รายงานสหกิจศึกษาฉบับสมบูรณ์

เว็บแอปพลิเคชันจาก Node.js โดยใช้การยืนยันตัวตนแบบ Token-based

Node.js Web Application with Token Based Authentication

นางสาวเพทาย พีรานนท์

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2562

ชื่อโครงงานสหกิจศึกษา เว็บแอปพลิเคชันจาก Node.js โดยใช้การยืนยันตัวตนแบบ Token-based

ชื่อ - สกุล นักศึกษา นางสาวเพทาย พีรานนท์
คณะ วิศวกรรมศาสตร์ **ภาควิชา** วิศวกรรมคอมพิวเตอร์
ชื่อ - สกุล อาจารย์นิเทศ อาจารย์เกียรติณรงค์ ทองประเสริฐ
ชื่อ - สกุล ผู้นิเทศงาน คุณศุภพงศ์ พุฒยางกูร
สถานประกอบการ บริษัท ดาต้าโปรคอมพิวเตอร์ ซิสเต็มส์ จำกัด

บทคัดย่อ

โครงงานนี้มีวัตถุประสงค์เพื่อพัฒนาเว็บแอปพลิเคชันที่สามารถสร้างโพสต์และแสดงความคิดเห็นได้ เพื่อเป็นแหล่งบันทึกแนวทางแก้ปัญหาที่เกิดขึ้นในการทำงาน และบันทึกความคืบหน้าในการทำงาน โดยมีระบบการเข้าสู่ระบบและยืนยันตัวตนที่สามารถนำไปใช้กับเว็บแอปพลิเคชันอื่นในองค์กรได้

ผู้จัดทำได้พัฒนาเว็บแอปพลิเคชัน Knowledge base ขึ้นมา ซึ่งใช้ NoSQL ในการเก็บข้อมูลในฐานข้อมูล, ใช้ Node.js ในการเขียนโปรแกรมฝั่งเซิร์ฟเวอร์, ใช้ React ในการเขียนโปรแกรมฝั่งผู้ใช้งาน และใช้ JSON Web Token (JWT) ซึ่งเป็น Token-based สำหรับการเข้าสู่ระบบและยืนยันตัวตน โดยการติดต่อระหว่างฝั่งผู้ใช้และฝั่งเซิร์ฟเวอร์ได้ใช้แนวคิด RESTful API ในพัฒนา เพื่อเพื่อให้ง่ายต่อการขยายระบบ

คำสำคัญ: เว็บแอปพลิเคชัน, Node.js, Token-based, การยืนยันตัวตน, RESTful API

Co-operative Title: Node.js Web Application with Token-based Authentication

Student Intern Name: Paytye Peeranon

Faculty: Engineering **Department:** Computer Engineering

Advisor Name: Kiatnarong Tongprasert

Mentor Name: Supapong Putthatangkura

Company: Datapro Computer Systems Co., Ltd

ABSTRACT

This thesis purpose to develop a web application that has an efficiency to create post and comment post. In order to be a source for recording solutions to case that occur during work and work progress. Furthermore, web application must have an authentication system that can be used with other web applications in the organization.

The author developed a web application that uses NoSQL for storing data in database, Node.js for the server-side, React for the client-side and JSON Web Token (JWT) that is an token-based for an authentication system. Finally, the communication between the client-side and the server-side uses the RESTful API concept for development, so the system has a scalability.

Keywords: Web application, Node.js, Token-based, Authentication, RESTful API

กิตติกรรมประกาศ

ปริญญานิพนธ์เว็บแอปพลิเคชันจาก Node.js โดยใช้การยืนยันตัวตนแบบ Token-based สำเร็จได้ด้วยดีด้วยเพราะได้รับการช่วยเหลือจากบุคคลหลาย ๆ ท่าน อาทิ

อาจารย์บัณฑิต พัสยา และอาจารย์จิระศักดิ์ สิทธิกร อาจารย์ผู้ดูแลโครงสหกิจศึกษา ที่คอยแจ้งกำหนดการของโครงการมาโดยตลอด และยังช่วยให้คำแนะนำเกี่ยวกับจุดบกพร่องในโครงงาน

อาจารย์เกียรติณรงค์ ทองประเสริฐ อาจารย์นิเทศที่ให้คำแนะนำเกี่ยวกับโครงงานและการนำเสนอ

คุณศุภพงศ์ พุฒยางกูร ผู้นิเทศงานที่คอยให้คำแนะนำและชี้จุดบกพร่องตั้งแต่ต้นจนจบการทำโครงงาน รวมไปถึงคอยช่วยเหลือแหละตอบคำถามข้อสงสัยต่าง ๆ ที่เกิดขึ้นด้วย

คุณอนันต์ คุณรสิก คุณธัญญาธร รวมไปถึงพนักงานทุกคนในแผนก NIG ที่ให้การดูแลตลอดระยะเวลาของสหกิจศึกษา

และสุดท้าย บิดา มารดา เพื่อนทุกคน รวมไปถึงบุคคลอื่นที่ไม่ได้กล่าวถึง ณ ทีนี้ทุกท่าน ที่คอยให้การสนับสนุนและเป็นกำลังใจในการทำโครงงานจนสำเร็จ

เพทาย พีรานนท์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

บริษัท ดาต้าโปร คอมพิวเตอร์ ซิสเต็มส์ จำกัด เป็นบริษัทที่นำเสนอระบบเทคโนโลยีสารสนเทศแบบครบวงจรให้กับองค์กรขนาดกลางไปจนถึงขนาดใหญ่ ซึ่งมักจะพบปัญหาต่าง ๆ ระหว่างการทำงานที่ทำให้ระบบเกิดความผิดพลาด ในบางกรณีก็เป็นปัญหาคล้ายกับที่พนักงานผู้อื่นเคยพบมาก่อน แต่ไม่มีสื่อกลางเอาไว้แบ่งปันแนวทางการแก้ปัญหา ทำให้ต้องเสียเวลาในการหาวิธีแก้ไขอีกครั้ง

นอกจากนี้ สำหรับการทำงานเป็นทีม สื่อกลางที่มีไว้เพื่อติดตามความคืบหน้าของงานเป็นสิ่งจำเป็นมาก หากพนักงานหลักผู้ดูแลโครงการติดภารกิจทำให้ไม่สามารถมาทำงานที่องค์กรได้ ก็ยังต้องมีผู้ที่รู้ความคืบหน้าและรายละเอียดของโครงการ เพื่อที่จะทำงานได้ต่ออย่างราบรื่นและเผื่อในกรณีที่ต้องตอบคำถามลูกค้าเกี่ยวกับความคืบหน้าของโครงการ

จากประเด็นดังกล่าว ทางองค์กรจึงได้วางแผนว่าจะพัฒนาเว็บแอปพลิเคชันขึ้นมา 2 ชิ้น คือ Knowledge base ที่มีลักษณะคล้ายเว็บบอร์ดซึ่งสามารถสร้างโพสต์และแสดงความคิดเห็นในโพสต์ได้ มีไว้สำหรับบันทึกปัญหาที่พบและวิธีแก้ไข หรือบันทึกความคืบหน้าของโครงการต่าง ๆ และเว็บแอปพลิเคชันอีกชิ้นที่จะพัฒนาคือ Case Management ซึ่งมีไว้สำหรับสร้างกรณีศึกษาเมื่อเจอปัญหาและติดตามผลจนกว่าจะสามารถแก้ไขปัญหาให้ลุล่วงได้ โดยต้องการให้เว็บแอปพลิเคชันทั้ง 2 ชิ้นใช้ฐานข้อมูลชุดเดียวกัน และต้องการให้มีตัวกลางสำหรับทำการเข้าสู่ระบบและยืนยันตัวตนที่ทั้ง 2 ชิ้นนี้สามารถนำไปใช้ร่วมกันได้

ด้วยเหตุนี้ ผู้จัดทำจึงได้รับมอบหมายให้พัฒนาเว็บแอปพลิเคชัน Knowledge base ซึ่งมีตัวกลางสำหรับการเข้าสู่ระบบและยืนยันตัวตนที่เว็บแอปพลิเคชันตัวอื่นสามารถเรียกใช้ได้ในอนาคต

1.2 วัตถุประสงค์ของโครงงาน

1. มีแหล่งสืบค้นแนวทางแก้ไขปัญหาที่เกิดในการทำงาน
2. มีแหล่งติดตามความคืบหน้าของงาน
3. มีตัวกลางที่ใช้ในการเข้าสู่ระบบและยืนยันตัวตน

1.3 ขอบเขตของโครงงาน

เว็บแอปพลิเคชันจาก Node.js ที่สามารถสร้างโพสต์และแสดงความคิดเห็นได้ โดยมีการเข้าสู่ระบบและยืนยันตัวตนโดยใช้ Token และส่วนต่าง ๆ ในระบบติดต่อกันโดยใช้แนวคิดของ RESTful API

1.4 วิธีดำเนินโครงงาน

1. ทำความเข้าใจถึงเหตุผลที่ต้องการทำโครงงานจากผู้นิเทศงาน
2. วางแผนขั้นตอนการดำเนินการทำโครงงาน
3. ศึกษาเครื่องมือที่นำมาใช้ในการทำโครงงาน
4. ออกแบบระบบทั้งหมด
5. ลงมือพัฒนาชิ้นงานในส่วนที่ใช้ติดต่อกับฐานข้อมูล
6. ลงมือพัฒนาชิ้นงานในส่วนที่ใช้ติดต่อกับผู้ใช้งาน
7. ทดสอบและแก้ไขชิ้นงาน
8. จัดทำรูปเล่มรายงานโครงงาน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ผู้ใช้งานสามารถหาแนวทางในการแก้ไขปัญหาที่เกิดในการทำงานได้ง่ายขึ้น
2. ผู้ใช้งานสามารถติดตามความคืบหน้าของงานได้ง่ายขึ้น

บทที่ 2
แนวคิดและทฤษฎี

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 JWT (JSON Web Tokens)

JWT เป็นมาตรฐานเปิด (RFC 7519) ที่ใช้กำหนดเส้นทางปลอดภัยเพื่อการส่งผ่านข้อมูลระหว่างส่วนต่าง ๆ ในรูปแบบของวัตถุเจซัน ข้อมูลที่ถูกส่งผ่านเป็นข้อมูลที่ได้รับการยืนยันว่าเชื่อถือได้ เพราะเป็นลายเซ็น (Signature) แบบดิจิทัลซึ่งมีการใช้ Secret ที่ใช้อัลกอริทึม HMAC หรือคู่ของกุญแจสาธารณะ หรือคู่ของกุญแจส่วนบุคคลที่ใช้อัลกอริทึม RSA หรืออัลกอริทึม ECDSA ในการ Sign

2.1.1.1 ตัวอย่างของสถานการณ์ที่เหมาะแก่การใช้ JSON Web Token เช่น

- การตรวจสอบสิทธิ (Authorization): เป็นสถานการณ์ที่พบบ่อยที่สุดสำหรับการใช้ JWT เมื่อผู้ใช้ทำการเข้าสู่ระบบแล้ว แต่ละ Request ที่ส่งมาจะประกอบด้วย JWT โดยการอนุญาตให้ผู้ใช้เข้าถึงเส้นทาง บริการ และทรัพยากรต่าง ๆ จะขึ้นอยู่กับโทเค็นที่ถูกส่งมา

การเข้าสู่ระบบเพียงครั้งเดียว (Single Sign On) เป็นคุณลักษณะที่ถูกใช้กับ JWT อย่างกว้างขวางในทุกวันนี้ เนื่องจาก Overhead ของ JWT มีขนาดเล็ก และสามารถใช้กับโดเมนคนละชนิดได้ง่าย

- การแลกเปลี่ยนข้อมูล (Information Exchange): JWT เป็นตัวเลือกที่ดีสำหรับการส่งผ่านข้อมูลอย่างปลอดภัยระหว่างส่วนต่าง ๆ เนื่องจาก JWT สามารถใส่ลายเซ็นเข้าไปได้ ผู้ส่งข้อมูลจะมั่นใจได้ว่าส่งไปให้ผู้รับได้ถูกคน นอกจากนั้น เมื่อลายเซ็นถูกนำไปคำนวณโดยใช้ส่วนหัวและเพย์โหลดจะสามารถตรวจสอบได้ว่าเนื้อหาที่ส่งไปไม่ได้ถูกปรับเปลี่ยนใด ๆ

2.1.1.2 โครงสร้างของ JSON Web Token

รูปแบบของ JWT ประกอบด้วย 3 ส่วนที่ถูกแบ่งโดยมหัพภาค (.) โดย 3 ส่วนที่ถูกกล่าวถึงคือ ส่วนหัว (Header), เพย์โหลด (Payload) และ ลายเซ็น (Signature) ที่เขียนอยู่ในรูปแบบ xxxxx.yyyyy.zzzzz ซึ่งส่วนประกอบแต่ละส่วนมีรายละเอียด ดังนี้

1) ส่วนหัว (Header): โดยปกติแล้วจะประกอบด้วย 2 ส่วน คือ ชนิดของโทเค็นซึ่งเป็น JWT และอัลกอริทึมที่ใช้ทำการ Sign จากนั้นนำไปเข้ารหัสแบบ Base64Url เช่น

{

"alg": "HS256",

"typ": "JWT"

}

2) ส่วนเพย์โหลด (Payload): ประกอบด้วย claim ซึ่งเป็นชุดข้อความที่เกี่ยวกับผู้ใช้หรือข้อมูลเพิ่มเติม จากนั้นนำไปเข้ารหัสแบบ Base64Url เช่น

{

“sub”: “1234567890”,

“name”: “John Doe”,

“admin”: true

}

3) ส่วนลายเซ็น (Signature): สำหรับการสร้างส่วนลายเซ็นจำเป็นต้องใช้ส่วนหัวที่ถูกเข้ารหัสแล้ว ส่วนเพย์โหลดที่ถูกเข้ารหัสแล้ว ส่วน Secret และอัลกอริทึมที่ใช้เพื่อเข้ารหัสส่วนหัวและส่วนเพย์โหลด จากนั้นนำไปเข้ารหัสแบบ Base64Url

ยกตัวอย่างเช่น หากต้องการใช้อัลกอริทึม HMAC SHA256 ส่วนลายเซ็นจะถูกสร้างขึ้นโดยมีรูปแบบดังนี้

HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload),

secret)

ส่วนลายเซ็นถูกใช้เพื่อตรวจสอบว่าเนื้อหาที่ได้รับมาไม่ได้ถูกปรับแต่งในระหว่างการส่งผ่าน และในกรณีที่โทเค็นถูก Sign ด้วยกุญแจส่วนบุคคลจะสามารถตรวจสอบได้ว่าผู้ที่ส่ง JWT มาเป็นตัวจริงหรือไม่

2.1.2.3 การรวมทุกส่วนของ JSON Web Token เข้าด้วยกัน

ผลลัพธ์ของการรวมทุกส่วนประกอบของ JWT เข้าด้วยกันคือสายอักขระ Base64-URL 3 ชุดซึ่งถูกแบ่งโดยมหัพภาค (.) โดยเมื่อเทียบกับ SAML ที่เป็นมาตรฐาน XML เห็นได้ว่า JWT มีขนาดเล็กกว่าและยังสามารถส่งผ่านทาง HTML และ HTTP ได้ง่ายกว่าอีกด้วย

ส่วนต่อไปนี้จะแสดงว่าส่วนหัว ส่วนเพย์โหลด และส่วนลายเซ็นที่ถูกเข้ารหัสซึ่งได้ยกตัวอย่างไปข้างต้น เมื่อ Sign ด้วย Secret แล้วจะมีรูปแบบดังนี้

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

ภาพที่ 2.1 รูปแบบของ JWT ที่ถูก Sign

2.1.2.4 JSON Web Token ทำงานอย่างไร

ในการการตรวจสอบสิทธิ เมื่อผู้ใช้เข้าสู่ระบบด้วยข้อมูลประจำตัวได้สำเร็จ JWT จะถูกสร้างและส่งกลับมา โทเค็นที่ได้รับมาจำเป็นต้องได้รับการดูแลอย่างดีเนื่องจากประกอบไปด้วยข้อมูลประจำตัวของผู้ใช้ ดังนั้นโทเค็นจึงไม่ควรถูกเก็บไว้นานเกินกว่าที่ต้องการ

เมื่อใดก็ตามหากผู้ใช้ต้องการเข้าถึงเส้นทาง บริการ หรือทรัพยากรต่าง ๆ ที่ถูกป้องกันไว้ User Agent จำเป็นต้องส่ง JWT ที่ถูกเก็บไว้ที่ส่วนหัวส่วน Authorization โดยถูกเก็บไว้ในรูปแบบของ Bearrer schema ซึ่งมีรูปแบบดังนี้

Authorization: Bearer <token>

แผนภาพข้างล่างนี้แสดงขั้นตอนการได้รับ JWT และนำ JWT ไปใช้เพื่อเข้าถึง API หรือทรัพยากรต่าง ๆ

ภาพที่ 2.2 ขั้นตอนการได้รับ JWT และนำไปใช้

1) แอปพลิเคชันหรือผู้รับบริการร้องขอการตรวจสอบสิทธิจากเครื่องบริการการตรวจสอบสิทธิ (Authorization Server)

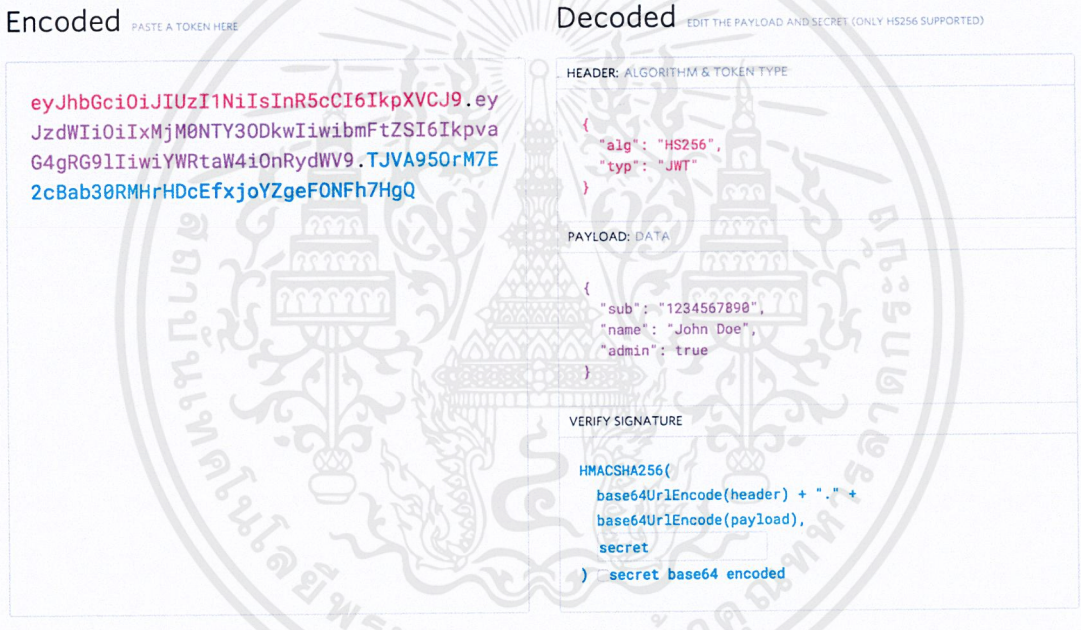
2) เมื่อการตรวจสอบสิทธิได้รับการอนุญาตแล้ว เครื่องบริการการตรวจสอบสิทธิจะส่งโทเค็นการเข้าถึงกลับมายังแอปพลิเคชัน

3) แอปพลิเคชันจะใช้โทเค็นการเข้าถึงเพื่อเข้าถึงทรัพยากรที่ถูกป้องกันไว้ อย่างเช่น API ต่าง ๆ

โปรดอย่าลืมว่า ถึงแม้ผู้อื่นไม่สามารถทำการแก้ไขข้อมูลในโทเค็นได้ แต่ข้อมูลทั้งหมดก็สามารถถูกเปิดเผยได้ ดังนั้นจึงไม่ควรเก็บข้อมูลที่เป็นความลับไว้ในโทเค็น

2.1.1.5 เหตุผลที่ควรใช้ JSON Web Token

เมื่อเปรียบเทียบระหว่าง JSON Web Tokens (JWT) กับ Simple Web Tokens (SWT) และ Security Assertion Markup Language Tokens (SAML) จากรูปที่ 2.4 และรูปที่ 2.5 จะเห็นได้ว่า JSON มีข้อมูลที่ไม่จำเป็นน้อยกว่า XML เมื่อ JSON ถูกเข้ารหัสแล้วจึงมาขนาดเล็กกว่า XML ซึ่งทำให้ JWT กระชับกว่า SAML จึงสรุปได้ว่า JWT เป็นตัวเลือกที่ดีที่จะถูกนำมาใช้ในสภาพแวดล้อมของ HTML และ HTTP



ภาพที่ 2.3 ความยาวของ JWT ที่ถูกเข้ารหัส

ภาพที่ 2.4 ความยาวของ SAML ที่ถูกเข้ารหัส

SWT สามารถทำได้เพียง Sign แบบสมมาตรโดยใช้ Secret และอัลกอริทึม HMAC อย่างไรก็ตาม JWT และ SAML สามารถใช้ได้ทั้งคู่ของกุญแจสาธารณะและคู่ของกุญแจส่วนบุคคลในรูปแบบของใบรับรอง X.509 สำหรับการ Sign โดยการ Sign XML ด้วยลายเซ็นดิจิทัล XML จะทำได้ยากกว่าเมื่อเทียบกับการ Sign ของ JSON

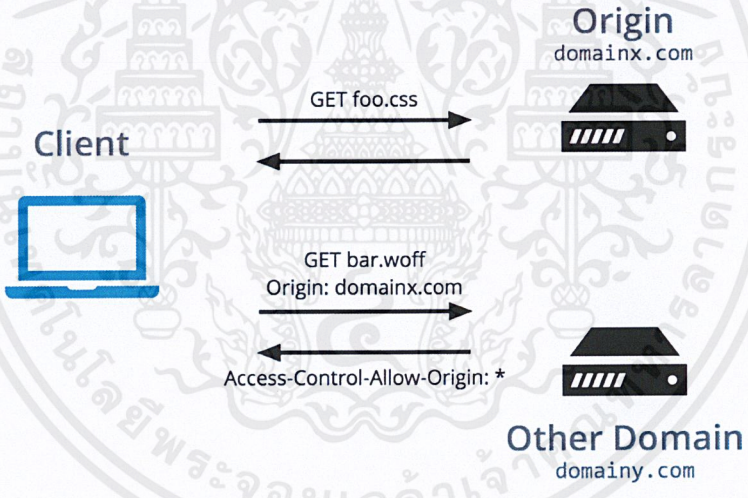
เนื่องจาก JSON parser จะ เป็นสิ่งที่พบเจอได้โดยทั่วไปในภาษาการเขียนโปรแกรม เพราะมีการ Map โดยตรงไปสู่วัตถุ ในทางตรงกันข้าม XML ไม่มีการ Map เอกสารและวัตถุอย่างธรรมชาติ จึงทำให้การทำงานร่วมกับ JWT ง่ายกว่า SAML

2.1.2 Node.js

Node.js เป็นซอฟต์แวร์ Open Source, Cross-platform, JavaScript runtime ที่ทำให้ JavaScript ที่ปกติทำงานในฝั่ง Front-end สามารถทำงานในฝั่ง Back-end ซึ่งก็คือฝั่งเซิร์ฟเวอร์ได้ โดยจะรันสคริปต์ในฝั่งเซิร์ฟเวอร์เพื่อสร้างเนื้อหาของหน้าเว็บแบบไดนามิกก่อนจะส่งไปยังเว็บบราวเซอร์ของผู้ใช้งาน

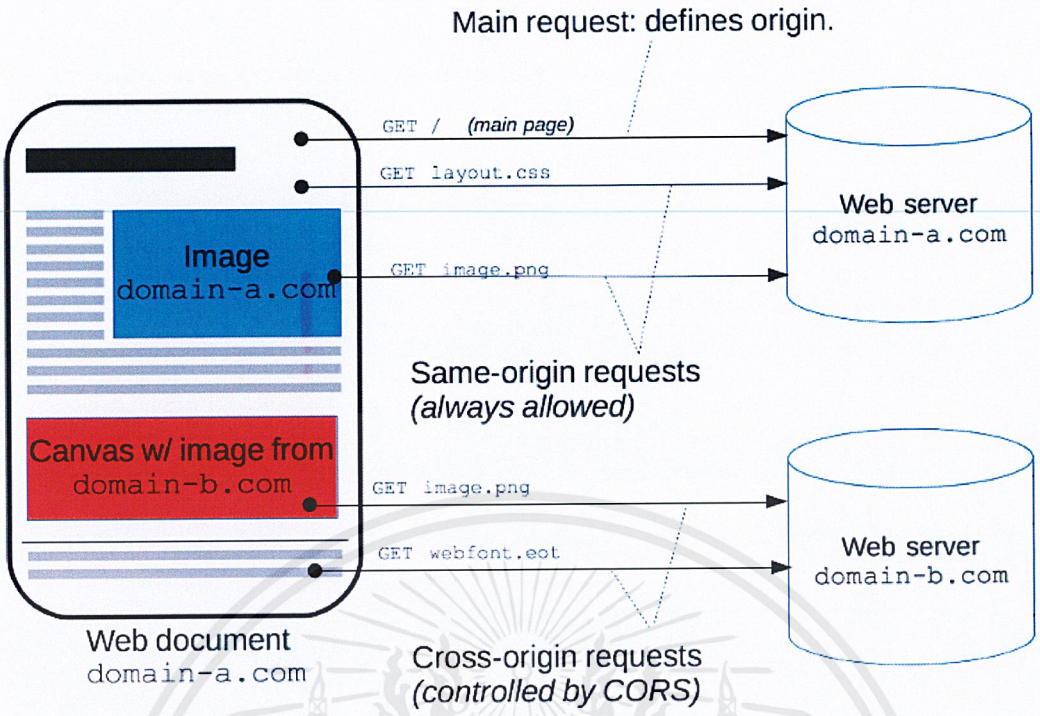
2.1.3 Cross-Origin Resource Sharing (CORS)

CORS เป็นกลไกที่ใช้ HTTP headers เพิ่มเติมเพื่อให้บราวเซอร์ได้รับสิทธิ์ในการเข้าถึงทรัพยากรที่เลือกจากเซิร์ฟเวอร์บนโดเมนอื่นมาแสดงบนหน้าเว็บบราวเซอร์ได้ เนื่องจากอินเทอร์เน็ตเป็นการสื่อสารระหว่างคอมพิวเตอร์ ดังนั้นคอมพิวเตอร์แต่ละเครื่องต้องมี Protocol ที่เหมือนกัน จึงจะสื่อสารกันรู้เรื่อง เว็บบราวเซอร์จะส่ง HTTP request เมื่อต้องการขอข้อมูลข้ามโดเมนหรือ Port ที่ต่างกัน และต้องทำตามข้อตกลงการสื่อสาร (Protocol)



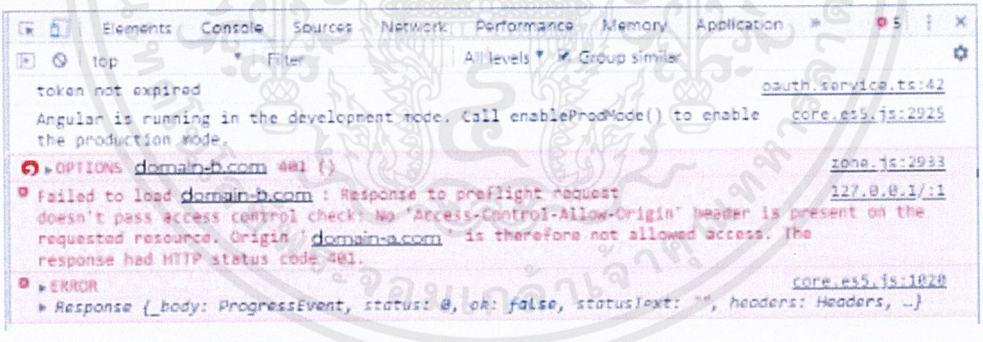
ภาพที่ 2.5 หลักการของ CORS

CORS จะกำหนดวิธีการที่เว็บบราวเซอร์และเซิร์ฟเวอร์สามารถโต้ตอบเพื่อกำหนดว่าจะให้อนุญาตในการขอข้อมูลข้ามโดเมนหรือไม่



ภาพที่ 2.6 การทำงานของ CORS

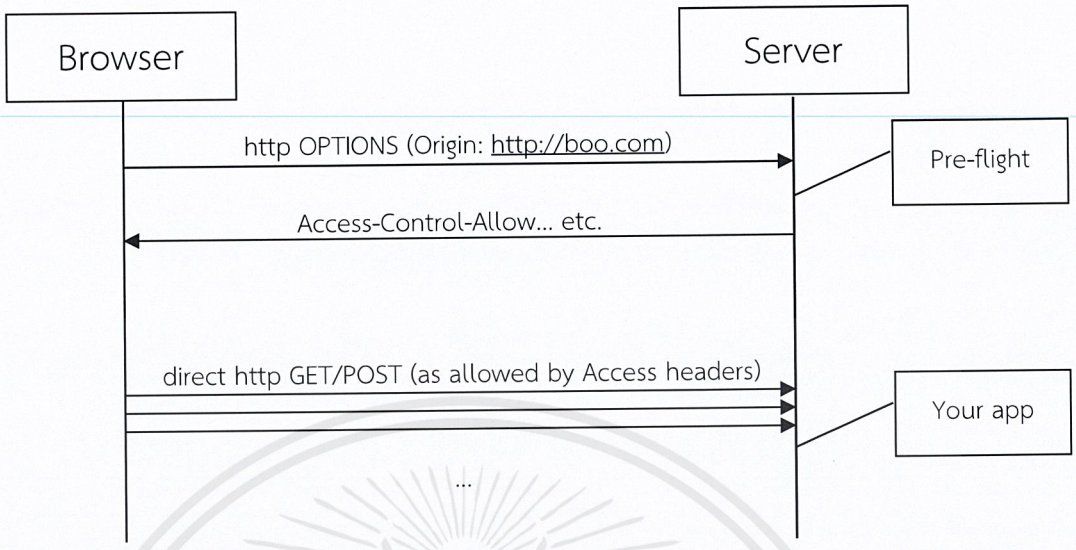
จากภาพที่ 2.6 หากต้องการร้องขอข้อมูลจาก domain-b.com จะพบข้อผิดพลาดซึ่งแสดงดังภาพที่ 2.7



ภาพที่ 2.7 domain-a.com ไม่ได้รับอนุญาติจาก domain-b.com

เนื่องจากปัจจุบันมักจะแยกฝั่ง Front-end และ Back-end ออกจากกันเป็นคนละโดเมนด้วยเหตุผลเรื่องความปลอดภัยของ HTTP บราวเซอร์ ดังนั้นการอนุญาตให้เข้าถึงแหล่งข้อมูลจะต้องอยู่ในโดเมนเดียวกันเท่านั้น เว้นแต่ว่าแหล่งข้อมูลนั้นจะอนุญาตให้โดเมนของบราวเซอร์สามารถเข้าถึงข้อมูลเหล่านั้นได้

2.1.3.1 การทำงานของ Browser กับ Server



ภาพที่ 2.8 การทำงานของ Browser กับ Server

เมื่อเริ่มการทำงานแอปพลิเคชัน เว็บบราวเซอร์จะส่ง request ไปหา Server โดยแบ่งออกเป็น 2 รอบ

รอบที่ 1 บราวเซอร์จะส่ง request method เป็น OPTIONS โดยข้างใน OPTIONS จะมีการแนบโดเมนต้นทาง หากโดเมนที่แนบมาได้รับการอนุญาตจาก Server ในรอบที่ 2 จะสามารถส่ง request ไปเรียกใช้ API เพื่อรับข้อมูลจาก Server ได้

2.1.3.2 วิธีการแก้ไขปัญหา CORS กับ Browser

หากเราเป็นเจ้าของ Server เราสามารถอนุญาตโดเมนที่ต้องการให้เข้าถึงแหล่งข้อมูลได้ โดยกำหนด Access-Control-Allow-Origin: ‘domain-a.com’

```
1  const express = require("express")
2  const app = express()
3  const PORT = process.env.PORT || 8445
4
5  app.use((req, res, next) => {
6        res.header('Access-Control-Allow-Origin', 'domain-a.com')
7        res.header('Access-Control-Allow-Methods','POST, GET, PUT, PATCH, DELETE, OPTIONS')
8        res.header('Access-Control-Allow-Headers','Content-Type, Option, Authorization')
9        return next()
10     })
11 }
12
13 app.use('/', (req,res) => res.send("test"))
14 app.listen(PORT, () => { console.info(`server started on port ${PORT}`)})
```

ภาพที่ 2.9 การกำหนด Access-Control-Allow-Origin

- Access-Control-Allow-Origin คือ ที่มาที่อนุญาตให้ใช้ทรัพยากรร่วมกันได้
- Access-Control-Allow-Methods คือ Methods ที่อนุญาตให้ใช้ในการติดต่อได้
- Access-Control-Allow-Headers คือ Headers ที่อนุญาตให้ใช้ในการติดต่อ

หากต้องการอนุญาตให้ทุกโดเมนสามารถเข้าถึงแหล่งข้อมูลได้ ให้กำหนด Access-Control-Allow-Origin เป็น '*' แต่ถ้าใน Source code มีการกำหนด Access-Control-Allow-Credentials เป็นtrue และไม่ควรใช้การกำหนด Access-Control-Allow-Origin เป็น '*'

หากต้องการอนุญาตให้เข้าถึงได้มากกว่า 1 ที่มา สามารถกำหนด Source code ได้ดังภาพที่ 2.10

```
const express = require("express")
const app = express()
const PORT = process.env.PORT || 8445

app.use((req, res, next) => {
    let ALLOW_ORIGIN = ['domain-a.com', 'domain-b.com', 'domain-c.com']
    let ORIGIN = req.headers.origin
      if (ALLOW_ORIGIN.includes(ORIGIN)) {
        res.header('Access-Control-Allow-Origin', ORIGIN)
      }
      res.header('Access-Control-Allow-Methods','POST, GET, PUT, PATCH, DELETE, OPTIONS')
      res.header('Access-Control-Allow-Headers','Content-Type, Option, Authorization')
      return next()
    })
}

app.use('/', (req,res) => res.send("test"))
app.listen(PORT, () => { console.info(`server started on port ${PORT}`)})
```

ภาพที่ 2.10 การกำหนด Access-Control-Allow-Origin มากกว่า 1 Origin

2.1.4 Express.js

Express.js เอ็กเพรสดอทเจเอส เป็น Web Application Framework ชื่อดังที่ได้รับความนิยมมากสำหรับทำงานบนแพลตฟอร์มของ Node.js ซึ่งเป็น Server ตัวหนึ่ง โดยทั้ง Express.js และ Node.js ต่างก็ใช้ภาษา JavaScript ในการพัฒนา ถ้าเป็น Web Application Framework ในสมัยก่อน คนที่พัฒนาจะต้องมีความรู้มากกว่า 1 ภาษา ภาษาที่ทำงานทางฝั่ง Server อย่าง PHP หรือ ASP และภาษาที่ทำงานทางฝั่ง Client อย่าง JavaScript เพื่อลดความยุ่งยากรวมถึงเวลาในการต้องเรียนรู้หลาย ๆ ภาษาทำให้เกิด Node.js กับ Express.js เพียงแค่มีความรู้ JavaScript ก็สามารถเขียนได้ทั้ง Server และ Client นอกจากนี้ JavaScript จะมีการตอบสนองที่รวดเร็ว ทำให้

Express.js มีข้อเด่นในเรื่องความเร็ว ทำให้การเขียน Express.js จะใช้รูปแบบที่ง่ายในการเรียนรู้มากที่สุด

สำหรับการพัฒนา Express.js จะพูดถึงการใช้ Routing (การกำหนดเส้นทางของระบบ) และ Middleware (การรับส่งข้อมูลของระบบ) สามารถเขียนได้ในรูปแบบ MVC ส่วนการเชื่อมต่อกับฐานข้อมูลสามารถใช้ MongoDB หรือ MySQL ซึ่งสกุลของไฟล์ Express.js คือ .js

2.1.5 ฟังก์ชัน Middleware

ฟังก์ชัน Middleware คือ ฟังก์ชันที่สามารถเข้าไปจัดการกับ Request object (req), Response object (res) และฟังก์ชัน next ที่อ้างถึง Middleware ตัวต่อไปที่อยู่ในวัฏจักรของ request-response ในแอปพลิเคชัน โดยฟังก์ชัน next ที่อ้างถึง Middleware ตัวต่อไป

หน้าที่ของฟังก์ชัน Middleware คือ

- รันคำสั่งต่าง ๆ ที่กำหนด
- แก้ไขข้อมูลของ request / response object
- จบการทำงานวัฏจักร request-response
- ใช้งานคำสั่ง next เพื่อทำงานฟังก์ชัน middleware ถัดไป

2.1.5 MongoDB

MongoDB คือ Open-Source Document Database รูปแบบหนึ่ง ที่ใช้เป็นฐานของข้อมูลแบบ NoSQL หรือก็คือการไม่มีความสัมพันธ์ของตารางแบบ SQL ทั่ว ๆ ไป แต่จะใช้วิธีการเก็บข้อมูลแบบ JSON (JavaScript Object Notation) แทน โดยการบันทึกข้อมูลทุกบันทึกใน MongoDB จะเรียกว่าเป็น Document ที่จะเก็บค่าเป็น Key และ Value ซึ่งมีรูปแบบดังตัวอย่างด้านล่าง

```
{
    "_id": ObjectId("554b8ee746e04bc5503aef47"),
    "name": "Chai"
}
```

การเก็บข้อมูล Document ใน MongoDB นั้น จะถูกเก็บไว้ใน Collections (เปรียบเทียบได้กับ Table ใน Relational Database ทั่วไป) แต่มีความแตกต่างกันตรงที่ Collection ไม่จำเป็นจะต้องมี Schema เหมือนกันก็สามารถบันทึกข้อมูลได้ นอกจากนั้น ใน MongoDB ข้อมูลของ Document ที่เก็บไว้ใน Collection จะมีคีย์ _id ที่ทำหน้าที่เปรียบเสมือน Primary Key อยู่ด้วย

จุดเด่นที่น่าสนใจของ MongoDB เช่น

- เก็บข้อมูลแบบ Document หมายถึงการจัดเก็บข้อมูลแบบมีโครงสร้างและมีหลากหลายมิติ
- รองรับการทำ Full Index ซึ่งช่วยในการค้นหาได้อย่างรวดเร็ว กับข้อมูลที่มีปริมาณมหาศาล
- รองรับการขยายขนาดและรองรับการทำงานหนัก
- ทำระบบสำรองได้ง่าย
- เนื่องจากมีการเก็บข้อมูลแบบโครงสร้าง ดังนั้น เมื่อมีการเรียกข้อมูลมาแสดงก็จะได้ทั้งโครงสร้างและข้อมูลออกมา
- แก้ไขข้อมูลได้รวดเร็ว
- มีการเก็บข้อมูลด้วยระบบ GridFS ซึ่งเป็นระบบการเก็บไฟล์บนพื้นที่ฮาร์ดดิสก์ ที่เก็บข้อมูลเป็นกลุ่ม และรองรับการเพิ่มหรือลดปริมาณข้อมูลได้

2.1.5 Mongoose

Mongoose เป็น ODM (Object Document Mapping) ตัวหนึ่งของ MongoDB ซึ่งทำหน้าที่เหมือนกับ ORM ในฝั่ง DBMS แต่ ODM จะทำงานกับ NoSQL ซึ่งเหตุผลที่ต้องเป็น Object Document Mapping เพราะว่า MongoDB เป็น NoSQL ที่เก็บข้อมูลแบบ Document Store ซึ่งเป็นรูปแบบหนึ่งของ NoSQL

การทำงานของ Object Mapping คือ การสร้าง Object ความสัมพันธ์กับโครงสร้างของข้อมูลในฐานข้อมูล เวลาเรียกใช้สามารถทำได้เหมือนการเข้าถึงคลาสหรือวัตถุ ทำให้ลดการเขียนคำสั่ง SQL หรือคำสั่งสำหรับ query ลง

โดยการใช้งานของ ODM เหมือนกับ ORM คือ กำหนด Schema, กำหนด Model และเรียกใช้งานผ่าน Model ซึ่งทำให้ไม่ต้องสนใจโครงสร้างของฐานข้อมูลข้างหลัง

2.1.6 RESTful

RESTful API คือ Application Program Interface (API) ที่ใช่ HTTP request เพื่อ GET, PUT, POST และ DELETE ข้อมูล RESTful API มีต้นกำเนิดมาจากเทคโนโลยี REST (Representational State Transfer) ซึ่งเป็นรูปแบบของสถาปัตยกรรมและวิธีการที่ใช้สื่อสารซึ่งมักใช้ในการพัฒนา web service

API สำหรับเว็บไซต์เป็นสิ่งที่ช่วยให้ 2 ซอฟต์แวร์สามารถสื่อสารกันได้ และเป็นวิธีที่เหมาะสมต่อนักพัฒนาในการเขียนโปรแกรมที่ร้องขอบริการจากระบบปฏิบัติการหรือแอปพลิเคชันอื่น

6 ข้อกำหนดของ RESTful API ซึ่งถือเป็นสิ่งที่สำคัญในการสร้าง RESTful API ตามมาตรฐานซึ่งทำให้ง่ายต่อการพัฒนา และทำให้เป็นที่ยอมรับ (หากไม่ทำตามให้ครบทั้ง 6 ข้อจะไม่ถือว่าเป็น RESTful API ยกเว้น optional)

1) Client-server architecture: Client ไม่จำเป็นต้องรู้อะไรเกี่ยวกับ Business logic ภายใน ไม่มีหน้าที่เกี่ยวกับการจัดเก็บข้อมูล ส่วน Server มีหน้าที่เก็บ Resource และไม่จำเป็นต้องรู้อะไรเกี่ยวกับ UI Frontend หรือสถานะของผู้เรียก

2) Statelessness: เมื่อส่ง Request รับ Response จากเซิร์ฟเวอร์เสร็จ จากนั้นยกเลิกการเชื่อมต่อ

3) Cacheability: สามารถ cache response ได้ การ Response จะต้องสามารถกำหนดได้ว่าจะ Cache หรือไม่ เพื่อป้องกันไม่ให้ User หรือ Client ได้รับข้อมูลเก่า

4) Layered system: ปกติ Client ไม่รู้ว่าที่ทำการเชื่อมต่อนั้น ได้เชื่อมต่อโดยตรงกับ Server ปลายทางหรือไปยังตัวกลางอื่น ๆ ระหว่างทาง ดังนั้น Server ตัวกลางควรสามารถปรับปรุงความสามารถในการขยายระบบได้ โดยการใช้งานการทำ Load balance

5) Code on demand (optional): Server สามารถขยายได้ชั่วคราว หรือปรับแต่งการทำงานของ Client ได้ ตัวอย่างเช่น ทำ client-side scripts ใน JavaScript

6) Uniform interface: ถือเป็นข้อสำคัญจะที่แยกระหว่าง REST API และ Non-REST API ซึ่งแสดงให้เห็นถึงวิธีการที่จะติดต่อกับ Server โดยไม่คำนึงถึงประเภทของอุปกรณ์ หรือประเภทของ application

2.1.7 React

React คือ JavaScript Library ที่ทีม Facebook เป็นผู้พัฒนาขึ้นมาและเปิดให้คนทั่วไปนำมาใช้ฟรี ซึ่งเว็บไซต์ในปัจจุบันของ Facebook.com ก็ใช้ React อยู่เช่นกัน โดย React มีไว้เพื่อใช้สร้างภาพ (View) ในส่วน User Interface

แนวคิดที่ต้องรู้เพื่อเขียน React

1) Component

ส่วนต่าง ๆ ในเว็บจะถูกมองเป็น Component โดยแนวคิดของ Component คือ การเขียนองค์ประกอบของ View ให้ย่อยที่สุดเท่าที่จะย่อยได้ และเน้นให้นำกลับมาใช้ซ้ำได้เยอะ

ข้อดีข้องการแบ่ง Component คือ

- มีการแบ่งส่วนกันเขียน ทำให้แต่ละส่วนแยกกันอย่างชัดเจน
- สามารถนำ component ไปใช้ซ้ำที่อื่นได้

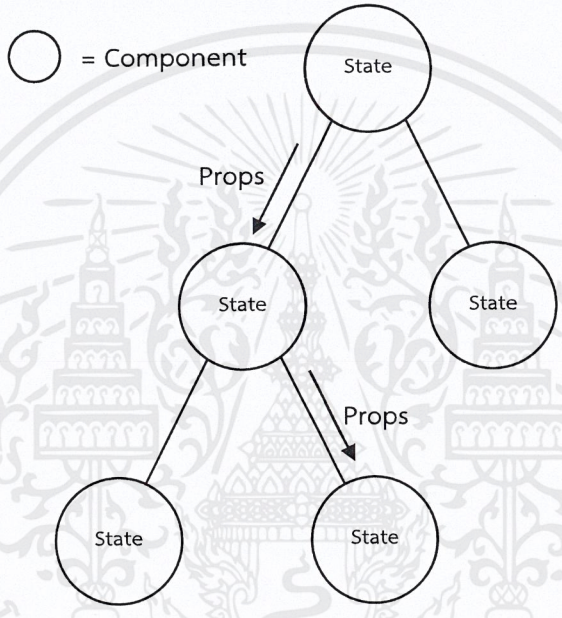
- แก้ไขได้ง่าย

2) State

State นั้นเสมือนเป็นข้อมูลที่มีการใช้แค่ภายใน Component นั้น ๆ ซึ่งการกระทำแก้ข้อมูลของ state จะทำผ่านฟังก์ชัน setState ของ React

3) Props

ข้อมูลที่ถูกส่งต่อจาก Component ชั้นบนลงไปชั้นล่าง เรียกว่า Props (Properties) ทำให้ React มีการไหลของข้อมูลเป็น Unidirectional คือ ไหลไปทางเดียวจากพ่อสู่ลูก



ภาพที่ 2.11 แนวคิดของ React

สำหรับการเขียน Component จะเหมือนกับการเขียน HTML โดย React ใช้สิ่งที่เรียกว่า JSX ในการแสดงผลเว็บไซต์ ซึ่งลักษณะจะเหมือน HTML แต่ต่างตรงที่ต้องเขียนในไฟล์ JavaScript แทนไฟล์ HTML ทำให้สามารถปรับแต่งสิ่งต่าง ๆ ได้มากกว่า

บทที่ 3
วิธีดำเนินโครงงาน

รายงานสหกิจฉบับนี้เป็นรายงานที่กล่าวถึงเว็บแอปพลิเคชัน Knowledge base ที่เป็นแหล่งแบ่งปันแนวทางการแก้ปัญหาที่เกิดระหว่างการทำงาน รวมถึงรายงานความคืบหน้าการทำงานของผู้ใช้ในองค์กรเดียวกัน โดยในบทนี้จะกล่าวถึงความต้องการของระบบ รวมไปถึงการวิเคราะห์และออกแบบโครงสร้างและการทำงานของระบบด้วย โดยรายละเอียดมีดังนี้

3.1 การเก็บรวบรวมความต้องการ (Get Requirement)

จากการที่ผู้จัดทำได้พูดคุยกับผู้นิเทศถึงความต้องการของเว็บแอปพลิเคชันที่จะพัฒนา ทำให้เกิดคุณสมบัติของเว็บแอปพลิเคชัน ดังนี้

1. สามารถทำการเข้าสู่ระบบได้
2. สามารถยืนยันตัวตนและบทบาทของผู้ใช้ได้
3. ผู้ใช้งานเว็บแอปพลิเคชันแบ่งได้เป็น 3 บทบาท โดยเรียงลำดับจากบทบาทที่สามารถเข้าถึงการทำงานของเว็บแอปพลิเคชันจากน้อยไปมากได้ดังนี้

3.1) ผู้เยี่ยมชม (Guest) สามารถดูโพสต์ ดูความคิดเห็นได้ และค้นหาโพสต์ตามเงื่อนไขที่สนใจ

3.2) ผู้ใช้งานในระบบ (User) สามารถเข้าถึงการทำงานได้เหมือนกับผู้เยี่ยมชม และเพิ่มการสร้างโพสต์, แสดงความคิดเห็น, กดถูกใจ, ดูข้อมูลส่วนตัวของตนเอง และแจ้งลบโพสต์ได้

3.3) ผู้ดูแลระบบ (Admin) สามารถเข้าถึงการทำงานได้เหมือนกับผู้ใช้งานในระบบและเพิ่มการดูโพสต์ที่ถูกแจ้งลบเข้ามาทั้งหมดและซ่อนโพสต์ออกจากระบบได้

4. สามารถเรียกดูหัวข้อของโพสต์ทั้งหมดในระบบ
5. สามารถเรียกดูรายละเอียดของแต่ละโพสต์
6. สามารถเรียกดูไฟล์ที่ถูกอัปโหลดในแต่ละโพสต์
7. สามารถเรียกดูความคิดเห็นในแต่ละโพสต์
8. สามารถสร้างโพสต์ที่มีฟอร์มให้กรอกหัวข้อโพสต์และเนื้อหาของโพสต์
9. สามารถจัดหมวดหมู่ของแต่ละโพสต์
10. สามารถอัปโหลดไฟล์ลงในโพสต์ได้ โดยไฟล์ที่ต้องการอัปโหลดเข้าไปมีจำนวนไม่เกิน 5 ไฟล์ใน 1 โพสต์

11. สามารถทำการแก้ไขและปรับปรุงรายละเอียดต่าง ๆ ในโพสต์ที่ตนเองเป็นผู้สร้างขึ้นมาได้

12. สามารถแสดงความคิดเห็นลงในแต่ละโพสต์

13. สามารถกดถูกใจโพสต์และความคิดเห็น

14. สามารถยกเลิกการกดถูกใจโพสต์และความคิดเห็น

15. สามารถทำการค้นหาโพสต์ตามเงื่อนไขที่ต้องการ

16. สามารถแจ้งลบโพสต์ที่ไม่เหมาะสม

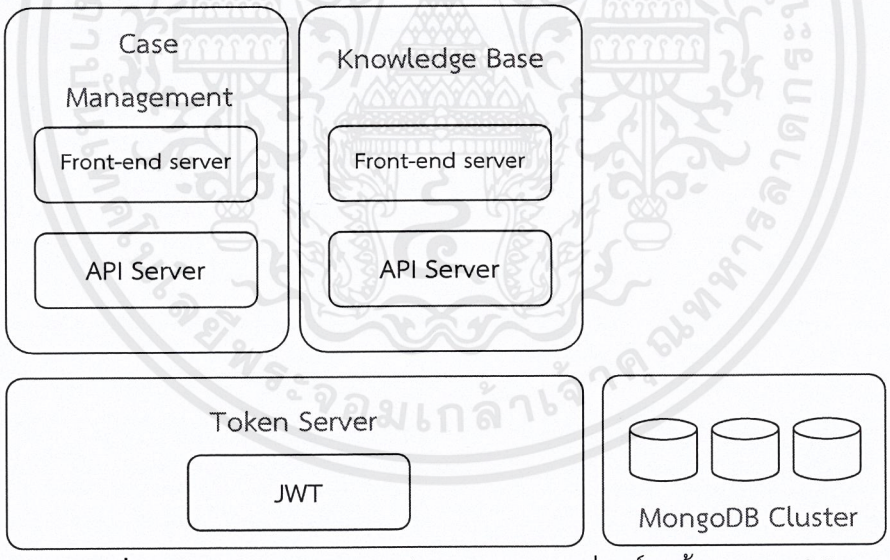
17. สามารถเรียกดูข้อมูลส่วนตัวของตนเองเมื่ออยู่ในระบบ

18. สามารถเรียกดูรายชื่อโพสต์ที่ถูกแจ้งลบเข้ามา

19. สามารถซ่อนโพสต์ออกจากระบบได้

20. แอปพลิเคชันอื่นในเครือข่ายเดียวกันสามารถเรียกใช้ระบบยืนยันตัวตนที่เขียนขึ้นมาได้โดยการเรียกใช้ผ่าน API

3.2 ภาพรวมของระบบ

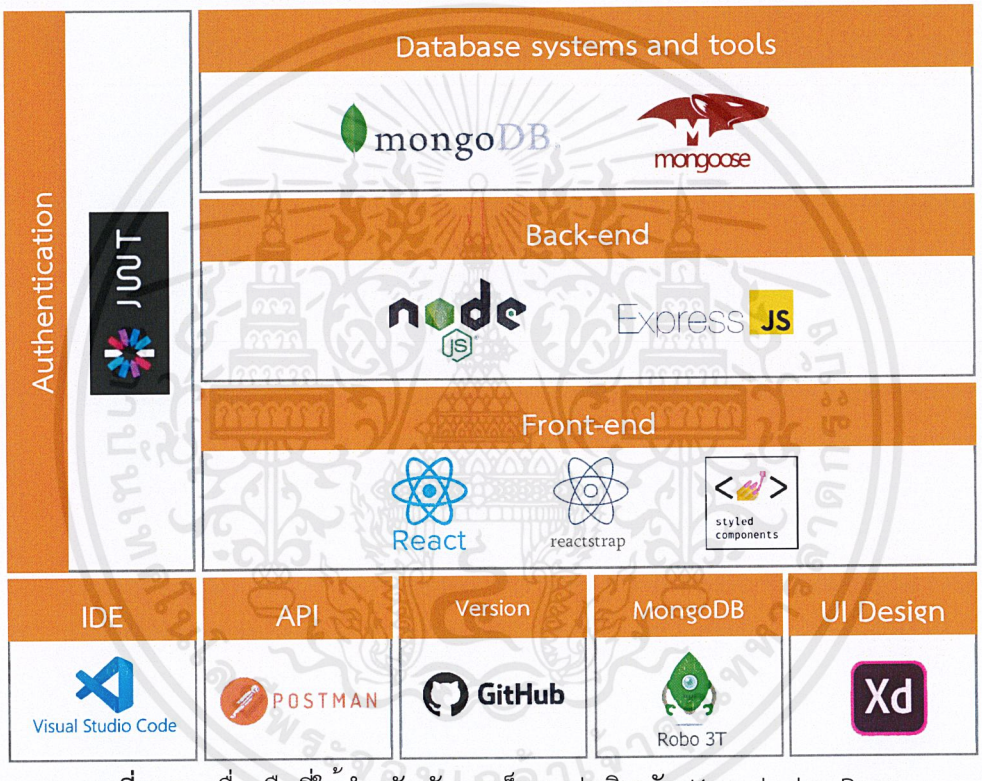


ภาพที่ 3.1 ภาพรวมของระบบเว็บแอปพลิเคชันที่องค์กรตั้งเป้าหมายไว้

จากภาพที่ 3.1 แสดงให้เห็นถึงภาพรวมของระบบเว็บแอปพลิเคชันที่องค์กรวางแผนจะพัฒนา ซึ่งมีเว็บแอปพลิเคชัน 2 ตัว คือ Case Management และ Knowledge Base โดยเว็บแอปพลิเคชันแต่ละตัวจะแบ่งระหว่างเซิร์ฟเวอร์ของ Front-end และ API อย่างชัดเจน เนื่องจากการแยกเซิร์ฟเวอร์ของ API ออกมาทำให้ระบบสามารถในการขยายตัวได้ง่าย ซึ่งเป็นการขยายตัวแนวกว้างที่ใช้วิธีเพิ่มจำนวน node แทนที่จะเพิ่มฮาร์ดแวร์

จะเห็นได้ว่า เซิร์ฟเวอร์ของ Token จะถูกแยกออกมาต่างมาเพื่อที่ระบบการยืนยันตัวตนสามารถใช้ได้กับเว็บแอปพลิเคชันทั้งสองอัน แต่เนื่องจากทางองค์กรณ์ยังไม่เริ่มพัฒนาเว็บแอปพลิเคชัน Case Management ผู้จัดทำจึงยังไม่ได้ทำการแยกเซิร์ฟเวอร์ของ Token ออกมา ส่งผลให้ตอนนี้เซิร์ฟเวอร์ Token และเซิร์ฟเวอร์ API ของแอปพลิเคชัน Knowledge Base ยังเป็นเซิร์ฟเวอร์เดียวกันอยู่ แต่มีการพัฒนาระบบยืนยันตัวตนไว้เป็นโมดูลเพื่อให้สามารถรองรับการแยกเซิร์ฟเวอร์ภายในอนาคต

3.3 ภาพรวมของเว็บแอปพลิเคชัน



ภาพที่ 3.2 เครื่องมือที่ใช้สำหรับพัฒนาเว็บแอปพลิเคชัน Knowledge Base

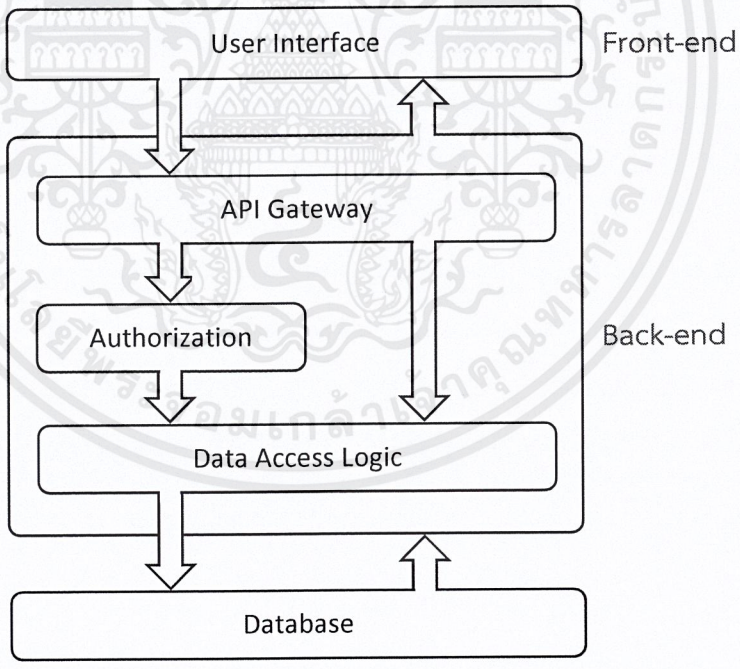
เว็บแอปพลิเคชัน Knowledge Base พัฒนาขึ้นมาจาก Reactwtj ซึ่งเป็น JavaScript Library โดยการใช้งาน React ทำให้เว็บแอปพลิเคชันแสดงผลได้ไวขึ้น เนื่องจากไม่จำเป็นต้องดาวน์โหลดหน้าเว็บแอปพลิเคชันใหม่ทั้งหน้า แต่สามารถดาวน์โหลดเฉพาะส่วนที่มีการเปลี่ยนแปลง นอกจากนั้น React ยังเหมาะสมที่จะนำมาใช้ในเว็บแอปพลิเคชัน Knowledge base เนื่องจาก React มีความสามารถในการใช้ซ้ำ และเว็บแอปพลิเคชันมีการเรียกใช้ Component เดิมมาแสดงผลซ้ำหลายครั้งทำให้ไม่ต้องเสียเวลาในการพัฒนาซ้ำ ๆ ซึ่งผู้จัดทำได้นำ Reactstrap มาใช้เพื่อเพิ่มความสวยงามให้หน้าแสดงผลของเว็บแอปพลิเคชัน และปรับแต่งแต่ละ Component ด้วยการใช้ Styled components

ในส่วนการเก็บข้อมูลผู้จัดทำเลือกใช้ Mongoose ที่เป็น ODM (Object Document Mapping) ของ MongoDB เนื่องจาก mongoDB เก็บข้อมูลในรูปแบบของ NoSQL ซึ่งไม่ต้องเขียนคำสั่ง SQL ทำให้การ query ข้อมูลสะดวกและทำได้ง่าย นอกจากนั้นยังสามารถเก็บข้อมูลที่มีฟิลด์ไม่เหมือนกันได้อีกด้วย

ในส่วนของฝั่งเซิร์ฟเวอร์ ผู้จัดทำเลือกใช้ Node.js มาใช้ในการพัฒนา เนื่องจากเป็น JavaScript ที่ใช้งานง่ายและมี library ให้เลือกใช้อย่างหลากหลาย

โดยการสื่อสารระหว่างเซิร์ฟเวอร์และฝั่งผู้ใช้ ผู้จัดทำเลือกใช้ RESTful API มาเป็นตัวช่วยในการพัฒนาเนื่องจากไม่ต้องการให้ฝั่งผู้ใช้ติดต่อกับฐานข้อมูลได้โดยตรง และการใช้ RESTful API จะส่งข้อมูลกลับมาในรูปแบบ JSON ซึ่งทำให้ข้อมูลที่ได้รับมีขนาดเล็กและใช้ได้ง่าย นอกจากนี้ยังทำการขยายระบบในแนวกว้างได้ง่าย

ในการเข้าสู่ระบบและยืนยันตัวตน ผู้จัดทำได้เลือกใช้ JWT (JSON Web Token) เนื่องจากมีขนาดที่เล็ก ใช้งานได้ง่าย และมีการเข้ารหัสที่ยากต่อการปลอมแปลง อีกทั้งยังสามารถตรวจสอบได้ว่าโทเค็นที่ได้รับมาถูกปลอมแปลงระหว่างการส่งผ่านหรือไม่

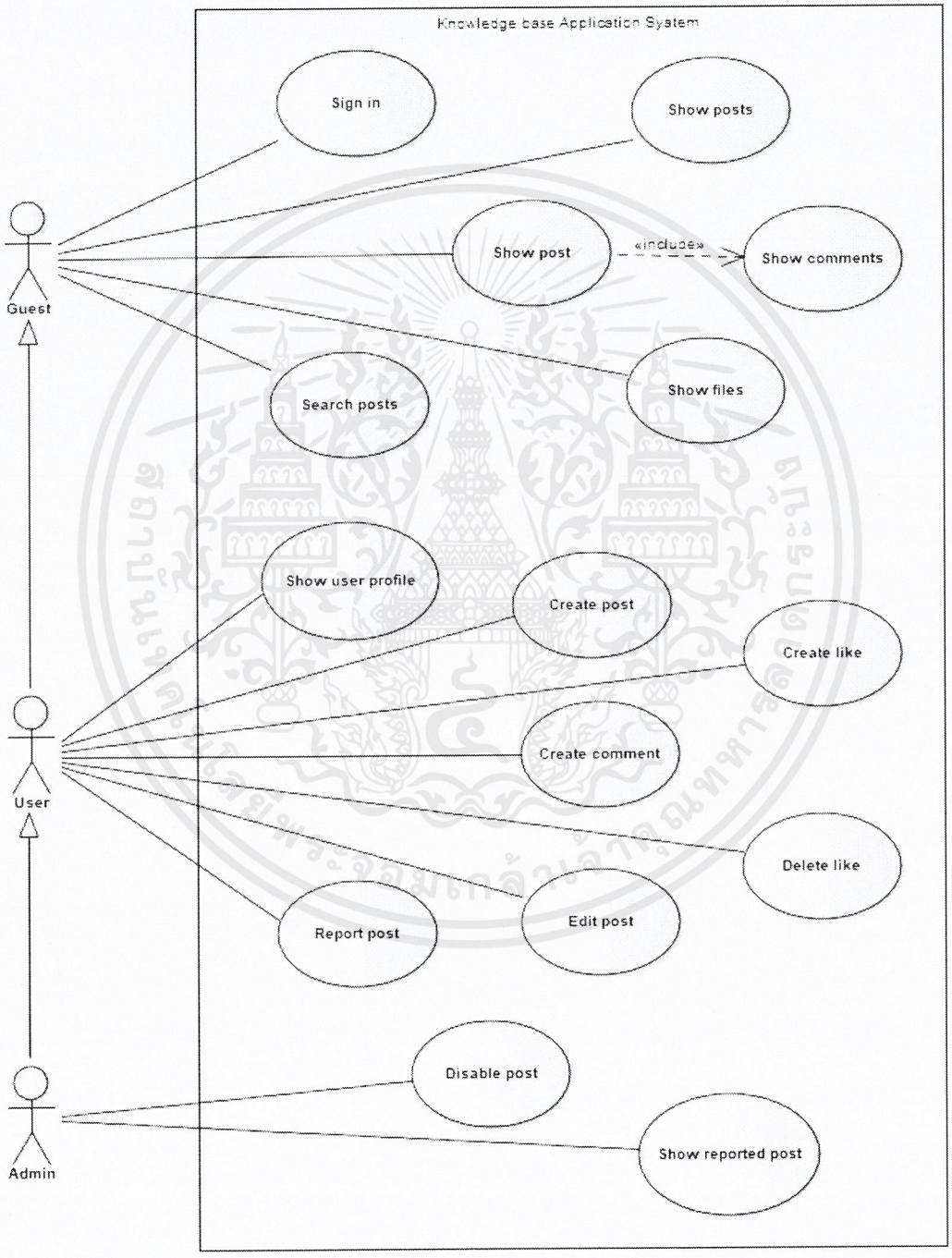


ภาพที่ 3.3 สถาปัตยกรรมของเว็บแอปพลิเคชัน Knowledge Base

3.4 แผนภาพอธิบายโครงสร้างและการทำงานของระบบ

3.4.1 Use case Diagram

เนื่องจากเว็บแอปพลิเคชัน Knowledge Base มีฟังก์ชันที่หลากหลาย ผู้จัดทำจึงออกแบบ Use case ของเว็บแอปพลิเคชันขึ้นมาเพื่อรวบรวมความสามารถของระบบ ดังนี้



ภาพที่ 3.4 Use case ของเว็บแอปพลิเคชัน Knowledge base

ตารางที่ 3.1 Sign in

Use Case Title: Sign in
Primary Actor: Guest
Brief Description: Use Case Sign in มีไว้เพื่อให้ผู้ใช้สามารถเข้าสู่ระบบ โดยใช้ username และ password ที่มีอยู่ ไปตรวจสอบกับ username และ password ที่เก็บไว้ในฐานข้อมูลว่าตรงกันหรือไม่ หากไม่ตรงกันจะต้องทำการกรอก username หรือ password ใหม่ หากตรงกันกับในฐานข้อมูลจะสามารถเข้าสู่ระบบได้
Related Use Case: -
Main Success Scenario: 1. ผู้ใช้เข้าสู่จอภาพเข้าสู่ระบบโดยการคลิกปุ่ม Sign in, กดปุ่ม New Post เมื่อยังไม่ได้เข้าสู่ระบบ หรือกดถูกใจเมื่อยังไม่ได้เข้าสู่ระบบ 2. ผู้ใช้กรอก username ที่ช่อง Username ในฟอร์ม 3. ผู้ใช้กรอก username ที่ช่อง Password ในฟอร์ม 4. ผู้ใช้คลิกปุ่ม Sign in
Extensions: 4. ผู้ใช้ไม่สามารถเข้าสู่ระบบได้ เนื่องจาก username หรือ password ที่กรอกไปไม่ตรงกับข้อมูลในฐานข้อมูล 1) ระบบแสดงข้อความแจ้งเตือนว่า username หรือ password ที่กรอกไปไม่ถูกต้อง 2) ผู้ใช้กรอก username หรือ password ใหม่อีกครั้ง 3) ผู้ใช้คลิกปุ่ม Sign in

ตารางที่ 3.2 Show posts

Use Case Title: Show posts
Primary Actor: Guest
Brief Description: Use Case Show posts ทำหน้าที่แสดงหัวข้อของโพสต์, หมวดหมู่ของโพสต์, วันที่สร้างโพสต์, ผู้สร้างโพสต์, จำนวนความคิดเห็นในโพสต์ และจำนวนผู้ใช้ที่กดถูกใจโพสต์ โดยจะนำโพสต์ทั้งหมดที่ไม่ถูกซ่อนมาแสดง โพสต์ทั้งหมดจะถูกแบ่งเป็นหลายหน้าและแสดงหน้าละ 10 โพสต์ เรียงลำดับจากโพสต์ใหม่ที่สุดไปยังโพสต์เก่าที่สุด
Related Use Case: -

Main Success Scenario: 1. ผู้ใช้เข้าสู่จอภาพหลักโดยการคลิกที่หัวข้อ Home บนแถบนำทาง เพื่อดูหัวข้อโพสต์ทั้งหมด 2. ระบบจะแสดงโพสต์หน้าที่ 1 ซึ่งก็คือ 10 โพสต์ที่ถูกสร้างล่าสุด 3. ผู้ใช้คลิกเลขหน้าหรือคลิกปุ่มหน้าแรก, หน้าก่อน, หน้าถัดไป และหน้าสุดท้าย บนแถบเลขหน้าเพื่อเลือกหน้าที่ต้องการดูโพสต์
Extensions: -

ตารางที่ 3.3 Search posts

Use Case Title: Search posts
Primary Actor: Guest
Brief Description: Use Case Search post มีไว้เพื่อค้นหาโพสต์ที่ผู้ใช้ต้องการแล้วนำมาแสดงผ่านฟังก์ชันการทำงาน Show posts โดยสามารถค้นหาได้จากหัวข้อโพสต์, หมวดหมู่, username ผู้ตั้งโพสต์ และวันที่สร้างโพสต์
Related Use Case: -
Main Success Scenario: 1. ผู้ใช้เข้าสู่จอภาพค้นหาโดยคลิกที่หัวข้อ Search บนแถบนำทาง 2. ผู้ใช้เลือกกรอกเงื่อนไขที่ต้องการทำการค้นหา โดยมีหัวข้อให้เลือกดังนี้ 1) ผู้ใช้กรอกหัวข้อของโพสต์ที่ต้องการค้นหาลงในช่อง Title 2) ผู้ใช้คลิกรายการ Category เพื่อเลือกหมวดหมู่ของโพสต์ที่ต้องการค้นหา 3) ผู้ใช้กรอก username ของผู้ตั้งโพสต์ที่ต้องการค้นหาลงในช่อง From user 4) ผู้ใช้คลิกรายการ From Date และ To Date เพื่อกำหนดช่วงเวลาของการสร้างโพสที่ต้องการค้นหา 3. ผู้ใช้คลิกปุ่ม Submit 4. ระบบเรียกใช้ฟังก์ชันการทำงาน Show posts เพื่อแสดงโพสต์ที่ค้นหาออกมาได้ตามเงื่อนไขที่ระบุไว้
Extensions: 3. ผู้ใช้ไม่สามารถกดปุ่ม Submit ได้ เนื่องจากยังไม่ได้กรอกเงื่อนไขที่ต้องการทำการค้นหา 1) ผู้ใช้เลือกกรอกหัวข้อที่ต้องการค้นหาในขั้นตอนที่ 2 2) ผู้ใช้คลิกปุ่ม Submit

ตารางที่ 3.4 Show post

Use Case Title: Show post
Primary Actor: Guest
Brief Description: Use Case Show post มีไว้เพื่อแสดงรายละเอียดของแต่ละโพสต์ โดยจะแสดงหัวข้อโพสต์, หมวดหมู่ของโพสต์, username ของผู้โพสต์, วันที่สร้างโพสต์, เนื้อหาในโพสต์, ไฟล์ที่ถูกอัปโหลดลงในโพสต์, จำนวนผู้ที่กดถูกใจโพสต์, วันที่อัปเดตโพสต์ล่าสุด และความคิดเห็นในโพสต์
Related Use Case: Show comments
Main Success Scenario: 1. ผู้ใช้เข้าสู่จอภาพโพสต์ได้โดยการคลิกที่หัวข้อโพสต์จากจอภาพหลักหรือจอภาพผลการค้นหา 2. ระบบจะแสดงรายละเอียดของโพสต์และความคิดเห็นในโพสต์ที่เลือก
Extensions: -

ตารางที่ 3.5 Show files

Use Case Title: Show files
Primary Actor: Guest
Brief Description: Use Case Show files มีไว้เพื่อแสดงไฟล์ที่ถูกอัปโหลดลงในโพสต์ โดยสามารถเรียกดูแต่ละไฟล์ได้โดยการคลิกชื่อไฟล์ที่ต้องการ
Related Use Case: Show post
Main Success Scenario: 1. ระบบแสดงจอภาพโพสต์ตาม Use Case Show post 2. ผู้ใช้คลิกที่ชื่อไฟล์ที่ต้องการเรียกดู 3. ระบบจะเปิดไฟล์ที่เลือกในแท็บใหม่ของบราวเซอร์
Extensions: -

ตารางที่ 3.6 Show comments

Use Case Title: Show comments
Primary Actor: -
Brief Description: Use Case Show comments มีไว้เพื่อแสดงรายละเอียดของแต่ละความคิดเห็น โดยจะแสดงลำดับของความคิดเห็น, เนื้อหาในความคิดเห็น, username ของผู้แสดงความคิดเห็น, วันที่แสดงความคิดเห็น และจำนวนผู้ที่กดถูกใจความคิดเห็น โดยจะนำความคิดเห็นทั้งหมดในโพสต์ที่เลือกมาแสดง ซึ่งความคิดเห็นทั้งหมดจะถูกแบ่งเป็นหลายหน้าและแสดงหน้าละ 10 ความคิดเห็น เรียงลำดับจากความคิดเห็นเก่าที่สุดไปยังความคิดเห็นใหม่ที่สุด
Related Use Case: -
Main Success Scenario: 1. ระบบแสดงจอภาพโพสต์ตาม Use Case Show post 2. ระบบจะแสดงรายละเอียดของแต่ละความคิดเห็นในโพสต์ที่เลือก 3. ผู้ใช้คลิกเลขหน้าหรือคลิกปุ่มหน้าแรก, หน้าก่อน, หน้าถัดไป และหน้าสุดท้าย บนแถบเลขหน้าเพื่อเลือกหน้าของความคิดเห็นที่ต้องการ
Extensions: -

ตารางที่ 3.7 Create post

Use Case Title: Create post
Primary Actor: User
Brief Description: Use Case Create post มีไว้เพื่อให้ผู้ใช้สร้างโพสต์ใหม่และบันทึกข้อมูลของโพสต์ลงในฐานข้อมูล โดยสามารถกรอกหัวข้อโพสต์, หมวดหมู่โพสต์, เนื้อหาของโพสต์ และสามารถแนบไฟล์ลงไปในโพสต์ได้สูงสุด 5 ไฟล์
Related Use Case: -
Main Success Scenario: 1. ผู้ใช้เข้าสู่จอภาพสร้างโพสต์โดยการคลิกที่ปุ่ม New Post ในจอภาพหลัก 2. ผู้ใช้กรอกหัวข้อโพสต์ลงในช่อง Title (จำเป็นต้องกรอก) 3. ผู้ใช้คลิกรายการ Category เพื่อเลือกหมวดหมู่ของโพสต์ที่ต้องการสร้าง (จำเป็นต้องเลือก)

4. ผู้ใช้กรอกเนื้อหาของโพสต์ลงในช่อง Detail (จำเป็นต้องกรอก) 5. ผู้ใช้คลิกที่ปุ่ม Browse ในช่อง Attachment 6. ผู้ใช้คลิกเลือกไฟล์ที่ต้องการอัปโหลด 7. ผู้ใช้คลิกปุ่ม Open เพื่อยืนยันการเลือกไฟล์ที่ต้องการอัปโหลด 8. ผู้ใช้คลิกปุ่ม Submit เพื่อยืนยันการสร้างโพสต์ 9. ระบบบันทึกข้อมูลที่กรอกไปลงในฐานข้อมูล 10. ระบบแสดงจอภาพโพสต์ที่ถูกสร้างใหม่ ตาม Use case Show post
Extensions: 1. ผู้ใช้ไม่สามารถเข้าสู่จอภาพสร้างโพสต์ได้ เนื่องจากยังไม่ได้เข้าสู่ระบบ 1) ผู้ใช้คลิกที่ปุ่ม Sign in บนแถบนำทาง 2) ผู้ใช้ทำการเข้าสู่ระบบตาม Use case Sign in 3) ระบบแสดงจอภาพสร้างโพสต์ 8. ผู้ใช้ไม่สามารถคลิกปุ่ม Submit ได้ เนื่องจากกรอกข้อมูลที่จำเป็นไม่ครบถ้วน 1) ผู้ใช้กรอกข้อมูลจากขั้นตอนที่ 2 ถึง 4 ลงไปในช่องที่ระบุให้ครบถ้วน 2) ผู้ใช้คลิกปุ่ม Submit เพื่อยืนยันการสร้างโพสต์ 3) ระบบบันทึกข้อมูลที่กรอกไปลงในฐานข้อมูล 4) ระบบแสดงจอภาพโพสต์ที่ถูกสร้างใหม่ ตาม Use case Show post 8. ผู้ใช้ไม่สามารถคลิกปุ่ม Submit ได้ เนื่องจากเลือกไฟล์ที่อัปโหลดจำนวนมากกว่า 5 ไฟล์ 1) ระบบแสดงข้อความแจ้งเตือนว่าไฟล์ที่เลือกมีจำนวนมากกว่า 5 ไฟล์ 2) ผู้ใช้เลือกไฟล์ที่ต้องการอัปโหลดใหม่ตาม Use case Upload files 3) ผู้ใช้คลิกปุ่ม Submit เพื่อยืนยันการสร้างโพสต์ 4) ระบบบันทึกข้อมูลที่กรอกไปลงในฐานข้อมูล 5) ระบบแสดงจอภาพโพสต์ที่ถูกสร้างใหม่ ตาม Use case Show post

ตารางที่ 3.8 Create comment

Use Case Title: Create comment
Primary Actor: User
Brief Description: Use Case Create comment มีไว้เพื่อทำการเพิ่มความคิดเห็นใหม่ในแต่ละโพสต์ และบันทึกข้อมูลของความคิดเห็นลงในฐานข้อมูล โดยสามารถกรอกได้เพียงรายละเอียดของความคิดเห็นเท่านั้น

Related Use Case: -
Main Success Scenario: 1. ระบบแสดงจอภาพโพสต์ตาม Use case Show post 2. ผู้ใช้กรอกความคิดเห็นที่ต้องการลงในช่องความคิดเห็นทางด้านล่างของจอภาพ 3. ผู้ใช้คลิกปุ่ม Submit เพื่อยืนยันการแสดงความคิดเห็นใหม่ 4. ระบบบันทึกข้อมูลที่กรอกไปลงในฐานข้อมูล
Extensions: 2. ระบบไม่แสดงช่องความคิดเห็น เนื่องจากยังไม่ได้เข้าสู่ระบบ 1) ผู้ใช้คลิกที่ปุ่ม Sign in บนแถบนำทาง 2) ผู้ใช้ทำการเข้าสู่ระบบตาม Use case Sign in 3) ระบบแสดงจอภาพโพสต์ที่มีช่องความคิดเห็น 3. ผู้ใช้ไม่สามารถคลิกปุ่ม Submit ได้ เนื่องจากยังไม่ได้กรอกความคิดเห็นลงในช่องความคิดเห็น 1) ผู้ใช้กรอกความคิดเห็นที่ต้องการลงในช่องความคิดเห็นทางด้านล่างของจอภาพ 2) ผู้ใช้คลิกปุ่ม Submit เพื่อยืนยันการแสดงความคิดเห็นใหม่ 3) ระบบบันทึกข้อมูลที่กรอกไปลงในฐานข้อมูล

ตารางที่ 3.9 Edit post

Use Case Title: Edit post
Primary Actor: User
Brief Description: Use Case Edit post มีไว้เพื่อทำการแก้ไขโพสต์ที่ถูกสร้างโดยผู้ใช้ที่อยู่ในระบบคนปัจจุบัน โดยสามารถแก้ไขหัวข้อโพสต์, หมวดหมู่ของโพสต์, เนื้อหาในโพสต์, ไฟล์ที่ถูกอัปโหลดลงในโพสต์ และระบบจะแก้ไขวันที่อัปเดตโพสต์ล่าสุดเป็นเวลาที่กด Submit โพสต์โดยอัตโนมัติ
Related Use Case: Upload file
Main Success Scenario: 1. ระบบแสดงจอภาพโพสต์ตาม Use case Show post 2. ผู้ใช้คลิกปุ่มสีส้มที่อยู่หลังหัวข้อโพสต์จากนั้นระบบจะแสดงรายการเมนูที่เลือกได้ 3. ผู้ใช้คลิกปุ่ม Edit 4. ผู้ใช้แก้ไขหัวข้อโพสต์ในช่อง Title (จำเป็นต้องกรอก) 5. ผู้ใช้คลิกรายการ Category เพื่อแก้ไขหมวดหมู่ของโพสต์ (จำเป็นต้องเลือก)

6. ผู้ใช้แก้ไขเนื้อหาของโพสต์ในช่อง Detail (จำเป็นต้องกรอก) 7. ผู้ใช้คลิกที่ปุ่ม Browse ในช่อง Attachment 8. ผู้ใช้คลิกเลือกไฟล์ที่ต้องการอัปโหลด 9. ผู้ใช้คลิกปุ่ม Open เพื่อยืนยันการเลือกไฟล์ที่ต้องการอัปโหลด 10. ผู้ใช้คลิกปุ่ม Delete หลังชื่อไฟล์ หากต้องการลบไฟล์นั้น 11. ผู้ใช้คลิกปุ่ม Submit เพื่อยืนยันการแก้ไขโพสต์ 12. ระบบบันทึกข้อมูลของโพสต์ที่ถูกแก้ไขลงในฐานข้อมูล 13. ระบบแสดงหน้าโพสต์ที่ถูกแก้ไข ตาม Use case Show post
Extensions: 2. ระบบไม่แสดงปุ่มสีส้มที่อยู่หลังหัวข้อโพสต์เนื่องจากยังไม่มีการเข้าสู่ระบบ 1) คลิกที่ปุ่ม Sign in บนแถบนำทางเพื่อทำการเข้าสู่ระบบ 2) ระบบแสดงจอภาพเข้าสู่ระบบ 3) ทำการเข้าสู่ระบบตาม Use case Sign in 4) ระบบแสดงจอภาพโพสต์ตาม Use case Show post 11. ผู้ใช้ไม่สามารถคลิกปุ่ม Submit ได้ เนื่องจากกรอกข้อมูลที่จำเป็นไม่ครบถ้วน 1) ผู้ใช้กรอกข้อมูลจากขั้นตอนที่ 4 ถึง 6 ลงไปในช่องที่ระบุให้ครบถ้วน 2) ผู้ใช้คลิกปุ่ม Submit เพื่อยืนยันการแก้ไขโพสต์ 3) ระบบบันทึกข้อมูลของโพสต์ที่ถูกแก้ไขลงในฐานข้อมูล 4) ระบบแสดงจอภาพโพสต์ที่ถูกแก้ไข ตาม Use case Show post 10. ผู้ใช้ไม่สามารถคลิกปุ่ม Submit ได้ เนื่องจากเลือกไฟล์ที่ต้องการอัปโหลดจำนวนมากกว่า 5 ไฟล์ 1) ระบบแสดงข้อความแจ้งเตือนว่าไฟล์ที่เลือกมีจำนวนมากกว่า 5 ไฟล์ 2) ผู้ใช้เลือกไฟล์ที่ต้องการอัปโหลดใหม่ตาม Use case Upload files 3) ผู้ใช้คลิกปุ่ม Submit เพื่อยืนยันการแก้ไขโพสต์ 4) ระบบบันทึกข้อมูลของโพสต์ที่ถูกแก้ไขลงในฐานข้อมูล 5) ระบบแสดงจอภาพโพสต์ที่ถูกแก้ไข ตาม Use case Show post

ตารางที่ 3.10 Create like

Use Case Title: Create like
Primary Actor: User
Brief Description: Use Case Create like มีไว้เพื่อให้ผู้ใช้ทำการคลิกถูกใจโพสต์หรือความคิดเห็นที่ชื่นชอบ โดยคลิกที่ไอคอนหัวใจสีขาว จากนั้นระบบจะบันทึกข้อมูลการกดถูกใจลงในฐานข้อมูล
Related Use Case: -
Main Success Scenario: 1. ระบบแสดงจอภาพโพสต์ตาม Use case Show post 2. ผู้ใช้คลิกไอคอนหัวใจสีขาวที่โพสต์หรือความคิดเห็นที่ชื่นชอบ 3. ระบบบันทึกข้อมูลการกดถูกใจของโพสต์หรือความคิดเห็นที่ถูกกดถูกใจลงในฐานข้อมูล 4. ไอคอนหัวใจสีขาวเปลี่ยนการแสดงผลเป็นไอคอนหัวใจสีแดง
Extensions: 2. หากยังไม่ได้เข้าสู่ระบบ เมื่อคลิกที่ไอคอนหัวใจสีขาวแล้วระบบจะแสดงจอภาพเข้าสู่ระบบขึ้นมา 1) ทำการเข้าสู่ระบบตาม Use case Sign in 2) ระบบแสดงจอภาพโพสต์ตาม Use case Show post

ตารางที่ 3.11 Delete like

Use Case Title: Delete like
Primary Actor: User
Brief Description: Use Case Delete like มีไว้เพื่อให้ผู้ใช้ยกเลิกการถูกใจโพสต์หรือความคิดเห็นที่เคยถูกใจไว้ โดยคลิกที่ไอคอนหัวใจสีแดง ระบบจะลบข้อมูลการกดถูกใจออกจากฐานข้อมูล
Related Use Case: -
Main Success Scenario: 1. ระบบแสดงจอภาพโพสต์ตาม Use case Show post 2. ผู้ใช้คลิกไอคอนหัวใจสีแดงที่โพสต์หรือความคิดเห็นที่ต้องการยกเลิกการกดถูกใจ 3. ระบบลบข้อมูลการกดถูกใจของโพสต์หรือความคิดเห็นที่เคยกดถูกใจออกจากฐานข้อมูล 4. ไอคอนหัวใจสีแดงเปลี่ยนการแสดงผลเป็นไอคอนหัวใจสีขาว
Extensions:

2. หากยังไม่ได้เข้าสู่ระบบ ระบบจะแสดงไอคอนหัวใจสีขาวแทนไอคอนหัวใจสีแดง และจะยกเลิกการกดถูกใจไม่ได้ 1) ทำการเข้าสู่ระบบตาม Use case Sign in 2) ระบบแสดงจอภาพโพสต์ตาม Use case Show post

ตารางที่ 3.12 Show profile

Use Case Title: Show user profile
Primary Actor: User
Brief Description: Use Case Show user profile ทำหน้าที่แสดงข้อมูลส่วนตัวของผู้ที่อยู่ในระบบคนปัจจุบัน โดยจะมีการแสดง username, ชื่อจริง, นามสกุล, อีเมล และบทบาทของผู้ใช้ (user / admin)
Related Use Case: -
Main Success Scenario: 1. ผู้ใช้เข้าสู่จอภาพข้อมูลส่วนตัวโดยการคลิกที่ username ของตนเองทางด้านขวาของแถบนำทาง 2. ระบบจะแสดงข้อมูลส่วนตัวของผู้ใช้
Extensions: 1. ระบบแสดงปุ่ม Sign in แทนที่จะแสดง username ของผู้ใช้เนื่องจากยังไม่มีการเข้าสู่ระบบ 1) คลิกที่ปุ่ม Sign in บนแถบนำทางเพื่อทำการเข้าสู่ระบบ 2) ระบบแสดงจอภาพเข้าสู่ระบบ 3) ทำการเข้าสู่ระบบตาม Use case Sign in 4) ระบบจะแสดงจอภาพข้อมูลส่วนตัวของผู้ใช้

ตารางที่ 3.13 Report post

Use Case Title: Report post
Primary Actor: User
Brief Description: Use Case Report post มีไว้เพื่อให้ผู้ใช้ทำการแจ้งลบโพสต์ที่ไม่เหมาะสม โดยการแจ้งลบจำเป็นต้องกรอกเหตุผลที่โพสต์ไม่เหมาะสมลงไปด้วย
Related Use Case: -

Main Success Scenario: 1. ระบบแสดงจอภาพโพสต์ตาม Use case Show post 2. ผู้ใช้คลิกปุ่มสีส้มที่อยู่หลังหัวข้อโพสต์จากนั้นระบบจะแสดงรายการเมนูที่เลือกได้ 3. ผู้ใช้คลิกปุ่ม Report 4. ระบบแสดงหน้าต่างการแจ้งลบโพสต์ขึ้นมา 5. ผู้ใช้กรอกเหตุผลที่ต้องการแจ้งลบโพสต์ 6. ผู้ใช้คลิกปุ่ม Done เพื่อยืนยันการแจ้งลบโพสต์ 7. ระบบจัดเก็บข้อมูลการแจ้งลบโพสต์ลงในฐานข้อมูล
Extensions: 2. ระบบไม่แสดงปุ่มสีส้มที่อยู่หลังหัวข้อโพสต์เนื่องจากยังไม่มีการเข้าสู่ระบบ 1) คลิกที่ปุ่ม Sign in บนแถบนำทางเพื่อทำการเข้าสู่ระบบ 2) ระบบแสดงจอภาพเข้าสู่ระบบ 3) ทำการเข้าสู่ระบบตาม Use case Sign in 4) ระบบแสดงจอภาพโพสต์ตาม Use case Show post

ตารางที่ 3.14 Show reported post

Use Case Title: Show reported post
Primary Actor: Admin
Brief Description: Use Case Show reported post ทำหน้าที่แสดงรายชื่อโพสต์ที่ถูกแจ้งลบเข้ามา หากคลิกที่ชื่อโพสต์จะสามารถไปยังจอภาพโพสต์นั้นได้ โดยผู้ที่สามารถใช้การทำงานนี้มีเพียงผู้ที่มีสถานะเป็นผู้ดูแลระบบเท่านั้น
Related Use Case: -
Main Success Scenario: 1. ผู้ใช้เข้าสู่จอภาพโพสต์ที่ถูกแจ้งลบโดยการคลิกที่ปุ่ม Reported post บนแถบนำทาง 2. ระบบจะแสดงโพสต์ที่ถูกแจ้งลบเข้ามาและผู้ดูแลระบบยังไม่ได้ทำการซ่อนออกจากระบบ โดยรายละเอียดจะมีเพียงชื่อโพสต์และเหตุผลที่ถูกแจ้งลบเข้ามา
Extensions: -

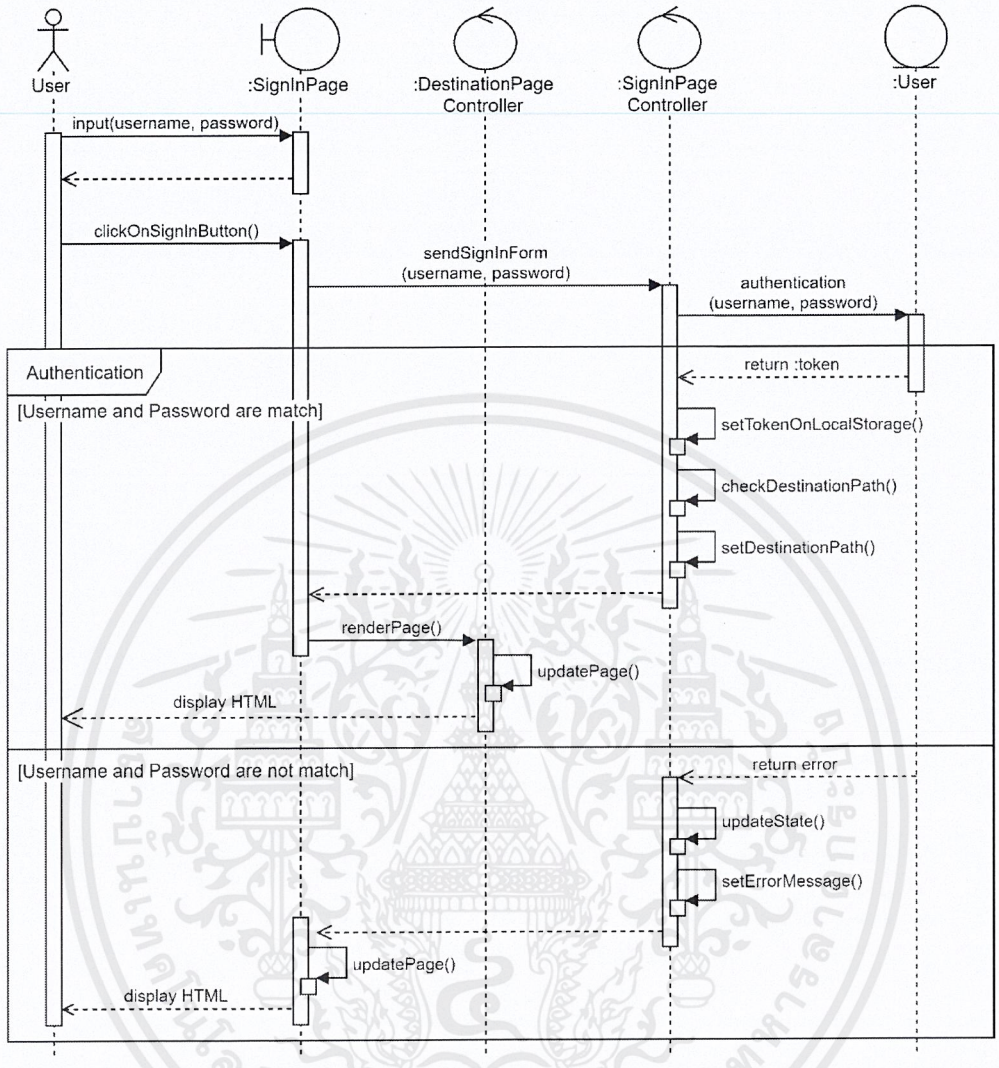
ตารางที่ 3.15 Disable post

Use Case Title: Disable post
Primary Actor: Admin
Brief Description: Use Case Disable post มีไว้เพื่อให้ผู้ดูแลระบบซ่อนโพสต์ออกจากระบบ โดยหลังจากซ่อนแล้ว โพสต์ที่ถูกซ่อนจะไม่แสดงในหน้ารวมโพสต์ และไม่สามารถดูโพสต์ที่ถูกซ่อนได้
Related Use Case: -
Main Success Scenario: 1. ระบบแสดงจอภาพโพสต์ตาม Use case Show post 2. ผู้ใช้คลิกปุ่มสีส้มที่อยู่หลังหัวข้อโพสต์จากนั้นระบบจะแสดงรายการเมนูที่เลือกได้ 3. ผู้ใช้คลิกปุ่ม Disable 4. ระบบแสดงหน้าต่างยืนยันการซ่อนโพสต์ขึ้นมา 5. ผู้ใช้คลิกปุ่ม Yes เพื่อยืนยันการแจ้งซ่อนโพสต์ 6. ระบบแก้ไขสถานะของโพสต์ที่ถูกซ่อนให้เป็น disable
Extensions: -

3.4.2 Sequence Diagram

เนื่องจากเว็บแอปพลิเคชัน Knowledge base มีวัตถุที่หลากหลาย ผู้จัดทำจึงทำการออกแบบระบบโดยใช้ Sequence Diagram เพื่อให้เห็นถึงความสัมพันธ์และฟังก์ชันที่ใช้ในแต่ละวัตถุ และยังสามารถดูลำดับการทำงานของแต่ละ Use case ได้ด้วย โดยผู้จัดทำออกแบบ Sequence Diagram ตาม Use case ได้ดังนี้

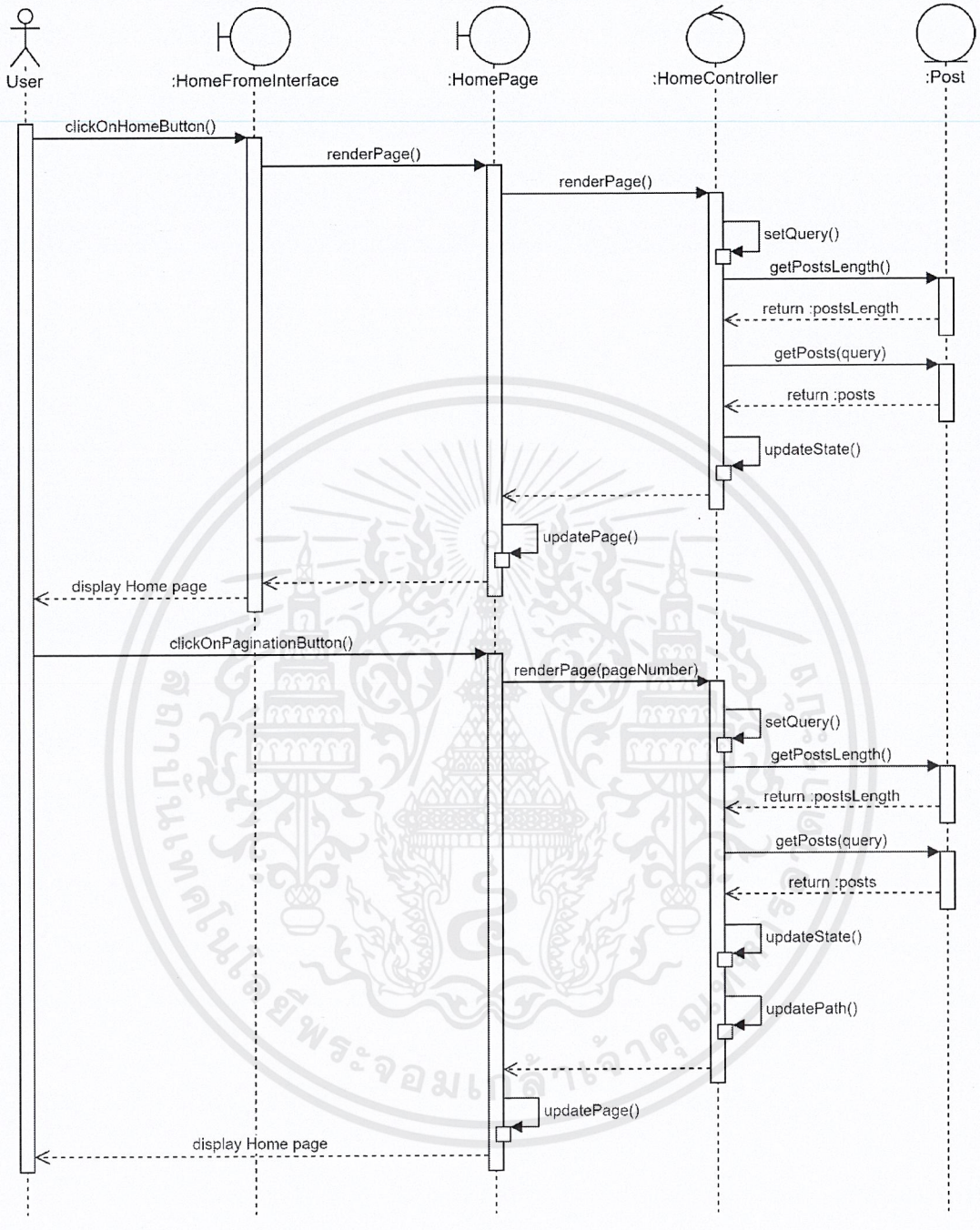
1. Sequence Diagram ของ Use case “Sign in”



ภาพที่ 3.5 Sequence Diagram ของ Use case “Sign in”

จากภาพที่ 3.5 ที่แสดงการทำงานของ Use case Sign in หลังจากที่ผู้ใช้กรอก username, password และคลิกปุ่ม Sign in แล้ว ระบบจะทำการส่งข้อมูลในฟอร์มไปเพื่อเรียกใช้ฟังก์ชัน authentication ที่ใช้ทำการตรวจสอบว่า username และ password ที่ผู้ใช้กรอกเข้ามาถูกต้องหรือไม่ โดยถ้า username และ password ที่กรอกเข้ามาตรงกับ username และ password ที่อยู่ในฐานข้อมูล ระบบจะส่งโทเค็นที่ใช้สำหรับการยืนยันตัวตนที่ได้จากฟังก์ชัน authentication มาเก็บไว้ใน Local storage หลังจากนั้นระบบจะตรวจสอบว่าผู้ใช้ต้องการเรียกดูจอภาพใดในเว็บแอปพลิเคชัน แล้วทำการแสดงจอภาพนั้นหน้าแก่ผู้ใช้ แต่หาก username และ password ที่กรอกเข้ามาไม่ตรงกับ username และ password ที่อยู่ในฐานข้อมูล ระบบจะทำการแสดงข้อความให้ผู้ใช้กรอก username และ password ใหม่อีกครั้ง

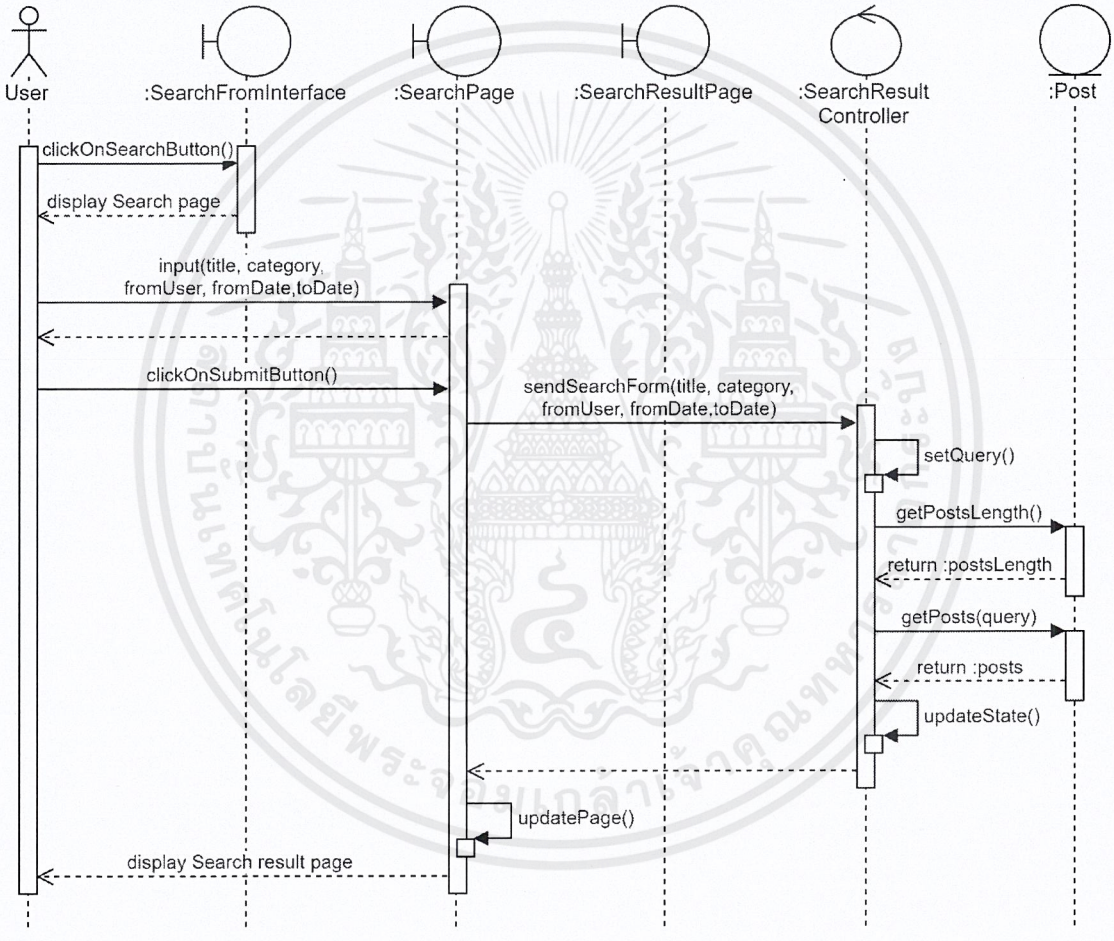
2. Sequence Diagram ของ Use case “Show posts”



ภาพที่ 3.6 Sequence Diagram ของ Use case “Show posts”

จากภาพที่ 3.6 ที่แสดงการทำงานของ Use case Show posts หลังจากที่ผู้ใช้คลิกปุ่ม Home บนแถบนำทางแล้ว ระบบจะเรียกใช้ฟังก์ชัน getPostsLength ที่ส่งค่าจำนวนโพสต์ทั้งหมดกลับมา เพื่อนำจำนวนโพสต์มาคำนวณหาจำนวนหน้าที่จะแสดงบนแถบเลขหน้าที่มีไว้เพื่อให้ผู้ใช้เลือกคลิกเลขหน้าที่ต้องการ โดยโพสต์ที่ถูกเรียกออกมาจะขึ้นอยู่กับ query ซึ่งอ้างอิงเลขหน้าที่ผู้ใช้ต้องการเรียกดู จากนั้นระบบจะทำการเรียกชุดของโพสต์มาจากฐานข้อมูลทีละชุดผ่านฟังก์ชัน getPosts เพื่อนำโพสต์มาแสดงบนจอภาพ

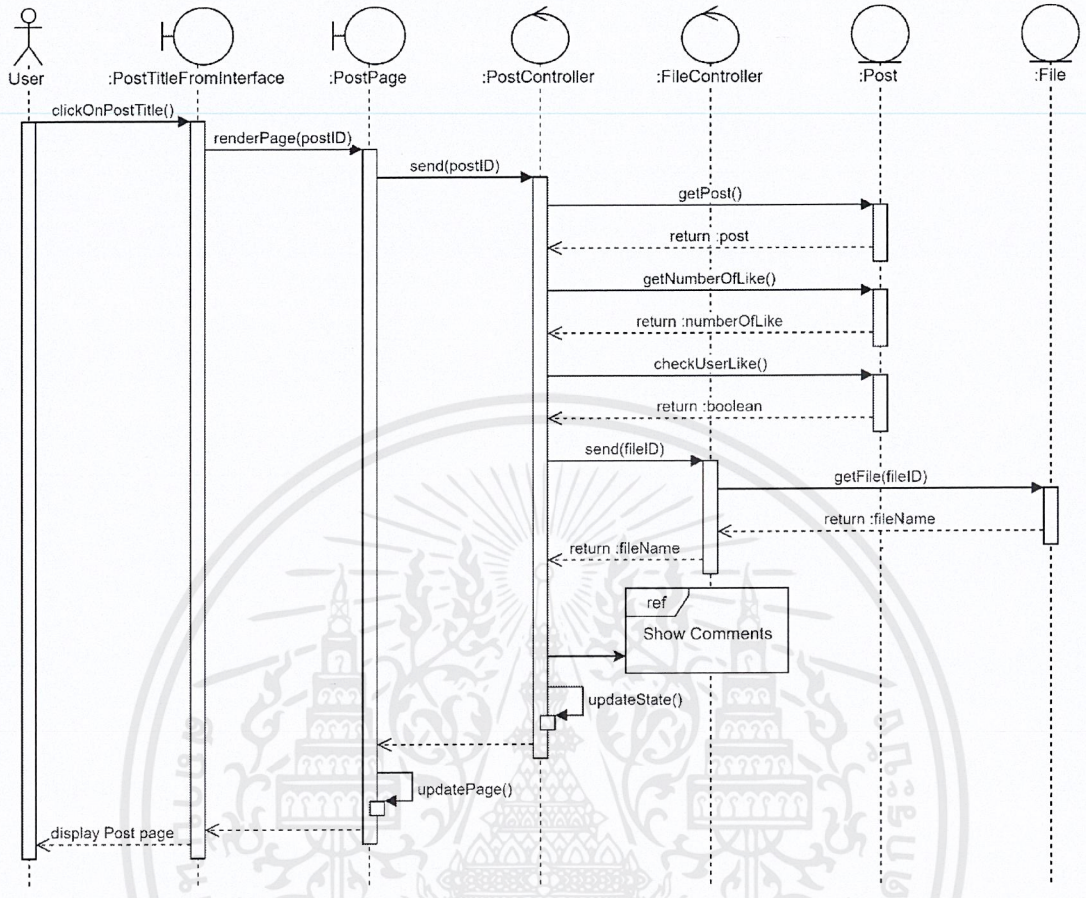
3. Sequence Diagram ของ Use case “Search post”



ภาพที่ 3.7 Sequence Diagram ของ Use case “Search post”

จากภาพที่ 3.7 ที่แสดงการทำงานของ Use case Search post หลังจากที่ผู้ใช้คลิกที่ปุ่ม Search บนแถบนำทางแล้ว ระบบจะทำการแสดงจอภาพค้นหาให้ผู้ใช้กรอกเงื่อนไขที่ต้องการทำการค้นหา โดยเมื่อกรอกเสร็จและคลิกที่ปุ่ม Search ระบบจะทำการจัดเก็บเงื่อนไขนั้นไว้ในตัวแปร query แล้วนำไปใช้ในการเรียกโพสต์ต่าง ๆ มาแสดงในจอภาพผลการค้นหา ตาม Use case Show posts

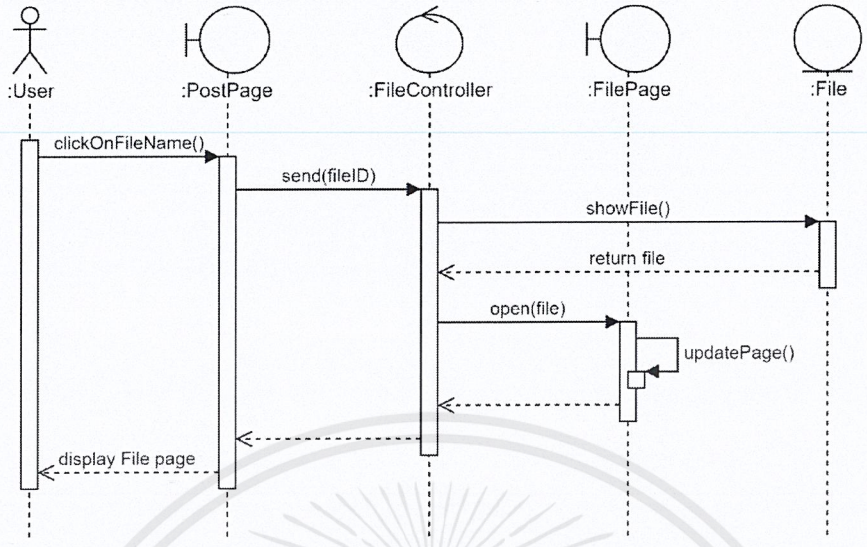
4. Sequence Diagram ของ Use case “Show post”



ภาพที่ 3.8 Sequence Diagram ของ Use case “Show post”

จากภาพที่ 3.8 ที่แสดงการทำงานของ Use case Show post หลังจากที่ผู้ใช้คลิกที่ชื่อโพสต์แล้ว ระบบจะทำการแสดงจอภาพโพสต์ตาม PostID ของชื่อโพสต์ที่คลิกไป จากนั้นระบบจะเรียกใช้ฟังก์ชัน getPost เพื่อเรียกรายละเอียดของแต่ละโพสต์มาแสดง จากนั้นจะเรียกใช้ฟังก์ชัน getNumberOfLike เพื่อเรียกจำนวนผู้ที่กดถูกใจมาแสดง และเรียกใช้ฟังก์ชัน checkUserLike เพื่อตรวจสอบว่าผู้ใช้ที่อยู่ในระบบได้ทำการถูกใจโพสต์นี้หรือไม่ จากนั้นระบบจะนำ fileID ที่เก็บไว้ในโพสต์ไปเรียกรายละเอียดของไฟล์แต่ละไฟล์ผ่านฟังก์ชัน getFile แล้วระบบจะทำการเรียกความคิดเห็นมาแสดงตาม Use case Show comments จากนั้นระบบจะทำการนำข้อมูลที่ได้มาไปเก็บไว้ใน state แล้วทำการแสดงผลจอภาพโพสต์แก่ผู้ใช้

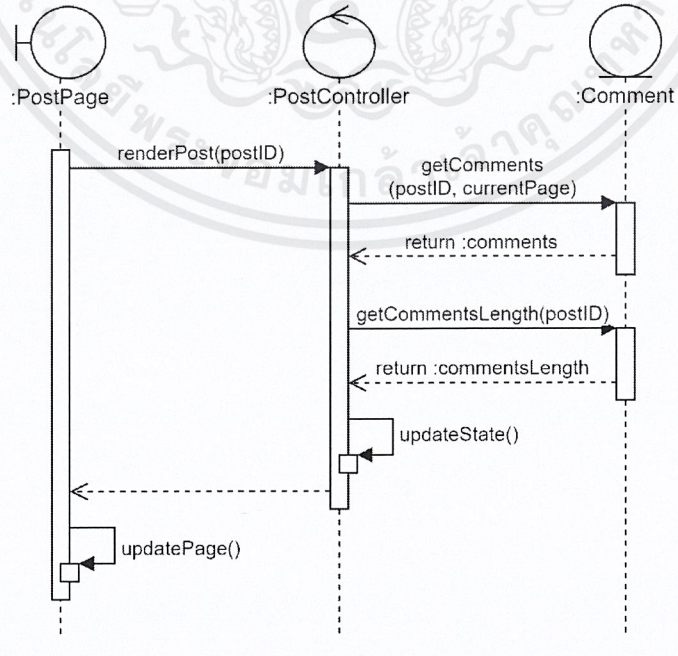
5. Sequence Diagram ของ Use case “Show files”



ภาพที่ 3.9 Sequence Diagram ของ Use case “Show files”

จากภาพที่ 3.9 ที่แสดงการทำงานของ Use case Show files หลังจากที่ผู้ใช้คลิกที่ชื่อไฟล์บนจอภาพโพสต์ ระบบจะนำ fileID ของไฟล์ที่ผู้ใช้เลือกไปเรียกใช้ฟังก์ชัน showFile เพื่อให้ได้ข้อมูลของไฟล์ แล้วเรียกใช้ฟังก์ชัน openFile เพื่อเปิดไฟล์ที่เลือกขึ้นมาที่แท็บใหม่บนบราวเซอร์

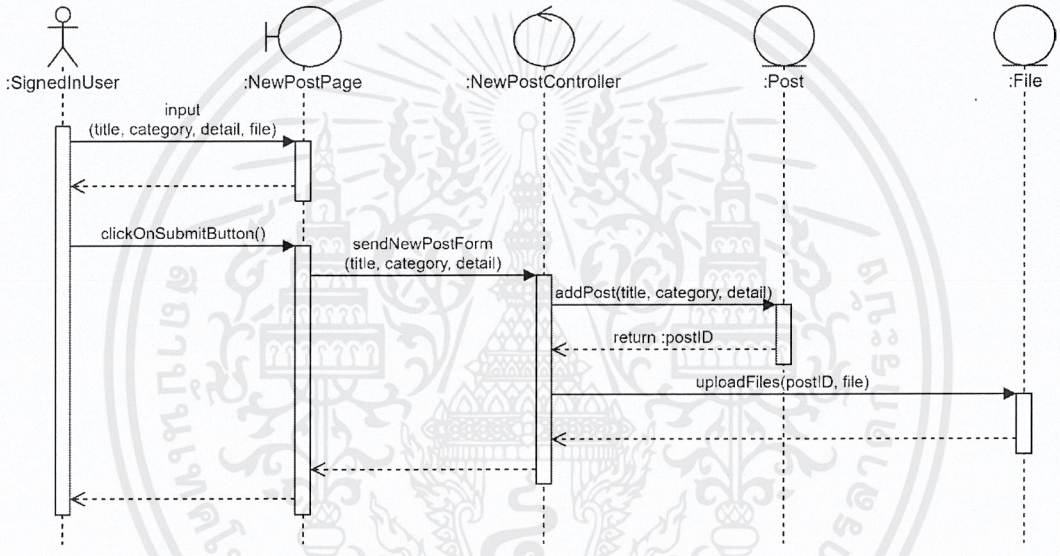
6. Sequence Diagram ของ Use case “Show comments”



ภาพที่ 3.10 Sequence Diagram ของ Use case “Show comments”

จากภาพที่ 3.10 ที่แสดงการทำงานของ Use case Show comments หลังจากที่ผู้ใช้เข้าสู่จอภาพโพสต์แล้ว ระบบจะเรียกข้อมูลของโพสต์มาแสดง ซึ่งมีการเรียกความคิดเห็นที่อยู่ในโพสต์มาแสดงโดยอัตโนมัติ โดยจะเรียกใช้ฟังก์ชัน getCommentsLength เพื่อนำจำนวนความคิดเห็นทั้งหมดในโพสต์มาคำนวณหาจำนวนหน้าที่จะแสดงบนแถบเลขหน้าที่มีไว้เพื่อให้ผู้ใช้เลือกคลิกเลขหน้าความคิดเห็นที่ต้องการ โดยความคิดเห็นที่ถูกเรียกออกมาจะขึ้นอยู่กับ query ซึ่งอ้างอิงจากเลขหน้าที่ผู้ใช้ต้องการเรียกดู จากนั้นระบบจะทำการเรียกชุดของความคิดเห็นมาจากฐานข้อมูลทีละชุดผ่านฟังก์ชัน getComments เพื่อนำความคิดเห็นมาแสดง

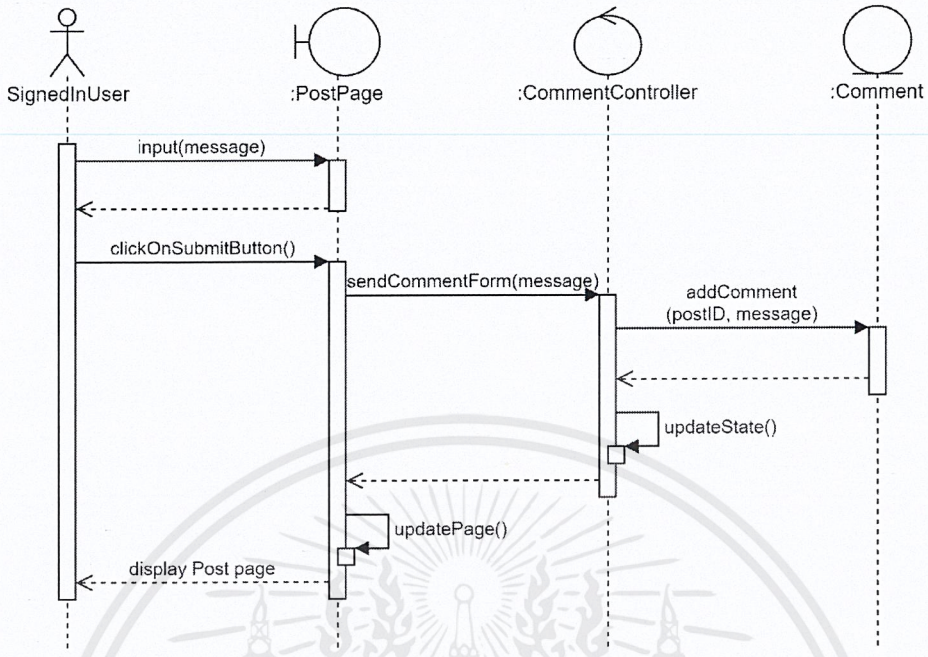
7. Sequence Diagram ของ Use case “Create post”



ภาพที่ 3.11 Sequence Diagram ของ Use case “Create post”

จากภาพที่ 3.11 ที่แสดงการทำงานของ Use case Create post หลังจากที่ผู้ใช้กรอกรายละเอียดของโพสต์ที่ต้องการสร้างใหม่ในจอภาพสร้างโพสต์และคลิกปุ่ม Submit แล้ว ระบบจะเรียกใช้ฟังก์ชัน addPost เพื่อเก็บรายละเอียดของโพสต์ไปยังฐานข้อมูล แล้วส่งค่า postID ของโพสต์ที่สร้างใหม่กลับมาเพื่อนำไปใช้ในฟังก์ชัน uploadFiles ซึ่งทำหน้าที่จัดเก็บไฟล์ลงไปยังฐานข้อมูล

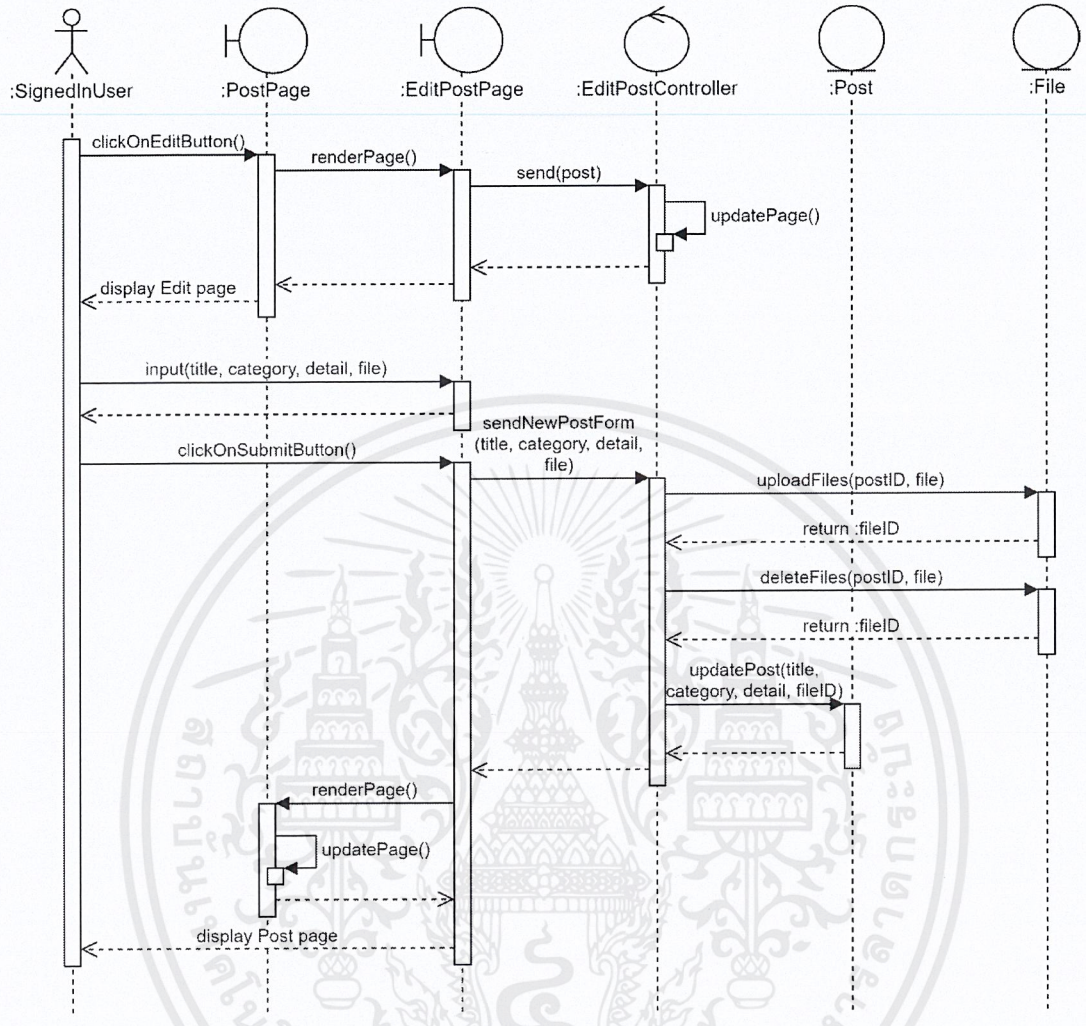
8. Sequence Diagram ของ Use case “Create comment”



ภาพที่ 3.12 Sequence Diagram ของ Use case “Create comment”

จากภาพที่ 3.12 ที่แสดงการทำงานของ Use case Create comment หลังจากที่ผู้ใช้กรอกรายละเอียดของความคิดเห็นที่ต้องการเพิ่มเข้าไปในโพสต์และคลิกปุ่ม Submit แล้ว ระบบจะเรียกใช้ฟังก์ชัน addComment เพื่อบันทึกรายละเอียดของความคิดเห็นไปยังฐานข้อมูล จากนั้นระบบแสดงหน้าโพสต์และความคิดเห็นที่ถูกเพิ่มเข้าไปใหม่

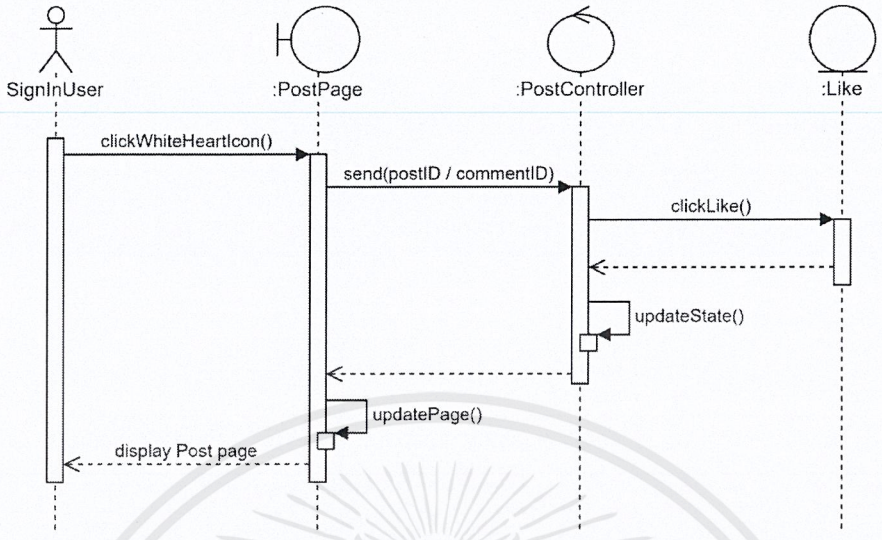
9. Sequence Diagram ของ Use case “Edit post”



ภาพที่ 3.13 Sequence Diagram ของ Use case “Edit post”

จากภาพที่ 3.13 ที่แสดงการทำงานของ Use case Edit post ผู้ใช้สามารถเข้าสู่จอภาพแก้ไขโพสต์ได้เมื่อคลิกปุ่ม Edit บนจอภาพโพสต์ที่ต้องการแก้ไข ระบบจะแสดงหน้าแก้ไขโพสต์นั้นขึ้นมา จากนั้น ผู้ใช้แก้ไขข้อมูลในส่วนที่ต้องการแล้วคลิกที่ปุ่ม Submit หากมีการเพิ่มไฟล์ระบบจะเรียกใช้ฟังก์ชัน uploadFiles เพื่อทำการเพิ่มไฟล์ไปยังฐานข้อมูล หากมีการลบไฟล์ระบบจะเรียกใช้ฟังก์ชัน deleteFiles เพื่อทำการลบไฟล์ออกจากฐานข้อมูล จากนั้นระบบจะเรียกใช้ฟังก์ชัน updatePost เพื่อทำการบันทึกส่วนที่ถูกแก้ไขลงในฐานข้อมูล และระบบจะแก้ไขค่า lastUpdated เป็นเวลาที่คลิกปุ่ม Submit โดยอัตโนมัติ จากนั้น ระบบจะจอภาพโพสต์ที่ถูกแก้ไขแล้วมายังผู้ใช้

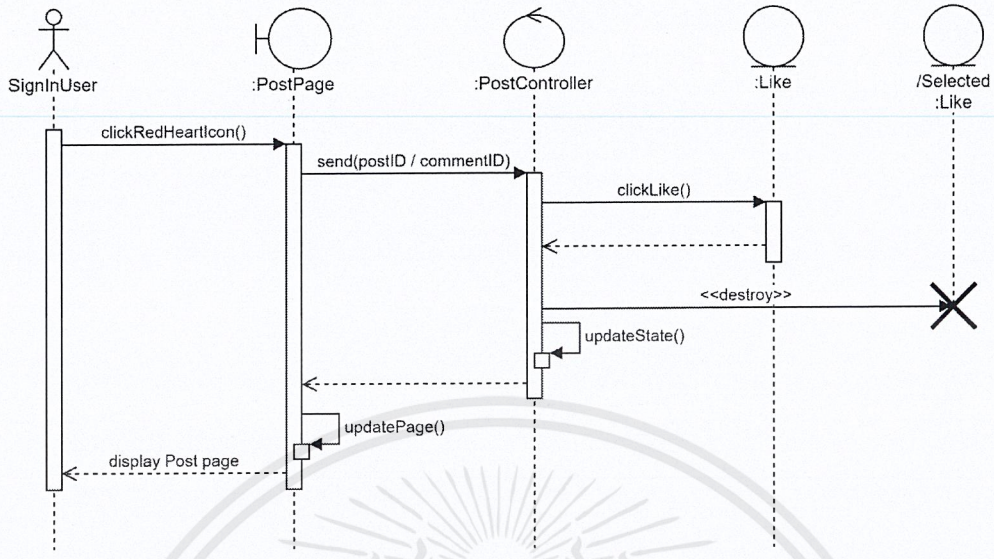
10. Sequence Diagram ของ Use case “Create like”



ภาพที่ 3.14 Sequence Diagram ของ Use case “Create like”

จากภาพที่ 3.14 ที่แสดงการทำงานของ Use case Create like ผู้ใช้ที่กำลังอยู่ในระบบสามารถคลิกถูกใจโพสต์หรือความคิดเห็นได้โดยการคลิกไอคอนรูปหัวใจสีขาวที่โพสต์หรือความคิดเห็นที่ถูกใจ หลังจากคลิกแล้วระบบจะเรียกใช้ฟังก์ชัน clickLike ที่มีการส่งค่า postID หรือ commentID ไปด้วย แล้วจัดเก็บค่าที่ส่งไปและ userID ของผู้ที่กดถูกใจลงไปยังฐานข้อมูล หลังจากทำการอัปเดตจอภาพโพสต์แล้วไอคอนหัวใจสีขาวจะเปลี่ยนเป็นไอคอนหัวใจสีแดง และจำนวนผู้ถูกใจจะเพิ่มขึ้น 1

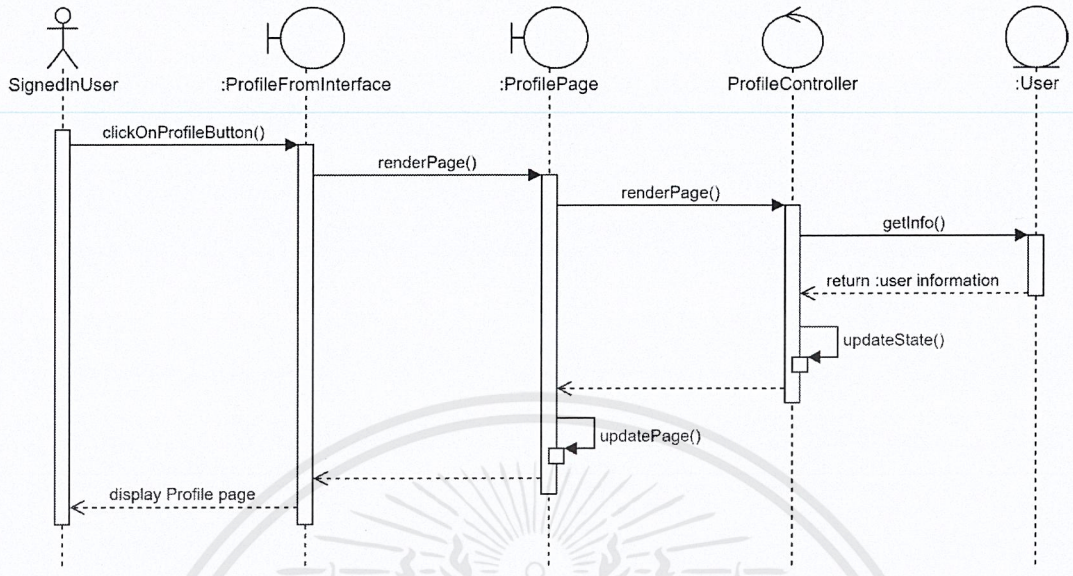
11. Sequence Diagram ของ Use case “Delete like”



ภาพที่ 3.15 Sequence Diagram ของ Use case “Delete like”

จากภาพที่ 3.15 ที่แสดงการทำงานของ Use case Delete like ผู้ใช้ที่กำลังอยู่ในระบบสามารถยกเลิกการถูกใจโพสต์หรือความคิดเห็นได้โดยการคลิกไอคอนรูปหัวใจสีแดงที่โพสต์หรือความคิดเห็นที่ต้องการยกเลิกการถูกใจ หลังจากคลิกแล้วระบบจะเรียกใช้ฟังก์ชัน clickLike ที่มีการส่งค่า postID หรือ commentID ไปด้วย แล้วลบการถูกใจที่ userID และ postID หรือ commentID ตรงกับค่าที่อยู่ในฐานข้อมูลออกจากฐานข้อมูล หลังจากทำการอัปเดตจอภาพโพสต์แล้วหัวใจสีแดงจะเปลี่ยนเป็นหัวใจสีขาว และจำนวนผู้ถูกใจจะลดลง 1

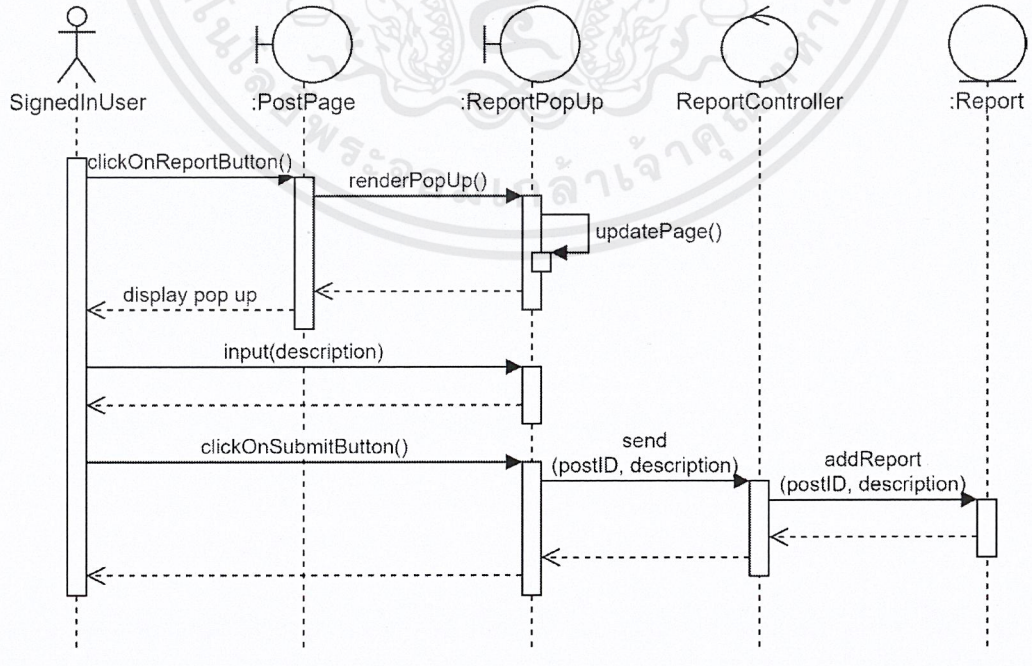
12. Sequence Diagram ของ Use case “Show profile”



ภาพที่ 3.16 Sequence Diagram ของ Use case “Show profile”

จากภาพที่ 3.16 ที่แสดงการทำงานของ Use case Show profile ผู้ใช้ที่กำลังอยู่ในระบบสามารถเรียกดูข้อมูลส่วนตัวของตนเองได้โดยการคลิกที่ปุ่ม Profile บนแถบนำทาง ระบบจะเรียกใช้ฟังก์ชัน getInfo เพื่อเรียกข้อมูลมาแสดงแก่ผู้ใช้ที่ต้องการดูข้อมูลส่วนตัว

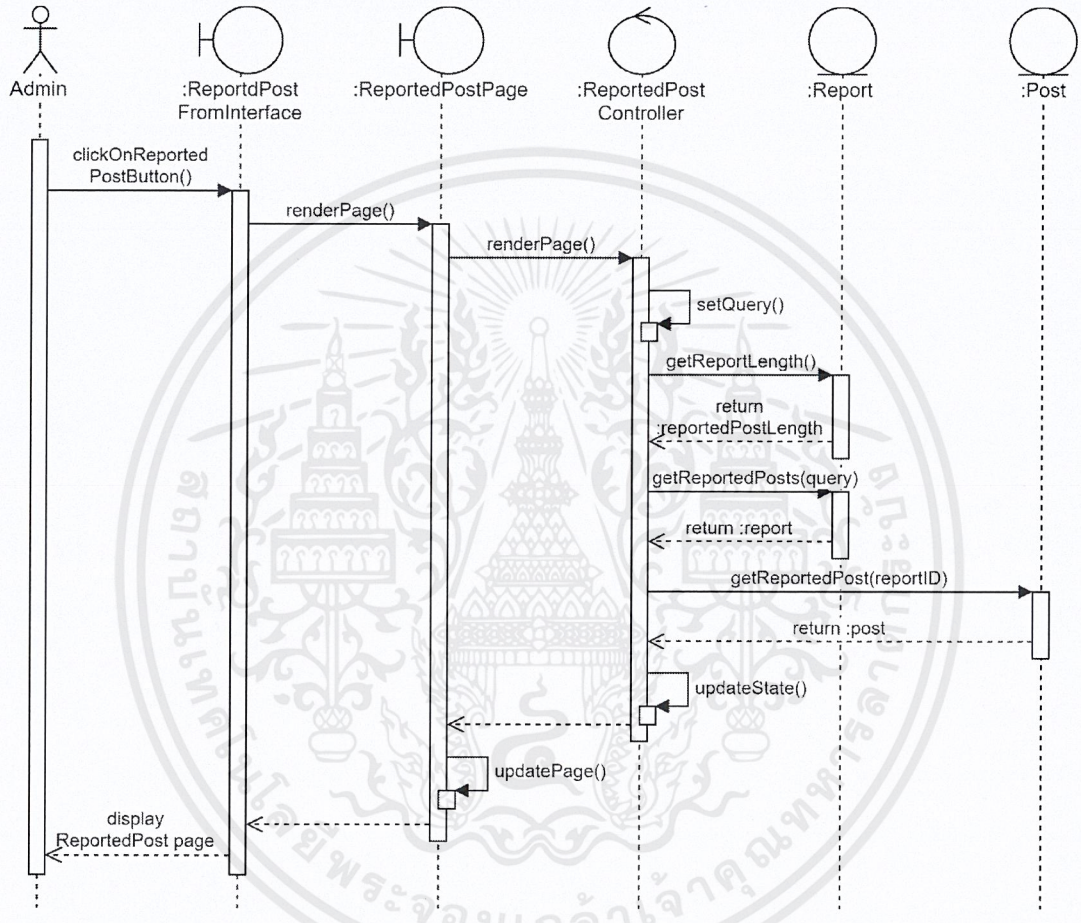
13. Sequence Diagram ของ Use case “Report post”



ภาพที่ 3.17 Sequence Diagram ของ Use case “Report post”

จากภาพที่ 3.17 ที่แสดงการทำงานของ Use case Report post ผู้ใช้ที่กำลังอยู่ในระบบสามารถคลิกที่ปุ่ม Report บนจอภาพของโพสต์ที่ต้องการแจ้งลบ ระบบจะหน้าต่างแจ้งลบโพสต์ขึ้นมา หลังจากกรอกเหตุผลที่ต้องการแจ้งลบและกด Submit แล้ว ระบบจะเพิ่ม postID, userID ที่ทำการแจ้งลบ และเหตุผลที่แจ้งลบไปยังฐานข้อมูล

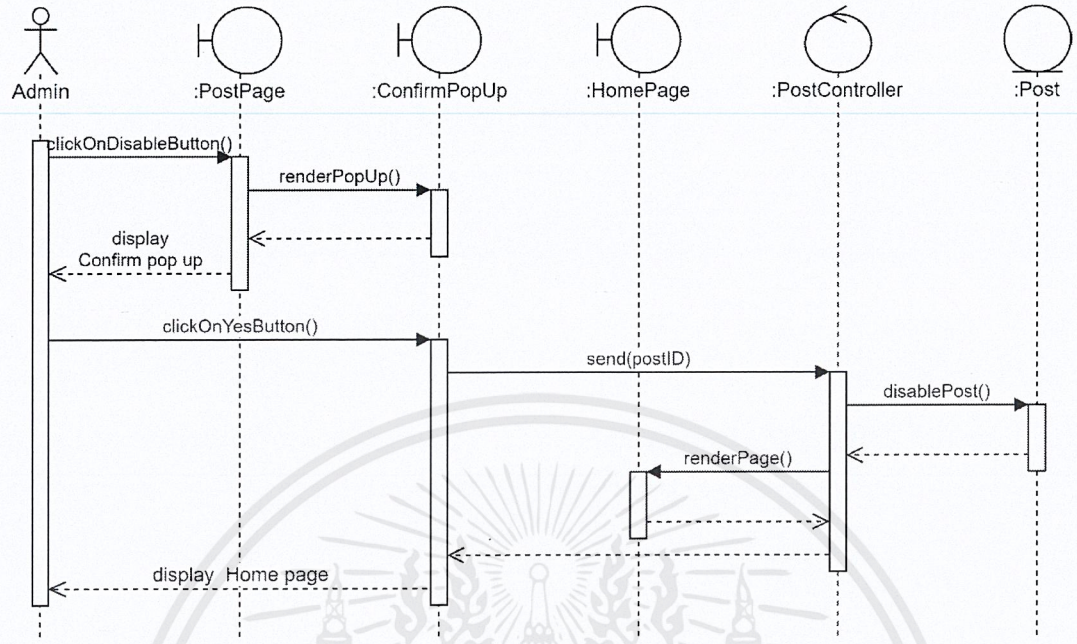
14. Sequence Diagram ของ Use case “Show reported post”



ภาพที่ 3.18 Sequence Diagram ของ Use case “Show reported post”

จากภาพที่ 3.18 ที่แสดงการทำงานของ Use case Show reported post ผู้ใช้ที่มีบทบาทเป็นผู้ดูแลระบบสามารถเรียกดูโพสต์ที่ถูกแจ้งลบเข้ามาได้โดยการคลิกที่ปุ่ม Reported Post บนแถบนำทาง ระบบจะเรียกใช้ฟังก์ชัน getReportLength ที่ส่งค่าจำนวนโพสต์ที่ถูกแจ้งลบกลับมา เพื่อนำจำนวนโพสต์มาคำนวณหาจำนวนหน้าที่จะแสดงบนแถบเลขหน้าที่มีไว้เพื่อให้ผู้ใช้เลือกคลิกเลขหน้าที่ต้องการ โดยโพสต์ที่ถูกเรียกออกมาจะขึ้นอยู่กับ query ซึ่งก็คือเลขหน้าที่ผู้ใช้ต้องการเรียกดู จากนั้นระบบจะทำการเรียกชุดของข้อมูลการแจ้งลบและหัวข้อของโพสต์ที่ถูกแจ้งลบผ่านฟังก์ชัน getReportedPosts เพื่อนำมาแสดงบนจอภาพโพสต์ที่ถูกแจ้งลบ

15. Sequence Diagram ของ Use case “Disable post”



ภาพที่ 3.19 Sequence Diagram ของ Use case “Disable post”

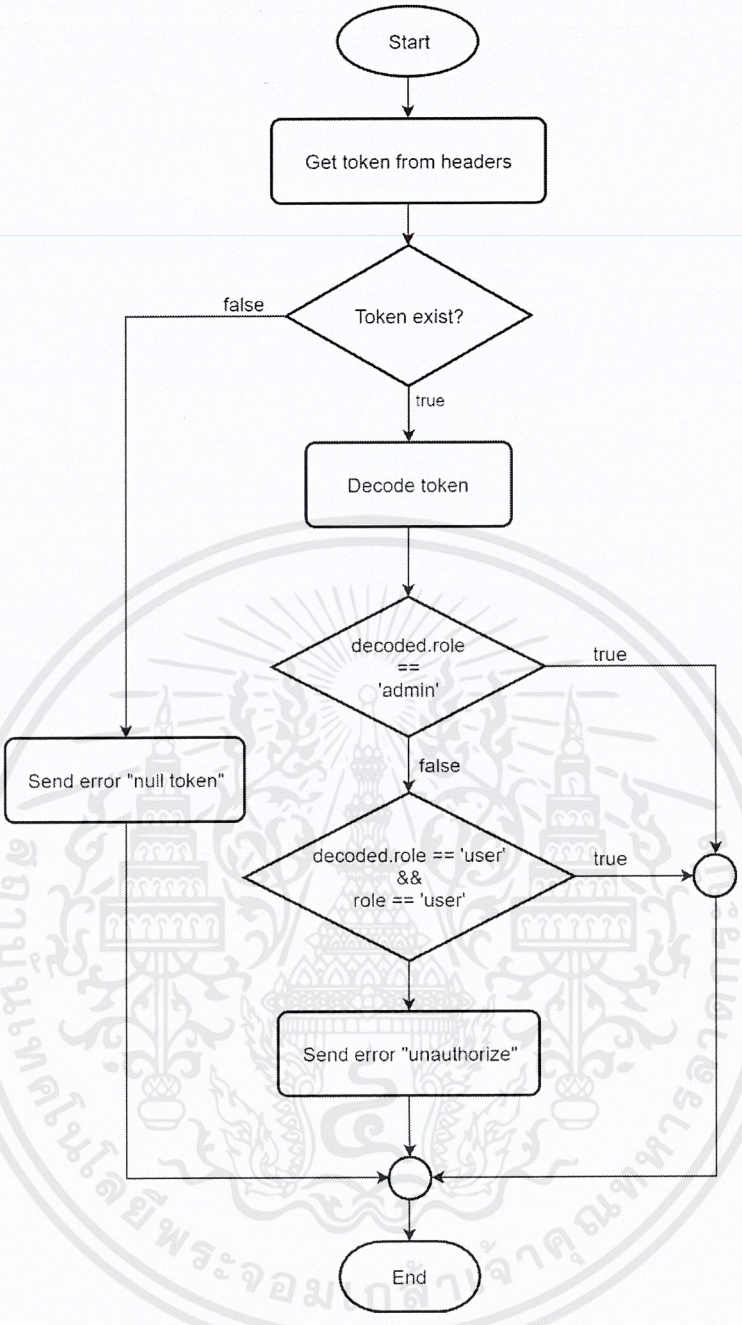
จากภาพที่ 3.9 ที่แสดงการทำงานของ Use case Disable post ผู้ใช้ที่มีบทบาทเป็นผู้ดูแลระบบสามารถซ่อนโพสต์ไม่ให้แสดงในระบบได้ โดยคลิกที่ปุ่ม Disable บนจอภาพของโพสต์ที่ต้องการซ่อน ระบบจะแสดงหน้าต่างยืนยันขึ้นมา เมื่อคลิก yes ระบบจะเรียกใช้ฟังก์ชัน disablePost เพื่อแก้ไขค่า disable ใน post ให้เป็น 1 จากนั้นระบบจะแสดงจอภาพหลักแก่ผู้ใช้

3.4.3 Flowchart

เนื่องจากเว็บแอพลิเคชัน Knowledge base มีวัตถุที่ฟังก์ชันการทำงานที่หลากหลาย ผู้จัดทำจึงทำการออกแบบฟังก์ชันการทำงานที่จะถูกเรียกไปใช้โดยใช้ Flowchart เพื่อให้เห็นถึงลำดับการทำงานของฟังก์ชัน โดยผู้จัดทำได้ออกแบบ Flowchart ไว้ ดังนี้

1. ฟังก์ชัน authorization(role)

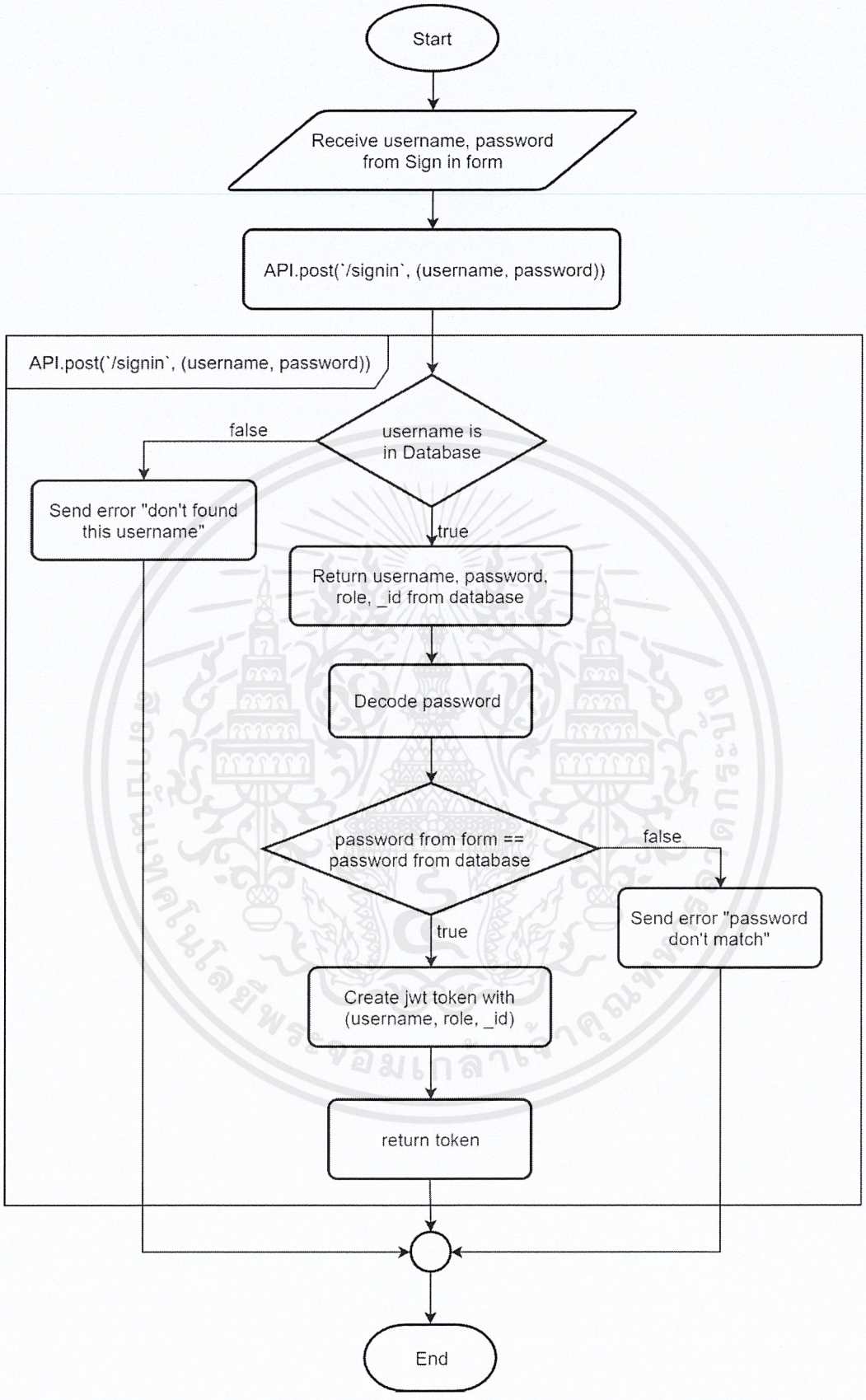
ฟังก์ชัน authorization เป็น Middleware ที่ถูกเรียกใช้ก่อนการเข้าถึง API หาก API นั้นกำหนดบทบาทของการถึง ซึ่งก็คือกำหนดว่าต้องเป็น User หรือ Admin หรือไม่ โดยจะทำการนำโทเค็นจากเฮดเดอร์ถอดรหัสและตรวจสอบบทบาทจากโทเค็นว่าตรงกับบทบาทที่ต้องการหรือไม่ หากไม่ตรงจะส่งค่า error กลับมา หากตรงจะสามารถเข้าถึง API ที่ต้องการได้



ภาพที่ 3.20 Flowchart ของฟังก์ชัน authorization(role)

2. ฟังก์ชัน authentication()

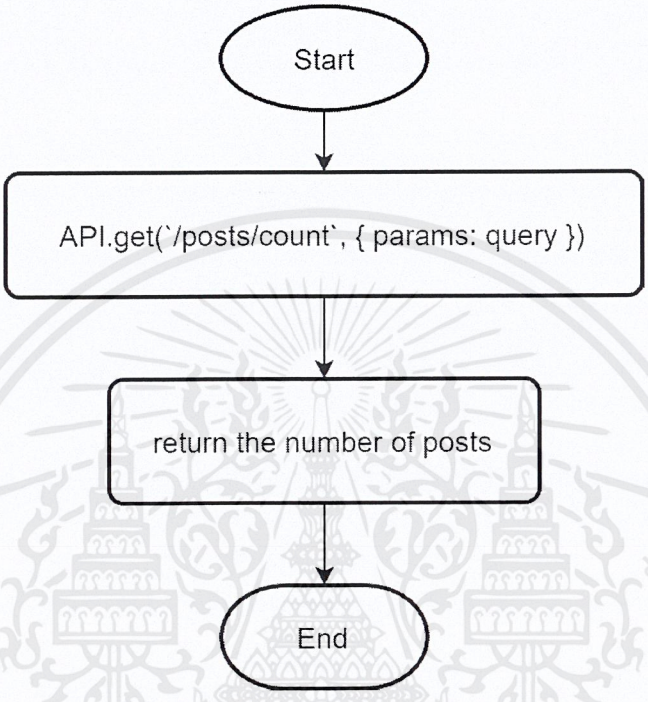
ฟังก์ชัน authentication ถูกเรียกใช้เพื่อตรวจสอบว่าในขั้นตอนการเข้าสู่ระบบนั้น username และ password ที่ผู้ใช้กรอกเข้ามาตรงกับ username และ password ที่เก็บไว้ในฐานข้อมูลหรือไม่ หากตรงกันจะทำการสร้าง JWT ที่มีการเก็บ username, บทบาทของผู้ใช้ และไอดีของผู้ใช้ แล้วนำโทเค็นที่ได้ไปเก็บไว้ใน Local storage เพื่อนำไปใช้ในการยืนยันตัวตน (Authorization) ต่อไป



ภาพที่ 3.21 Flowchart ของฟังก์ชัน authentication()

3. ฟังก์ชัน getPostsLength()

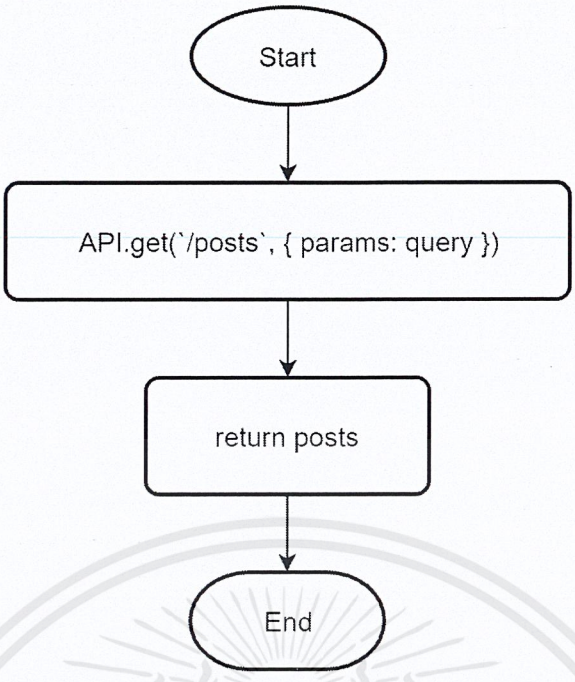
ฟังก์ชัน getPostsLength ถูกเรียกใช้เมื่อต้องการจำนวนโพสต์ในระบบที่สามารถนำมาแสดงได้ เพื่อใช้ในการคำนวณจำนวณหน้าที่ต้องการให้แสดงบนแถบเลขหน้า ฟังก์ชันนี้สามารถใช้ได้กับการแสดงโพสต์ในจอภาพหลักและจอภาพผลการค้นหา โดยความแตกต่างของผลลัพธ์ที่นำมาแสดงจะขึ้นอยู่กับ query ของแต่ละการใช้งาน



ภาพที่ 3.22 Flowchart ของฟังก์ชัน getPostsLength()

4. ฟังก์ชัน getPosts()

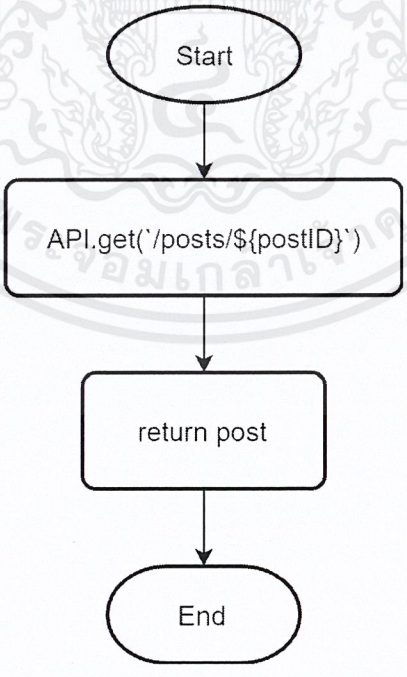
ฟังก์ชัน getPosts ถูกเรียกใช้เมื่อต้องการข้อมูลของโพสต์จำนวนหลายโพสต์โดยไม่ลงรายละเอียด ฟังก์ชันนี้สามารถใช้ได้กับการแสดงโพสต์ในจอภาพหลักและจอภาพผลการค้นหา โดยความแตกต่างของผลลัพธ์ที่นำมาแสดงจะขึ้นอยู่กับ query ของแต่ละการใช้งาน



ภาพที่ 3.23 Flowchart ของฟังก์ชัน getPosts()

5. ฟังก์ชัน getPost()

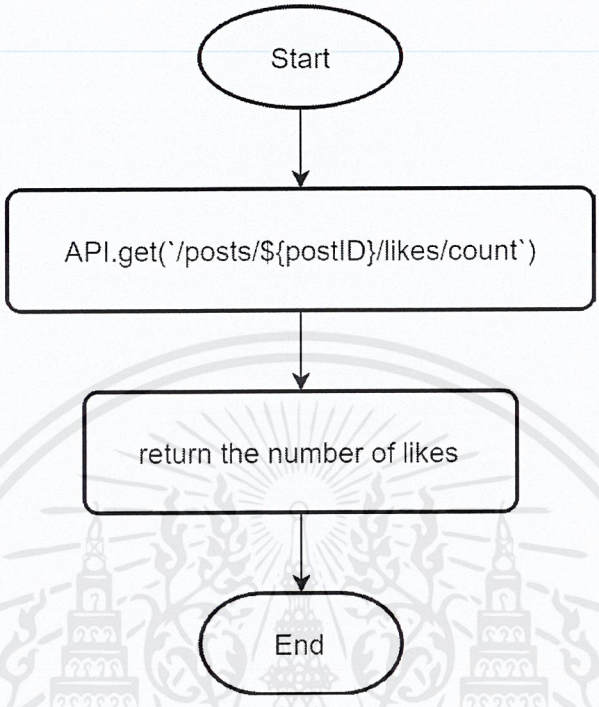
ฟังก์ชัน getPost() ถูกเรียกใช้เมื่อต้องการรายละเอียดของแต่ละโพสต์ที่ถูกเก็บไว้ในฐานข้อมูล



ภาพที่ 3.24 Flowchart ของฟังก์ชัน getPost()

6. ฟังก์ชัน getNumberOfLike()

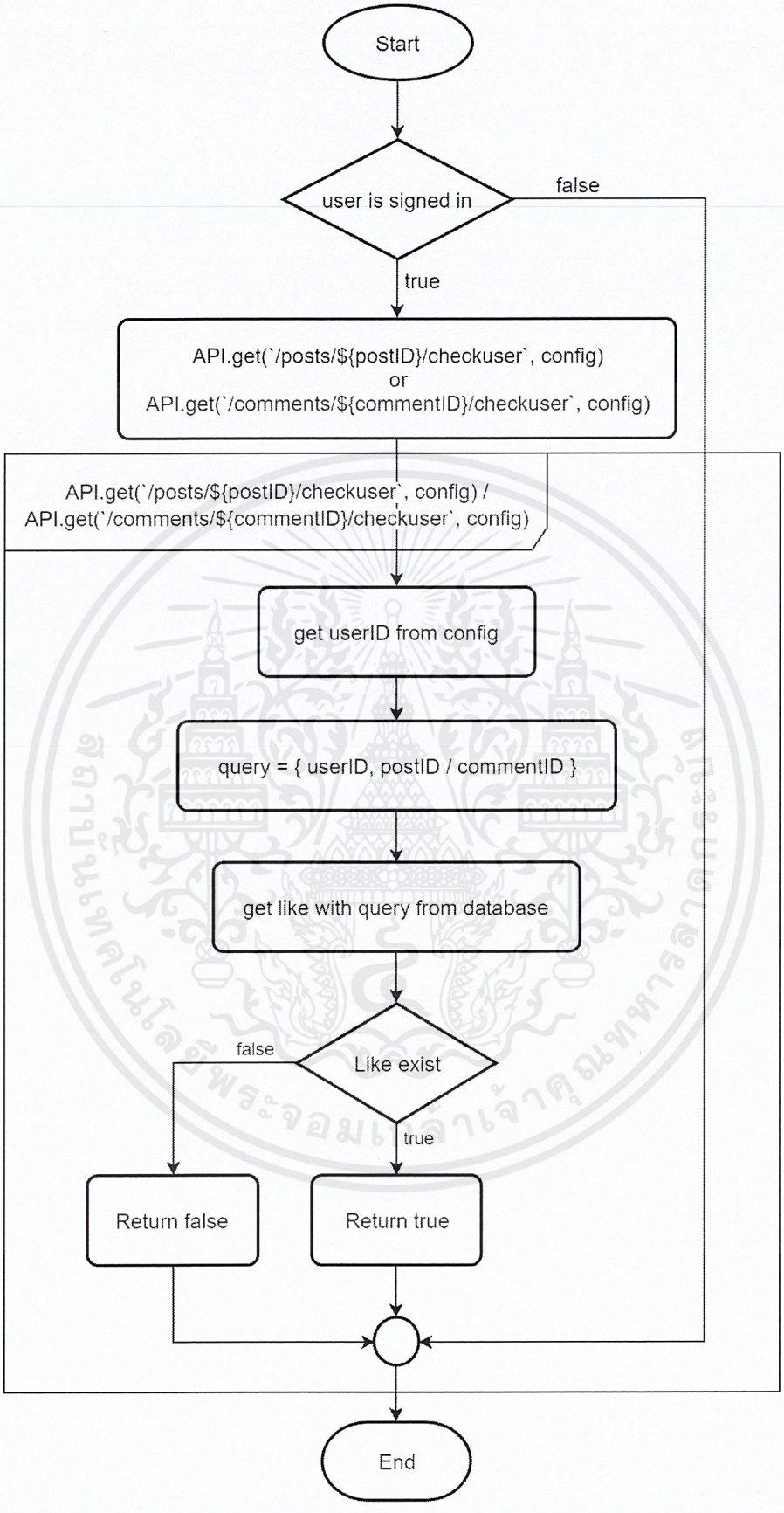
ฟังก์ชัน getNumberOfLike() ถูกเรียกใช้เมื่อต้องการจำนวนผู้ที่กดถูกใจโพสต์และความคิดเห็นแต่ละอัน



ภาพที่ 3.25 Flowchart ของฟังก์ชัน getNumberOfLike()

7. ฟังก์ชัน checkUserLike()

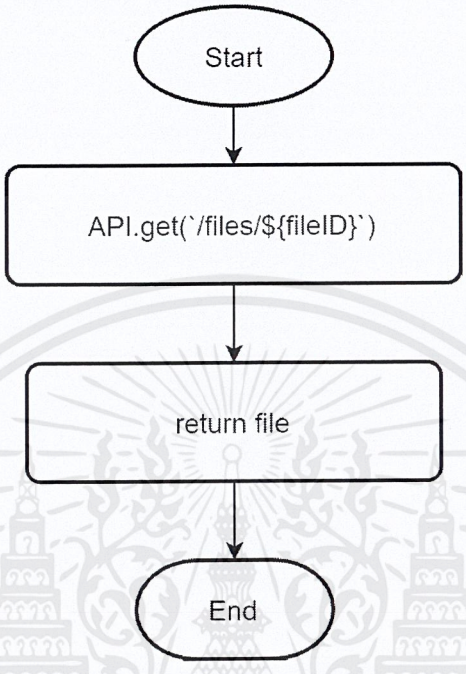
ฟังก์ชัน checkUserLike ถูกเรียกใช้เพื่อตรวจสอบว่าผู้ใช้ที่กำลังใช้งานอยู่ได้ทำการเข้าสู่ระบบแล้วหรือไหม และหากกำลังอยู่ในระบบ ผู้ใช้ได้กดถูกใจโพสต์หรือความคิดเห็นนั้น ๆ หรือไม่ เพื่อนำมาใช้แสดงสีของไอคอนหัวใจในการกดถูกใจ ค่าที่ถูกส่งกลับมาจะเป็นจริงหากผู้ใช้กำลังอยู่ในระบบและได้ทำการถูกใจโพสต์หรือความคิดเห็น ค่าที่ถูกส่งกลับมาจะเป็นเท็จหากผู้ใช้ยังไม่ได้ทำการเข้าสู่ระบบหรือยังไม่ได้ทำการกดถูกใจโพสต์หรือความคิดเห็น



ภาพที่ 3.26 Flowchart ของฟังก์ชัน checkUserLike()

8. ฟังก์ชัน getFile()

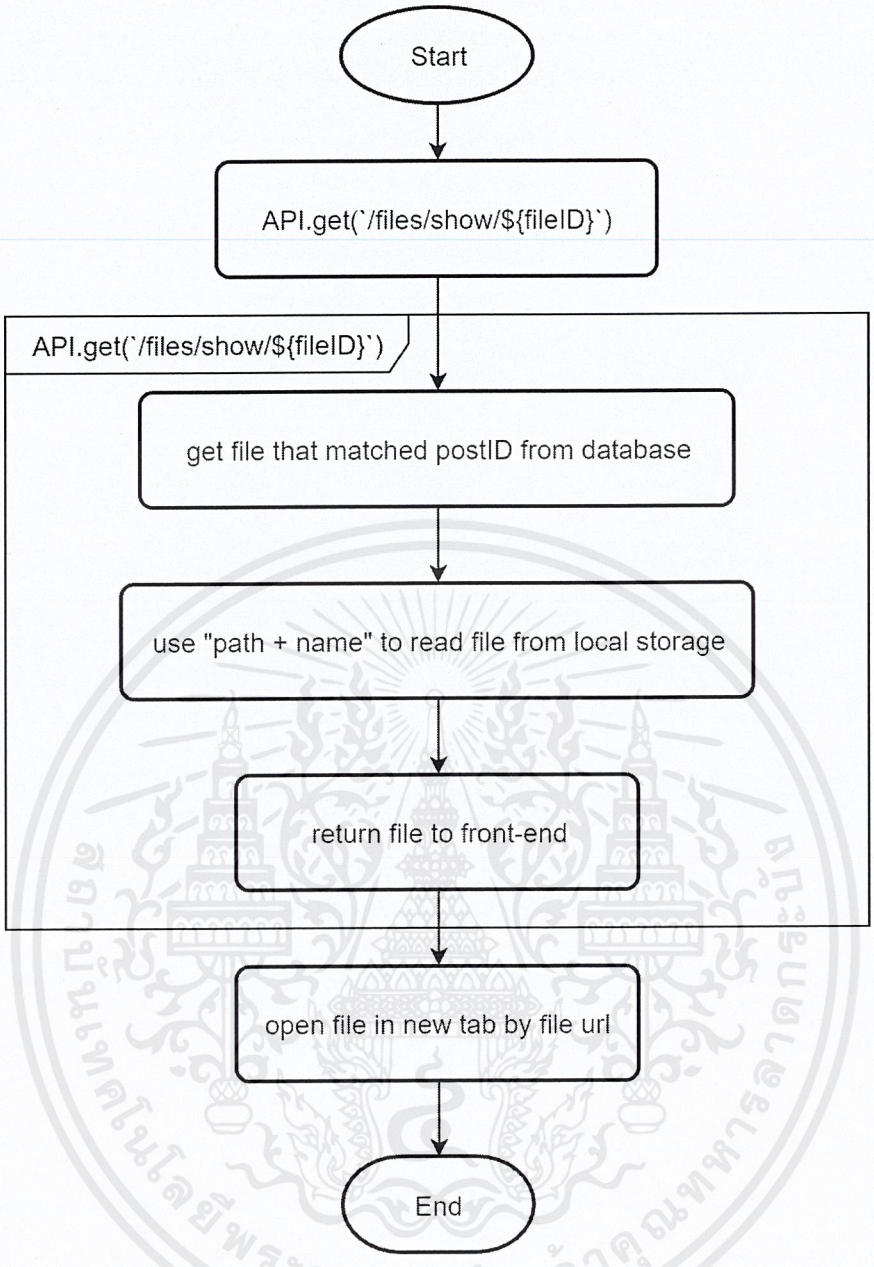
ฟังก์ชัน getFile ถูกเรียกใช้เมื่อต้องการเรียกข้อมูลไฟล์ที่ถูกเก็บไว้ในแต่ละโพสต์ ซึ่งใช้ไอดีของไฟล์ในการเรียกข้อมูลจากฐานข้อมูล



ภาพที่ 3.27 Flowchart ของฟังก์ชัน getFile()

9. ฟังก์ชัน showFile ()

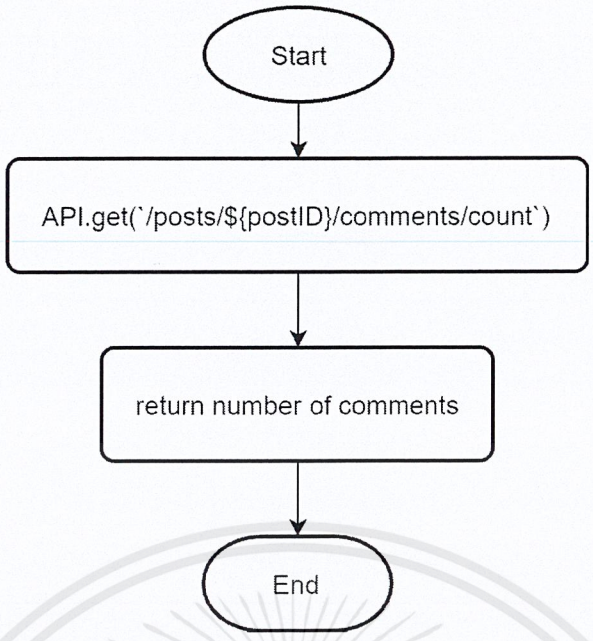
ฟังก์ชัน showFile ถูกเรียกใช้เมื่อต้องการเปิดไฟล์ที่ถูกอัปโหลดในแต่ละโพสต์ ซึ่งทำการเรียกข้อมูลของไฟล์มาจากฐานข้อมูลโดยใช้ไอดีของไฟล์ แล้วนำชื่อไฟล์ที่ได้มาไปเปิดไฟล์จากใน local storage



ภาพที่ 3.28 Flowchart ของฟังก์ชัน showFile()

10. ฟังก์ชัน getCommentsLength()

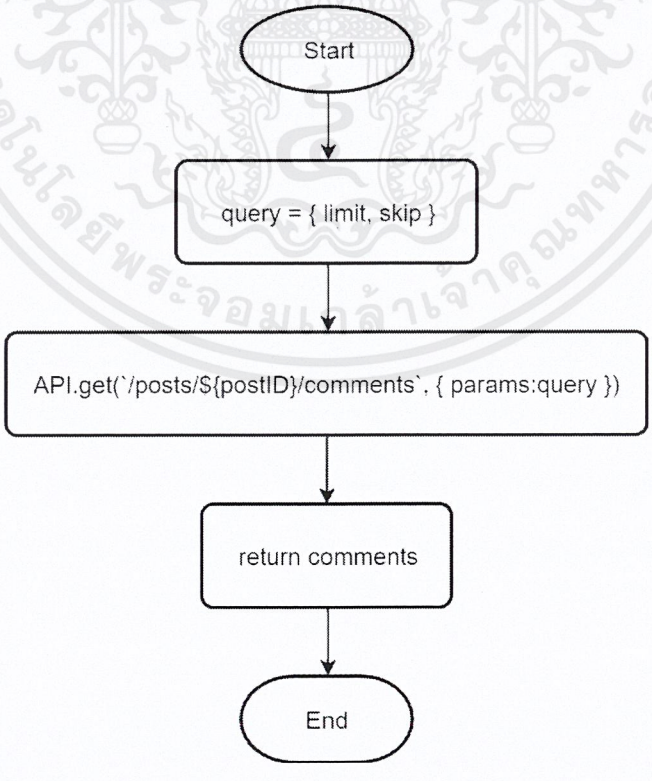
ฟังก์ชัน getCommentsLength ถูกเรียกใช้เมื่อต้องการจำนวนความคิดเห็นทั้งหมดในแต่ละโพสต์ เพื่อใช้ในการคำนวณจำนวณหน้าที่ต้องการให้แสดงบนแถบเลขหน้า



ภาพที่ 3.29 Flowchart ของฟังก์ชัน getCommentsLength()

11. ฟังก์ชัน getComments()

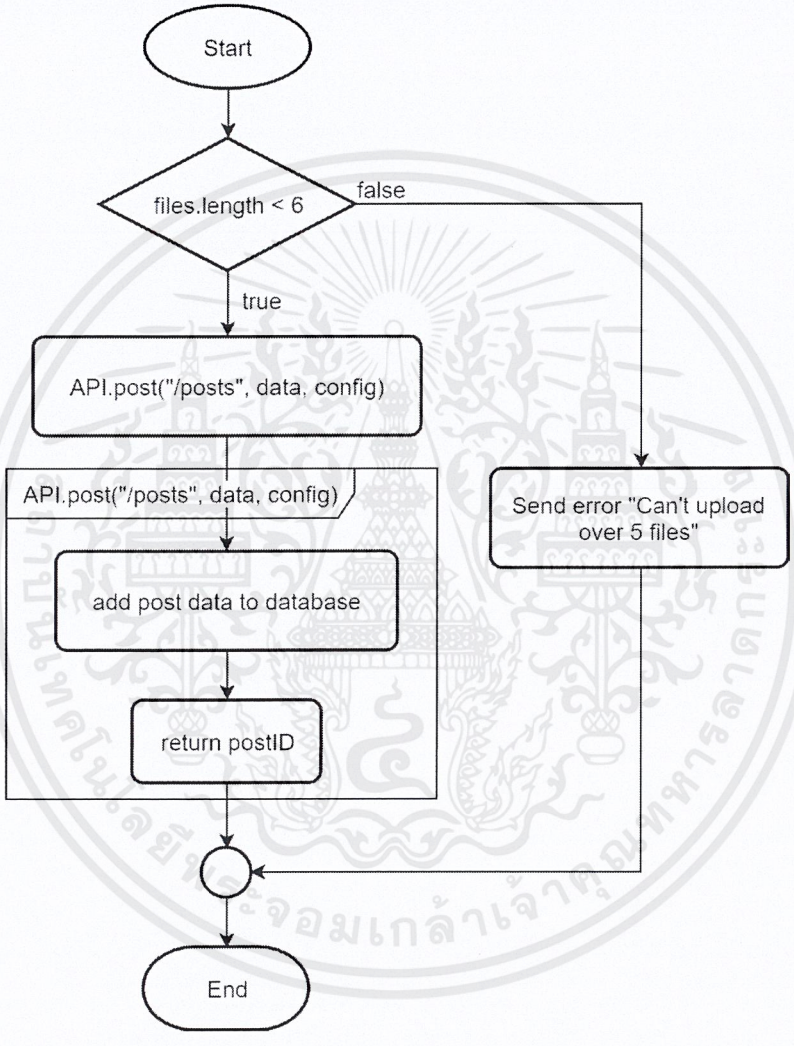
ฟังก์ชัน getComments ถูกเรียกใช้เมื่อต้องการเรียกความคิดเห็นทั้งหมดใน แต่ละโพสต์มาแสดงในแต่ละหน้าของความคิดเห็น



ภาพที่ 3.30 Flowchart ของฟังก์ชัน getComments()

12. ฟังก์ชัน addPost()

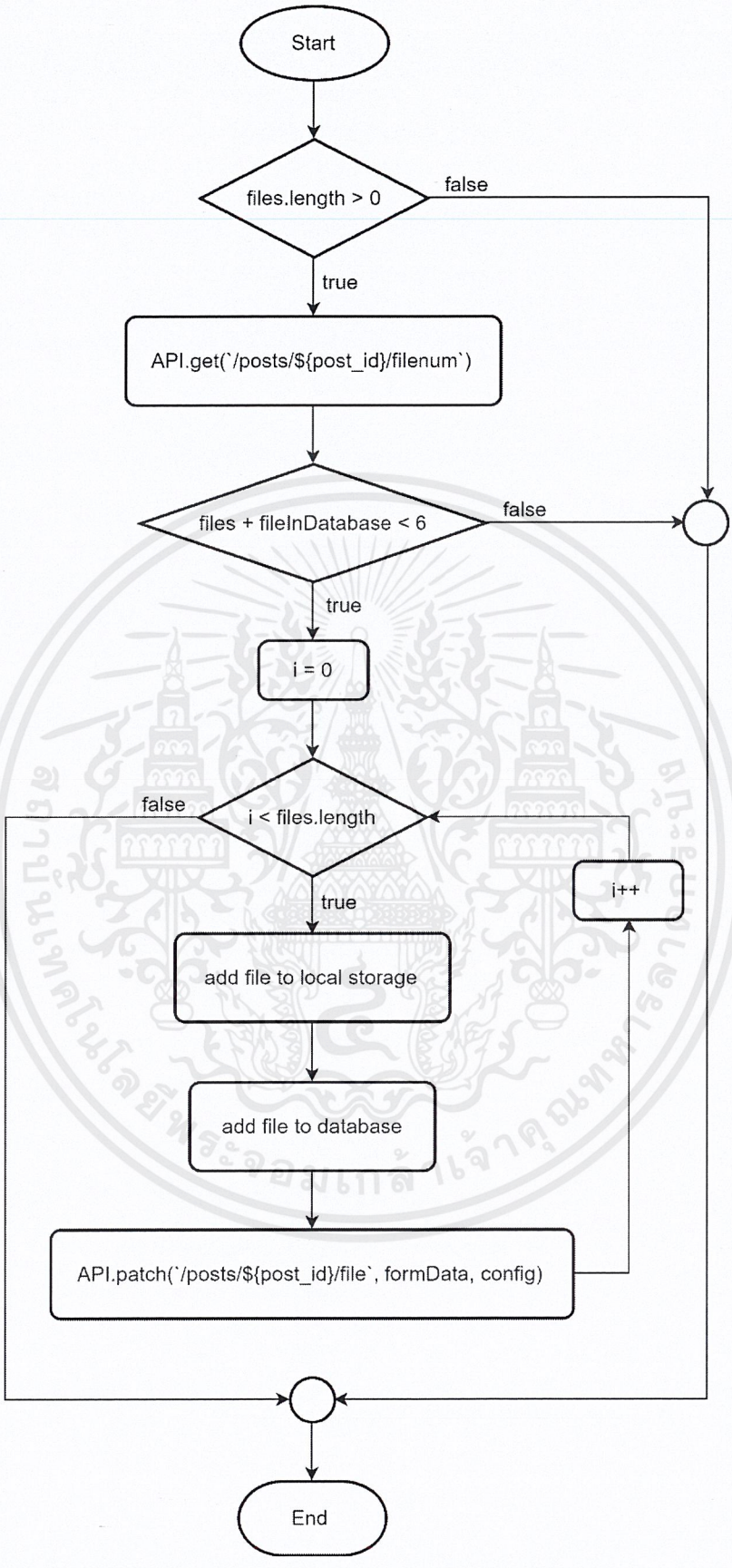
ฟังก์ชัน addPost ถูกเรียกใช้เมื่อมีการสร้างโพสต์ใหม่เข้าสู่ระบบ โดยจะต้องมีการตรวจสอบก่อนว่าไฟล์ที่ถูกอัปโหลดเข้ามามีจำนวนเกิน 5 ไฟล์หรือไม่ หากมีจำนวนไม่เกิน 5 ไฟล์ระบบจะตรวจสอบว่าผู้ใช้ที่สร้างโพสต์ได้ทำการเข้าสู่ระบบแล้วหรือไม่ หากกำลังอยู่ในระบบจะสามารถเพิ่มข้อมูลของโพสต์ใหม่เข้าไปในฐานข้อมูลได้



ภาพที่ 3.31 Flowchart ของฟังก์ชัน addPost()

13. ฟังก์ชัน addFile()

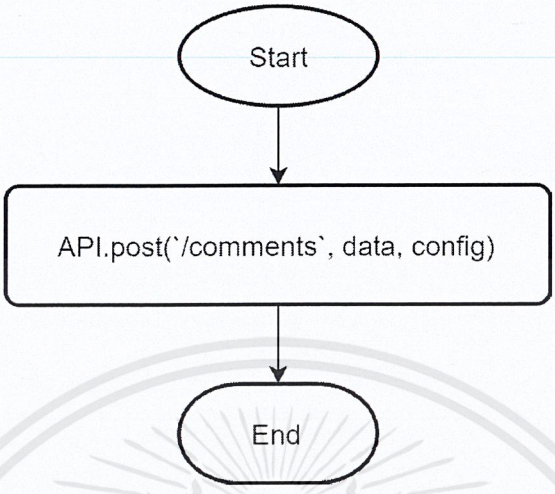
ฟังก์ชัน addFile ถูกเรียกใช้เมื่อมีการสร้างโพสต์หรือแก้ไขโพสต์ทุกครั้ง โดยจะมีการตรวจสอบว่าจำนวนของไฟล์ในระบบรวมกับไฟล์ที่ต้องการเพิ่มเข้าไปมีจำนวนเกิน 5 ไฟล์หรือไม่ หากไม่เกินจะทำการบันทึกไฟล์ไปยัง local storage แล้วนำข้อมูลของไฟล์ไปจัดเก็บในฐานข้อมูล จากนั้นไอดีของไฟล์ไปเพิ่มใน fileID ของโพสต์ที่ไฟล์ถูกอัปโหลดเข้าไป



ภาพที่ 3.32 Flowchart ของฟังก์ชัน addFile()

14. ฟังก์ชัน addComment()

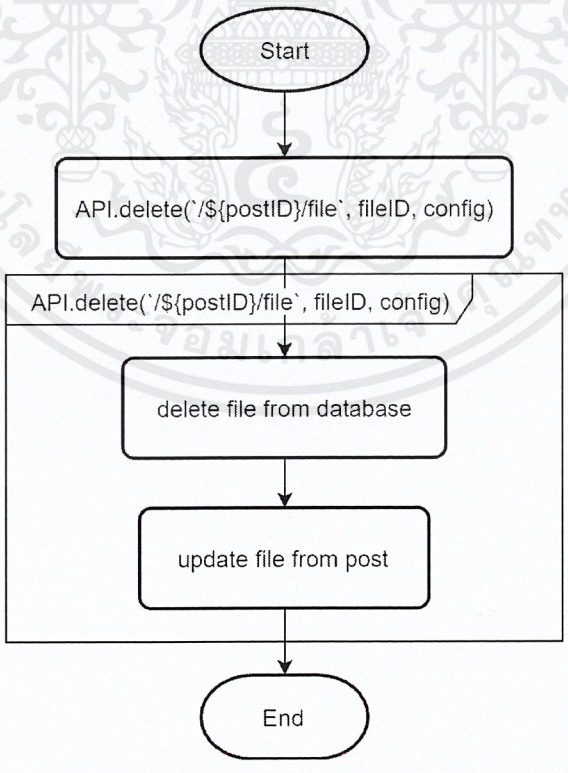
ฟังก์ชัน addComment ถูกเรียกใช้เมื่อมีการเพิ่มความคิดเห็นใหม่ในโพสต์



ภาพที่ 3.33 Flowchart ของฟังก์ชัน addComment()

15. ฟังก์ชัน deleteFiles()

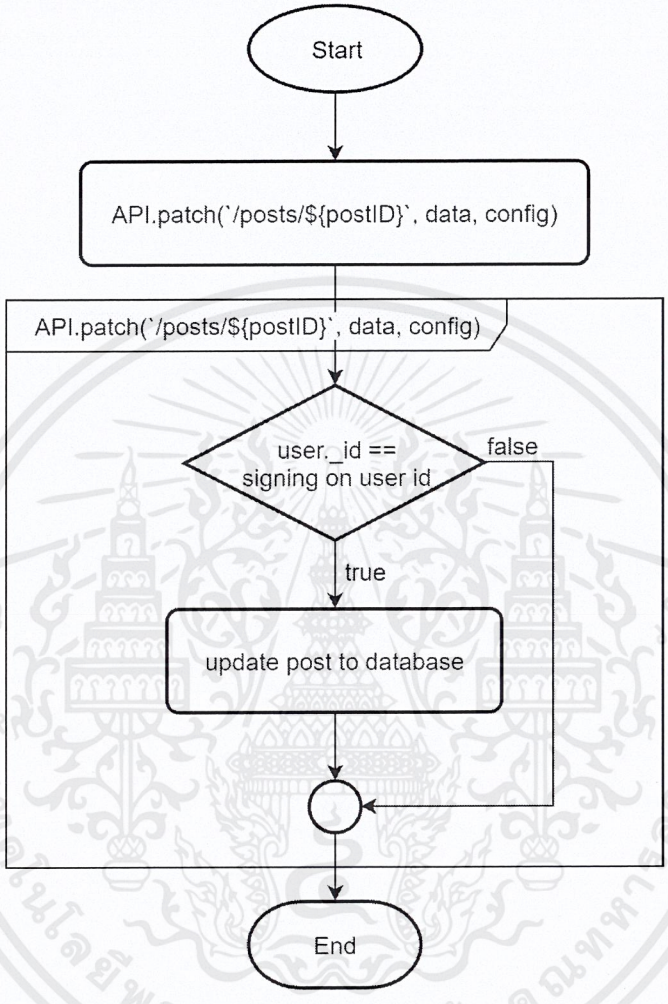
ฟังก์ชัน deleteFiles ถูกเรียกใช้เมื่อมีการแก้ไขโพสต์แล้วต้องการลบไฟล์ที่ถูกอัปโหลดไว้แล้ว โดยจะทำการลบไฟล์ออกจากฐานข้อมูล และลบ fileID ออกจากโพสต์



ภาพที่ 3.34 Flowchart ของฟังก์ชัน deleteFiles()

16. ฟังก์ชัน updatePost()

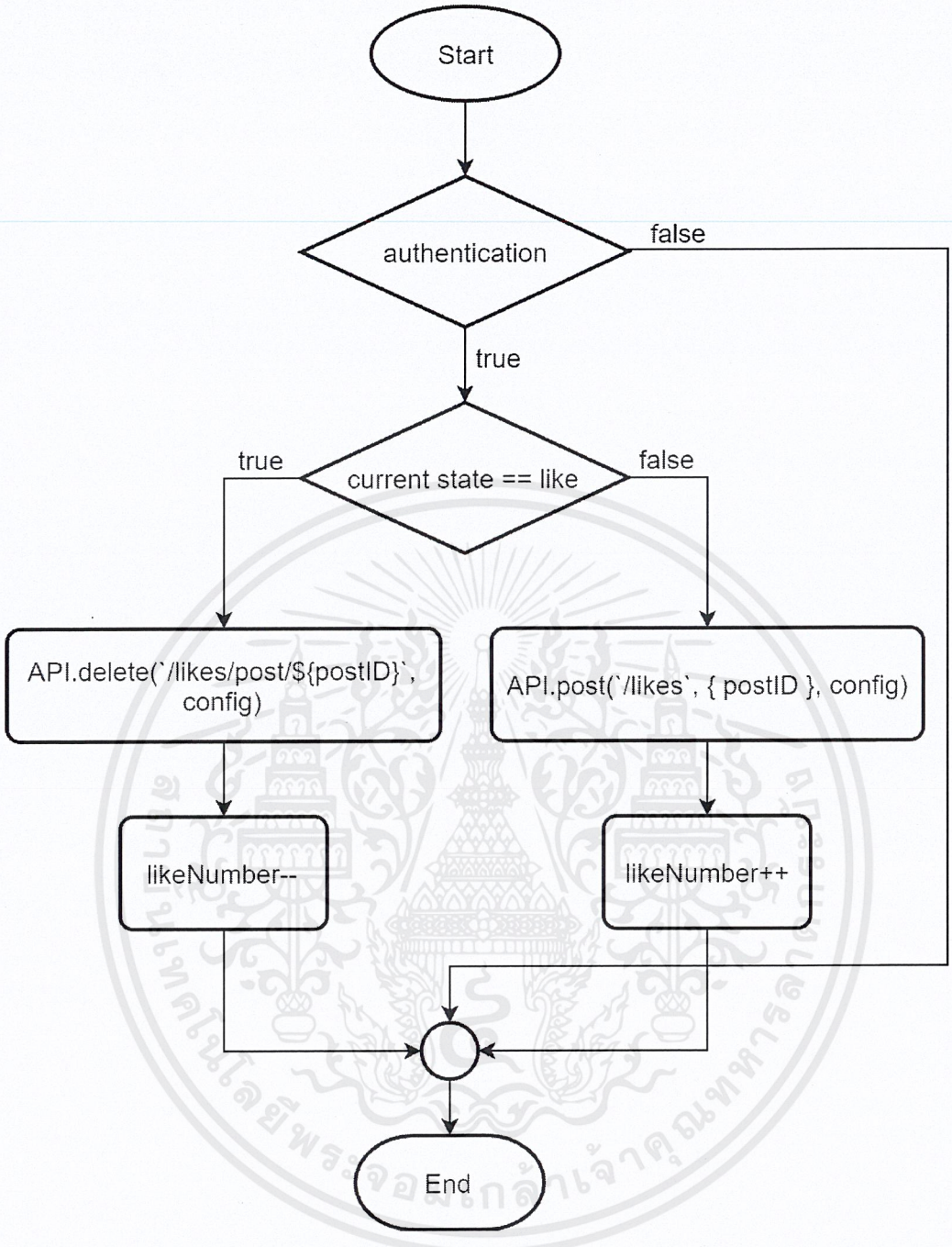
ฟังก์ชัน updatePost ถูกเรียกใช้เมื่อมีการแก้ไขโพสต์ รวมถึงการเพิ่มและลบไฟล์ในแต่ละโพสต์ด้วย



ภาพที่ 3.35 Flowchart ของฟังก์ชัน updatePost()

17. ฟังก์ชัน clickLike()

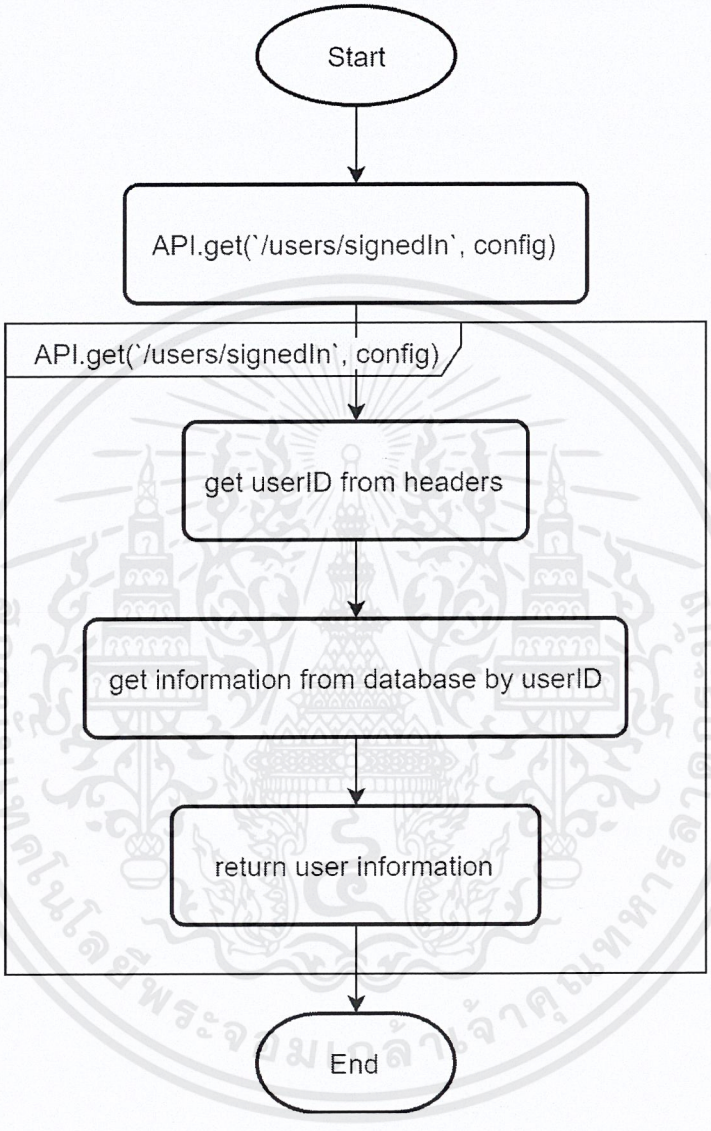
ฟังก์ชัน clickLike ถูกเรียกใช้เมื่อมีการคลิกไอคอนหัวใจในโพสต์หรือความคิดเห็น หากตอนนี้ผู้ใช้กำลังอยู่ในระบบ ระบบจะทำการตรวจสอบสถานะที่ผู้ใช้คลิกไอคอนหัวใจว่าผู้ใช้คนปัจจุบันถูกใจโพสต์หรือความคิดเห็นนี้หรือไม่ หากสถานะปัจจุบันเป็นถูกใจระบบจะทำการลบ like จากฐานข้อมูลและจำนวนถูกใจจะลดลง แต่หากสถานะปัจจุบันไม่ได้เป็นถูกใจระบบจะทำการเพิ่ม like เข้าไปยังฐานข้อมูล และจำนวนถูกใจจะเพิ่มขึ้น



ภาพที่ 3.36 Flowchart ของฟังก์ชัน clickLike()

18. ฟังก์ชัน getInfo()

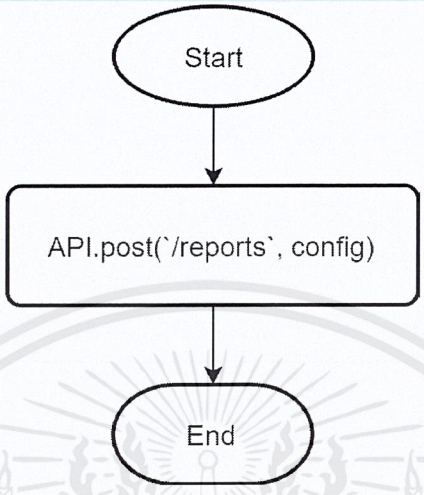
ฟังก์ชัน getInfo ถูกเรียกใช้เมื่อต้องการเรียกข้อมูลส่วนตัวของผู้ใช้ที่กำลังอยู่ในระบบมาแสดง



ภาพที่ 3.37 Flowchart ของฟังก์ชัน getInfo()

19. ฟังก์ชัน addReport()

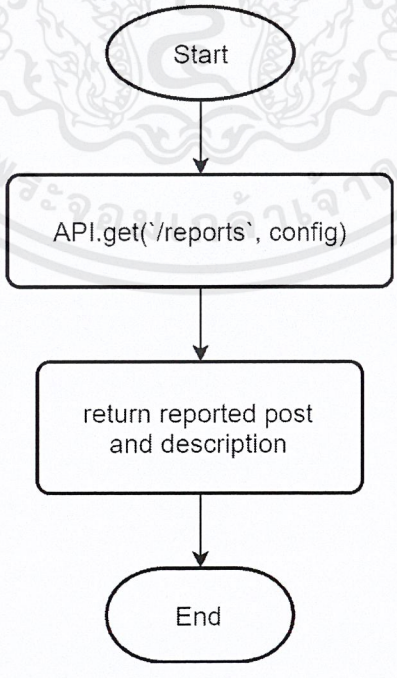
ฟังก์ชัน addReport ถูกเรียกใช้เพื่อจัดเก็บข้อมูลการแจ้งลบไปยังฐานข้อมูล โดยผู้ที่สามารถเรียกใช้ได้คือผู้ดูแลระบบเท่านั้น



ภาพที่ 3.38 Flowchart ของฟังก์ชัน addReport()

20. ฟังก์ชัน getReportedPosts()

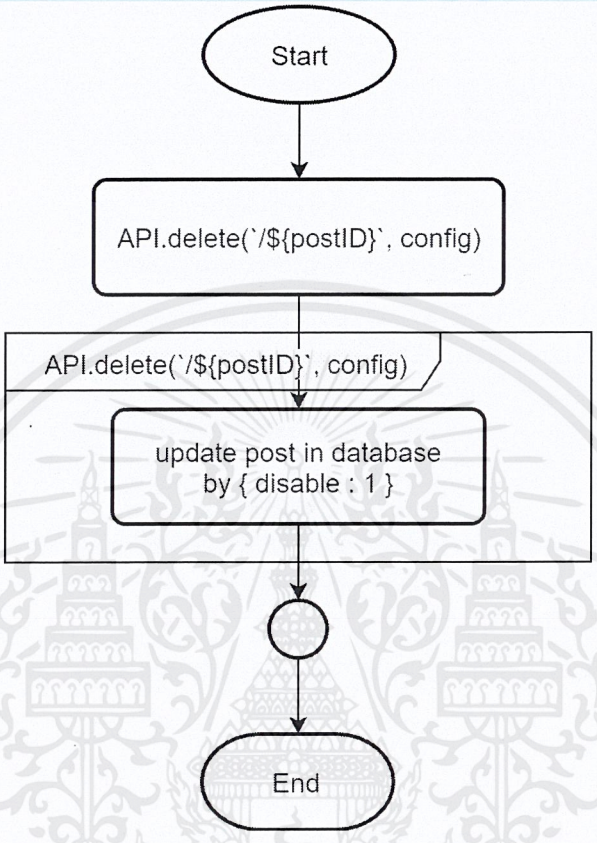
ฟังก์ชัน getReportedPosts ถูกเรียกใช้เมื่อต้องการเรียกดูโพสต์ที่ถูกแจ้งลบเข้ามายังระบบ โดยผู้ที่สามารถเรียกใช้ได้คือผู้ดูแลระบบเท่านั้น



ภาพที่ 3.39 Flowchart ของฟังก์ชัน getReportdPost()

21. ฟังก์ชัน disablePost()

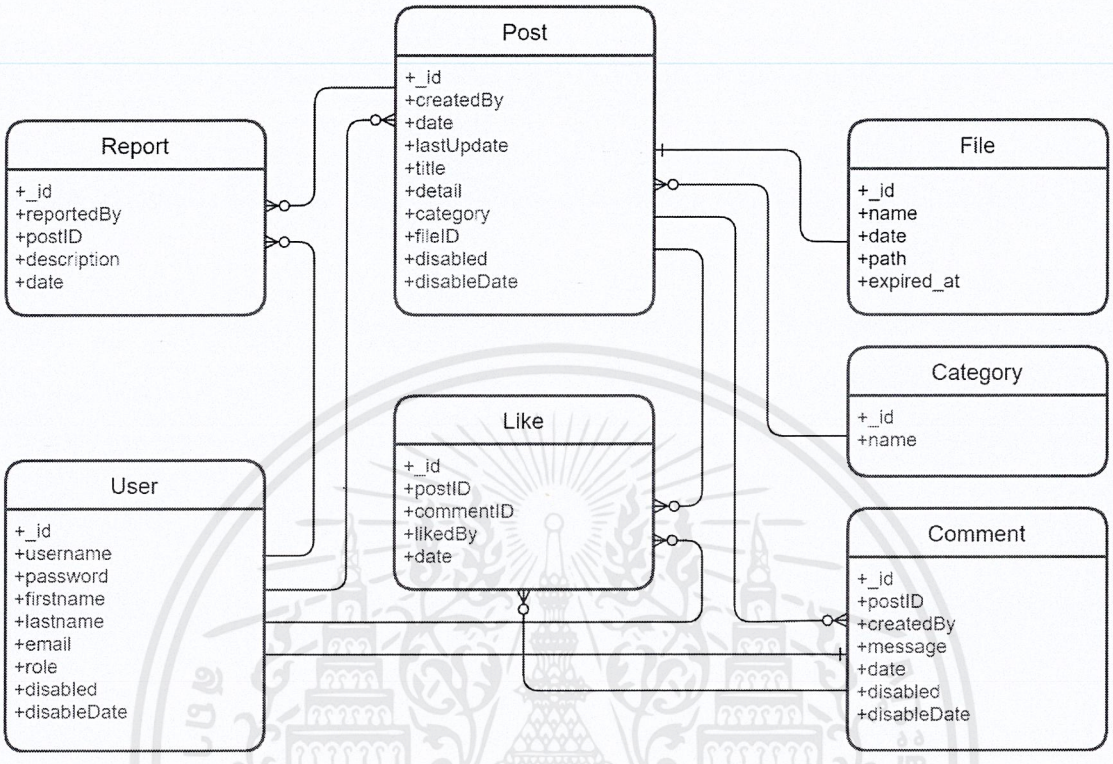
ฟังก์ชัน disablePost ถูกเรียกใช้เมื่อผู้ดูแลระบบต้องการซ่อนโพสต์ออกจากระบบ โดยจะทำการแก้ไขค่า disable ของโพสต์ให้กลายเป็น 1 ผู้ที่สามารถเรียกใช้ได้คือผู้ดูแลระบบเท่านั้น



ภาพที่ 3.40 Flowchart ของฟังก์ชัน disablePost()

3.4.4 Database Schema Diagram

ความสัมพันธ์ของโมเดลแต่ละตัวในระบบ สามารถสรุปได้ดังแผนภาพนี้



ภาพที่ 3.41 Database Schema Diagram ของเว็บแอปพลิเคชัน Knowledge base

1. Model User

แบบจำลอง User ทำหน้าเก็บข้อมูลส่วนตัวของผู้ใช้ โดยมีรายละเอียดดังนี้

ตารางที่ 3.16 รายละเอียด model ของ User

Feild	Datatype	Description
_id	ObjectId	ไอดีประจำตัวผู้ใช้
username	String	Username ของผู้ใช้
password	String	Password ของผู้ใช้
firstname	String	ชื่อจริงของผู้ใช้
Lastname	String	นามสกุลของผู้ใช้
email	String	อีเมลของผู้ใช้
role	String	บทบาทของผู้ใช้

disable	number	สถานะของผู้ใช้ ปกติมีค่าเป็น 0 หากไม่มีความเคลื่อนไหวแล้วจะมีค่าเป็น 1
delDate	Date	วันที่ผู้ใช้ถูก disable

2. Model Post

แบบจำลอง Post ทำหน้าเก็บรายละเอียดของแต่ละโพสต์ ดังนี้

ตารางที่ 3.17 รายละเอียดของ model Post

Feild	Datatype	Description
_id	ObjectId	ไอดีประจำโพสต์
createdBy	String	ไอดีของผู้ใช้ที่สร้างโพสต์
date	Date	วันที่สร้างโพสต์
lastUpdate	Date	วันที่อัปเดตโพสต์ล่าสุด
title	String	หัวข้อโพสต์
detail	String	เนื้อหาของโพสต์
category	String	หมวดหมู่ของโพสต์
fileID	String	ไอดีของไฟล์ที่เก็บอยู่ในโพสต์
disable	Number	สถานะของโพสต์ ปกติมีค่าเป็น 0 หากไม่มีความเคลื่อนไหวแล้วจะมีค่าเป็น 1
delDate	Date	วันที่โพสต์ถูก disable

3. Model Comment

แบบจำลอง Comment ทำหน้าเก็บรายละเอียดของแต่ละความคิดเห็น ดังนี้

ตารางที่ 3.18 รายละเอียดของ model Comment

Feild	Datatype	Description
_id	ObjectId	ไอดีประจำความคิดเห็น
postID	String	ไอดีของโพสต์ที่ความคิดเห็นนี้อยู่
createdBy	String	ไอดีของผู้ใช้ที่แสดงความคิดเห็น
message	String	เนื้อหาของความคิดเห็น
date	Date	วันที่สร้างความคิดเห็น

disable	Number	สถานะของความคิดเห็น ปกติมีค่าเป็น 0 หากไม่มีความเคลื่อนไหวแล้วจะมีค่าเป็น 1
delDate	Date	วันที่ความคิดเห็นถูก disable

4. Model Like

แบบจำลอง Like ทำหน้าเก็บรายละเอียดของแต่ละข้อมูลการถูกใจ ดังนี้

ตารางที่ 3.19 รายละเอียดของ model Like

Feild	Datatype	Description
_id	ObjectId	ไอดีประจำการถูกใจ
postID	String	ไอดีของโพสต์ที่มีการถูกใจ
commentID	String	ไอดีของความคิดเห็นที่มีการกดถูกใจ
likedBy	String	ไอดีของผู้ใช้ที่ทำการถูกใจ
date	String	วันที่ทำการการถูกใจ

5. Model File

แบบจำลอง File ทำหน้าเก็บรายละเอียดของแต่ละข้อมูลการถูกใจ ดังนี้

ตารางที่ 3.20 รายละเอียดของ model File

Feild	Datatype	Description
_id	ObjectId	ไอดีประจำไฟล์
name	String	ชื่อไฟล์
expired_at	Date	วันที่ที่ไฟล์จะถูกลบออกจากระบบ
path	String	เส้นทางที่ใช้ในการเก็บไฟล์
date	Date	วันที่ทำการอัปโหลดไฟล์

6. Model Report

แบบจำลอง Report ทำหน้าเก็บรายละเอียดของแต่ละการแจ้งลบที่มีผู้ใช้ได้แจ้งเข้ามา ดังนี้

ตารางที่ 3.21 รายละเอียดของ model Report

Feild	Datatype	Description
_id	ObjectId	ไอดีประจำการแจ้งลบ
reportedBy	String	ไอดีของผู้ใช้ที่ทำการแจ้งลบ
postID	Date	ไอดีของโพสต์ที่ถูกแจ้งลบ
description	String	รายละเอียดที่ทำการแจ้งลบ
date	Date	วันที่ทำการแจ้งลบ

7. Model Category

แบบจำลอง Category ทำหน้าเก็บหมวดหมู่ที่โพสต์สามารถเลือกได้ ดังนี้

ตารางที่ 3.22 รายละเอียดของ model Category

Feild	Datatype	Description
_id	ObjectId	ไอดีประจำหมวดหมู่
name	String	ชื่อของหมวดหมู่

3.5 API ที่ใช้ภายในระบบ

เนื่องจากระบบใช้การติดต่อระหว่างฝั่งเซิร์ฟเวอร์และฝั่งผู้ใช้ผ่านการเรียกใช้ API จึงได้มีการออกแบบ API ต่าง ๆ ที่ต้องใช้ภายในเว็บแอปพลิเคชัน โดยแบ่งหมวดหมู่ ดังนี้

3.5.1 User เป็นส่วนที่ใช้สำหรับติดต่อกับฐานข้อมูลในส่วนที่เกี่ยวข้องกับผู้ใช้

ตารางที่ 3.23 API ที่ใช้ติดต่อกับส่วนที่เกี่ยวข้องกับผู้ใช้ในฐานข้อมูล

Method	Path	Description	Role		
			guest	user	admin
GET	/users	ร้องขอข้อมูลของผู้ใช้	-	-	✓
GET	/users/signedIn	ร้องขอข้อมูลของผู้ใช้ที่กำลังอยู่ในระบบ	-	✓	✓
GET	/users/{{user_id}}	ร้องขอข้อมูลของผู้ใช้โดยใช้ไอดี	-	✓	✓
POST	/users	เพิ่มผู้ใช้	-	-	✓
PATCH	/users/{{user_id}}	แก้ไขข้อมูลผู้ใช้	-	-	✓
DELETE	/users/{{user_id}}	ซ่อนผู้ใช้ออกจากระบบ	-	-	✓

3.5.2 Post เป็นส่วนที่ใช้สำหรับติดต่อกับฐานข้อมูลในส่วนที่เกี่ยวข้องกับโพสต์

ตารางที่ 3.24 API ที่ใช้ติดต่อกับส่วนที่เกี่ยวข้องกับโพสต์ในฐานข้อมูล

Method	Path	Description	Role		
			guest	user	admin
GET	/posts	ร้องขอข้อมูลโพสต์	✓	✓	✓
GET	/posts/count	ร้องข้อจำนวนโพสต์	✓	✓	✓
GET	/posts/{{post_id}}	ร้องขอข้อมูลโพสต์โดยใช้ไอดี	✓	✓	✓
GET	/posts/{{post_id}}/comments	ร้องขอข้อมูลความคิดเห็นที่อยู่ในโพสต์	-	✓	✓
GET	/posts/{{post_id}}/comments/count	ร้องขอจำนวนความคิดเห็นที่อยู่ในโพสต์	-	✓	✓

GET	/posts/{{post_id}}/likes	ร้องขอข้อมูลการถูกใจในโพสต์	✓	✓	✓
GET	/posts/{{post_id}}/likes/count	ร้องขอจำนวนการถูกใจในโพสต์	✓	✓	✓
GET	/posts/{{post_id}}/files	ร้องขอข้อมูลไฟล์ในโพสต์	✓	✓	✓
GET	/posts/{{post_id}}/filenum	ร้องขอจำนวนไฟล์ในโพสต์	-	✓	✓
GET	/posts/{{post_id}}/checkuser	ตรวจสอบว่าผู้ใช้ที่กำลังอยู่ในระบบได้ทำการถูกใจโพสต์หรือไม่	-	✓	✓
POST	/posts	เพิ่มโพสต์	-	✓	✓
PATCH	/posts/{{post_id}}	แก้ไขโพสต์	-	✓	✓
PATCH	/posts/{{post_id}}/file	เพิ่มไฟล์ลงในโพสต์	-	✓	✓
DELETE	/posts/{{post_id}}	ซ่อนโพสต์ออกจากระบบ	-	-	✓
DELETE	/posts/{{post_id}}/file/{{file_id}}	ลบไฟล์ออกจากโพสต์	-	✓	✓

3.5.3 Comment เป็นส่วนที่ใช้สำหรับติดต่อกับฐานข้อมูลในส่วนที่เกี่ยวข้องกับความคิดเห็น

ตารางที่ 3.25 API ที่ใช้ติดต่อกับส่วนที่เกี่ยวข้องกับความคิดเห็นในฐานข้อมูล

Method	Path	Description	Role		
			guest	user	admin
GET	/comments	ร้องขอข้อมูลความคิดเห็นทั้งหมด	-	-	✓
GET	/comments/{{comment_id}}	ร้องขอข้อมูลความคิดเห็นโดยใช้ไอดี	✓	✓	✓
GET	/comments/{{comment_id}}/likes	ร้องขอข้อมูลการถูกใจในความคิดเห็น	-	-	✓

GET	/comments/{{comment_id}}/likes/count	ร้องขอจำนวนการถูกใจในความคิดเห็น	✓	✓	✓
GET	/comments/{{comment_id}}/checkuser	ตรวจสอบว่าผู้ใช้ที่กำลังอยู่ในระบบได้ทำการถูกใจความคิดเห็นหรือไม่	-	✓	✓
POST	/comments	เพิ่มความคิดเห็น	-	✓	✓
DELETE	/comments/{{comment_id}}	ซ่อนความคิดเห็นออกจากระบบ	-	-	✓

3.5.4 File เป็นส่วนที่ใช้สำหรับติดต่อกับฐานข้อมูลในส่วนที่เกี่ยวข้องกับไฟล์

ตารางที่ 3.26 API ที่ใช้ติดต่อกับส่วนที่เกี่ยวข้องกับไฟล์ในฐานข้อมูล

Method	Path	Description	Role		
			guest	user	admin
GET	/files	ร้องขอข้อมูลไฟล์ทั้งหมด	-	-	✓
GET	/files/{{file_id}}	ร้องขอข้อมูลไฟล์โดยใช้ไอดี	✓	✓	✓
GET	/files/show/{{file_id}}	เปิดไฟล์ในแท็บใหม่	✓	✓	✓

3.5.5 Like เป็นส่วนที่ใช้สำหรับติดต่อกับฐานข้อมูลในส่วนที่เกี่ยวข้องกับการถูกใจ

ตารางที่ 3.27 API ที่ใช้ติดต่อกับส่วนที่เกี่ยวข้องกับการถูกใจในฐานข้อมูล

Method	Path	Description	Role		
			guest	user	admin
GET	/likes	ร้องขอข้อมูลการถูกใจทั้งหมด	-	-	✓
GET	/likes/{{like_id}}	ร้องขอข้อมูลการถูกใจโดยใช้ไอดี	-	-	✓
POST	/likes	เพิ่มการถูกใจ	-	✓	✓
DELETE	/likes/comment/{{comment_id}}	ยกเลิกการถูกใจออกจากความคิดเห็น	-	✓	✓
DELETE	/likes/post/{{post_id}}	ยกเลิกการถูกใจออกจากโพสต์	-	✓	✓

3.5.6 Category เป็นส่วนที่ใช้สำหรับติดต่อกับฐานข้อมูลในส่วนที่เกี่ยวข้องกับหมวดหมู่

ตารางที่ 3.28 API ที่ใช้ติดต่อกับส่วนที่เกี่ยวข้องกับหมวดหมู่ในฐานข้อมูล

Method	Path	Description	Role		
			guest	user	admin
GET	/categories	ร้องขอหมวดหมู่ทั้งหมด	✓	✓	✓
POST	/categories	เพิ่มหมวดหมู่	-	-	✓
PATCH	/categories/{{category_id}}	แก้ไขหมวดหมู่โดยใช้ไอดี	-	-	✓
DELETE	/categories/{{category_id}}	ลบหมวดหมู่	-	-	✓

3.5.7 Report เป็นส่วนที่ใช้สำหรับติดต่อกับฐานข้อมูลในส่วนที่เกี่ยวข้องกับการแจ้งลบโพสต์

ตารางที่ 3.29 API ที่ใช้ติดต่อกับส่วนที่เกี่ยวข้องกับการแจ้งลบโพสต์ในฐานข้อมูล

Method	Path	Description	Role		
			guest	user	admin
GET	/reports	ร้องขอข้อมูลการแจ้งลบ	-	-	✓
GET	/reports/count	ร้องขอจำนวนการแจ้งลบ	-	-	✓
GET	/reports/{{report_id}}	ร้องขอข้อมูลการแจ้งลบโดยใช้ไอดี	-	-	✓
POST	/reports/{{post_id}}	เพิ่มการแจ้งลบ	-	✓	✓

3.5.8 SignIn เป็นส่วนที่ใช้สำหรับติดต่อกับฐานข้อมูลในส่วนที่เกี่ยวข้องกับการเข้าสู่ระบบ

ตารางที่ 3.30 API ที่ใช้ติดต่อกับส่วนที่เกี่ยวข้องกับการเข้าสู่ระบบในฐานข้อมูล

Method	Path	Description	Role		
			guest	user	admin
POST	/signin	ตรวจสอบการเข้าสู่ระบบ	✓	✓	✓

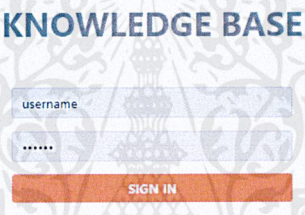
บทที่ 4
ผลการทำโครงงาน

จากการออกแบบและพัฒนาเว็บแอปพลิเคชันตามที่กล่าวไปในบทที่ 3 ผู้จัดทำได้พัฒนาเว็บแอปพลิเคชันได้สำเร็จ โดยมีส่วนแสดงผลและความสามารถ ดังนี้

4.1 ส่วนแสดงผลสำหรับผู้เยี่ยมชม ผู้ใช้งานในระบบ และผู้ดูแลระบบ

1. ส่วนของการเข้าสู่ระบบ (Sign in)

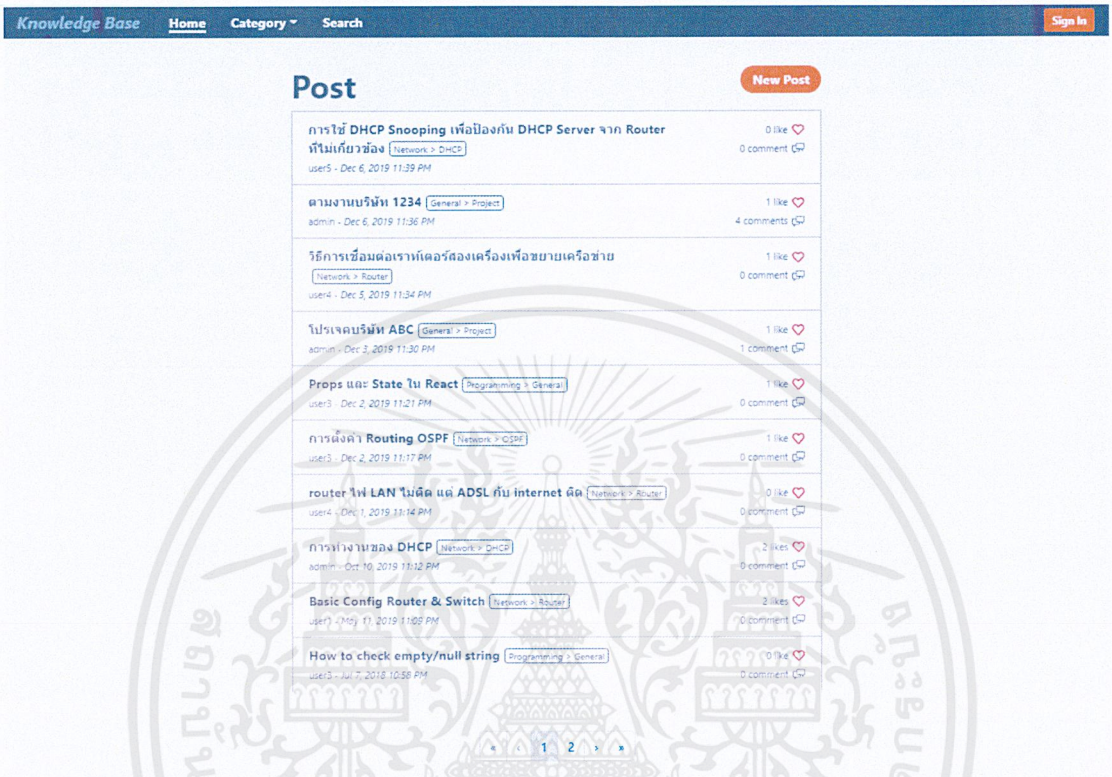
ผู้ใช้กรอก username และ password เพื่อทำการเข้าสู่ระบบ



ภาพที่ 4.1 จอภาพเข้าสู่ระบบ

2. ส่วนของการแสดงโพสต์ในระบบ

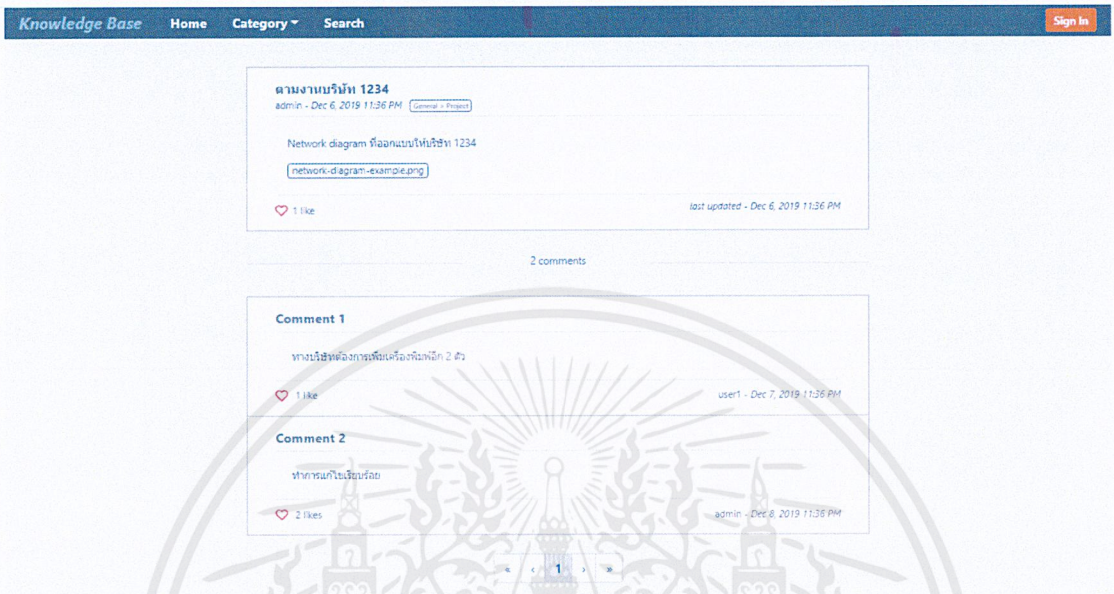
แสดงโพสต์ทั้งหมดที่ผู้ใช้สามารถเรียกดูได้ โดยโพสต์จะถูกแบ่งเป็นหลายหน้า ผู้ใช้สามารถคลิกที่แถบเลขหน้าด้านล่างเพื่อเลือกหน้าของโพสต์ที่ต้องการเรียกดูได้



ภาพที่ 4.2 จอภาพหลัก

3. ส่วนของการแสดงรายละเอียดโพสต์

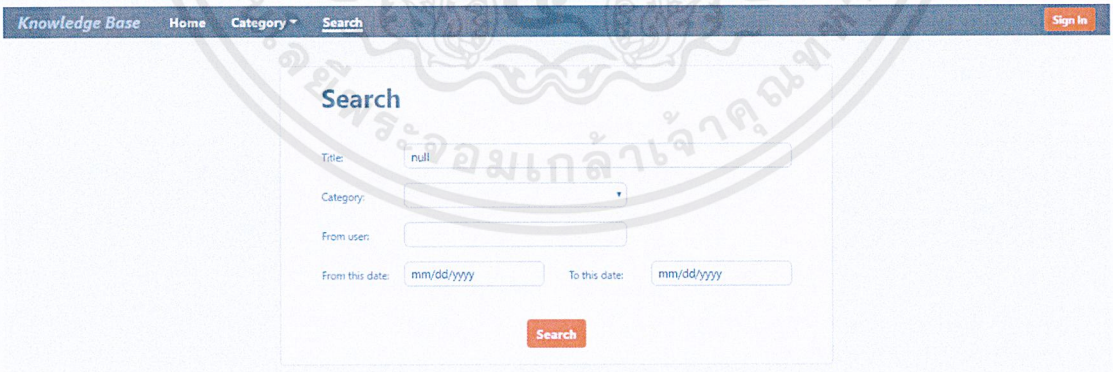
แสดงโพสต์ที่ต้องการเรียกดู โดยจะแสดงรายละเอียดของโพสต์และความคิดเห็นภายในโพสต์นี้



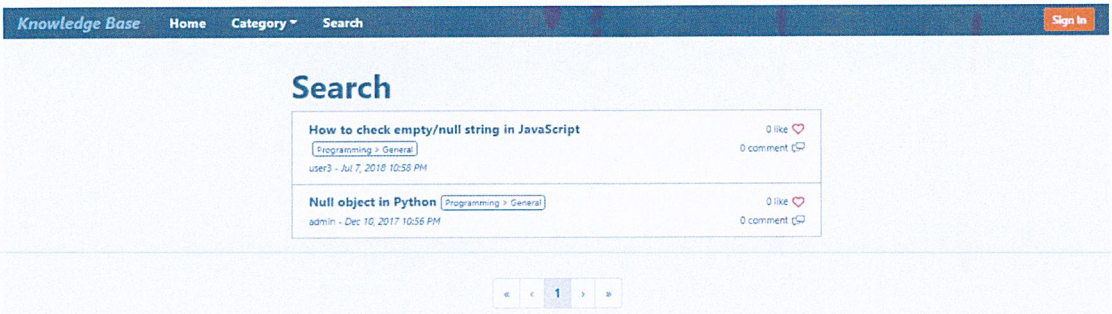
ภาพที่ 4.3 จอภาพโพสต์

4. ส่วนของการค้นหาโพสต์

สำหรับให้ผู้ใช้ค้นหาโพสต์ที่ต้องการตามเงื่อนไขที่สนใจ



ภาพที่ 4.4 จอภาพค้นหา

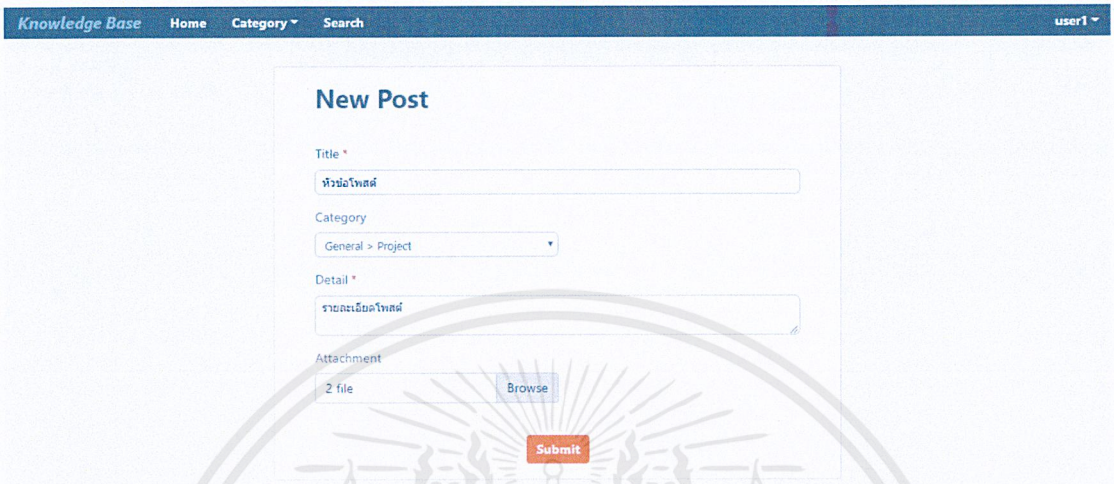


ภาพที่ 4.5 จอภาพผลการค้นหา

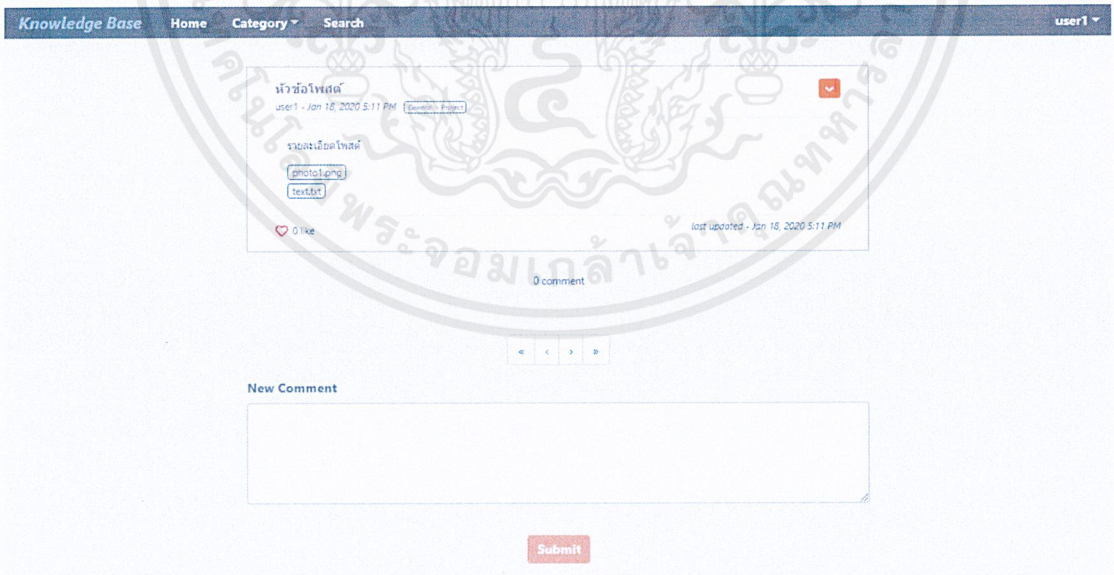
4.2 ส่วนแสดงผลสำหรับผู้ใช้งานในระบบ และผู้ดูแลระบบ

1. ส่วนของการสร้างโพสต์

กรอกรายละเอียดของโพสต์เพื่อทำการสร้างโพสต์ใหม่



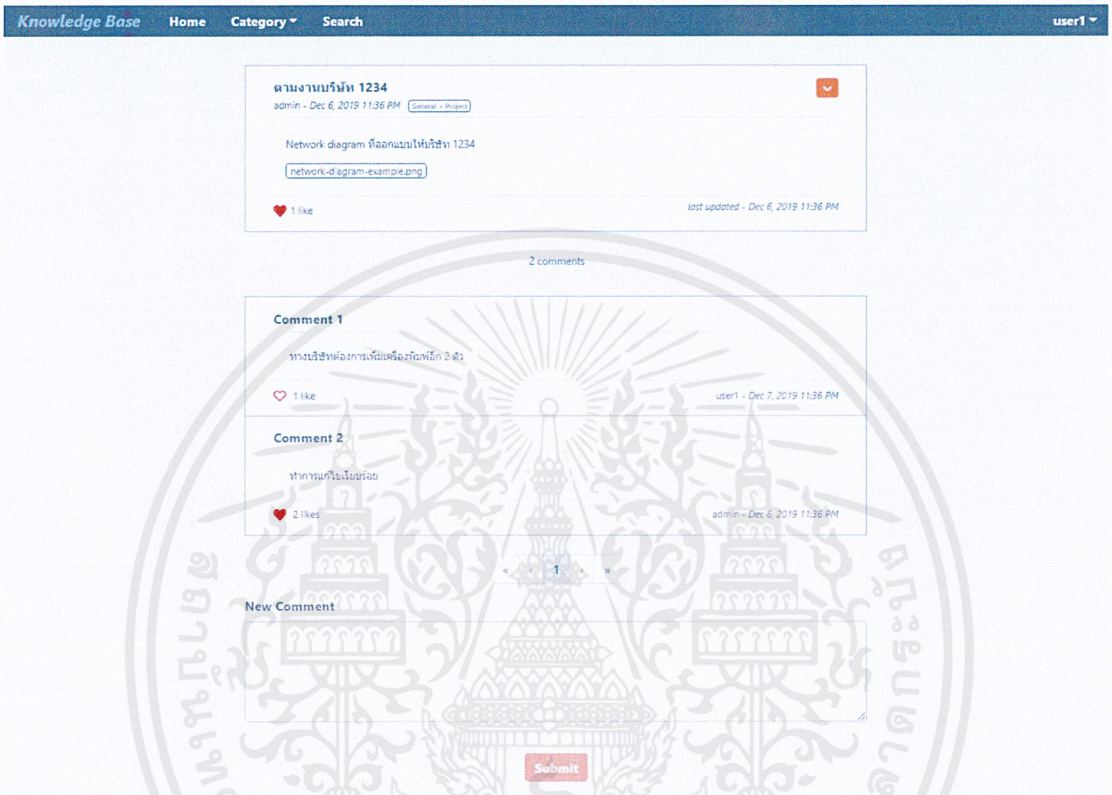
ภาพที่ 4.6 จอภาพสร้างโพสต์



ภาพที่ 4.7 จอภาพโพสต์ที่ถูกสร้างใหม่

2. ส่วนของการแสดงรายละเอียดโพสต์ที่ผู้ใช้งานในระบบทำการถูกใจโพสต์

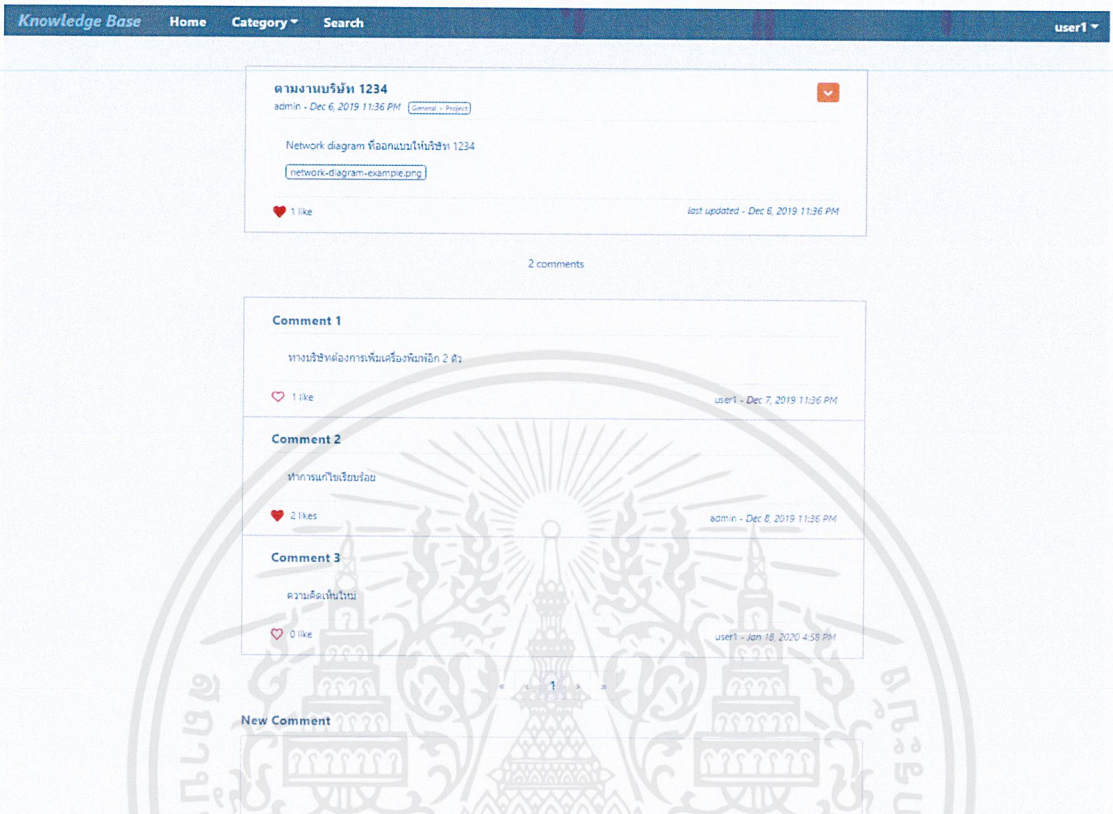
หากผู้ที่กำลังอยู่ในระบบเคยทำการถูกใจโพสต์หรือความคิดเห็นในโพสต์นี้ ไอคอนหัวใจของโพสต์หรือความคิดเห็นที่ถูกทำการถูกใจจะเป็นสีแดง ซึ่งสามารถคลิกไอคอนหัวใจสีขาวเพื่อทำการถูกใจ และคลิกไอคอนหัวใจสีแดงเพื่อยกเลิกการถูกใจ



ภาพที่ 4.8 จอภาพโพสต์ที่มีการถูกใจโพสต์และความคิดเห็น

3. ส่วนของการเพิ่มความคิดเห็นในโพสต์

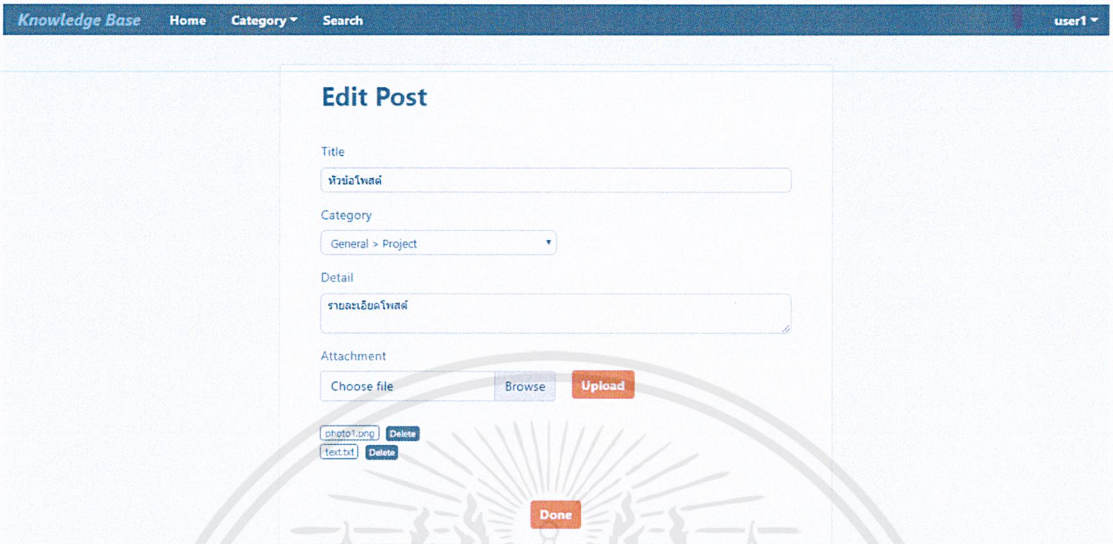
ความคิดเห็นที่ต้องการลงในช่อง New Comment เพื่อเพิ่มความคิดเห็นใหม่



ภาพที่ 4.9 จอภาพโพสต์ที่มีการเพิ่มความคิดเห็น

4. ส่วนของการแก้ไขโพสต์

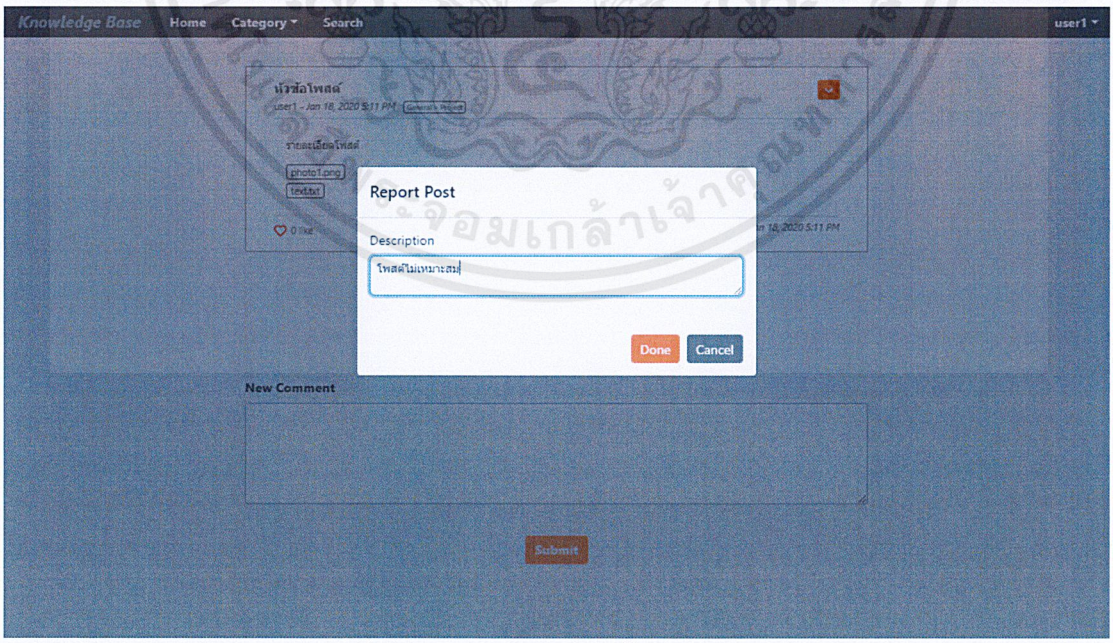
ผู้สร้างโพสต์คลิกรายการ Edit ระบบจะแสดงจอภาพแก้ไขโพสต์



ภาพที่ 4.10 จอภาพแก้ไขโพสต์

5. ส่วนของการแจ้งลบโพสต์

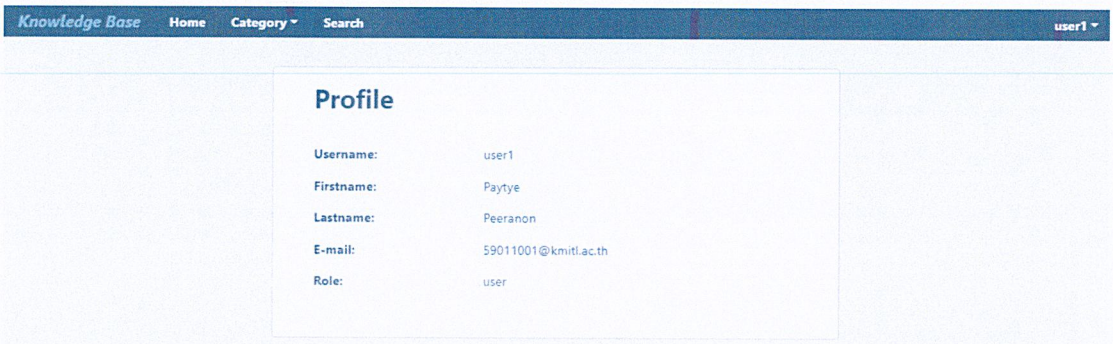
คลิกรายการ Report Post และกรอกเหตุผลที่ต้องการลบโพสต์เพื่อทำการแจ้งลบ



ภาพที่ 4.11 จอภาพแจ้งลบโพสต์

6. ส่วนของการแสดงข้อมูลผู้ใช้งาน

คลิก username ที่ด้านขวามือของแถบนำทางเพื่อเรียกดูข้อมูลผู้ใช้งาน

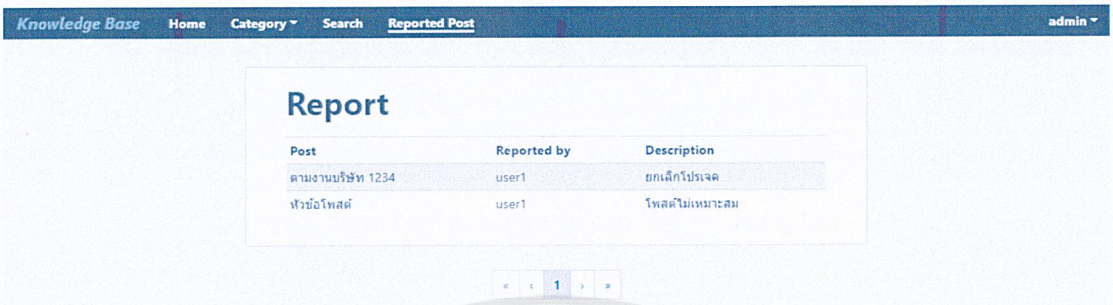


ภาพที่ 4.12 จอภาพข้อมูลผู้ใช้งาน

4.3 ส่วนแสดงผลสำหรับผู้ดูแลระบบ

1. ส่วนของการแสดงโพสต์ที่ถูกแจ้งลบ

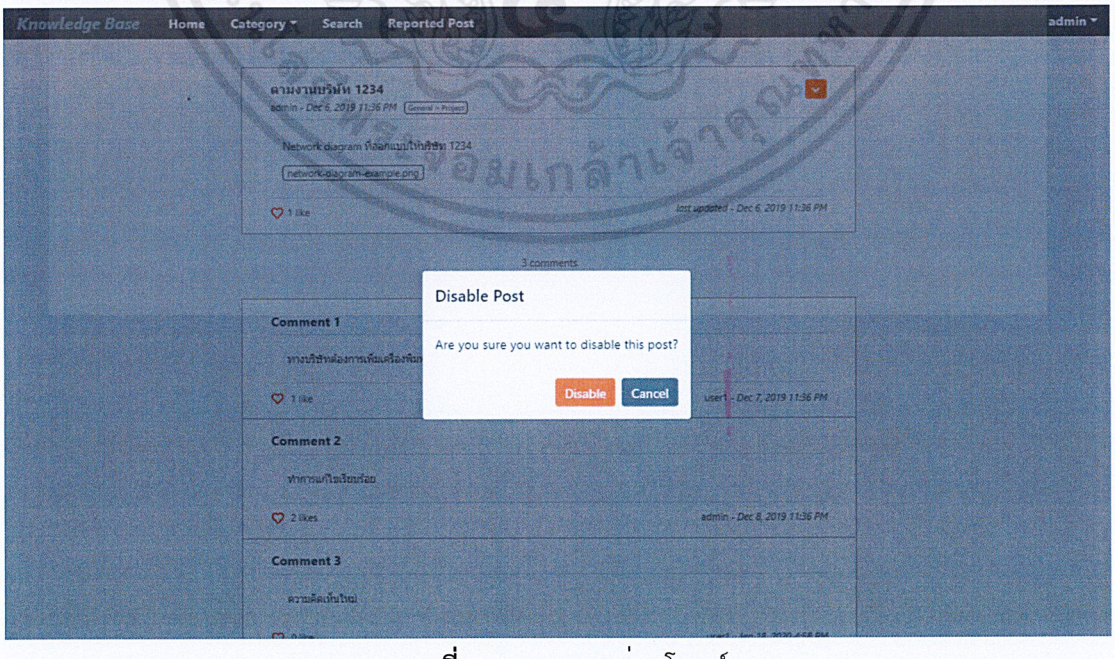
แสดงโพสต์ที่ถูกแจ้งลบและเหตุผลที่ถูกแจ้งลบ



ภาพที่ 4.11 จอภาพโพสต์ที่ถูกแจ้งลบ

2. ส่วนของการซ่อนโพสต์ออกจากระบบ

คลิกรายการ Disable Post เพื่อทำการซ่อนโพสต์ออกจากระบบ



ภาพที่ 4.12 จอภาพซ่อนโพสต์

บทที่ 5
สรุปผลการทำโครงงานและข้อเสนอแนะ

5.1 สรุปผลการทำโครงงาน

เว็บแอปพลิเคชันที่พัฒนาขึ้นมามีความสามารถในการทำงานครบถ้วนตามวัตถุประสงค์ที่ต้องการ ซึ่งก็คือ สามารถสร้างโพสต์ที่บันทึกแนวทางการแก้ปัญหาหรือสามารถติดตามความคืบหน้าของงานได้ และสามารถนำระบบยืนยันตัวตนไปใช้กับเว็บแอปพลิเคชันอื่นที่อยู่ในระบบได้

5.2 ข้อเสนอแนะ

1. หากเว็บแอปพลิเคชันมี Text Editor ที่ใช้ในการเขียนรายละเอียดของโพสต์หรือการเขียนความคิดเห็นจะทำให้ใช้งานได้อย่างมีประสิทธิภาพมากขึ้น เช่น ใส่ตัวหนาให้ตัวหนังสือได้ หรือใส่ตารางลงไปในรายละเอียดของโพสต์ได้ เป็นต้น

2. เว็บแอปพลิเคชันในขณะนี้ยังไม่มีระบบแจ้งเตือนเมื่อโพสต์ที่สนใจมีการเปลี่ยนแปลง เช่น มีผู้ใช้มาแสดงความคิดเห็นในโพสต์ หากมีระบบแจ้งเตือนจะทำให้สามารถติดตามโพสต์ที่สนใจได้ง่ายขึ้น

บรรณานุกรม

[1] Nodejs and Mongoose MongoDB ODM. สืบค้นเมื่อวันที่ 15 กรกฎาคม 2562 จาก https://khasathan.in.th/archives/237/nodejs-and-mongoose-mongodb-odm

[2] Express.js เอ็กเพรส ดอทเจเอส คืออะไร. สืบค้นเมื่อวันที่ 16 กรกฎาคม 2562 จาก https://www.mindphp.com/คู่มือ/73-คืออะไร/3874-what-is-express-js.html

[3] จัดการฐานข้อมูลได้ง่ายๆ ด้วย mongoDB. สืบค้นเมื่อวันที่ 25 กรกฎาคม 2562 จาก https://www.nipa.cloud/blogs/จัดการฐานข้อมูลได้ง่าย/

[4] REST กับ RESTful API ต่างกันนะรู้ยัง. สืบค้นเมื่อวันที่ 29 กรกฎาคม 2562 จาก https://medium.com/@iamgique/restful-api-กับ-rest-api-ต่างกันนะรู้ยัง-2c70c42990e3

[5] Introduction to JSON Web Tokens. สืบค้นเมื่อวันที่ 20 สิงหาคม 2562 จาก https://jwt.io/introduction/

[6] Using middleware. สืบค้นเมื่อวันที่ 23 สิงหาคม 2562 จาก https://expressjs.com/en/guide/using-middleware.html

[7] React คืออะไร?. สืบค้นเมื่อวันที่ 9 กันยายน 2562 จาก https://www.designil.com/react-คืออะไร.html

[8] Cross-Origin Resource Sharing (CORS) เป็นสิ่งที่ Web Developer ต้องควรรู้. สืบค้นเมื่อวันที่ 18 กันยายน 2562 จาก https://medium.com/nellika/cors-เป็นสิ่งที่-web-developer-ต้องควรรู้-c906b1b47958