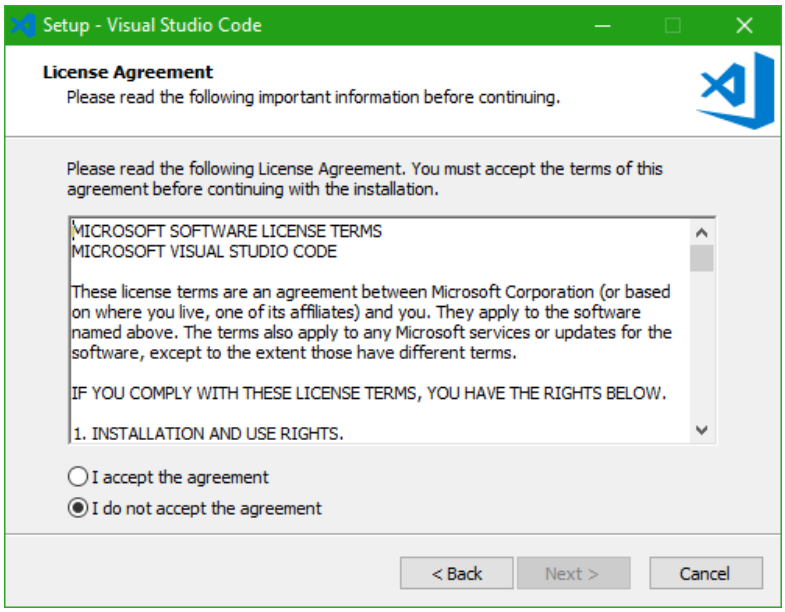
รูปที่ ก.8: หน้าเว็บดาวน์โหลด Visual Studio Code

2. เมื่อเปิดตัวติดตั้งขึ้นมาแล้ว จะแสดงหน้าจอ Welcome to the Visual Studio Code Setup Wizard ให้กดปุ่ม Next เพื่อเริ่มกระบวนการติดตั้ง ดังแสดงในรูปที่ ก.9



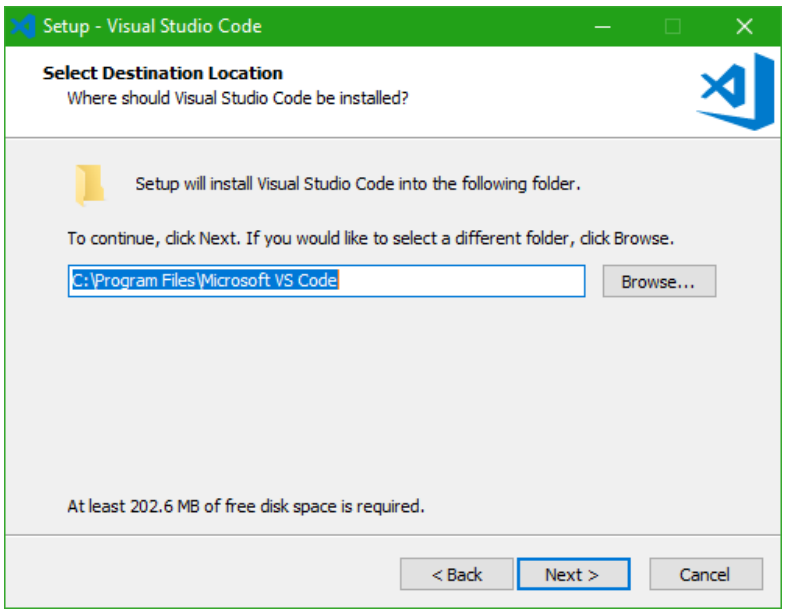
รูปที่ ก.9: หน้าต่างต้อนรับของ Visual Studio Code

3. หลังจากนั้นจะแสดงหน้าต่างข้อตกลงการใช้งาน Visual Studio Code ทำการติ๊กที่ I accept the areement แล้วกด Next ดังแสดงในรูปที่ ก.10



รูปที่ ก.10: หน้าต่างข้อตกลงการใช้งาน Visual Studio Code

4. จากนั้นจะแสดงหน้าต่างที่จัดเก็บไฟล์ต่างๆ ของ Visual Studio Code ทำการกด Next ดังแสดงในรูปที่ ก.11

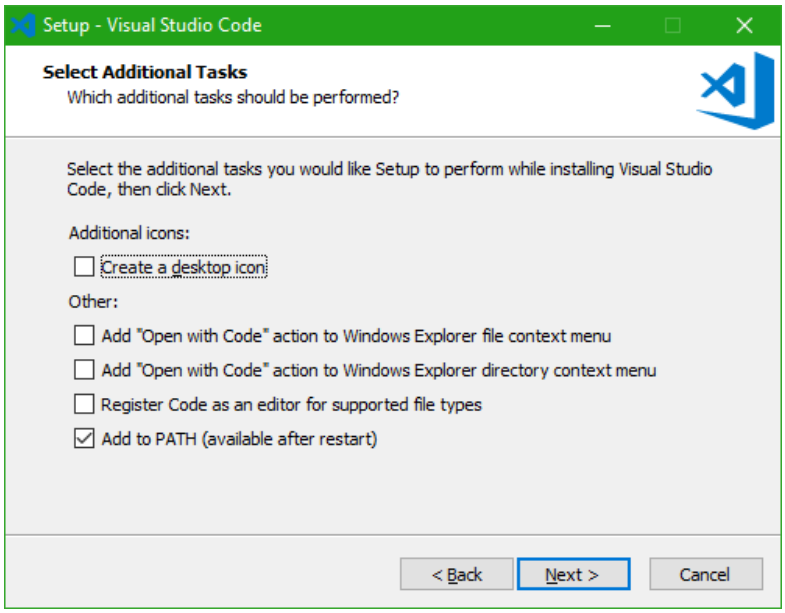


รูปที่ ก.11: หน้าต่างที่จัดเก็บไฟล์ต่างๆ ของ Visual Studio Code

5. จากนั้นจะแสดงหน้าต่างการจัดการซอร์ดคัทของ Visual Studio Code ทำการกด Next ดังแสดงในรูปที่ ก.12

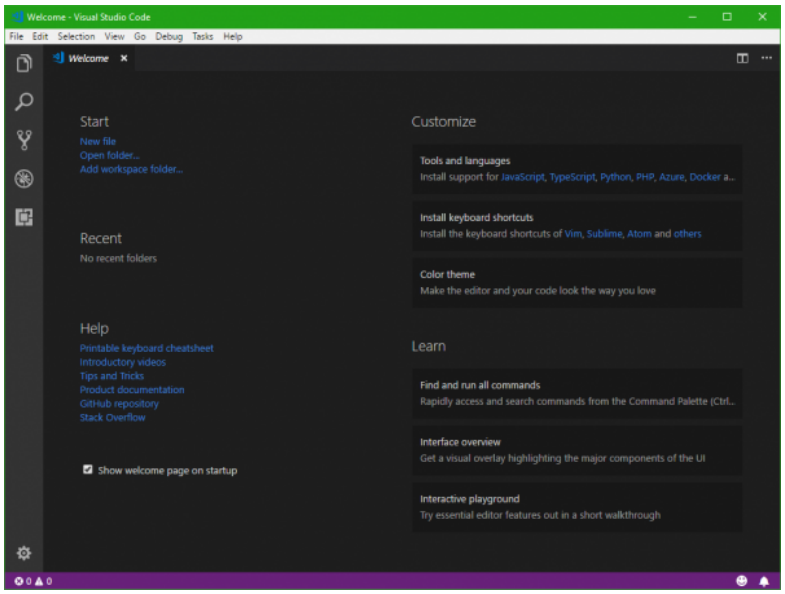
รูปที่ ก.12: หน้าต่างการจัดการซอร์ดคัท ของ Visual Studio Code

6. จากนั้นแสดงหน้าต่างเริ่มทำการติดตั้งทำการกด Next ดังแสดงในรูปที่ ก.13



รูปที่ ก.13: หน้าต่างเริ่มทำการติดตั้งทำการกด ของ Visual Studio Code

7. จากนั้นจะแสดงหน้าต่างเมื่อเข้าโปรแกรมหลังติตั้งเสร็จ ดังแสดงในรูปที่ ก.14



รูปที่ ก.14: หน้าต่างเมื่อเข้าโปรแกรมหลังติตั้งเสร็จ ของ Visual Studio Code

ภาคผนวก ข

คู่มือการติดตั้งระบบ

ในการติดตั้งเพื่อใช้งานเว็บแอปพลิเคชันระบบจองคิวร้านเสริมสวยสามารถทำได้โดยมีขั้นตอนดังนี้

1. สามารถดาวน์โหลด และติดตั้งระบบ ได้ที่ http://projectcs.sci.ubu.ac.th/senior-prj-62/59110440259.git ดังแสดงในรูปที่ ข.1 และรูปที่ ข.2

```
git clone http://projectcs.sci.ubu.ac.th
   /senior-prj-62/59110440259.git
```

รูปที่ ข.1: คำสั่งดาวน์โหลด

```
npm install หรือnpm i
```

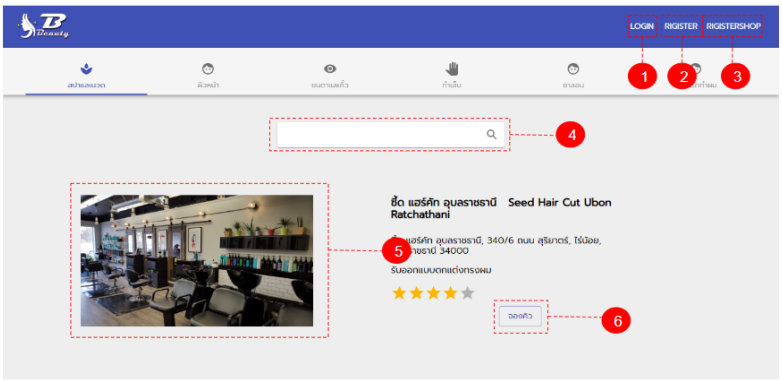
รูปที่ ข.2: คำสั่งติดตั้ง

ภาคผนวก ค

คู่มือการใช้งานระบบ

คู่มือการใช้งานทั้งหมดของระบบ สามารถแบ่งออกเป็น 3 ส่วน ดังนี้

1. ส่วนของหน้าเมนูเว็บแอปพลิเคชันสำหรับผู้ใช้บริการ
   - หน้าจอหลักแสดงผลทุกครั้งเมื่อผู้ใช้ทำการเปิดใช้งานเว็บแอปพลิเคชัน ดังแสดงในรูปที่ ค.1

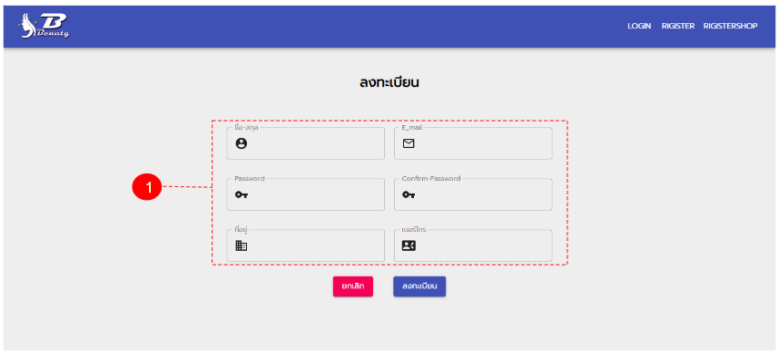


รูปที่ ค.1: หน้าจอหลัก

จากรูปที่ ค.2 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ ปุ่มเข้าสู่ระบบ
- หมายเลข 2 คือ ปุ่มลงทะเบียนสำหรับผู้ใช้งานทั่วไป
- หมายเลข 3 คือ ปุ่มลงทะเบียนสำหรับเจ้าของร้าน
- หมายเลข 4 คือ ฟอร์มสำหรับค้นหาร้าน
- หมายเลข 5 คือ คลิ๊กที่รูปภาพเพื่อนเปิดหน้าต่างรายละเอียดร้าน
- หมายเลข 6 คือ ปุ่มสำหรับจองคิว

   - เมื่อผู้ใช้งานกดที่ปุ่ม Register ระบบจะทำการแสดงหน้าต่างลงทะเบียน ดังแสดงในรูปที่ ค.2



รูปที่ ค.2: หน้าต่างลงทะเบียน

จากรูปที่ ค.2 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ กรอกข้อมูลในการลงทะเบียนแล้วกดยืนยัน

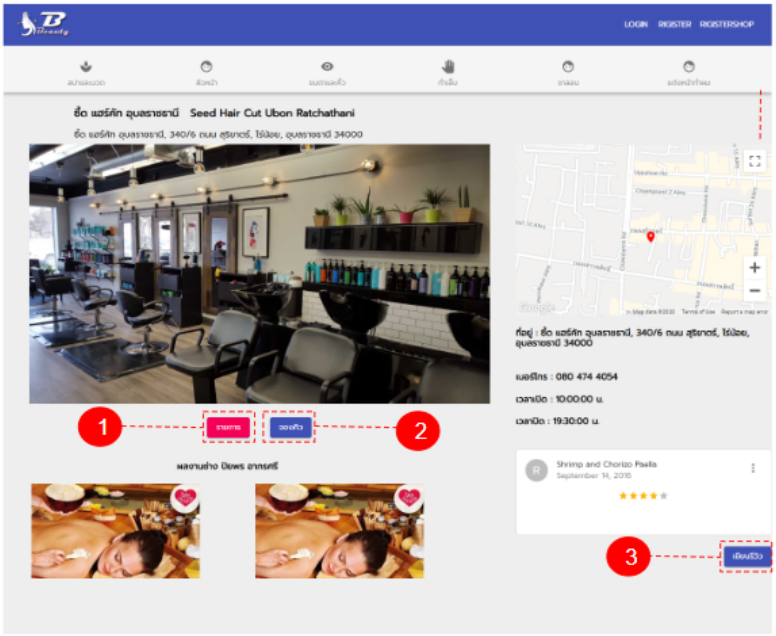
- ระบบทำการตรวจสอบทุกครั้งเมื่อผู้ใช้งานกดปุ่มเข้าสู่ระบบ ระบบจะแสดงหน้าจอเข้าสู่ระบบโดยผู้ใช้งานจำเป็นต้องทำการกรอกข้อมูลคือ อีเมลและรหัสผ่าน ดังแสดงในรูปที่ ค.3

รูปที่ ค.3: หน้าจอเข้าสู่ระบบ

จากรูปที่ ค.3 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ ส่วนของฟอร์มในการกรอกข้อมูลอีเมลและรหัสผ่านและ ปุ่มกดเข้าสู่ระบบ

- หน้าแสดงรายละเอียดข้อมูลร้าน ดังแสดงในรูปที่ ค.4

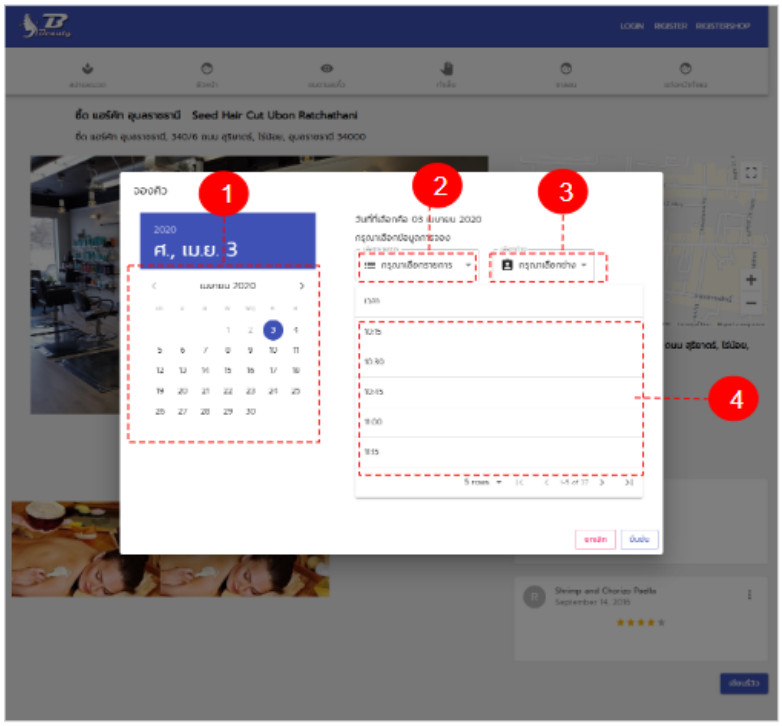


รูปที่ ค.4: หน้าแสดงรายระเอียดร้าน

จากรูปที่ ค.4 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ ปุ่มสำหรับดูรายการร้าน
- หมายเลข 2 คือ ปุ่มสำหรับจองคิว
- หมายเลข 3 คือ ปุ่มสำหรับเขียนรีวิว

• เมื่อผู้ใช้กดปุ่มจองคิว ระบบจะแสดง model จองคิว ดังแสดงในรูปที่ ค.5

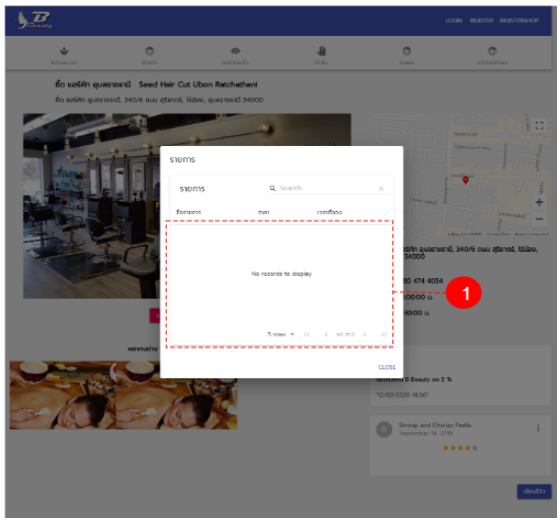


รูปที่ ค.5: หน้าการจองคิว

จากรูปที่ ค.5 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ สำหรับเลือกวันที่ในการจองคิว
- หมายเลข 2 คือ เลือกรายการสำหรับจองคิว
- หมายเลข 3 คือ เลือกช่างที่ต้องการจองคิว
- หมายเลข 4 คือ เลือกเวลาในการจองคิว

• เมื่อผู้ใช้กดปุ่มรายการ ระบบจะแสดง model รายการ ดังแสดงในรูปที่ ค.6

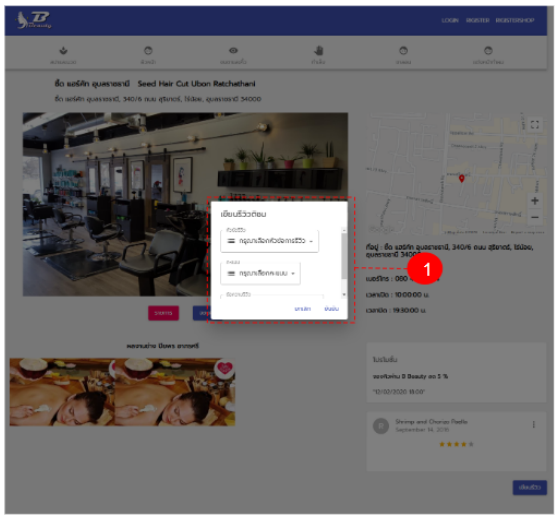


รูปที่ ค.6: เมนูดูรายการร้าน

จากรูปที่ ค.6 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ รายการประจำร้าน

- เมื่อผู้ใช้กดปุ่มเขียนรีวิว ระบบจะแสดง model เพื่อให้ผู้ใช้งานเขียนรีวิว ดังแสดงใน รูปที่ ค.7



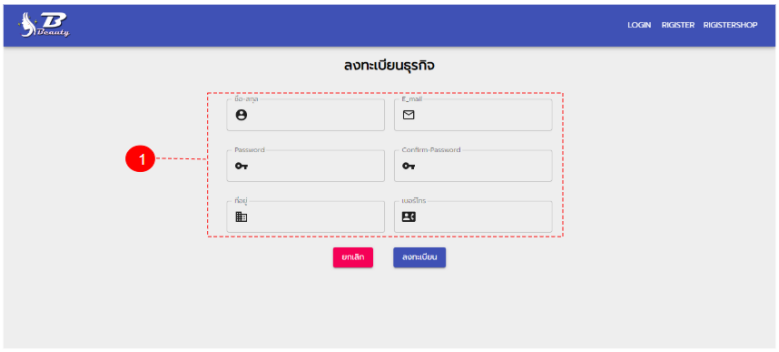
รูปที่ ค.7: หน้าจอเขียนรีวิว

จากรูปที่ ค.7 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ กรอกข้อมูลการรีวิวและกด submit

2. ส่วนของหน้าเมนูเว็บแอปพลิเคชันสำหรับเจ้าของร้าน

- เมื่อผู้ใช้งานกดที่ปุ่ม RegisterShop ระบบจะทำการแสดงหน้าต่างลงทะเบียน ดังแสดงในรูปที่ ค.8



รูปที่ ค.8: หน้าต่างลงทะเบียนธุรกิจ

จากรูปที่ ค.8 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ กรอกข้อมูลในการลงทะเบียนแล้วกดยืนยัน

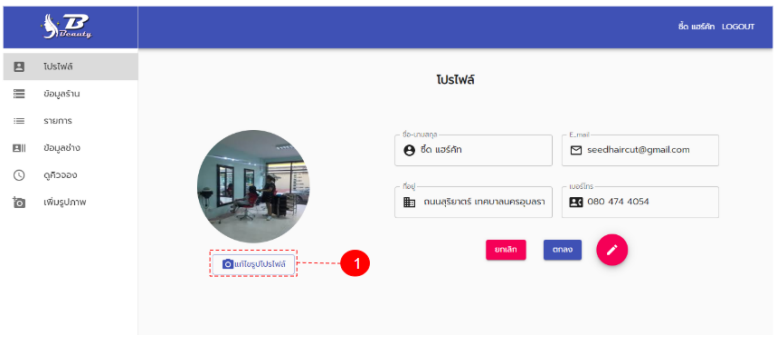
- ระบบทำการตรวจสอบทุกครั้งเมื่อผู้ใช้งานกดปุ่มเข้าสู่ระบบ ระบบจะแสดงหน้าจอเข้าสู่ระบบโดยผู้ใช้งานจำเป็นต้องทำการกรอกข้อมูลคือ อีเมลและรหัสผ่าน ดังแสดงในรูปที่ ค.9

รูปที่ ค.9: หน้าจอเข้าสู่ระบบ

จากรูปที่ ค.9 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ ส่วนของฟอร์มในการกรอกข้อมูลอีเมลและรหัสผ่านและ ปุ่มกดเข้าสู่ระบบ

- เมื่อเจ้าของร้านเข้าสู่ระบบสำเร็จ ดังแสดงในรูปที่ ค.10

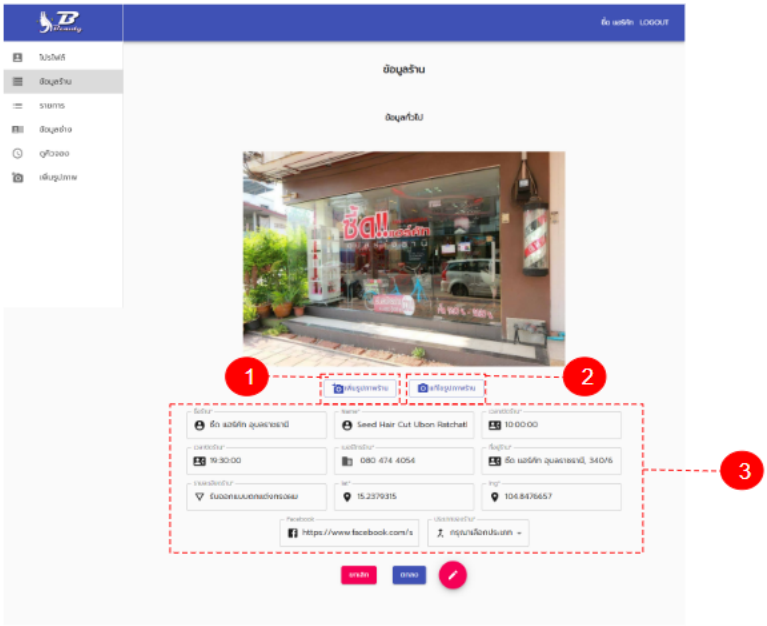


รูปที่ ค.10: หน้าจอโปรไฟล์

จากรูปที่ ค.10 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ ปุ่มสำหรับอัพโหลดรูปภาพ

- เมื่อเจ้าของร้านคลิ๊กที่เมนูข้อมูลร้าน ระบบจะแสดงหน้าข้อมูลร้าน ดังแสดงในรูปที่ ค.11



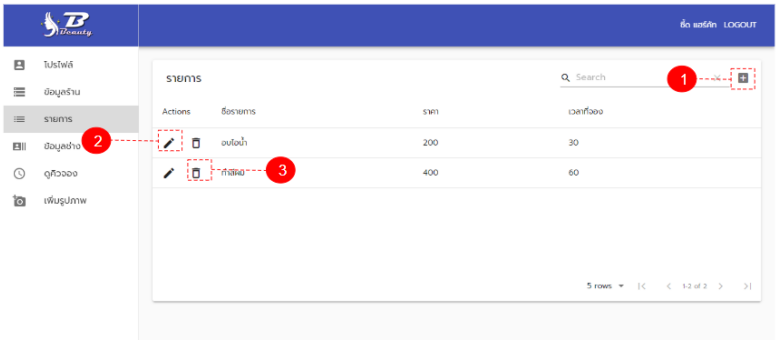
รูปที่ ค.11: หน้าจอข้อมูลร้าน

จากรูปที่ ค.11 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ ปุ่มสำหรับเพิ่มรูปภาพ

- หมายเลข 2 คือ ปุ่มสำหรับแก้ไขรูปภาพ
- หมายเลข 3 คือ กรอกข้อมูลร้านทั้งหมด

- เมื่อเจ้าของร้านคลิ๊กที่เมนูรายการ ระบบจะแสดงหน้ารายการ ดังแสดงในรูปที่ ค.12

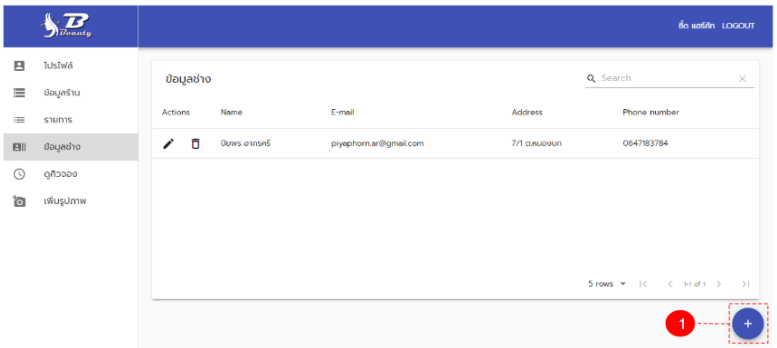


รูปที่ ค.12: หน้าจอรายการร้าน

จากรูปที่ ค.12 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ ปุ่มบวกใช้เพื่อเพิ่มข้อมูลรายการ
- หมายเลข 2 คือ ปุ่มสำหรับแก้ไขข้อมูลรายการ
- หมายเลข 3 คือ ปุ่มสำหรับลบข้อมูลรายการ

- เมื่อเจ้าของร้านกดที่ปุ่มข้อมูลช่าง ระบบจะทำการแสดงหน้าข้อมูลช่าง ดังแสดงในรูปที่ ค.13

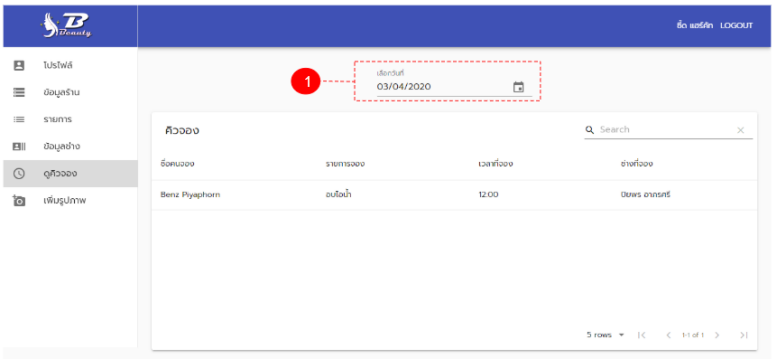


รูปที่ ค.13: หน้าจอข้อมูลช่าง

จากรูปที่ ค.13 สามารถอธิบายการใช้งานได้ดังนี้

- หมายเลข 1 คือ กดปุ่มเพื่อเพิ่มข้อมูลช่าง

- เมื่อเจ้าของร้านคลิ๊กที่เมนูดูคิวจอง ระบบจะแสดงหน้าข้อมูลการจองคิว ดังแสดงในรูปที่ ค.14

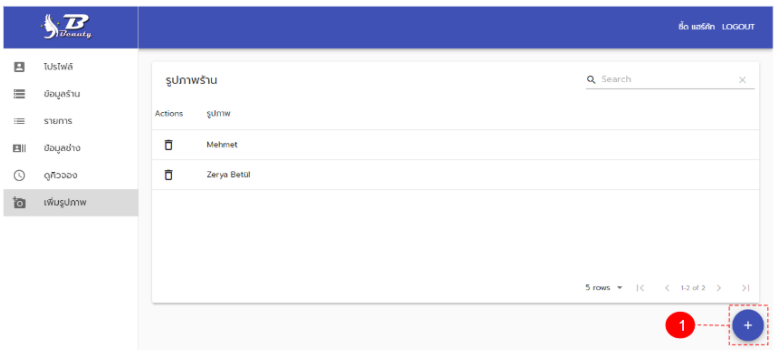


รูปที่ ค.14: หน้าจอดูการจองคิว

จากรูปที่ ค.14 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ เลือกวันที่ที่ต้องการดูการจองคิว

- เมื่อเจ้าของร้านคลิ๊กที่เมนูเพิ่มรูปภาพ ดังแสดงในรูปที่ ค.15



รูปที่ ค.15: หน้าจอเพิ่มรูปภาพ

จากรูปที่ ค.15 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ กดปุ่มเพื่อเพิ่มรูปภาพ

3. ส่วนของหน้าเมนูเว็บแอปพลิเคชันสำหรับช่าง

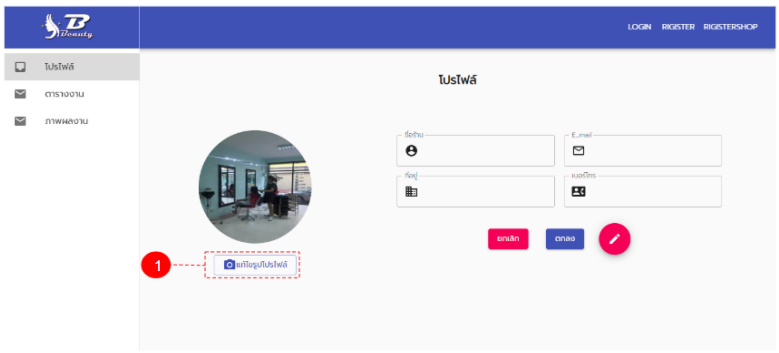
- ระบบทำการตรวจสอบทุกครั้งเมื่อผู้ใช้งานกดปุ่มเข้าสู่ระบบ ระบบจะแสดงหน้าจอเข้าสู่ระบบโดยผู้ใช้งานจำเป็นต้องทำการกรอกข้อมูลคือ อีเมลและรหัสผ่าน ดังแสดงในรูปที่ ค.16

รูปที่ ค.16: หน้าจอเข้าสู่ระบบ

จากรูปที่ ค.16 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ ส่วนของฟอร์มในการกรอกข้อมูลอีเมลและรหัสผ่านและ ปุ่มกดเข้าสู่ระบบ

- หน้าช่างเข้าสู่ระบบสำเร็จ ระบบจะแสดงหน้าโปรไฟล์ ดังแสดงในรูปที่ ค.17

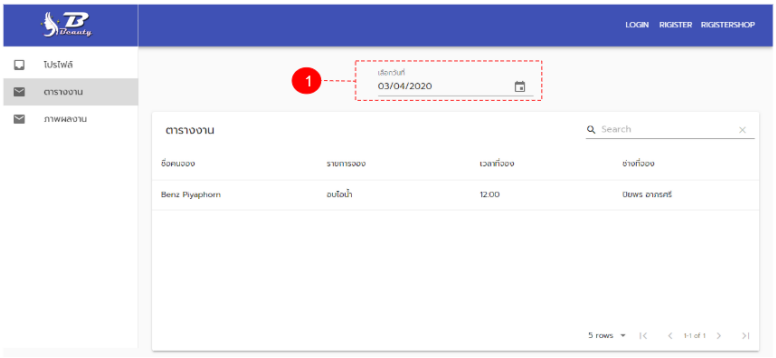


รูปที่ ค.17: หน้าโปรไฟล์

จากรูปที่ ค.17 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ กดปุ่มเพื่ออัพโหลดรูปภาพ

- เมื่อช่างคลิ๊กที่เมนูตารางงาน ระบบจะแสดงหน้าดูตารางงาน ดังแสดงในรูปที่ค.18

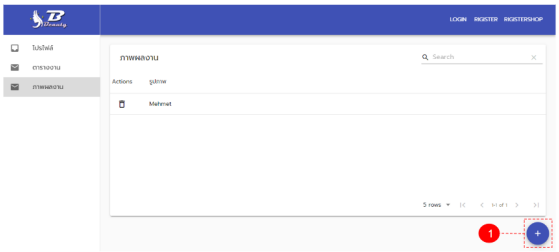


รูปที่ ค.18: หน้าตารางงาน

จากรูปที่ ค.18 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ เลือกวันที่ที่ต้องการให้แสดงตารางงาน

- เมื่อช่างคลิ๊กที่เมนูภาพผลงาน ระบบจะแสดงหน้าเพิ่มผลงานช่าง ดังแสดงในรูปที่ ค. 19



รูปที่ ค.19: หน้าจอภาพผลงานช่าง

จากรูปที่ ค.19 สามารถอธิบายการใช้งานได้ดังนี้

– หมายเลข 1 คือ กดปุ่มเพื่อเพิ่มรูปภาพผลงานช่าง

ประวัติผู้พัฒนา

ชื่อ-สกุล: นางสาวปิยพร อาภรศรี

รหัสประจำตัวนักศึกษา: 59110440259

วัดเกิด: 12 06 2540

ที่อยู่ที่สามารถติดต่อได้: 7 ม.1 ต.หนองบก อ.เหล่าเสือโก้ก จ.อุบลราชธานี 34000

เบอร์โทรศัพท์: (+66) 99 468 2013

อิเมลล์: piyaphorn.ar.59@ubu.ac.th

ระดับมัธยมต้น: โรงเรียนหกสิบพรรษาวิทยาคมอุบลราชธานี จังหวัดอุบลราชธานี

ระดับมัธยมปลาย: โรงเรียนหกสิบพรรษาวิทยาคมอุบลราชธานี จังหวัดอุบลราชธานี

ระดับอุดมศึกษา: ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ สาขาวิทยาการ คอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี