

CS 256/2

เอกสารโครงการคอมพิวเตอร์

การพัฒนาระบบการจัดการเนื้อหาของเว็บไซต์สำหรับนักพัฒนาโปรแกรม

Content Management System: Website Development

for Programmer

โดย

633020042-1 นายอนิรุทธิ์ ชาวกล้า

633020419-0 นายศักดิ์สิทธิ์ ศิริศักดา

อาจารย์ที่ปรึกษา : ผศ.ดร. ชิตสุชา สุ่มเล็ก

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา SC314775 โครงการวิทยาการคอมพิวเตอร์ 2

ภาคเรียนที่ 2 ปีการศึกษา 2566 สาขาวิชาการคอมพิวเตอร์

วิทยาลัยการคอมพิวเตอร์ มหาวิทยาลัยขอนแก่น

(เดือน มกราคม พ.ศ. 2567)

อนิรุท ชาวกล้า และ ศักดิ์สิทธิ์ ศิริศักดา. 2566. การพัฒนาระบบการจัดการเนื้อหาของเว็บไซต์สำหรับนักพัฒนาโปรแกรม. โครงการนักคอมพิวเตอร์ ปริญญาวิทยาศาสตรบัณฑิต สาขาวิชาการคอมพิวเตอร์ วิทยาลัยการคอมพิวเตอร์ มหาวิทยาลัยขอนแก่น.

อาจารย์ที่ปรึกษา: ผศ.ดร. ชิดสุชา สุ่มเล็ก

บทคัดย่อ

วัตถุประสงค์ในการพัฒนาระบบการจัดการเนื้อหาของเว็บไซต์สำหรับนักพัฒนาโปรแกรม คือเพื่อลดขั้นตอนในการฝังโค้ดบล็อก หรือ “Code block” เข้าไปในบทความของบล็อก (Post) โดยโค้ดบล็อกนี้สามารถทำ การคอมpileได้ เพื่อให้ได้ผลลัพธ์ออกมา ซึ่งโดยปกติแล้วบริการของระบบการจัดการเนื้อหารูปแบบบล็อกทั่วไป จำเป็นจะต้องเพิ่มส่วนประกอบเสริม (Extension) เข้าไปในตัวของบล็อกด้วย โดยบางครั้งจะทำให้เกิดความไม่ สะดวก เนื่องด้วยจะต้องไปใช้บริการจากบุคคลที่สาม หรือตัวบล็อกเองไม่สามารถเพิ่มส่วนประกอบเสริมโดยง่ายได้ และไม่เป็นมิตร (User - friendly) ต่อนักพัฒนาโปรแกรมเมื่อใหม่อีกด้วย

ผู้พัฒนาจึงได้นำเทคโนโลยีคอนเทนเนอร์ หรือ “Container” ที่ทำหน้าที่คล้ายกับเครื่องเสมือน (Virtual machine, VM) แต่ใช้ทรัพยากรน้อยกว่า แม้จะมีความยืดหยุ่นน้อยกว่า แต่สามารถใช้งานกับการสร้างเว็บไซต์ เพื่อให้สามารถคอมpileโค้ดภาษาต่าง ๆ ได้ มาใช้ในการดำเนินโครงการ ดังนั้น โครงการนี้ จะใช้หลักการของ Container เพื่อจำลองเป็นสิ่งแวดล้อม (Environment) ให้กับการคอมpileโค้ดแต่ละภาษาและจะนำเสนอใน รูปแบบของการทำเว็บไซต์ระบบการจัดการเนื้อหารูปแบบบล็อก โดยมีเครื่องมือในการสร้างโค้ดบล็อก และคอมpile โค้ดได้ในตัว

คำสำคัญ: บล็อก, โค้ดบล็อก, คอนเทนเนอร์

Anirut Chaogla and Saksit Sirisakda. **Content Management System: Website Development for Programmer.** Bachelor of Science Project in Computer Science, College of Computing, Khon Kaen University.

Project Advisor: Asst. Prof. Chitsutha Soomlek, Ph.D.

ABSTRACT

The purpose of developing a content management system tailored for programmers is to streamline the process of embedding code blocks into blog articles or posts. These code blocks can encompass compiled code, yielding specific results. Traditionally, integrating code into blog platforms necessitates additional components (extensions) within the block's body. However, this approach can lead to inconvenience, especially when relying on third-party services, making it challenging to add extensions seamlessly. This complexity poses challenges for new programmers.

To address these issues, the developer has harnessed container technology, functioning akin to a virtual machine but consuming fewer resources. Although container technology offers less flexibility, it proves invaluable for building websites that can compile code in diverse programming languages. Consequently, this project employs container technology to emulate environments for code compilation across various languages. This innovative approach will be presented as a content management system within a platform, equipped with user-friendly tools to effortlessly create code blocks and compile code autonomously.

Keywords: Blog, Code block, Container

กิตติกรรมประกาศ

รายงานความก้าวหน้าโครงการฉบับนี้ สำเร็จลุล่วงได้ด้วยความเมตตากรุณา และความเอื้อเฟื้อจาก
อาจารย์ที่ปรึกษา ผศ.ดร. ชิตสุชา สุมเล็ก ที่ได้ให้ความช่วยเหลือ คำแนะนำต่าง ๆ ตลอดจนให้ความรู้แก่ข้าพเจ้า

ขอขอบคุณเพื่อน ๆ ทุกคนที่เคยให้กำลังใจ และความช่วยเหลือในด้านต่าง ๆ เสมอมา แม้ในยามสิ้นหวัง
ท้อแท้ และหมดกำลังใจ

สุดท้ายนี้ ขอขอบคุณตัวข้าพเจ้าเองที่มีความพยายาม พากเพียร อุตสาหะ และไม่ย่อท้อต่ออุปสรรคใด ๆ

อนิรุทธิ์ ชาวกล้า

ศักดิ์สิทธิ์ ศิริศักดา

สารบัญ

บทคัดย่อ	ก
ABSTRACT	ข
กิตติกรรมประกาศ.....	ค
สารบัญตาราง	ฉ
สารบัญภาพ.....	ช
บทที่ 1 บทนำ.....	1
1.1 ความเป็นมาและความสำคัญของปัญหา	1
1.2 วัตถุประสงค์ของโครงการ	1
1.3 ขอบเขต และความต้องการของโครงการ	2
1.4 ประโยชน์ที่คาดว่าจะได้รับ.....	4
บทที่ 2 ทฤษฎีและผลงานวิจัยที่เกี่ยวข้อง	5
2.1 ระบบการจัดการเนื้อหา	5
2.2 หลักการทำงานของคอมไฟเลอร์	6
2.3 สถาปัตยกรรมระบบไมโครเซอร์วิส	8
บทที่ 3 วิธีการดำเนินงาน.....	14
3.1 เครื่องมือที่ใช้ในการพัฒนา.....	14
3.2 ขั้นตอนการดำเนินงาน	15
3.3 แผนการดำเนินงาน.....	16
บทที่ 4 ผลการดำเนินงาน	17
4.1 ภาพรวมของระบบ	17
4.1.1 การยืนยันตัวตนของผู้ใช้	17
4.1.2 การจัดการข้อมูลผู้ใช้.....	19
4.1.3 การจัดการบทความ	22
4.2 ส่วนติดต่อผู้ใช้	26

สารบัญ (ต่อ)

4.3 สถาปัตยกรรมของระบบเบท้าบล็อก	35
4.3.1 บริการจัดการเกตเวย์ของ API (API Gateway service)	37
4.3.2 บริการจัดการบัญชีและยืนยันตัวตนผู้ใช้ (Identity service).....	38
4.3.3 บริการจัดการไฟล์ (Media Service).....	39
4.3.4 บริการจัดการส่วนติดต่อผู้ใช้ (Web UI service).....	40
4.3.5 บริการจัดการบทความ (Article Service).....	41
4.3.6 บริการจัดการความคิดเห็น	42
4.3.7 บริการจัดการข้อมูลผู้ใช้ (User information service).....	43
4.3.8 บริการรันโค้ด (Code runner service).....	44
บทที่ 5 สรุปผลการดำเนินงาน	45
5.1 สรุปการดำเนินงานโครงการ	45
5.2 ข้อจำกัดของระบบ	50
5.3 ปัญหาอุปสรรค และแนวทางแก้ไข.....	51
5.4 ข้อเสนอแนะ	51
เอกสารอ้างอิง.....	52

สารบัญตาราง

ตารางที่ 1 ตารางแสดงแผนการดำเนินงานโครงการ.....	16
ตารางที่ 2 แผนภาพกรณีการใช้งานสำหรับการเข้าสู่ระบบ (Login)	17
ตารางที่ 3 แผนภาพกรณีการใช้งานสำหรับการลงทะเบียน (Register)	18
ตารางที่ 4 แผนภาพกรณีการใช้งานสำหรับการออกจากระบบ (Logout)	19
ตารางที่ 5 แผนภาพกรณีการใช้งานสำหรับการดูโปรไฟล์ผู้ใช้ (View user profile)	20
ตารางที่ 6 แผนภาพกรณีการใช้งานสำหรับการค้นหาผู้ใช้ด้วยชื่อ (Find user by name).....	20
ตารางที่ 7 แผนภาพกรณีการใช้งานสำหรับการแก้ไขโปรไฟล์ (Edit profile information).....	20
ตารางที่ 8 แผนภาพกรณีการใช้งานสำหรับการติดตามผู้ใช้ (Follow another user)	21
ตารางที่ 9 แผนภาพกรณีการใช้งานสำหรับการยกเลิกติดตามผู้ใช้ (Unfollow user)	21
ตารางที่ 10 แผนภาพกรณีการใช้งานสำหรับการร่างบทความ (Draft article).....	22
ตารางที่ 11 แผนภาพกรณีการใช้งานสำหรับการจัดการบทความ (Manage article)	23
ตารางที่ 12 แผนภาพกรณีการใช้งานสำหรับการค้นดูบทความ (Discover articles).....	23
ตารางที่ 13 แผนภาพกรณีการใช้งานสำหรับการอ่านบทความ (Read an article)	24
ตารางที่ 14 แผนภาพกรณีการใช้งานสำหรับการรันโค้ด (Run code).....	24
ตารางที่ 15 แผนภาพกรณีการใช้งานสำหรับการรันโค้ด (Run code).....	24
ตารางที่ 16 แผนภาพกรณีการใช้งานสำหรับการบันทึกบทความไว้ดูภายหลัง (Save article)	25
ตารางที่ 17 แผนภาพกรณีการใช้งานสำหรับการดูบทความที่นิยม (View a trending)	25
ตารางที่ 18 รายงาน และความก้าวหน้า	45
ตารางที่ 18 รายงาน และความก้าวหน้า (ต่อ)	46
ตารางที่ 18 รายงาน และความก้าวหน้า (ต่อ)	47
ตารางที่ 18 รายงาน และความก้าวหน้า (ต่อ)	48
ตารางที่ 18 รายงาน และความก้าวหน้า (ต่อ)	49
ตารางที่ 18 รายงาน และความก้าวหน้า (ต่อ)	50

สารบัญภาพ

ภาพที่ 1	ขั้นตอนในการทำงานของคอมเพล่อร์.....	7
ภาพที่ 2	แผนภาพกรณีการใช้งาน (Use case diagram) สำหรับการยืนยันตัวตนผู้ใช้ของระบบเบต้าบล็อก	17
ภาพที่ 3	แผนภาพกรณีการใช้งานสำหรับการจัดการข้อมูลผู้ใช้ของระบบเบต้าบล็อก.....	19
ภาพที่ 4	แผนภาพกรณีการใช้งานสำหรับการจัดการบทความของระบบเบต้าบล็อก.....	22
ภาพที่ 5	ส่วนติดต่อผู้ใช้ หน้าหลักของผู้ใช้ที่ยังไม่เข้าสู่ระบบ	26
ภาพที่ 6	ส่วนติดต่อผู้ใช้ หน้าหลักของผู้ใช้ที่เข้าสู่ระบบแล้ว.....	27
ภาพที่ 7	ส่วนติดต่อผู้ใช้ หน้าเข้าสู่ระบบ	27
ภาพที่ 8	ส่วนติดต่อผู้ใช้ หน้าลงทะเบียน	28
ภาพที่ 9	ส่วนติดต่อผู้ใช้ หน้าเขียนบทความ.....	29
ภาพที่ 10	ส่วนติดต่อผู้ใช้ หน้าบันทความ.....	30
ภาพที่ 11	ส่วนแสดงความคิดเห็น.....	31
ภาพที่ 12	ส่วนแสดงหน้าໂປຣໄຟລ்	32
ภาพที่ 13	ส่วนติดต่อผู้ใช้ในการเขียนโค้ด	33
ภาพที่ 14	หน้าสำหรับเขียนโค้ดในໂປສຕ്	33
ภาพที่ 15	แผนภาพสถาปัตยกรรมของระบบ Beta blog.....	35
ภาพที่ 16	สถาปัตยกรรมของบริการจัดการเกตเวย์ API	37
ภาพที่ 17	สถาปัตยกรรมของบริการจัดการบัญชีและยืนยันตัวตนผู้ใช้.....	38
ภาพที่ 18	สถาปัตยกรรมของบริการจัดการไฟล์	39
ภาพที่ 19	สถาปัตยกรรมของบริการจัดการส่วนติดต่อผู้ใช้	40
ภาพที่ 20	สถาปัตยกรรมของระบบจัดการบทความ	41
ภาพที่ 21	สถาปัตยกรรมของบริการจัดการความคิดเห็น.....	42
ภาพที่ 22	สถาปัตยกรรมของบริการจัดการข้อมูลผู้ใช้	43
ภาพที่ 23	สถาปัตยกรรมของระบบรันโค้ด	44

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

อาชีพนักพัฒนาโปรแกรม (Programmer) และวิชาชีพที่เกี่ยวข้อง เป็นส่วนหนึ่งที่ช่วยขับเคลื่อนความเจริญก้าวหน้าทางเทคโนโลยีและนวัตกรรมใหม่ ๆ ในทุกวันนี้ไม่ว่าจะเป็น ปัญญาประดิษฐ์หรือ AI (Artificial intelligence), XR (Extended reality) ส่วนหนึ่งก็มาจากการป้อนข้อมูล หรือการเขียนโปรแกรมจากนักพัฒนาโปรแกรม [1] ซึ่งอาจจะมีข้อผิดพลาดระหว่างการเขียนโปรแกรม โดยการจะหาทางแก้ข้อผิดพลาดเหล่านี้ จึงจำเป็นต้องศึกษาค้นคว้าจากแหล่งข้อมูลที่สามารถสอบถามความวิธีการแก้ปัญหาข้อผิดพลาดได้หรือศึกษาจากบทความที่ผู้เชี่ยวชาญ หรือบุคคลที่เกี่ยวข้องกับการเขียนโปรแกรม ได้สร้างขึ้นมาเพื่อให้ บุคคลอื่นในสาขาวิชาชีพนี้ ได้นำไปประยุกต์ใช้กับโปรแกรมของตนเองที่เขียนขึ้น เปรียบเสมือนชุมชน (Community) สำหรับนักพัฒนาโปรแกรมที่สร้างไว้เพื่อสอบถามปัญหา หรือศึกษารณีตัวอย่างได้ ดังเช่น เว็บไซต์ Stack Overflow ที่สร้างขึ้นเพื่อใช้สร้างกระดูกสำหรับสอบถามปัญหาเกี่ยวกับการเขียนโปรแกรม แล้วกับนักพัฒนาโปรแกรมอื่นที่ได้เข้าเยี่ยมชมเว็บไซต์แล้วเห็นกระดูกดังกล่าว ก็สามารถตอบกระทู้นั้น เพื่อช่วยแก้ปัญหาได้ [2] แต่ผู้จัดทำได้เลิงเห็นแล้วว่า เมื่อมีความจำเป็นที่จะต้องใส่ส่วนประกอบรหัสคำสั่ง หรือโค้ดบล็อก (Code block) ที่ใช้สำหรับพิมพ์รหัสคำสั่ง หรือโค้ด (Code) ลงไว้แล้วให้ผู้อ่านกระดูกสามารถประมวลผล หรือคอมไพล์ (Compile) โค้ดเหล่านั้นเพื่อให้แสดงผลลัพธ์ (Output) ออกมานอกจากนี้ใน Stack overflow ผู้ใช้จำเป็นต้องนำส่วนประกอบเสริม (Extension) ภายนอกเว็บไซต์เพิ่มเข้ามาเพื่อให้สามารถใช้ตัวโค้ดบล็อกได้ แต่ในบางครั้ง ส่วนประกอบเสริมเหล่านั้น ไม่สามารถใช้งานได้อย่างมีประสิทธิภาพ เพียงพอ เช่น จำเป็นต้องเปิดหน้าใหม่เพื่อเข้าไปยังตัวโค้ดบล็อก หรือผู้ใช้ใหม่ที่ยังไม่มีความรู้เกี่ยวกับส่วนประกอบเสริม ก็ต้องไปศึกษาตรงจุดนั้นก่อนเพื่อที่จะใช้งาน

ดังนั้นปัญหาจากที่กล่าวมาข้างต้น แนวทางแก้ไขปัญหานั้น คือการพัฒนาเว็บบล็อกให้มีความสามารถในการคอมไпал์ตัวโค้ดบล็อกให้สามารถใช้งานในตัวเว็บได้เลย โดยยังสามารถเพิ่มความสามารถด้านอื่น ๆ ให้เหมาะสมกับการใช้งานของผู้ใช้ที่เป็นนักพัฒนาโปรแกรมได้สมอ่อนการใช้เว็บบล็อกได้ตามปกติ

จากแนวคิดที่ได้กล่าวมา ผู้จัดทำจึงสนใจที่จะพัฒนาเว็บบล็อกให้มีความสามารถเฉพาะเจาะจงสำหรับการใช้งานของนักพัฒนาโปรแกรม เพื่อให้นักพัฒนาโปรแกรมสามารถใช้คอมไප์ล์ในเว็บไซต์โดยไม่ต้องใช้ส่วนประกอบเสริมจากแหล่งอื่น โดยนำมาใช้ในรูปแบบเว็บบล็อก ไม่ว่าจะเป็นการสอบถามปัญหา หรือการสร้างบทความ หรือบทเรียนทางการศึกษาเกี่ยวกับการเขียนโปรแกรม

1.2 วัตถุประสงค์ของโครงงาน

เพื่อพัฒนาระบบจัดการเนื้อหาของเว็บไซต์สำหรับนักพัฒนาโปรแกรมรูปแบบบล็อกให้สามารถสร้างโค้ดบล็อก และคอมไಪล์โค้ดบล็อกเหล่านั้นโดยใช้ระบบออนไลน์คอมไປ์เลอร์ของเว็บไซต์โดยไม่ต้องไปหาส่วนประกอบเสริมจากบริการของบุคคลที่สาม ภายในเว็บไซต์ได้เลย โดยไม่ต้องไปหาส่วนประกอบเสริมจากเครื่องคอมพิวเตอร์ที่ต่อเน็ต

1.3 ขอบเขต และความต้องการของโครงงาน

1.3.1) ผู้ใช้ระบบ

- บุคคลทั่วไปที่ต้องการเขียนและอ่านบทความ
- ผู้ที่ต้องการเขียนบทความ และแนบตัวอย่างโค้ดอย่างง่าย พร้อมตัวแก้ไขโค้ดและผลการรันที่ตอบสนองได้แบบเรียลไทม์

1.3.2) เครื่องมือและอุปกรณ์ที่จำเป็นต่อการใช้งาน

- สำหรับอุปกรณ์ที่มีขนาดหน้าจอ 10 นิ้วขึ้นไป (Desktop, Laptop)
- สำหรับเบราว์เซอร์ Chrome, Edge รุ่นไม่ต่ำกว่าปีค.ศ. 2021
- เครื่องแม่ข่ายที่ทำการรันระบบต้องมีหน่วยประมวลผล 64 bit 3 GHz ไม่น้อยกว่า 4 core หน่วยความจำไม่น้อยกว่า 4 GB และพื้นที่เก็บข้อมูลไม่น้อยกว่า 50 GB

1.3.3) ความต้องการที่เป็นหน้าที่หลัก (Functional requirement)

- ผู้ใช้ระบบสามารถสมัครและเข้าสู่ระบบด้วย ชื่อผู้ใช้และรหัสผ่านที่ผู้ใช้กำหนดเองได้ โดยที่ชื่อผู้ใช้จะต้องไม่ซ้ำกันในระบบ และประกอบด้วยอักษรภาษาอังกฤษและตัวเลข 4-20 อักษร และรหัสผ่านจะต้องประกอบด้วยอักษรภาษาอังกฤษตัวพิมพ์ใหญ่ ตัวพิมพ์เล็ก อักษรพิเศษ และตัวเลข รวมกันอย่างน้อย 6 อักษร
- เพื่อความสะดวกและรวดเร็วในการลงทะเบียนเข้าใช้ จะต้องสามารถลงทะเบียนเข้าใช้ด้วยบัญชี Facebook, Google หรือ GitHub ได้
- ผู้ใช้ระบบสามารถเพิ่มและแก้ไขประวัติโดยย่อได้ทั้ง ชื่อแสดงผล ภาพประจำตัวผู้ใช้ และข้อมูลน้ำหน้าแบบสั้น
- ผู้ใช้ระบบไม่สามารถแก้ไขชื่อผู้ใช้ได้ แต่ต้องสามารถแก้ไขรหัสผ่าน และต้องสามารถออกจากระบบได้
- ผู้ใช้ระบบสามารถเลือกดูตัวอย่างบทความทั้งหมดโดยเรียงตามเวลาได้ในหน้าหลักของ เว็บไซต์ โดยจะต้องสามารถเลือกได้ว่าต้องการดูบทความทั้งหมดในระบบ หรือ บทความของผู้ที่กำลังติดตามอยู่
- ระบบสามารถแสดงรายการหมวดหมู่ที่มีจำนวนบทความมากที่สุด 20 หมวดหมู่ และบทความที่มีจำนวนการกดถูกใจมากที่สุดจำนวน 15 บทความ ในหน้าหลัก
- ผู้ใช้ระบบสามารถเลือกเพื่ออ่านบทความ หรือบันทึกเก็บไว้อ่านทีหลังได้ โดยบทความที่ผู้ใช้ได้บันทึกไว้จะถูกรวบรวมไว้เป็นรายการส่วนตัวที่ผู้อื่นไม่สามารถมองเห็นรายการนี้ได้
- ผู้ใช้ระบบสามารถกดถูกใจต่อบทความเพื่อให้คะแนนความนิยมได้ โดยรายชื่อผู้ที่กดถูกใจ จะไม่เป็นสาธารณะเพียงแต่เป็นจำนวนคนที่กดถูกใจเท่านั้น

- ผู้ใช้ระบบสามารถแสดงความคิดเห็นต่อบทความ และตอบกลับการแสดงความคิดเห็นได้โดยผู้ใช้ทุกคนสามารถแสดงความคิดเห็นและตอบกลับได้ และแต่ละความคิดเห็นจะถูกเผยแพร่เป็นสาธารณะ
- ผู้ใช้ระบบสามารถเลือกดูบทความตามหมวดหมู่ได้ โดยที่หมวดหมู่จะถูกกำหนดโดยผู้เขียนบทความ
- ผู้ใช้ระบบสามารถใช้แบบค้นหา เพื่อค้นหาข้อมูลความ รายการหมวดหมู่ และ ชื่อผู้ใช้ได้
- ผู้ใช้ระบบสามารถเผยแพร่บทความสู่สาธารณะได้ โดยองค์ประกอบในบทความที่จำเป็นต้องระบุได้แก่ ชื่อบทความ คำนำ และบทความ ซึ่งผู้เขียนสามารถเพิ่มเนื้อหาได้ 8 ประเภท ประกอบด้วย หัวข้อ 6 ระดับ ย่อหน้า รูปภาพ เส้นแบ่งบทความแนวโน้ม ตาราง รายการ ลิงก์ และ โค้ดบล็อก โดยภายในย่อหน้าสามารถจัดรูปแบบตัวอักษรแบบ ตัวหนา ตัวเอียง และเน้นคำได้ ส่วนองค์ประกอบเสริม ผู้เขียนสามารถเพิ่มภาพปกเพื่อความสวยงาม และเพิ่มหมวดหมู่ที่กำหนดเองได้ไม่เกิน 5 หมวดหมู่ แต่ละหมวดหมู่มีความยาวไม่เกิน 20 ตัวอักษร
- ผู้ใช้ระบบสามารถลบหรือแก้ไขความขอมูลของตนเองได้
- ผู้ใช้ระบบสามารถติดตามหรือเลิกติดตามผู้ใช้อื่นเพื่อรับอ่านบทความที่ผู้ใช้นั้นเผยแพร่
- ผู้ใช้ระบบสามารถแก้ไขและทดลองรันโค้ดที่ถูกแนบไว้ในบทความได้ โดยไม่จำเป็นต้องเข้าสู่ระบบก่อน และการแก้ไขดังกล่าวเป็นการแก้ไขโค้ดเพื่อทดลองรันเท่านั้น จึงจะไม่มีการบันทึกการเปลี่ยนแปลงของโค้ด และผู้ใช้สามารถมีปฏิสัมพันธ์กับผลลัพธ์ได้แบบเรียลไทม์ ทั้ง ข้อมูลสองออก ข้อมูลนำเข้า และข้อมูลผลลัพธ์ โดยโค้ดที่รันได้จะถูกจำกัดจำนวนไม่เกิน 200 บรรทัด และต้องเป็นคอนโซลแอปพลิเคชัน ภาษา C, Python และ Java เท่านั้น

1.3.4) ความต้องการที่ไม่ใช่หน้าที่หลัก (Non-functional requirement)

- ระบบถูกออกแบบด้วยสถาปัตยกรรมแบบไมโครเซอร์วิส และแต่ละบริการควรบรรจุเป็นคอนเทนเนอร์ เพื่อให้ง่ายต่อการขยายขนาดในแนวโน้มได้ง่าย
- ระบบปรับปรุงตัวเองได้โดยอัตโนมัติ ผ่านการฝึกอบรมและการเรียนรู้ จึงสามารถปรับปรุงการทำงานได้ต่อไปอย่างต่อเนื่อง
- ระบบปรับปรุงตัวเองได้โดยอัตโนมัติ ผ่านการฝึกอบรมและการเรียนรู้ จึงสามารถปรับปรุงการทำงานได้ต่อไปอย่างต่อเนื่อง
- ระบบปรับปรุงตัวเองได้โดยอัตโนมัติ ผ่านการฝึกอบรมและการเรียนรู้ จึงสามารถปรับปรุงการทำงานได้ต่อไปอย่างต่อเนื่อง
- ระบบปรับปรุงตัวเองได้โดยอัตโนมัติ ผ่านการฝึกอบรมและการเรียนรู้ จึงสามารถปรับปรุงการทำงานได้ต่อไปอย่างต่อเนื่อง
- เพื่อป้องกันไม่ให้ผู้ใช้รันโค้ดถึงเกินไปจนทำให้เครื่องเซิร์ฟเวอร์ทำงานหนัก ระบบจำกัดการรันโค้ดของผู้ใช้ให้สามารถรันได้ 100 ครั้ง ต่อ 1 วัน ต่อ 1 IP address
- เพื่อลดการทำงานของระบบรันโค้ด ระบบสามารถบันทึกผลลัพธ์ของบางโปรแกรมลงในแคช (Cache) และสามารถนำผลลัพธ์ที่บันทึกไว้มาใช้งานได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

ผู้ใช้เว็บบล็อกในกลุ่มของนักพัฒนาโปรแกรมสามารถเขียนโค้ดบล็อกบนเว็บไซต์ได้ และใช้ออนไลน์คอมไพล์เตอร์ที่มีอยู่แล้วในระบบ เพื่อทำการรันโค้ดในโค้ดบล็อกนั้น โดยไม่ต้องใช้ส่วนเสริม (Extension) จากแหล่งบริการอื่นในการรันโค้ด

บทที่ 2

ทฤษฎีและผลงานวิจัยที่เกี่ยวข้อง

2.1 ระบบการจัดการเนื้อหา

หลังจากอินเทอร์เน็ตได้ถือกำเนิดขึ้นมา โลกก็มีการเปลี่ยนแปลงไปมากmany โดยเฉพาะเรื่องการติดต่อสื่อสารที่รวดเร็วขึ้น และการพัฒนาของภาษา หรือซอฟต์แวร์ที่เกี่ยวข้องกับอินเทอร์เน็ต ก็มีการเติบโตขึ้นมาก เช่นกัน ตลอดจนมีการแข่งขันพัฒนาซอฟต์แวร์พร้อมเผยแพร่บนโลกอินเทอร์เน็ตมากขึ้น [3]

ในอดีต ภาษา HTML (Hypertext markup language) เป็นภาษายอดนิยมที่ใช้ในการสร้างเว็บเพจ หรือเว็บไซต์ขึ้นมา ต่อมามีการพัฒนาภาษาในการสร้างเว็บไซต์เพิ่มขึ้นมากมาย ซึ่งภาษาหนึ่งที่ได้รับความนิยมมากตัวหนึ่ง คือ ภาษา PHP (Personal home page, ปัจจุบันคือ PHP Hypertext preprocessor) เนื่องจากภาษานี้มีความสามารถในการทำงานสูง ใช้งานได้ฟรี และมีตัวอย่างให้เลือกใช้มากมาย ซึ่งสิ่งนี้เป็นต้นกำเนิดของการทำเว็บไซต์รูปแบบใหม่ เรียกว่า ระบบการจัดการเนื้อหา (Content management system, CMS) [3]

CMS เป็นระบบที่นำมายังในการสร้างเว็บไซต์ และบริหารจัดการเร็วๆ จังหวะ โดยในการใช้งาน CMS ผู้ใช้งานไม่จำเป็นต้องมีความรู้ในการเขียนโปรแกรม สามารถเขียนเว็บไซต์ได้ด้วยตนเอง โดยที่ CMS มีโปรแกรมประยุกต์พร้อมใช้งานมากมาย เช่น ระบบการจัดการบุคลากรและข่าวสาร ระบบการจัดการห้องสมุด ระบบจัดการสมาชิก เป็นต้น [4]

CMS สามารถแบ่งออกเป็น 8 ประเภท ได้แก่

1. เว็บล็อก (Weblog) เป็น CMS สำหรับบันทึก หรือเขียนบันทึกเผยแพร่ส่วนบุคคล นิยมเรียกันว่า บล็อก (Blog) โดยแต่ละบันทึกจะเรียกว่า โพสต์ (Post) มากใช้สำหรับบรรยายเหตุการณ์ส่วนตัว หรือความรู้ที่ต้องการแบ่งปันให้กับผู้อื่น

2. พานิชย์อิเล็กทรอนิกส์ หรืออีคอมเมิร์ซ (E-commerce) เป็น CMS สำหรับทำร้านค้าออนไลน์ มีความสามารถในการซื้อขาย เพิ่มลดรายการสินค้า และราคา

3. อีเลิร์นนิ่ง (E-learning) เป็น CMS สำหรับทำสื่อการเรียนการสอน หรือบทเรียนคอมพิวเตอร์ช่วยสอน บนเว็บ ทำเป็นระบบออนไลน์ และสามารถสร้างแบบทดสอบได้

4. กระดานข่าว (Forum) เป็น CMS สำหรับถามตอบปัญหา หรือสร้างเป็นชุมชน (Community) โดยแบ่งเป็นหัวข้อต่าง ๆ ตามความสนใจของผู้เข้าชม ซึ่งส่วนใหญ่จะติดตั้งไปพร้อมกับ CMS ประเภทอื่น ๆ

5. โปรแกรมสนับสนุนการทำงานเป็นองค์กร (Groupware) เป็น CMS สำหรับการทำงานในองค์กร หรือหน่วยงานที่มีความสัมพันธ์ต่อเนื่องกัน มีความรวดเร็ว สามารถช่วยเหลือกันได้ ทำงานเป็นทีม และควบคุมการทำงานได้ ทำงานบนระบบเครือข่าย ซึ่งสามารถติดต่อสื่อสารเป็นทั้งแบบกลุ่ม หรือบุคคลก็ได้

6. คลังภาพ (Image galleries) เป็น CMS สำหรับจัดการคลังภาพ

7. เว็บท่า (Web portal) เป็น CMS เพื่อใช้เป็นหน้าหลักของเว็บไซต์ สามารถนำ CMS ตัวอื่นมาติดตั้งเพิ่มเติมได้

8. วิกิ (Wiki) เป็น CMS สำหรับให้ผู้ใช้เพิ่ม หรือแก้ไขเนื้อหาได้โดยง่าย เมื่อการเขียนบทความร่วมกันนำเสนอเนื้อหาสาระทางด้านสารานุกรม หรือแหล่งความรู้จำนวนมาก โดยรวมความเห็นจากหลาย ๆ คน [5]

2.2 หลักการทำงานของคอมไพล์เลอร์

คอมไпал์เลอร์เป็นโปรแกรมที่ใช้ในการแปลโค้ดที่เขียนด้วยภาษา源ต้นที่ต้องการ ไปเป็นภาษาเครื่อง ซึ่งสามารถแบ่งกระบวนการการอ่านเป็นสองช่วงหลัก ๆ ดังนี้

1) ช่วงวิเคราะห์ (Analysis) เป็นขั้นตอนส่วนหน้าของคอมไпал์เลอร์ที่สร้างรูปแบบระดับกลาง (Intermediate representation) จากตัวโค้ดที่ป้อนเข้ามา ข้อมูลจากโปรแกรมต้นทางจะถูกเก็บ และบันทึกในรูปแบบของตารางสัญลักษณ์ (Symbol table) โดยรูปแบบการวิเคราะห์จะมีอยู่ด้วยกัน 3 ลักษณะ ได้แก่

1.1) แบบตรง (Linear) เป็นการสแกนตามแน่นที่อ่านอักขระจากซ้ายไปขวา และจัดกลุ่มให้เป็นโทเคนที่สื่อความหมาย

1.2) แบบลำดับชั้น (Hierarchical) จะทำให้โทเคนถูกจัดหมวดหมู่ตามลำดับชั้นในกลุ่มช้อน

1.3) แบบประเมินความหมาย (Semantic) จะทำการประเมินความหมายของส่วนประกอบในโปรแกรมต้นทาง

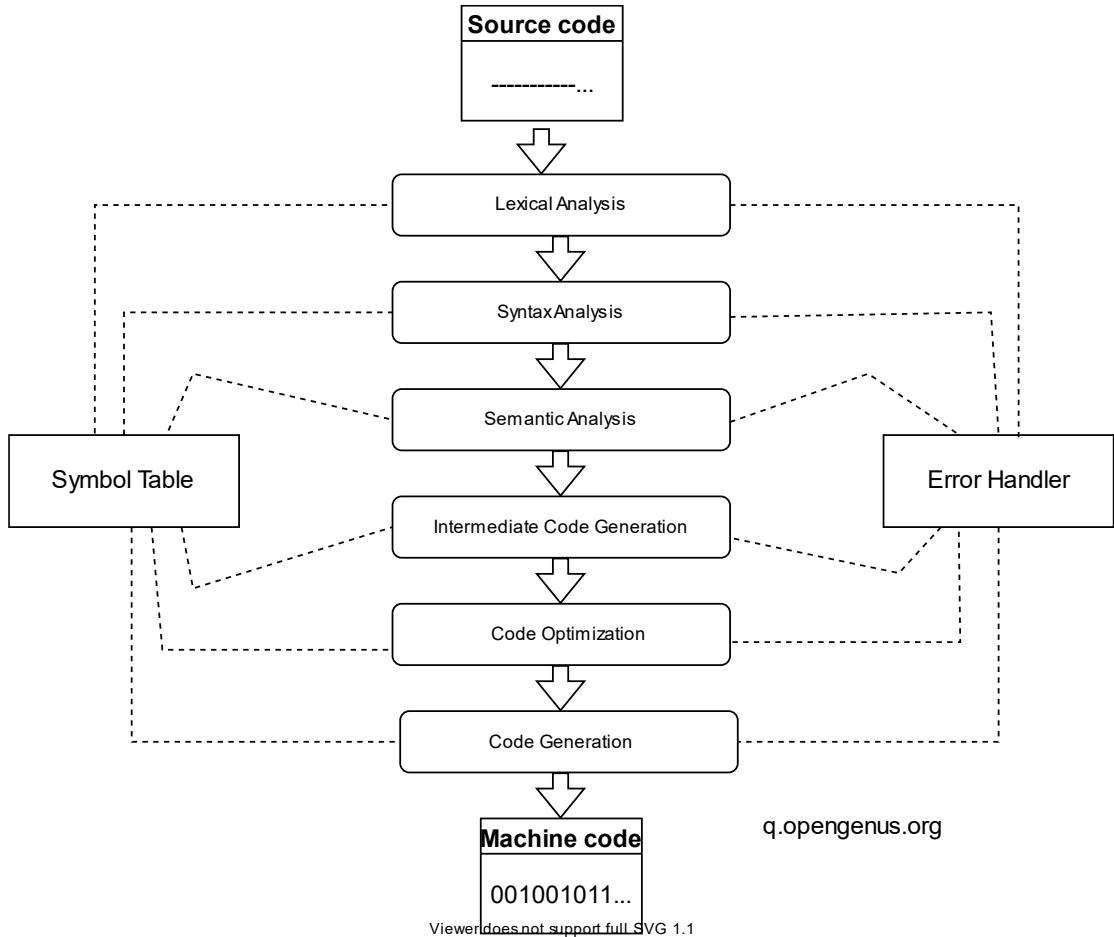
2) ช่วงสังเคราะห์ (Synthesis) เป็นขั้นตอนส่วนหลังของคอมไпал์เลอร์ซึ่งเป็นที่ที่โปรแกรมเป้าหมาย stemmed ถูกสร้างขึ้นจากโค้ดภาษา源ต้นที่ต้องการ

ศัพท์เฉพาะบางคำจะอธิบายต่อจากนี้

โทเคน (Token) หมายถึงลำดับของอักขระที่นำเสนอในรูปแบบของหน่วยศัพท์ที่ตรงกับรูปแบบของตัวดำเนินการ ตัวระบุ หรือคำส่วน

เล็กซิเม (Lexeme) หมายถึงลำดับของอักขระในโปรแกรมต้นทางที่ตรงกับรูปแบบของโทเคน หรือคือเป็น อินสแตนซ์ของโทเคน [6]

ในช่วงเหล่านี้ ทำให้สามารถแยกได้เป็นขั้นตอน 6 ขั้นตอนได้ ดังภาพที่ 1



ภาพที่ 1 ขั้นตอนในการทำงานของคอมไปเลอร์

2.2.1 การวิเคราะห์ศัพท์ (Lexical analysis) จะทำการสแกนโปรแกรมต้นทางจากช้ายไปขวา อักขระต่ออักขระ และจัดกลุ่มให้เป็นໂທເຄນ ກຮແສອກຂະ (Character stream) จะຖືກຈັດກລຸ່ມເປັນລຳດັບທີ່ສື່ອຄວາມໝາຍຕາມໂທເຄນທີ່ຮະບູ (Lexemes)

2.2.2 การวิเคราะห์ວາຍສົມພັນ (Syntax analysis) จะทำการແປງກຮແສຂອງໂທເຄນເປັນຕົ້ນໄໝການກຮຈາຍ (Parse tree) ໂດຍໂທເຄນທັງໝົດຈະຖືກຕຽບສອບຜ່ານໄວຍາກຮົນຂອງໂຄດຕັ້ນທາງວ່າມີຄວາມຖືກຕ້ອງ

2.2.3 การວິເຄາະໜ້າ (Semantic analysis) ຈະທຳນໍາທີ່ເປັນຕົວຈະກຳສອບຄວາມຖືກຕ້ອງຂອງຕົ້ນໄໝກຮຈາຍ ໂດຍຈະໄດ້ຜລັກພົບຈາກຂັ້ນຕອນນີ້ເປັນຕົ້ນໄໝໄວຍາກຮົນອີບາຍ (Annotated syntax tree) [6, 7]

2.2.4) การສ້າງໂຄດໃນພາຫະນະກັບກລາງ (Intermediate code generation) ຂັ້ນຕອນນີ້ຈະທຳການສ້າງໂຄດສາມສ່ວນ ທີ່ຄລ້າຍຄໍາສົ່ງແລ້ວສ່າມບລື່ມື້ມີຕົວຖືກດຳເນີນການ 3 ຕົວ ໂດຍແຕ່ລະຕົວຖືກດຳເນີນກາຈະທຳນໍາທີ່ເໝືອນ

รีจิสเตอร์ โค้ดนี้จะถูกเรียกว่าโค้ดระดับกลางซึ่งจะนำไปใช้แปลงเป็นโค้ดภาษาเครื่อง และขั้นตอนนี้ถือเป็นสะพานเชื่อมระหว่างการวิเคราะห์ และการสังเคราะห์

2.2.5) การเพิ่มประสิทธิภาพโค้ด (Code optimizer) จะทำการลดขนาดของโปรแกรมโดยลดจำนวนบรรทัดที่ไม่จำเป็นในโค้ดสามส่วน ดังนั้นโปรแกรมจะใช้ความจำน้อยที่สุดที่เท่าจะได้ และทำให้เวลาปฏิบัติการ (Execution time) ลดลง ซึ่งช่วยให้เพิ่มประสิทธิภาพโค้ดได้ ในขั้นตอนนี้จะยังไม่สูญเสียความหมายของโค้ด

2.2.6) การสร้างโค้ดในภาษาระดับเครื่อง (Code generator) ในขั้นตอนนี้จะสร้างโค้ดแอสเซมบลี โดยตัวแปรแต่ละตัวจะไปจ่อพื้นที่หน่วยความจำ [6, 7]

2.3 สถาปัตยกรรมระบบไมโครเซอร์วิส

ไมโครเซอร์วิสเป็นรูปแบบสถาปัตยกรรมระบบที่มีการแบ่งระบบทั้งหมดออกเป็นเซอร์วิส (Service) ย่อยโดยแต่ละเซอร์วิสจะเป็นอิสระต่อกันอย่างเด็ดขาด หากจะต้องติดต่อกับเซอร์วิสอื่น ก็จะต้องติดต่อผ่านตัวกลางที่เรียกว่า API (Application interface) หากจะส่งมอบ (Deploy) ก็จะส่งมอบแยกจากกัน [8]

สถาปัตยกรรมไมโครเซอร์วิสนิยมนำมาเปรียบเทียบกับสถาปัตยกรรมดั้งเดิม (Monolithic) ซึ่งเป็นรูปแบบที่ทุกรอบวนการจะทำงานอยู่ที่เดียวกัน หากระบบมีขนาดใหญ่ขึ้น การใช้สถาปัตยกรรมแบบดั้งเดิม จะทำให้เกิดความซับซ้อนสูงขึ้น ซึ่งยากต่อการพัฒนาต่อหากไม่มีการจัดการที่ดี และใช้เวลาในการส่งมอบนาน หลังจากการพัฒนา เพราะจะต้องส่งมอบทั้งระบบ ถึงแม้จะต้องแก้ไขเพียงบางระบบย่อยเท่านั้น [8]

เนื่องจากไมโครเซอร์วิสเป็นรูปแบบสถาปัตยกรรมระบบที่เพิ่งเกิดขึ้นมาใหม่ จึงยังไม่มีรูปแบบที่ชัดเจน ซึ่งในที่นี้ จะขอยกตัวอย่างรูปแบบที่ผู้เชี่ยวชาญด้านการออกแบบซอฟต์แวร์ อย่าง Martin Fowler และ James Lewis อธิบายไว้ ดังนี้

1) แบ่งองค์ประกอบผ่านเซอร์วิส (Componentization via services)

ในไมโครเซอร์วิส ซอฟต์แวร์จะถูกแบ่งออกเป็นเซอร์วิสเล็ก ๆ หลายเซอร์วิสที่เป็นอิสระจากกันโดยจะติดต่อกันผ่านทาง API ทำให้แต่ละเซอร์วิสทำหน้าที่เฉพาะงานของตัวเอง และเนื่องจากเซอร์วิสเป็นอิสระจากกัน ทำให้สามารถนำเข้า หรือถอนออกจากระบบได้ง่าย และด้วยการแบ่งย่อยดังกล่าว ยังทำให้ผู้พัฒนามีอิสระในการออกแบบแต่ละฟีเจอร์ของระบบ โดยไม่จำเป็นต้องทราบว่าระบบหรือ เซอร์วิสอื่น ๆ จะทำงานอย่างไร นั่นคือเป็นแบบกล่องดำ (Black box)

การลดภาระผู้พัฒนาช่วยเพิ่มความคล่องตัวในการทำงาน ด้วยการลดความต้องการการพัฒนาจากเซอร์วิสหนึ่ง ด้วยการรับการพัฒนาจากอีกเซอร์วิสหนึ่งให้เสร็จก่อน เมื่อมีการใช้คอนเทนเนอร์ใน การพัฒนาซอฟต์แวร์ อิมเมจ (Image) ในคอนเทนเนอร์สามารถสลับกับอิมเมจอื่นได้ อิมเมจที่ได้อาจเป็นเวอร์ชันที่แตกต่างกันจากอิมเมจเดิม หรือแตกต่างจากอิมเมจเดิมอย่างสิ้นเชิงก็ได้ ทราบที่ขอบเขตและหน้าที่การทำงานยังอยู่

ปัจจัยสำคัญที่มีบทบาทในการแบ่งองค์ประกอบผ่านเซอร์วิส ได้แก่

การขึ้นต่อ กัน (Dependencies) – มีการขึ้นต่อ กันอยู่เพียงภายในเซอร์วิสเอง และเป็น อิสระจากเซอร์วิสอื่น

ความสามารถในการถอด (Disposability) – สามารถถอดออกจากคอนเทนเนอร์ได้อย่าง ง่าย (ขึ้นอยู่กับการใช้ประโยชน์ของคอนเทนเนอร์) และกำจัดได้เมื่อหยุดทำงาน โดยไม่กระทบกับ ระบบส่วนอื่น

การทำงานพร้อมกัน (Concurrency) – การทำงานพร้อมกันประกอบด้วยงาน หรือ พอด (Pod) ซึ่งสร้างจากคอนเทนเนอร์ที่ทำงานร่วมกัน ที่สามารถปรับขนาดได้อย่างเป็นอัตโนมัติ โดยมี การจัดการประสิทธิภาพหน่วยความจำ [5, 8]

2) จัดระเบียบงานโดยคำนึงถึงความสามารถทางธุรกิจ (Organized around business capabilities)

การกำหนดว่า อะไรคือไมโครเซอร์วิส เป็นสิ่งสำคัญมากสำหรับทีมพัฒนาที่จะต้องตกลงกัน ขอบเขตคืออะไร และพลิกเคชันเป็นไมโครเซอร์วิสหรือไม่ หรือส่วนประกอบที่ใช้ร่วมกันถือเป็นไมโคร เซอร์วิสหรือไม่ ก่อนหน้าการมาของไมโครเซอร์วิส สถาปัตยกรรมของระบบจะถูกจัดการผ่าน ความสามารถของเทคโนโลยี เช่น ส่วนติดต่อ กับผู้ใช้ ฐานข้อมูล และตระกรากของตัวเซิร์ฟเวอร์ ตาม นิยามพื้นฐานของไมโครเซอร์วิส ทีมพัฒนาแต่ละทีม จะทำหน้าที่เป็นเจ้าของเซอร์วิส และทำงานที่ เกี่ยวข้องกับเซอร์วิสของทีมตนเอง ตลอดจนถึงการทำงานกับฝ่ายลูกค้าด้วย เช่น ในทีมนั้น ๆ จะต้อง พัฒนา ส่งมอบ ให้บริการ และเก็บเสียงตอบรับ (Feedback) ระบบที่ตนเองเป็นเจ้าของ

ในองค์กรที่ขับเคลื่อนด้วยไมโครเซอร์วิส ทีมย่อยจะทำการสร้าง ส่งมอบ และจัดการระบบ อย่างเป็นอัตโนมัติ สิ่งนี้จะทำให้ทีมทำงานอย่างเป็นระบบเพื่อส่งมอบฟีเจอร์ที่ได้รับมอบหมาย การ รับผิดชอบต่องานที่ได้รับมอบหมาย และการรับผิดชอบต่อผลที่เกิดขึ้น จะช่วยส่งเสริมวัฒนธรรมการ เป็นเจ้าของ และช่วยพัฒนาองค์กรให้ถึงเป้าหมายที่วางไว้ และได้รับประสิทธิภาพที่ดียิ่งขึ้น

หากสถาปัตยกรรมระบบและความสามารถถูกจัดระเบียบเป็นหน้าที่ธุรกิจอยู่ ๆ ภาระผูกพัน ระหว่างเซอร์วิสของระบบจะลดลงมากขึ้น ทำให้แต่ละทีมเจ้าของเซอร์วิส สามารถทำงานได้เร็วขึ้น เนื่องด้วยความอิสระ สิ่งที่เคยซ้อนกัน ก็จะถอนกันไป มือิสระในการเลือกเครื่องมือพัฒนา เช่น ส่วน ติดต่อผู้ใช้ เชียนด้วย Javascript และ HTML ส่วนหลังเขียนด้วย Java การประมวลผลข้อมูลใช้ Python สิ่งนี้หมายถึงว่า แต่ละเซอร์วิสจะขับเคลื่อนผ่านการตัดสินใจจากการพัฒนาได้

ปัจจัยสำคัญที่มีบทบาทในการจัดระเบียบงานโดยคำนึงถึงความสามารถทางธุรกิจ ได้แก่

ใช้โค้ดเป็นตัวตั้ง (Codebase) – แต่ละเซอร์วิสเป็นเจ้าของโค้ดเบสในพื้นที่เก็บข้อมูลแยก ต่างหาก และวงจรการเปลี่ยนแปลงของโค้ด [5, 8]

ส่งมอบ (Build, release, run) – แต่ละเซอร์วิสมีแนวทางการส่งมอบ และความถี่ในการส่งมอบเป็นของตนเอง ทำให้ทีมพัฒนาส่งมอบเซอร์วิสด้วยความเร็วที่ต่างกันได้ ขึ้นอยู่กับความต้องการของผู้ใช้

กระบวนการ (Process) - แต่ละเซอร์วิสมีหน้าที่หนึ่งอย่าง และหน้าที่อื่นๆได้เป็นอย่างดี เซอร์วิสจะต้องถูกออกแบบเพื่อแก้ไขปัญหาที่เกิดขึ้นให้ที่สุดที่เป็นไปได้

กระบวนการของผู้ดูแล (Admin process) แต่ละเซอร์วิส จะต้องมีการดูแล และจัดการงานตามหน้าที่ที่ออกแบบไว้ [5, 8]

3) ผลิตภัณฑ์ไม่ใช่โครงการ (Products not projects)

เพื่อให้องค์กรมีการจัดการที่เรียบง่าย และสามารถเพิ่มประสิทธิภาพได้ องค์กรจะต้องเบรียบให้องค์ประกอบของซอฟต์แวร์เป็นผลิตภัณฑ์ ที่สามารถปรับปรุงช้า ๆ และมีการวิวัฒน์ ซึ่งตรงข้ามกับกลยุทธ์ของการเบรียบซอฟต์แวร์เป็นโครงการ ที่วางแผนกับซอฟต์แวร์นั้นหลังจากการ เมื่อเป็นไปโครงการ มันจึงเป็นไปได้ที่จะเบรียบให้แต่ละเซอร์วิสเป็นผลิตภัณฑ์ส่งมอบ

ประโยชน์สำคัญของการเบรียบซอฟต์แวร์เป็นผลิตภัณฑ์ คือการเพิ่มประสิทธิภาพผู้ใช้ เมื่อซอฟต์แวร์ได้ทำการปรับปรุงเรื่อย ๆ จะทำให้การแก้ไขในอนาคตดี และง่ายขึ้น ส่งผลต่อผู้ใช้ที่ได้ใช้งานด้วย

ปัจจัยสำคัญที่มีบทบาทในการเบรียบซอฟต์แวร์เป็นผลิตภัณฑ์ไม่ใช่โครงการ ได้แก่

ส่งมอบ (Build, release, run) – ผู้พัฒนาได้มีการทำ Development operation (DevOps)

ตั้งค่า (Config) – ผู้พัฒนาสร้างตัวจัดการการตั้งค่าที่ดีขึ้นให้ซอฟต์แวร์ ผ่านการประเมินจากผู้ใช้ที่ได้ปรับปรุงมา

ความเท่าเทียมของการพัฒนา และผลิตภัณฑ์ (Dev/Prod parity) - ซอฟต์แวร์ที่ถือเป็นผลิตภัณฑ์สามารถพัฒนาช้า ๆ เป็นชิ้นเล็ก ๆ ซึ่งใช้เวลาในการดำเนินการและปรับใช้น้อยกว่าโครงการที่ใช้เวลานาน ซึ่งช่วยให้การพัฒนาและการผลิตมีความเท่าเทียมกันมากขึ้น [5, 8]

4) สมาร์ทเอนด์พอยนต์ และดัมป์ไบเปิล (Smart endpoints and dump pipes)

เมื่อเปลี่ยนผ่านจากสถาปัตยกรรมดั้งเดิมมาเป็นไปโครงการเซอร์วิส สิ่งที่ต้องเข้าใจคือวิธีที่เซอร์วิสสื่อสารกัน ซึ่งมีรูปแบบหลัก ๆ ในการสื่อสารอยู่สองแบบดังนี้

Request/Response – เมื่อเซอร์วิสหนึ่งต้องการติดต่อกับอีกเซอร์วิส จะสร้างการร้องขอ (Request) เพื่อให้ข้อมูลเอาไปเก็บ หรือขอข้อมูลมาในรูปแบบการตอบสนอง (Response)

Publish/Subscribe – ถูกใช้เมื่อเซอร์วิสหนึ่งมีการกระตุนอีเวนต์ แล้วอีกเซอร์วิสจะต้องรับทราบ เพื่อทำงานบางอย่าง ตัวส่งอีเวนต์จะถูกเรียกว่า ผู้เผยแพร่ (Publisher) และตัวรับอีเวนต์จะเรียกว่า ผู้ติดตาม (Subscriber)

ปัจจัยสำคัญที่มีบทบาทในการทำสมาร์ทเอนด์พอยนต์ และดัมป์ไปป์ ได้แก่

ผูกพอร์ต (Port binding) – เซอร์วิสจะผูกกับพอร์ตเพื่อรอการร้องขอ หรือส่งคำขอ ผ่านพอร์ตของอีเซอร์วิส ท่อส่งน้ำหนึ่งเป็นเพียงโปรโตคอลเครือข่าย เช่น HTTP

เซอร์วิสเบื้องหลัง (Backing service) – ท่อส่งจะอนุญาตให้เซอร์วิสเบื้องหลังแนบไป กับเซอร์วิสอื่น ในทางเดียวกับที่แนบฐานข้อมูล

การทำงานพร้อมกัน (Concurrency) – การออกแบบแนวทางการสื่อสารระหว่าง เซอร์วิสจะช่วยให้เซอร์วิสทำงานพร้อมกันได้ [5, 8]

5) การปกครองแบบกระจายอำนาจ (Decentralized governance)

เมื่องคุณมีการใช้กระบวนการที่ขับเคลื่อนด้วยโคด (Code-driven) หนึ่งความท้าทายที่ต้อง เมนูชิญ คือการปรับขนาดของทีมพัฒนา และให้ทำงานแบบขนาน การใช้ชีวิธีแบบกระจายอำนาจ เป็น สิ่งหนึ่งที่ดีสำหรับสถาปัตยกรรมไมโครเซอร์วิส ที่ใช้ต่อกรับความท้าทายนี้ เมื่อทีมเริ่มต้นโครงการ ด้วยจำนวนคนเล็กน้อยมาทำงานด้วยกัน หลังจากผ่านช่วง Greenfield ตัวธุรกิจจะเร่งค้นหาโอกาส เพื่อขยายการส่งมอบเวอร์ชันแรก เสียงตอบรับจากลูกค้าจะสร้างไอเดียให้เกิดฟีเจอร์ใหม่ ๆ เพื่อใช้ใน การปรับปรุงตัวเดิมที่มี ในช่วงนี้ ผู้พัฒนาจะเริ่มทำการลงมือพัฒนา และองค์กรจะมีการแบ่งทีมให้ ทำงานในแต่ละเซอร์วิส

การปกครองแบบกระจายอำนาจ ในที่นี้ หมายถึงว่า แต่ละทีมสามารถตัดสินใจที่จะใช้เครื่องมือ สำหรับแก้ปัญหา การบังคับให้ทุกทีมใช้ฐานข้อมูลเดียวกัน หรือการใช้ภาษาโปรแกรมเดียวกันนั้น เป็นสิ่งที่ไม่สมเหตุสมผล แต่การปกครองแบบกระจายอำนาจใช้ว่าจะไม่มีขอบเขต การกำหนด มาตรฐานองค์กรก็ยังเป็นสิ่งสำคัญในการสร้างและพัฒนาซอฟต์แวร์ เพียงแค่ต้องเลือกให้เหมาะสม กับความต้องการองค์กร

ปัจจัยสำคัญที่มีบทบาทในการการปกครองแบบกระจายอำนาจ ได้แก่

การขึ้นต่อ กัน (Dependencies) – มีการให้ทีมเลือก stack ของตัวเองที่เหมาะสมกับ งานที่ได้รับมอบหมาย

การส่งมอบ (Build, Release, Run) – มีการให้ทีมเลือกกระบวนการสร้างเพื่อเลือก เครื่องมือ

เซอร์วิสเบื้องหลัง (Backing service) – ถ้าทรัพยากรที่ถูกใช้งาน เปรียบกับเซอร์วิส แบบบุคคลที่สาม (Third party) และการกระจายอำนาจ จะช่วยให้ทรัพยากรสามารถถูกใช้ในทางที่ ต่างกันได้ ทราบเท่าที่ยังมีติดต่อกับเซอร์วิสอื่น [8, 5]

6) การจัดการข้อมูลแบบกระจายอำนาจ (Decentralized data management)

สถาปัตยกรรมระบบแบบดั้งเดิม นักจะใช้ฐานข้อมูลแบบแบ่งปัน (Shared database) ที่เป็นที่ เก็บข้อมูลเพียงที่เดียวในระบบทั้งหมด สิ่งนี้นำไปสู่ความซับซ้อนในการเปลี่ยนแปลงรูปแบบข้อมูล

การอัปเกรด การล่มของระบบ และการจัดการกับความเสี่ยง แต่สำหรับไมโครเซอร์วิส จะมีการใช้ฐานข้อมูลเป็นของตนเอง ไม่มีการแบ่งปันฐานข้อมูลกับระบบอื่น ๆ ซึ่งเป็นการเพิ่มประสิทธิภาพของการออกแบบระบบด้วยการเก็บข้อมูลที่ดีที่สุดของงานที่จะใช้มัน และสามารถล้างงานที่ยกลำบากในการใช้ฐานข้อมูลแบบแบ่งปันด้วย

ปัจจัยสำคัญที่มีบทบาทในการการปกคล้องแบบกระจายอำนาจ ได้แก่

ความสามารถในการถอด (Disposability) – เชอร์วิสควรจะทนทาน และเป็นอิสระจากภายนอก หลักการนี้จะช่วยให้เชอร์วิสทำงานไปได้ด้วยความสามารถที่จำกัด ได้ถ้าไม่องค์ประกอบใดองค์ประกอบหนึ่งล้ม

เชอร์วิสเบื้องหลัง (Backing services) – เชอร์วิสเบื้องหลังเป็นเชอร์วิสที่ใช้ทรัพยากรบนเครือข่าย โดยทั่วไป เชอร์วิสเบื้องหลังจะถูกจัดการโดยการดำเนินการ ระบบไม่ควรสร้างความแตกต่างระหว่างเชอร์วิสภายใน และภายนอก [8, 5]

7) ระบบอัตโนมัติของโครงสร้างระบบ (Infrastructure automation)

ประโยชน์อย่างหนึ่งในการทำให้โครงสร้างระบบเป็นโค้ด คือเมื่อมองไปยังฐานข้อมูล โหลดบาลานเซอร์ หรือเครือข่าย ที่เขียนขึ้นในโค้ด และสร้างอินสแตนซ์ขึ้นมา เมื่อเทียบกับภาพวาดของสถานปิน ก็คือพิมพ์เขียวที่มีกำแพง หน้าต่าง ประตู ที่จะเปลี่ยนแปลงไปเป็นอาคารนั่นเอง

ปัจจัยสำคัญที่มีบทบาทในการการสร้างระบบอัตโนมัติ ได้แก่

โค้ดเบส (Codebase) – เพราะโครงสร้างระบบถูกอธิบายในรูปแบบโค้ด ก็เทียบโค้ดเหล่านั้น และเก็บไว้ในเชอร์วิส

การตั้งค่า (Configuration) – สภาพแวดล้อมควรจะมีลักษณะเฉพาะของมัน

การส่งมอบ (Build, Release, Run) – หนึ่งสภาพแวดล้อมหนึ่งจุดประสงค์

ความสามารถในการถอด (Disposability) – ปัจจัยนี้จะอยู่เหนือกระบวนการ และลงไปที่ดาวน์สตรีม (Downstream) เช่น คอนเทนเนอร์ เครื่องเสมือน และคลาวด์ส่วนตัว

ความเท่าเทียมของการพัฒนา และการผลิต (Dev/Prod parity) - รักษาการพัฒนาการจัดการเวอร์ชัน มองเป็นการผลิตเท่าที่เป็นไปได้ [8, 5]

8) การออกแบบเพื่อความล้มเหลว (Design for failure)

การออกแบบเพื่อความล้มเหลว หมายถึง การทดสอบสิ่งที่ออกแบบและทำให้ครอบคลุมทุกเงื่อนไขที่เกิดขึ้นได้ ไม่ใช่ทุกเทคโนโลยีที่สามารถทำตามหลักการนี้ แต่อย่างให้มีป้อย ๆ

การออกแบบเพื่อความล้มเหลวทำให้มicroservices ที่สามารถซ่อมแซมตัวเองได้ซึ่งทำหน้าที่ได้เต็มที่ตามที่คาดหวังจากปริมาณงานล่าสุด การป้องกันการโทรศัพท์รับประกันความพึงพอใจระดับพื้นฐานสำหรับทีมที่เป็นเจ้าของบริการ นอกจากนี้ยังช่วยจัดระดับความเครียดที่อาจถูกนำไปเป็น

การออกแบบระบบเร่งดีด้วย การออกแบบเพื่อรับความล้มเหลวจะช่วยให้ผลิตภัณฑ์มีเวลาทำงานมากขึ้น สามารถป้องบีริษัทจากการหยุดการทำงานที่อาจก่อกร่อนความไว้วางใจของลูกค้า

ปัจจัยสำคัญที่มีบทบาทในการออกแบบเพื่อความล้มเหลว ได้แก่

ความสามารถในการถอด (Disposability) – ผลิตภัณฑ์ที่สามารถถอดออกได้โดยไม่ต้องรีสตาร์ท หรือหยุดได้ในเวลาแบบทันท่วงที

บันทึก (Logs) – ถ้าส่วนหนึ่งของระบบเกิดความล้มเหลว การแก้ไขปัญหาเป็นสิ่งที่จำเป็นโดยต้องตรวจสอบให้แน่ใจว่าสามารถแก้ไขอย่างไรก็ได้

ความเท่าเทียมของการพัฒนา และการผลิต (Dev/Prod parity) - ความเท่าเทียมของการพัฒนา และการผลิต (Dev/Prod parity) - รักษาการพัฒนา การจัดการเรอร์ชัน มองเป็นการผลิตเท่าที่เป็นไปได้ [8, 5]

9) การออกแบบเพื่อการพัฒนาในอนาคต (Evolutionary design)

ในการออกแบบสถาปัตยกรรมระบบสมัยใหม่ จำเป็นต้องตระหนักรู้ถึงความต้องการไม่มีที่สิ้นสุด เชอร์วิสจะต้องมีการปรับปรุงซ้ำ ๆ และเชอร์วิสจะต้องใช้การเรียนรู้จากการใช้งานจริงที่ช่วยให้ความสามารถเกิดวิวัฒนา

ปัจจัยสำคัญที่มีบทบาทในการออกแบบเพื่อการพัฒนาในอนาคต ได้แก่

โค๊ดเบส (Codebase) – ช่วยให้พัฒนาความสามารถได้เร็วขึ้น เมื่อมีเสียงตอบรับกลับมา

การขึ้นต่อ กัน (Dependencies) – มีการออกแบบซ้ำ ๆ แบบเร็ว ๆ เมื่อมีความสามารถใหม่ผุดขึ้นมา

การตั้งค่า (Configuration) - ทุกสิ่งที่มีแนวโน้มที่จะแตกต่างกันระหว่างการปรับใช้งาน (การจัดเตรียม การผลิต สภาพแวดล้อมของนักพัฒนา ฯลฯ) การกำหนดค่าจะแตกต่างกันอย่างมากในการปรับใช้ แต่โค๊ดจะไม่แตกต่างกัน ด้วยการกำหนดค่าที่จัดเก็บไว้ภายนอกโค๊ด การออกแบบสามารถพัฒนาได้โดยไม่คำนึงถึงสภาพแวดล้อม

การส่งมอบ (Build, Release, Run) – ช่วยเบิดตัวฟีเจอร์ใหม่ ๆ โดยใช้เทคนิคการปรับใช้งานต่างๆ แต่ละรุ่นมีรหัสเฉพาะและสามารถใช้เพื่อเพิ่มประสิทธิภาพการออกแบบและคุณภาพจากผู้ใช้ [8, 5]

บทที่ 3

วิธีการดำเนินงาน

3.1 เครื่องมือที่ใช้ในการพัฒนา

3.1.1) ASP.NET เป็นเฟรมเวิร์กสำหรับพัฒนาเว็บไซต์ในส่วนหลัง (Backend) โดยใช้ภาษา C# เป็นหลัก โดยในโครงงานจะใช้เวอร์ชันที่ 7.0 [9]

3.1.2) Ocelot เป็นเครื่องมือสำหรับสร้างเกตเวย์ (Gateway) เพื่อเข้าถึงแต่ละบริการ (Service) ที่อยู่ในสถาปัตยกรรมไมโครเซอร์วิส (Microservices) [10]

3.1.3) Identity framework เป็นเครื่องมือสำหรับจัดการระบบบัญชีผู้ใช้ และการให้สิทธิ์เข้าถึงของผู้ใช้ (Authorization) [11]

3.1.4) OpenIddict เป็นเครื่องมือสำหรับให้บริการในการสร้าง OpenID Connect (OIDC) และการตรวจสอบโทคেน [12]

3.1.5) MongoDB เป็นเครื่องมือที่ใช้ในการจัดการฐานข้อมูลเชิงไม่สัมพันธ์ (Non-relational database) [13]

3.1.6) Docker เป็นเครื่องมือที่ใช้สำหรับจัดการสภาพแวดล้อม (Environment) โดยบรรจุไว้ในหน่วย มาตรฐานที่เรียกว่า คอนเทนเนอร์ (Container) ซึ่งใช้บรรจุทุกสิ่งที่ซอฟต์แวร์นั้นต้องใช้งาน รวมถึงไลบรารี และ เครื่องมืออื่น ๆ ที่จำเป็นในซอฟต์แวร์ โดยในโครงงานจะนำไปใช้บรรจุทุกบริการของเว็บไซต์ เพื่อให้สะดวกต่อการ สร้าง ทดสอบ และติดตั้งเว็บไซต์ [14]

3.1.7) Angular เป็นเฟรมเวิร์กที่สำหรับพัฒนาเว็บไซต์ในส่วนหน้า (Frontend) โดยใช้ภาษา Typescript เป็นหลัก โดยในโครงงานจะใช้เวอร์ชัน 16 [15]

3.1.8) EditorJS เป็นเครื่องมือสำหรับเป็นตัวเขียนบทความหลักในเว็บไซต์ โดยมีความสามารถในการ นำเข้าและส่งออกข้อมูลเป็น JSON ทำให้สะดวกในการนำข้อมูลไปใช้งานต่อ ซึ่งพัฒนาโดยภาษา Javascript [16]

3.1.9) Monaco editor เป็นเครื่องมือสำหรับเป็นตัวเขียนโค้ดของเว็บไซต์ โดยมีความสามารถในการ ตรวจสอบภาษาสัมพันธ์ของแต่ละภาษาโปรแกรมได้ [17]

3.1.10) Xterm.js เป็นเครื่องมือสำหรับเป็นส่วนติดต่อผู้ใช้ในการแสดงผลผลลัพธ์ของการรันโค้ด

3.2 ขั้นตอนการดำเนินงาน

3.2.1) ออกแบบภาพรวมของระบบล็อกที่จะพัฒนา ซึ่งจะต้องออกแบบโดยคำนึงถึงความสามารถ หรือ หน้าที่ของแต่ละระบบอย่าง ว่ามีส่วนใดที่สามารถแยกจากกันได้หรือไม่ หากสามารถแยกจากกันได้ จะให้หน้าที่อะไร และจะมีวิธีการ หรือกฎเกณฑ์ในการติดต่อกันระหว่างระบบย่อยได้อย่างไร ซึ่งแต่ละระบบย่อย จะต้องออกแบบให้เป็นบริการย่อย ตามหลักการเบื้องต้นของสถาปัตยกรรมไมโครเซอร์วิส

3.2.2) ออกแบบการติดต่อกับบริการย่อยแต่ละบริการตามขอบเขตของระบบ ตามที่ได้ออกแบบไว้ใน ขั้นตอนที่ 3.2.1) โดยคำนึงถึงหน้าที่ที่ต้องรับมือหมายให้ทำตามภาพรวมระบบ

3.2.3) พัฒนาบริการในส่วนที่เกี่ยวข้องกับการจัดการเนื้อหาทั้งในส่วนหน้าและส่วนหลังของระบบ ไม่ว่า จะเป็นในส่วนที่เกี่ยวข้องกับผู้ใช้ เนื้อหา หรือระบบอื่น ๆ ที่เกี่ยวข้องการจัดการเนื้อหา

3.2.4) พัฒนาบริการในส่วนที่เกี่ยวข้องกับการคอมไพล์โค้ดทั้งในส่วนหน้าและส่วนหลังของระบบ และ ออกแบบข้อตกลง และกฎเกณฑ์ในการคอมไпал์โค้ด ว่าผู้ใช้สามารถทำอะไรได้บ้าง และทำอะไรไม่ได้บ้าง เกี่ยวกับ การคอมไпал์โค้ด

3.2.5) ปรับปรุงความมั่นคงในการคอมไпал์โค้ด เพื่อเพิ่มความมั่นคงของระบบ โดยป้องกันไม่ให้ผู้ใช้ที่ไม่ ประสงค์ดีกระทำการที่เป็นอันตรายต่อระบบผ่านทางการคอมไпал์โค้ดด้วยวิธีนี้

3.2.6) ทดสอบระบบกับกลุ่มตัวอย่าง เพื่อหาจุดอ่อนของระบบ หรือการตอบกลับจากการทดสอบระบบ เพื่อนำมา ปรับปรุงระบบให้ได้ตามเป้าหมายหลังจากการทดสอบระบบ

3.3 แผนการดำเนินงาน

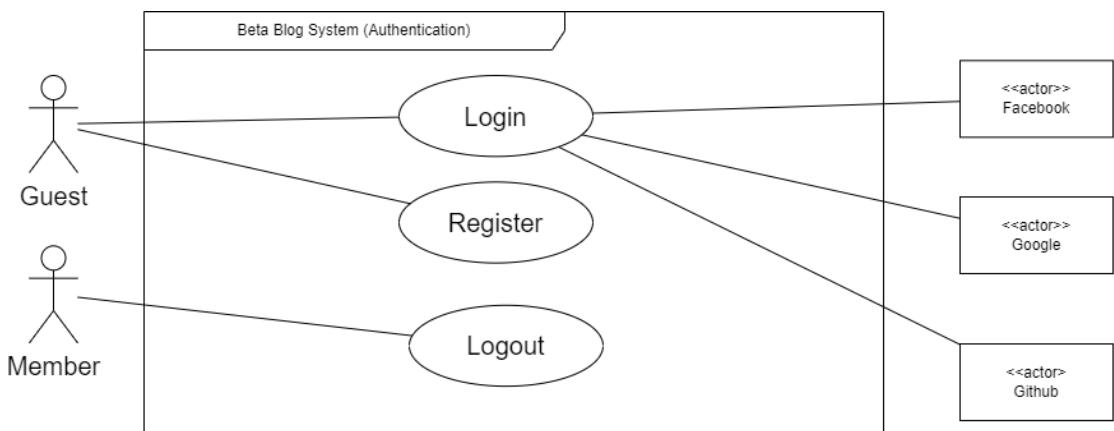
ตารางที่ 1 ตารางแสดงแผนการดำเนินงานโครงการ

บทที่ 4

ผลการดำเนินงาน

4.1 ภาพรวมของระบบ

4.1.1 การยืนยันตัวตนของผู้ใช้



ภาพที่ 2 แผนภาพกรณีการใช้งาน (Use case diagram) สำหรับการยืนยันตัวตนผู้ใช้ของระบบเบต้าบล็อก

จากรายละเอียดส่วนยืนยันตัวตนของระบบเบต้าบล็อกที่แสดงในภาพที่ 2 แสดงถึงแต่ละกรณีการใช้งาน ซึ่งอธิบายดังตารางต่อไปนี้

ตารางที่ 2 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการเข้าสู่ระบบ (Login)

Use case name	Login
Actor	ผู้ใช้ที่ยังไม่เป็นสมาชิก (Guest)
Precondition	สมัครสมาชิก Beta blog หรือ Facebook หรือ GitHub หรือ Google และ
Stakeholder Actor	ระบบ OAuth ของ Facebook, GitHub และ Google
Main flow	ผู้ใช้เข้าสู่หน้าลงชื่อเข้าใช้

ตารางที่ 2 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการเข้าสู่ระบบ (ต่อ)

Alternate flow 1	ผู้ใช้กรอกชื่อผู้ใช้และรหัสผ่านเพื่อทำการเข้าสู่ระบบด้วยบัญชีผู้ใช้ของระบบ Beta blog จากนั้นระบบจะนำทางให้ผู้ใช้สู่หน้าแรกของ Beta blog
Alternate flow 2	ผู้ใช้เลือกลงชื่อเข้าใช้ด้วยผู้ให้บริการภายนอกอย่าง Facebook, GitHub หรือ Google จากนั้นระบบจะนำทางผู้ใช้ให้เข้าสู่หน้าลงชื่อเข้าใช้ของระบบดังกล่าว เมื่อเสร็จแล้ว ระบบจะนำทางให้ผู้ใช้เข้าสู่หน้าแรกของ Beta blog
Exception flow	หากชื่อผู้ใช้หรือรหัสผ่านที่ผู้ใช้ลงชื่อเข้าใช้ไม่ถูกต้อง ระบบจะแสดงข้อความ “ชื่อผู้ใช้ หรือรหัสผ่านไม่ถูกต้อง” และให้ลังชื่อเข้าใช้ใหม่อีกครั้ง

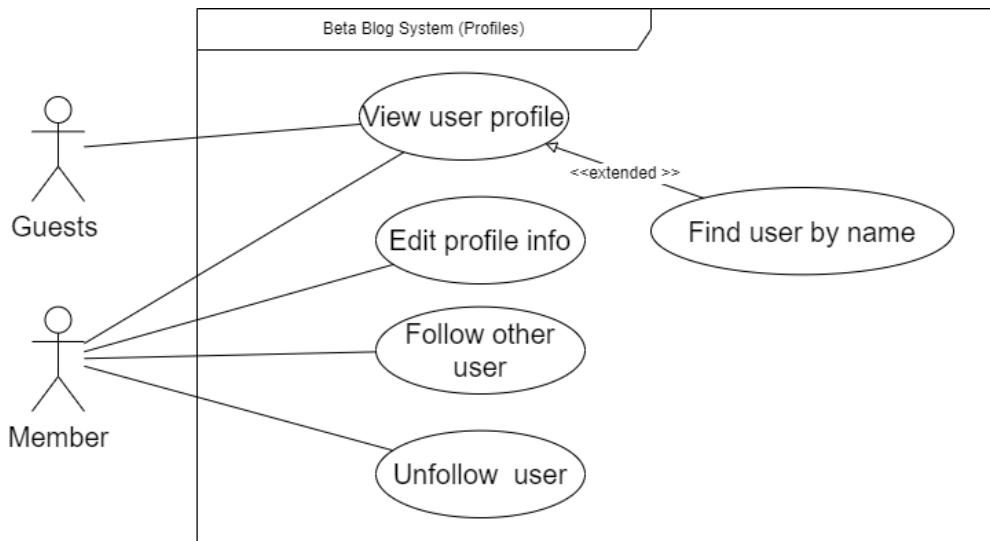
ตารางที่ 3 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการลงทะเบียน (Register)

Use case name	Register
Actor	ผู้ใช้ที่ยังไม่เป็นสมาชิก (Guest)
Main flow	ผู้ใช้เข้าสู่หน้าลงทะเบียนเพื่อกรอกชื่อผู้ใช้และรหัสผ่าน จากนั้นกดลงทะเบียน และกลับสู่หน้าเข้าสู่ระบบเมื่อทำการลงทะเบียนสำเร็จ
Exception flow	<ol style="list-style-type: none"> กรณีชื่อผู้ใช้ซ้ำ ระบบจะแสดงข้อความเพื่อให้ผู้ใช้กรอกชื่อผู้ใช้ใหม่อีกครั้ง กรณีรหัสผ่านน้อยกว่า 6 ตัวระบบจะแสดงข้อความให้ผู้ใช้กรอกรหัสมากกว่า 5 ตัว กรณีรหัสไม่มีตัวอักษรพิมพ์ใหญ่และพิมพ์เล็กผสมกัน ระบบจะแสดงข้อความให้ผู้ใช้กรอกรหัสผ่านที่มีทั้งตัวพิมพ์ใหญ่และพิมพ์เล็กผสมกัน กรณีรหัสผ่านไม่มีตัวเลขและอักษรพิเศษ ระบบจะแสดงข้อความให้ผู้ใช้กรอกรหัสผ่านที่ประกอบด้วยตัวเลขและอักษรพิเศษ กรณีผู้ใช้ยืนยันรหัสผ่านไม่ตรงกัน ระบบจะแสดงให้ผู้ใช้ยืนยันด้วยรหัสผ่านที่เหมือนกัน

ตารางที่ 4 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการออกจากระบบ (Logout)

Use case name	Logout
Actor	ผู้ใช้ที่เป็นสมาชิก (Member)
Precondition	ผู้ใช้เข้าสู่ระบบแล้ว
Main flow	ผู้ใช้ทำการกดปุ่มออกจากระบบ จากนั้นระบบจะนำทางให้ผู้ใช้ไปที่หน้าแรก
Exception flow	-

4.1.2 การจัดการข้อมูลผู้ใช้



ภาพที่ 3 แผนภาพกรณีการใช้งานสำหรับการจัดการข้อมูลผู้ใช้ของระบบเบต้าบล็อก

จากรายละเอียดส่วนการจัดการข้อมูลผู้ใช้ของระบบเบต้าบล็อกที่แสดงในภาพที่ 3 แสดงถึงแต่ละกรณีการใช้งาน ซึ่งอธิบายดังตารางต่อไปนี้

ตารางที่ 5 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการดูโปรไฟล์ผู้ใช้ (View user profile)

Use case name	View user profile
Actor	ผู้ใช้ที่ยังไม่เป็นสมาชิก (Guest) และ ผู้ใช้ที่เป็นสมาชิก (Member)
Main flow	ผู้ใช้กดเลือกชื่อผู้ใช้หรือภาพโปรไฟล์เพื่อไปสู่หน้าแสดงรายละเอียดของผู้ใช้คนดังกล่าว
Exception flow	-

ตารางที่ 6 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการค้นหาผู้ใช้ด้วยชื่อ (Find user by name)

Use case name	Find user by name
Actor	ผู้ใช้ที่ยังไม่เป็นสมาชิก (Guest) และ ผู้ใช้ที่เป็นสมาชิก (Member)
Main flow	ผู้ใช้ทำการค้นหาโดยระบุชื่อผู้ใช้ที่ช่องค้นหา จากนั้นระบบจะแสดงรายการผู้ใช้ที่ใกล้เคียงกัน
Exception flow	หากไม่มีชื่อผู้ใช้ที่ใกล้เคียงระบบจะแสดงข้อความ “ไม่พบชื่อผู้ใช้”

ตารางที่ 7 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการแก้ไขโปรไฟล์ (Edit profile information)

Use case name	Edit profile information
Actor	ผู้ใช้ที่เป็นสมาชิก (Member)
Precondition	ผู้ใช้เข้าสู่ระบบแล้ว
Main flow	ผู้ใช้เข้าสู่หน้าแก้ไขข้อมูลส่วนตัว จากนั้นสามารถแก้ไขชื่อที่แสดง ข้อความแนะนำตัวแบบสั้น และภาพโปรไฟล์ หลังจากผู้ใช้ทำการบันทึกการเปลี่ยนแปลงจะถูกนำทางไปยังหน้าโปรไฟล์ของผู้ใช้เอง
Exception flow	-

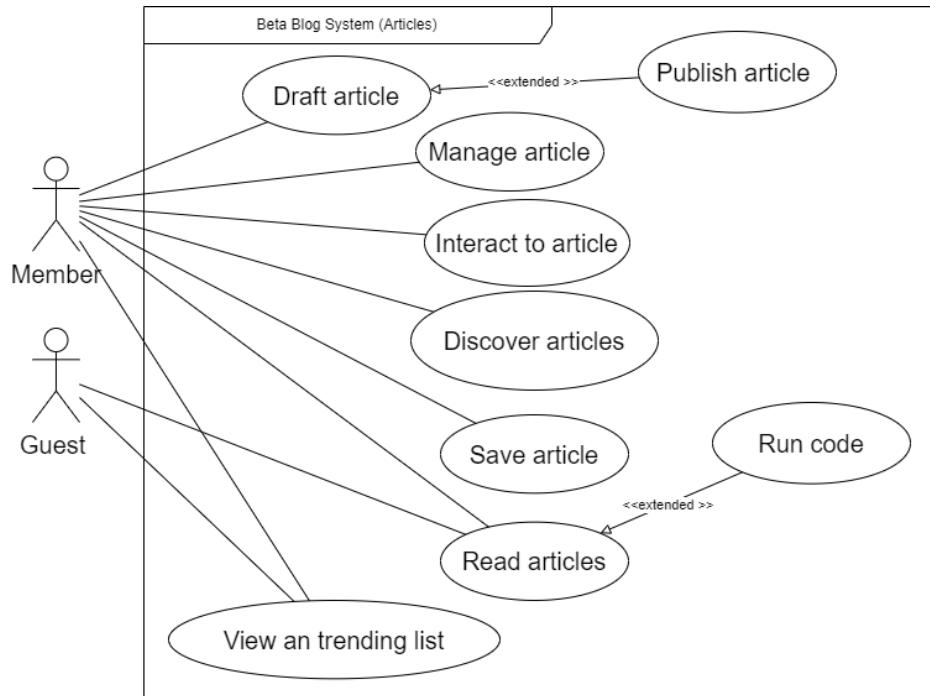
ตารางที่ 8 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการติดตามผู้ใช้ (Follow another user)

Use case name	Follow another user
Actor	ผู้ใช้ที่เป็นสมาชิก (Member)
Precondition	ผู้ใช้เข้าสู่ระบบแล้วและยังไม่ได้ติดตามผู้ใช้ดังกล่าว
Main flow	ผู้ใช้สามารถติดตามผู้ใช้อื่นได้ที่หน้าໂປຣໄຟລ໌ หรือที่หน้าบุทความของผู้ใช้ดังกล่าว
Exception flow	-

ตารางที่ 9 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการยกเลิกติดตามผู้ใช้ (Unfollow user)

Use case name	Unfollow user
Actor	ผู้ใช้ที่เป็นสมาชิก (Member)
Precondition	ผู้ใช้เข้าสู่ระบบและติดตามผู้ใช้ดังกล่าวแล้ว
Main flow	ผู้ใช้กดยกติดตามผู้ใช้ที่หน้าໂປຣໄຟລ໌ หรือที่หน้าบุทความของผู้ใช้ดังกล่าว และกดปุ่มยืนยัน การยกติดตามอีกครั้งเพื่อบันทึกการเปลี่ยนแปลง
Exception flow	-

4.1.3 การจัดการบทความ



ภาพที่ 4 แผนภาพกรณีการใช้งานสำหรับการจัดการบทความของระบบเบต้าบล็อก

จากรายละเอียดส่วนการจัดการข้อมูลผู้ใช้ของระบบเบต้าบล็อกที่แสดงในภาพที่ 4 แสดงถึงแต่ละกรณีการใช้งาน ซึ่งอธิบายดังตารางต่อไปนี้

ตารางที่ 10 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการร่างบทความ (Draft article)

Use case name	Draft article
Actor	ผู้ใช้ที่เป็นสมาชิก (Member)
Precondition	ผู้ใช้เข้าสู่ระบบแล้ว
Main flow	ผู้ใช้กดปุ่มเพิ่มบทความ จากนั้นผู้ใช้จะสามารถเขียนเนื้อหาและบันทึกเป็นฉบับร่างได้
Alternate flow	ผู้ใช้กดปุ่มเพิ่มบทความ จากนั้นผู้ใช้จะสามารถเขียนเนื้อหาและเผยแพร่เป็นสาธารณะได้หลังจากเผยแพร่ ผู้ใช้จะเข้าสู่หน้าบทความสาธารณะของบทความดังกล่าว

ตารางที่ 11 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการจัดการบทความ (Manage article)

Use case name	Manage article
Actor	ผู้ใช้ที่เป็นสมาชิก (Member)
Precondition	ผู้ใช้เข้าสู่ระบบแล้ว
Main flow	ผู้ใช้เข้าสู่หน้าจัดการบทความเพื่อ
Alternate flow 1	ผู้ใช้สามารถเปลี่ยนสถานะการเผยแพร่ได้ ระหว่างสถานะเผยแพร่และฉบับร่าง หลังจากทำการบันทึกการเปลี่ยนแปลงผู้ใช้จะเข้าสู่หน้าบทความดังกล่าว
Alternate flow 2	ผู้ใช้สามารถแก้ไขเนื้อหาบทความได้ หลังจากทำการบันทึกการเปลี่ยนแปลงผู้ใช้จะเข้าสู่หน้าบทความดังกล่าว
Alternate flow 3	ผู้ใช้สามารถลบบทความได้ หลังจากทำการบันทึกการเปลี่ยนแปลงผู้ใช้จะเข้าสู่หน้าก่อนหน้า
Exception flow	-

ตารางที่ 12 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการค้นดูบทความ (Discover articles)

Use case name	Discover articles
Actor	ผู้ใช้ที่ยังไม่เป็นสมาชิก (Guest) และ ผู้ใช้ที่เป็นสมาชิก (Member)
Precondition	-
Main flow	ผู้ใช้เข้าสู่หน้าแรกเพื่อดูรายการบทความทั้งหมดที่เรียงตามเวลาล่าสุด
Alternate flow 1	ผู้ใช้เลือกดูเฉพาะรายการบทความของผู้ใช้ที่กำลังติดตามอยู่
Alternate flow 2	ผู้ใช้เลือกดูเฉพาะรายการบทความในหมวดหมู่ต่าง ๆ
Alternate flow 3	ผู้ใช้เลือกดูเฉพาะรายการบทความที่ตรงกับคำค้นหาของผู้ใช้
Exception flow	หากรายการบทความที่ผู้ใช้เลือกแสดงไม่มีข้อมูลในระบบจะแสดงข้อความ “ไม่พบบทความที่ตรงกับความต้องการดังกล่าว”

ตารางที่ 13 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการอ่านบทความ (Read an article)

Use case name	Read an article
Actor	ผู้ใช้ที่ยังไม่เป็นสมาชิก (Guest) และ ผู้ใช้ที่เป็นสมาชิก (Member)
Precondition	-
Main flow	ผู้ใช้เลือกบทความจากหน้าต่าง ๆ เพื่อเข้าสู่หน้าเนื้อหาของบทความเพื่ออ่าน
Exception flow	-

ตารางที่ 14 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการรันโค้ด (Run code)

Use case name	Run code
Actor	ผู้ใช้ที่ยังไม่เป็นสมาชิก (Guest) และ ผู้ใช้ที่เป็นสมาชิก (Member)
Precondition	-
Main flow	ผู้ใช้เลือกบทความจากหน้าต่าง ๆ เพื่อเข้าสู่หน้าเนื้อหาของบทความ จากนั้นหากบทความมีเนื้อหาชนิด Code box ผู้ใช้จะสามารถกดดันโค้ด และมีปฏิสัมพันธ์กับผลลัพธ์ได้
Exception flow 1	หากโค้ดมีความยาวเกิน 200 บรรทัดระบบจะไม่อนุญาตให้รัน
Exception flow 2	หากผู้ใช้รันโค้ดครบ 100 โค้ดแล้วจะไม่สามารถรันได้อีกในวันนั้น

ตารางที่ 15 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการรันโค้ด (Run code)

Use case name	Interact to article
Actor	ผู้ใช้ที่เป็นสมาชิก (Member)
Precondition	ผู้ใช้ทำการลงชื่อเข้าใช้แล้ว
Main flow	ผู้ใช้เลือกบทความจากหน้าต่าง ๆ เพื่อเข้าสู่หน้าเนื้อหาของบทความ
Alternate flow	ผู้ใช้เลือกดูถูกใจบทความ เพื่อให้คะแนนบทความ
Alternate flow	ผู้ใช้แสดงความคิดเห็น และตอบกลับความคิดเห็นได้ที่ท้ายบทความ
Exception flow	-

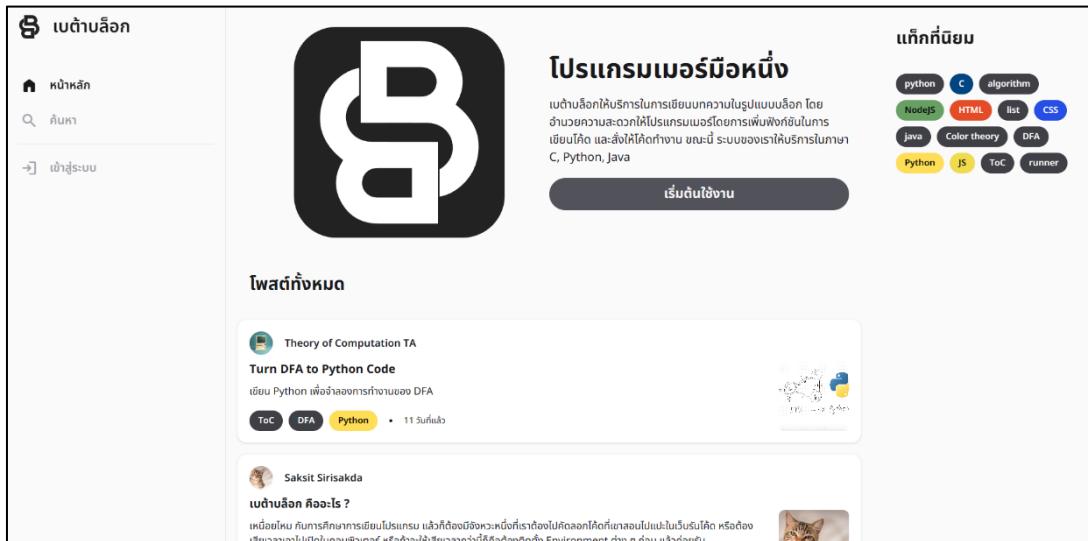
ตารางที่ 16 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการบันทึกบทความไว้ดูภายหลัง (Save article)

Use case name	Save article
Actor	ผู้ใช้ที่เป็นสมาชิก (Member)
Precondition	ผู้ใช้ทำการลงชื่อเข้าใช้แล้ว
Main flow	ผู้ใช้เลือกบทความจากหน้าต่าง ๆ เพื่อเข้าสู่หน้านี้อพาร์ทของบทความ จากนั้นกดปุ่มบันทึกบทความ และบทความดังกล่าวจะปรากฏในรายการที่บันทึกของผู้ใช้
Exception flow	-

ตารางที่ 17 คำอธิบายแผนภาพกรณีการใช้งานสำหรับการดูบทความที่นิยม (View a trending)

Use case name	View a trending
Actor	ผู้ใช้ที่ยังไม่เป็นสมาชิก (Guest) และ ผู้ใช้ที่เป็นสมาชิก (Member)
Precondition	-
Main flow	ผู้ใช้เข้าสู่หน้าแรก จะเห็นรายการหมวดหมู่ที่มีบทความมากที่สุด และบทความที่มียอดกดถูกใจเยอะที่สุด เมื่อกดแล้วผู้ใช้จะถูกนำทางไปยังหน้ารายการดังกว่า
Exception flow	-

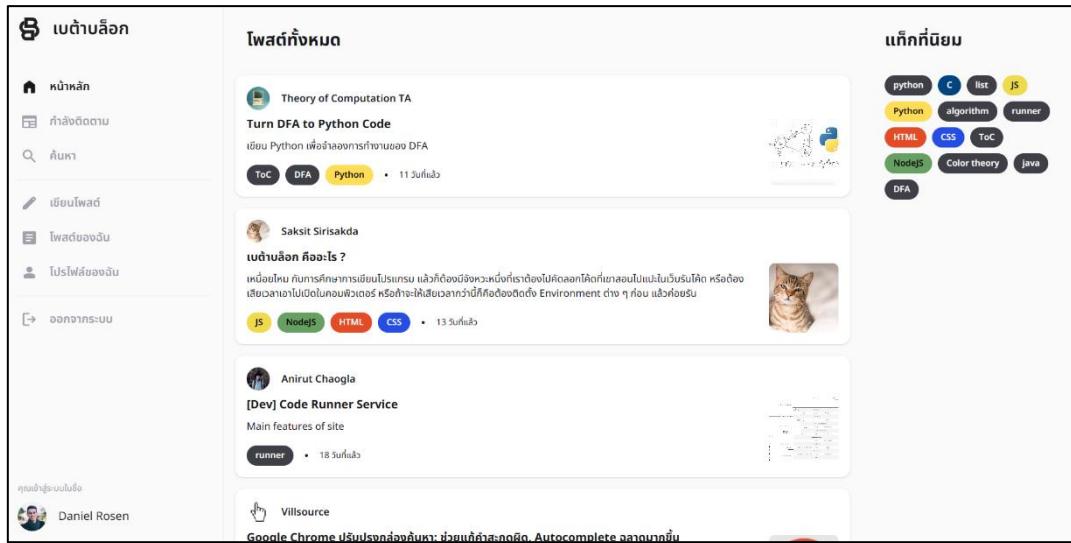
4.2 ส่วนติดต่อผู้ใช้



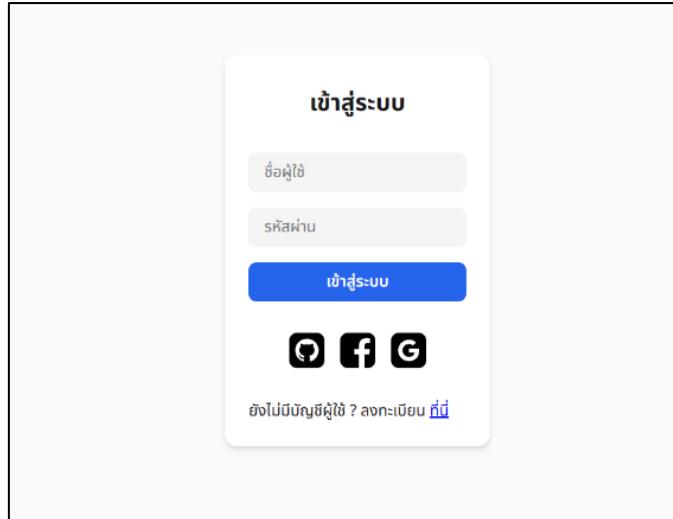
ภาพที่ 5 ส่วนติดต่อผู้ใช้ หน้าหลักของผู้ใช้ที่ยังไม่เข้าสู่ระบบ

จากภาพที่ 5 แสดงถึงหน้าหน้าหลักของผู้ใช้ที่ยังไม่เข้าสู่ระบบ ซึ่งมีองค์ประกอบ ดังนี้

- 1) แบรนด์ แสดงชื่อระบบ และใช้เป็นปุ่มเพื่อกดกลับมายังหน้าหลัก
- 2) แบบค้นหา แสดงรายการค้นหา ตามอักษรที่ใส่
- 3) แบบนำทาง แสดงรายการเมนูในระบบ ในที่นี้มีเพียงการเขียนโพสต์
- 4) ปุ่มเข้าสู่ระบบ ใช้เป็นปุ่มเพื่อไปยังหน้าเข้าสู่ระบบ
- 5) แบบเนอร์ แสดงโลโก้ และข้อความต้อนรับ จะไม่แสดงเมื่อทำการเข้าสู่ระบบแล้ว
- 6) รายการโพสต์ แสดงรายการโพสต์ย่อ โดยเรียงเวลาจากใหม่สุด ไปเก่าสุด หากกดที่ตัวบทความ จะนำทางไปยังหน้าโพสต์ตัวเต็ม และหากกดที่ชื่อผู้โพสต์จะนำทางไปยังหน้าข้อมูลส่วนตัวของผู้โพสต์
- 7) แบบแท็กที่นิยม แสดงแท็กที่มีการใช้งานในระบบมากที่สุด



ກາພທີ 6 ສ່ວນຕິດຕ່ອຜູ້ໃຊ້ ມີຫັນຫຼັກຂອງຜູ້ໃຊ້ທີ່ເຂົ້າສູ່ຮະບບແລ້ວ

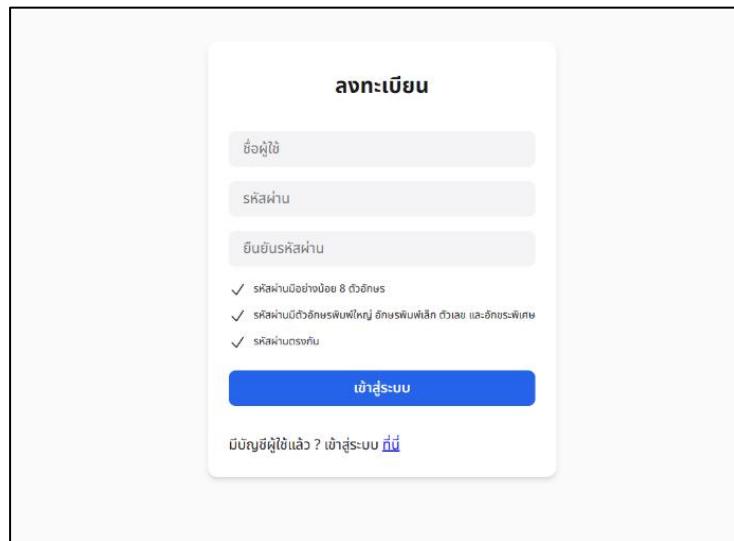


ກາພທີ 7 ສ່ວນຕິດຕ່ອຜູ້ໃຊ້ ມີຫັນເຂົ້າສູ່ຮະບບ

ຈາກກາພທີ 6 ແສດງລຶ່ງໜັນເຂົ້າສູ່ຮະບບສໍາຮັບຜູ້ທີ່ຍິ່ງໄມ້ເຂົ້າສູ່ຮະບບ ທີ່ມີອົງກົດປະກອບ ດັ່ງນີ້

- 1) ພິລົດອິນພຸດໜີ່ອັນພູ້ໃຊ້ ສໍາຮັບກຣອກໜີ່ອັນພູ້ໃຊ້
- 2) ພິລົດອິນພຸດຮ້ສັ່ນ ສໍາຮັບກຣອກຮ້ສັ່ນ

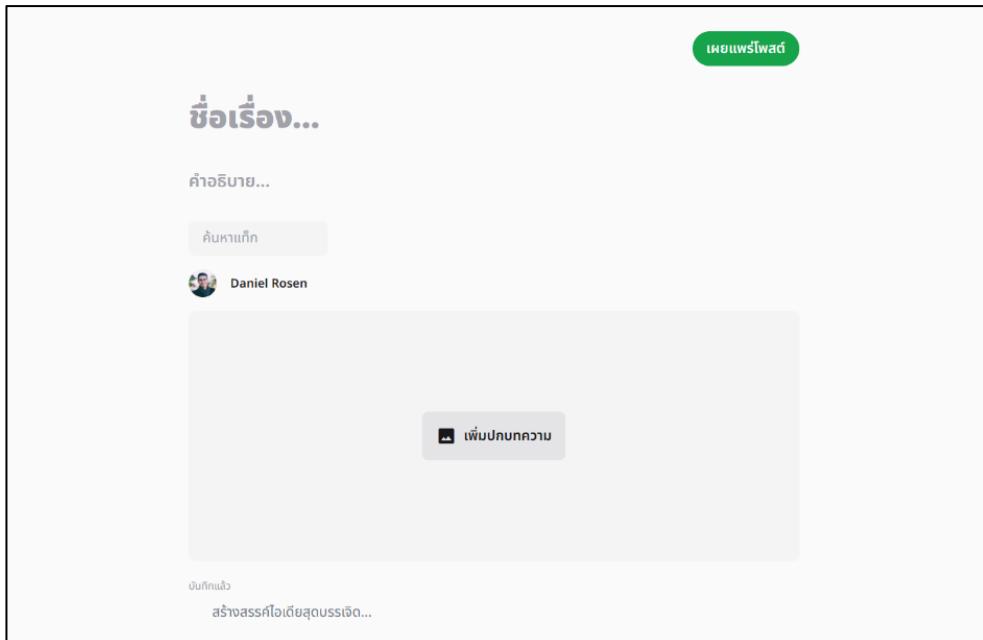
- 3) ปุ่มเข้าสู่ระบบ เพื่อหวานเช็คข้อมูลอินพุต
- 4) ข้อความลงทะเบียน สำหรับผู้ที่ต้องการลงทะเบียน
- 5) ปุ่ม External login สำหรับเข้าสู่ระบบโดยใช้บริการภายนอก



ภาพที่ 8 ส่วนติดต่อผู้ใช้ หน้าลงทะเบียน

จากภาพที่ 8 แสดงถึงหน้าลงทะเบียนสำหรับผู้ที่ยังไม่เข้าสู่ระบบ ซึ่งมีองค์ประกอบดังนี้

- 1) ฟิลด์อินพุตชื่อผู้ใช้ สำหรับกรอกชื่อผู้ใช้
- 2) ฟิลด์อินพุตรหัสผ่าน สำหรับกรอกรหัสผ่าน
- 3) ฟิลด์อินพุตรหัสผ่านอีกครั้ง สำหรับยืนยันรหัสผ่าน
- 4) ข้อความแสดงความเหมาะสม สำหรับตรวจสอบรหัสผ่านเบื้องต้น
- 5) ปุ่มลงทะเบียน เพื่อหวานเช็คข้อมูลอินพุต
- 6) ข้อความเข้าสู่ระบบ สำหรับผู้ที่ต้องการเข้าสู่ระบบ



ภาพที่ 9 ส่วนติดต่อผู้ใช้ หน้าเขียนบุทความ

จากภาพที่ 9 แสดงถึงหน้าเขียนบุทความ ซึ่งมีองค์ประกอบดังนี้

- 1) ฟิล์ดอินพุตชื่อเรื่อง สำหรับใส่ชื่อเรื่อง (จำเป็นต้องใส่)
- 2) ฟิล์ดอินพุตคำอธิบาย สำหรับใส่คำอธิบาย
- 3) ฟิล์ดค้นหาแท็ก สำหรับเลือก หรืออินพุตเลือกแท็ก
- 4) ฟิล์ดรูปภาพปก สำหรับแสดงเป็นปกบุทความ
- 5) ฟิล์ดเนื้อหา สำหรับเนื้อหา ซึ่งมีคัดลอกอยู่ด้วย
- 6) ปุ่มเผยแพร่ สำหรับเผยแพร่บุทความ

Building a simple shell in C - Part 1

A basic shell that prints a prompt, waits for user to enter command and prints what they entered on the next line

 HermesCS



One of the projects I previously worked on as part of my software engineering training at ALX Arica was building a simple shell that mimics the Bash shell but with limited features.

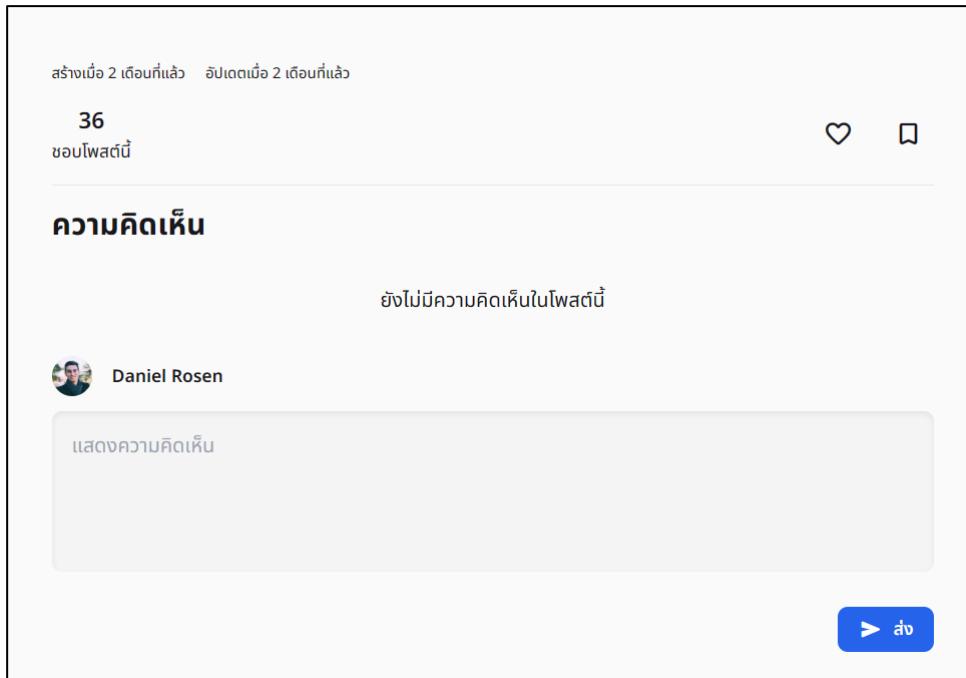
Recently, I have had a few people reach out to ask specific questions about the project, and others want me to organize a tutorial session on it.

Thinking about it, I realized that I had forgotten most of the things because I haven't

ภาพที่ 10 ส่วนติดต่อผู้ใช้ หน้าบุทความ

จากภาพที่ 10 แสดงถึงหน้าบุทความ ซึ่งมีองค์ประกอบดังนี้

- 1) ชื่อเรื่อง สำหรับแสดงชื่อเรื่อง
- 2) คำอธิบาย สำหรับแสดงคำอธิบาย
- 3) แท็ก สำหรับแสดงแท็ก
- 4) รูปภาพปก สำหรับแสดงปกบุทความ



ภาพที่ 11 ส่วนแสดงความคิดเห็น

จากภาพที่ 11 แสดงถึงส่วนแสดงความคิดเห็น ซึ่งมีองค์ประกอบ ดังนี้

- 1) กลุ่มของความคิดเห็นที่ได้แสดงความคิดเห็นในโพสต์นั้น ๆ
- 2) ช่องแสดงความคิดเห็นสำหรับผู้ใช้ที่เข้าสู่ระบบแล้ว
- 3) ปุ่มกดส่งความคิดเห็น

The screenshot shows a user profile for 'Anirut Chaogla' on a platform. The profile picture is a circular photo of a person wearing a hat and a backpack outdoors. Below the profile is the name 'Anirut Chaogla' and the tagline 'I am god'. It also shows two 'ผู้ติดตาม' (followers) and two 'กำลังติดตาม' (following). A button labeled 'ติดตามแล้ว' (Followed) is visible.

Three articles by this user are listed:

- [Dev] Code Runner Service**: Main features of site. Includes a 'runner' badge and 18 วันที่แล้ว (18 days ago).
- Python Program for QuickSort**: Describes QuickSort as a divide and conquer algorithm. Includes a 'python' badge and 2 เดือนที่แล้ว (2 months ago). To the right is a diagram illustrating the QuickSort algorithm with numbers 3, 6, 7, 8, 5 partitioned around a pivot '1'.
- Python Lists**: Lists are used to store multiple items in a single variable. Includes a 'python' badge and 2 เดือนที่แล้ว (2 months ago). To the right is a diagram showing a list of elements [C, S, F, O, R, G] being sorted into [S, F, O, R, C, G].

ภาพที่ 12 ส่วนแสดงหน้าໂປຣໄຟລ໌

จากภาพที่ 12 แสดงถึงหน้าໂປຣໄຟລ໌ສໍາຫັບແສດງຜລື້ອງຜູ້ໃໝ່ ຊຶ່ງມີອົກປະກອບ ດັ່ງນີ້

- 1) ຮູບໂປຣໄຟລ໌ຂອງຜູ້ໃໝ່
- 2) ຜ່ອນຂອງຜູ້ໃໝ່
- 3) ຄໍາອົບຍາຍຜູ້ໃໝ່
- 4) ປຸ່ມກົດຕິດຕາມຜູ້ໃໝ່ ອາກທໍາການເຂົ້າສູ່ຮບບແລ້ວ ແລະໄຟຜູ້ໃໝ່ທີ່ເປັນຕົວເອງ
- 5) ກລຸ່ມຂອງໂພສຕໍ່ທີ່ຜູ້ໃໝ່ຄົນນັ້ນໄດ້ແຜຍແພຣ່ມາແລ້ວ

```

9     nextState = transition[int(a)]
10    |
11    return yeild(nextState, string[1:])

```

* * *

จะได้ผลลัพธ์การทำงานตามภาษา Python ดังนี้

DFA with python

ไปหน้ารันโค้ด

```

1
2 # δ(state, input) = next state
3 transition = [
4     [0,1],
5     [0,2],
6     [1,0]
7 ]
8
9 def yeild(state, string):
10
11     print("(q%u, %s)"%(state, string))
12
13     if(len(string)==0):
14         | return state
15
16     a = string[0:1]
17     nextState = transition[state][int(a)]
18
19     return yeild(nextState, string[1:])
20
21
22 print(yeild(0, "1011101010101110"))

```

ภาพที่ 13 ส่วนติดต่อผู้ใช้ในการเขียนโค้ด

```

python
1
2 # δ(state, input) = next state
3 transition = [
4     [0,1],
5     [0,2],
6     [1,0]
7 ]
8
9 def yeild(state, string):
10
11     print("(q%u, %s)"%(state, string))
12
13     if(len(string)==0):
14         | return state
15
16     a = string[0:1]
17     nextState = transition[state][int(a)]
18
19     return yeild(nextState, string[1:])
20
21
22 print(yeild(0, "1011101010101110"))

```

(q0, 101110101010101110)
(q1, 0110101010101110)
(q0, 11101010101110)
(q1, 1101010101110)
(q2, 101010101110)
(q0, 01010101110)
(q0, 1010101110)
(q1, 010101110)
(q0, 10101110)
(q1, 011110)
(q0, 01110)
(q0, 1110)
(q1, 110)
(q2, 10)
(q0, 0)
(q0,)
0

Exit with status code 0

ภาพที่ 14 หน้าสำหรับเขียนโค้ดในโพสต์

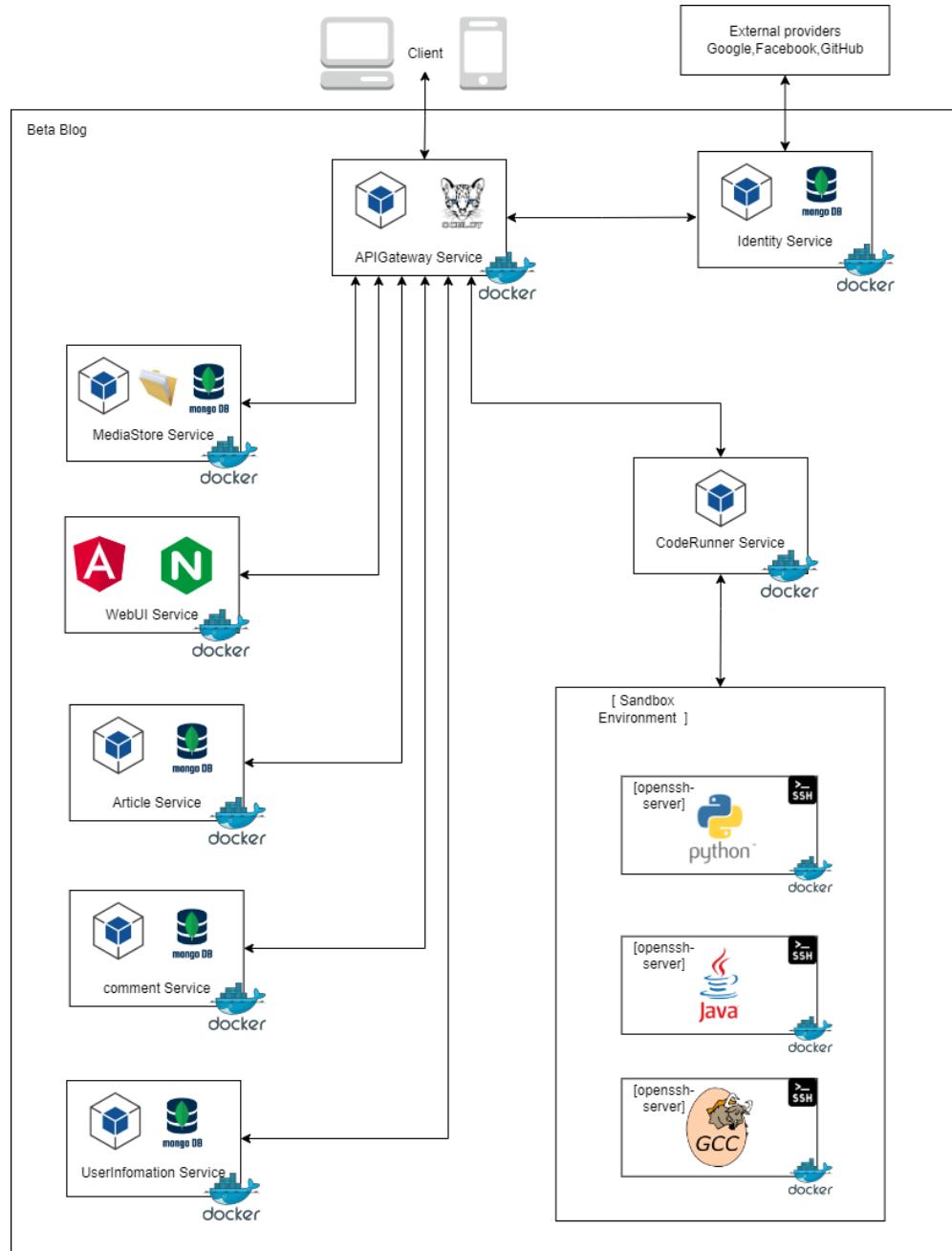
จากภาพที่ 13 แสดงถึงส่วนติดต่อผู้ใช้สำหรับใช้ในการเขียนโค้ด โดยสามารถทำการเขียนได้ แต่ไม่สามารถรันในที่นี้ได้โดยตรงได้ ซึ่งมีองค์ประกอบดังนี้

- 1) ชื่อของโปรแกรม
- 2) ภาษาโปรแกรมที่ใช้
- 3) ปุ่มสำหรับกดเพื่อไปยังหน้ารันโค้ดตัวเต็ม
- 4) พื้นที่สำหรับเขียนโค้ด

จากภาพที่ 14 แสดงถึงหน้าเพจสำหรับคอมไพล์โค้ด โดยจะมีการเพิ่มส่วนของหน้าแสดงผลลัพธ์ เพิ่มจากเครื่องมือดังภาพที่ 13 ซึ่งมีองค์ประกอบดังนี้

- 1) ชื่อของโปรแกรม
- 2) ภาษาโปรแกรมที่ใช้
- 3) ปุ่มสำหรับคอมไпал์โค้ด
- 4) ปุ่มสำหรับยกเลิกการคอมไпал์โค้ด
- 5) ปุ่มสำหรับปิดหน้าคอมไпал์โค้ด (พร้อมกับยกเลิกการคอมไпал์)
- 6) พื้นที่สำหรับเขียนโค้ด (ไม่มีผลกับตัวโค้ดต้นฉบับ)
- 7) พื้นที่สำหรับแสดงผลลัพธ์คำสั่ง และสามารถใส่ข้อมูลนำเข้าได้หากมีการรับค่านำเข้า

4.3 สถาปัตยกรรมของระบบเบต้าบล็อก

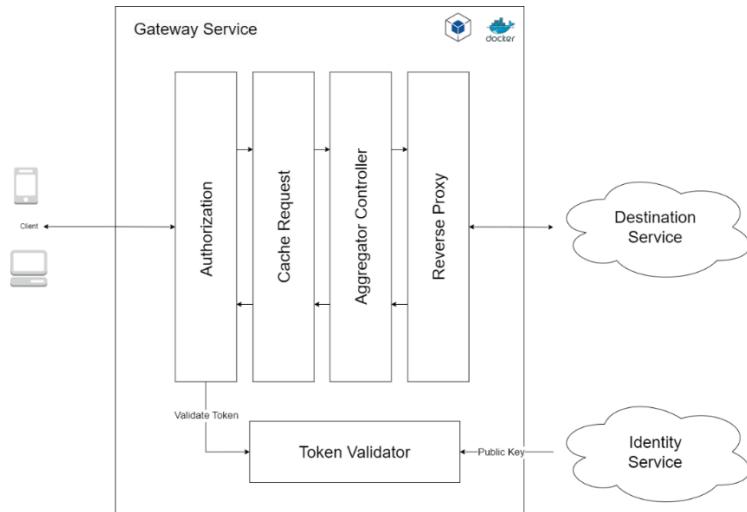


ภาพที่ 15 แผนภาพสถาปัตยกรรมของระบบ Beta blog

ระบบเบต้าบล็อกจะถูกออกแบบโดยสถาปัตยกรรมไมโครเซอร์วิสที่แบ่งเซอร์วิสออกจากกันด้วยคุณลักษณะ จากราพที่ 15 โดยจะแบ่งออกเป็น 8 เซอร์วิส ซึ่งประกอบด้วย 1) API Gateway service ให้บริการ เป็น Forward proxy เพื่อให้สะ粿ต่อการเรียกใช้บริการอื่น ๆ ด้วยโภสต์ในเมเดียกัน การแคนช และรวมข้อมูล จากหลายเซอร์วิสเป็นรีเควสต์เดียว เพื่อลดความซ้ำซ้อนของเครือข่ายผ่านการเรียกใช้จากคลาวน์ 2) Identity service ให้บริการในด้านบัญชีผู้ใช้ การกำหนดสิทธิ์ผู้ใช้ และการยืนยันตัวตน 3) Media store service ให้บริการ กีบไฟล์รูปภาพต่าง ๆ ในระบบ 4) Web UI service ให้บริการโภสต์สำหรับทำหน้าเว็บเพจเดียว (Single page application, SPA) ซึ่งพัฒนาโดย Angular 5) Article service ให้บริการจัดเก็บบทความ 6) Comment service ให้บริการในการเก็บความคิดเห็นของแต่ละโพสต์ 7) User information service ให้บริการในการเก็บข้อมูลผู้ใช้ และความสัมพันธ์ระหว่างผู้ใช้ 8) Code runner service ให้บริการในการรันโค้ดพร้อมรับข้อมูลนำเข้า และข้อมูลส่งออก ในการคอมไพล์โค้ดแบบทันที และมีเซอร์วิสอยเพื่อทำการจำลองสภาพแวดล้อมในการรันโค้ดเพื่อ ป้องกันไม่ให้โค้ดที่ผู้ใช้ส่งเข้ามากระทบต่อระบบหลัก

4.3.1 บริการจัดการเกตเวย์ของ API (API Gateway service)

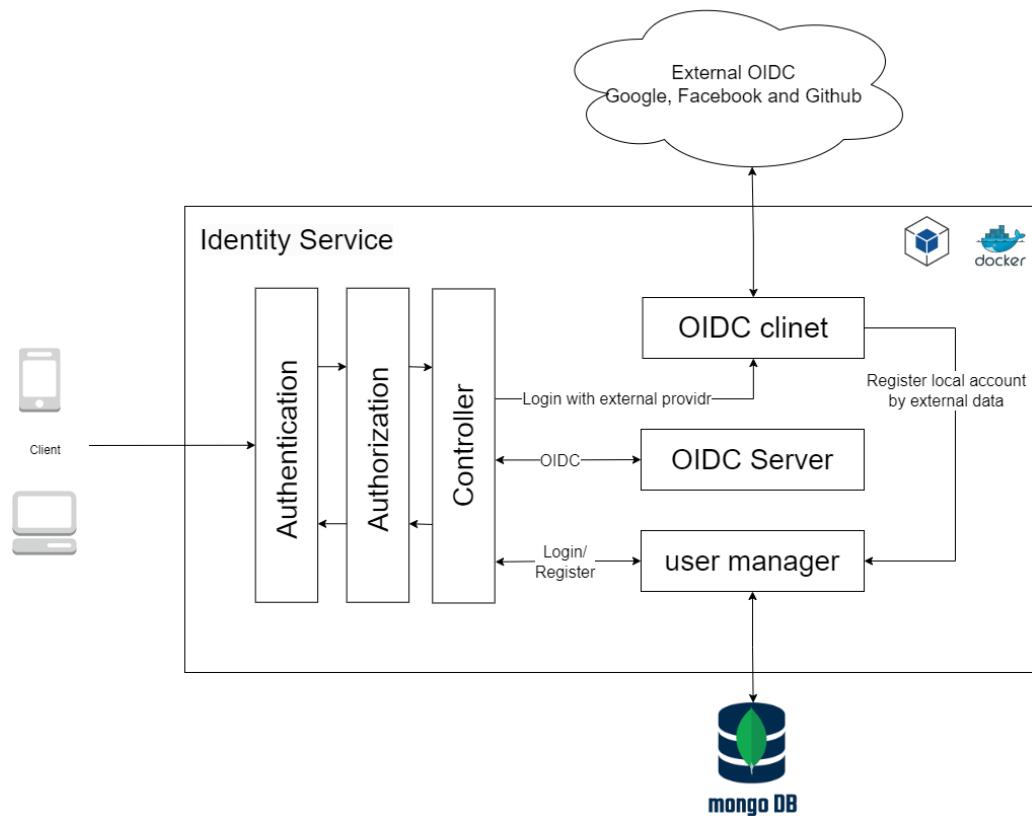
บริการจัดการเกตเวย์ของ API ให้บริการเป็น Forward proxy เพื่อให้สະดาກต่อการเรียกใช้บริการอื่น ๆ ด้วยโถสต์เนมเดียวกัน การแคช และรวบรวมข้อมูลจากหลายเซอร์วิสเป็นรีเควสต์เดียว เพื่อลดความซ้ำซ้อนของเครือข่ายผ่านการเรียกใช้จากไคลเอนต์



ภาพที่ 16 สถาปัตยกรรมของบริการจัดการเกตเวย์ API

จากภาพที่ 16 จะแสดงขั้นตอนการทำงานของบริการจัดการเกตเวย์ โดยก่อนที่คำร้องขอจะถูกส่งต่อให้บริการปลายทาง ตัวเกตเวย์จะทำการยืนยันตัวตนและตรวจสอบสิทธิ์ของผู้ใช้ด้วย JWT (JSON web token) ที่ผู้ใช้ส่งมาพร้อมกับคำร้องขอพร้อมแนบข้อมูลผู้ใช้ให้บริการปลายทางได้ทำการประมวลผลต่อไป และเพื่อลดการทำงานของบริการจัดการบัญชีและยืนยันตัวตนผู้ใช้ บริการจัดการเกตเวย์จะบันทึกคุณลักษณะที่ใช้ถูกเข้ารหัส JWT

4.3.2 บริการจัดการบัญชีและยืนยันตัวตนผู้ใช้ (Identity service)

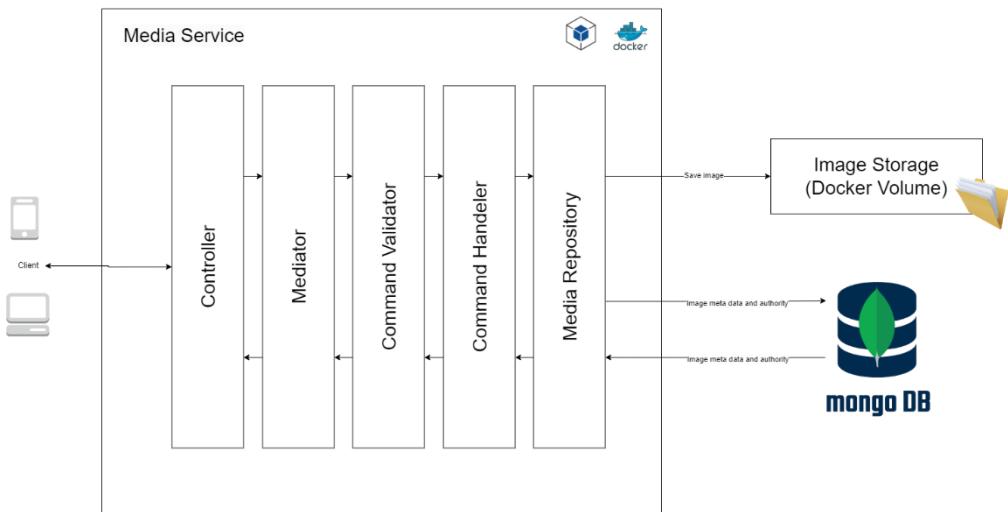


ภาพที่ 17 สถาปัตยกรรมของบริการจัดการบัญชีและยืนยันตัวตนผู้ใช้

จากภาพที่ 17 แสดงถึงสถาปัตยกรรมของบริการจัดการบัญชีและยืนยันตัวตนผู้ใช้ ที่มีหน้าที่ในการจัดการบัญชีของผู้ใช้ ไม่ว่าจะเป็นชื่อผู้ใช้ หรือรหัสผ่านสำหรับเข้าสู่ระบบ หรือการสมัครลงทะเบียนของผู้ใช้ใหม่ และยังรวมไปถึงบริการยืนยันตัวตนจากบริการภายนอก โดยไม่จำเป็นจะต้องสมัครบัญชีผ่านทางระบบโดยตรง ซึ่งในที่นี้ได้ใช้บริการจาก Facebook, Google และ Github

4.3.3 บริการจัดการไฟล์ (Media Service)

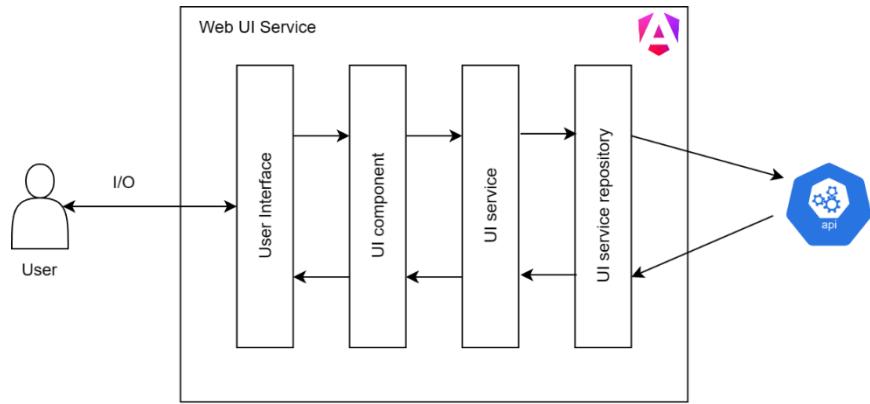
ระบบจัดการบุคลากรเป็นบริการย่อยของระบบเบต้าบล็อกซึ่งมีหน้าที่ให้บริการจัดเก็บและเข้าถึงไฟล์รูปภาพ รวมถึงผู้ใช้สามารถจัดการไฟล์ของผู้ใช้งานได้ เช่น การลบ การเพิ่ม



ภาพที่ 18 สถาปัตยกรรมของบริการจัดการไฟล์

จากภาพที่ 18 แสดงให้เห็นถึงสถาปัตยกรรมของระบบจัดการไฟล์ ซึ่งทำหน้าที่บันทึกไฟล์ลงในระบบจัดเก็บไฟล์ และบันทึกข้อมูลรายละเอียด เช่น ชื่อภาพ ขนาด วันที่จัดเก็บ และข้อมูลทางสังคมิตรต่าง ๆ ในฐานข้อมูล เพื่อให้สามารถวิเคราะห์ได้ หากต้องการ

4.3.4 บริการจัดการส่วนติดต่อผู้ใช้ (Web UI service)

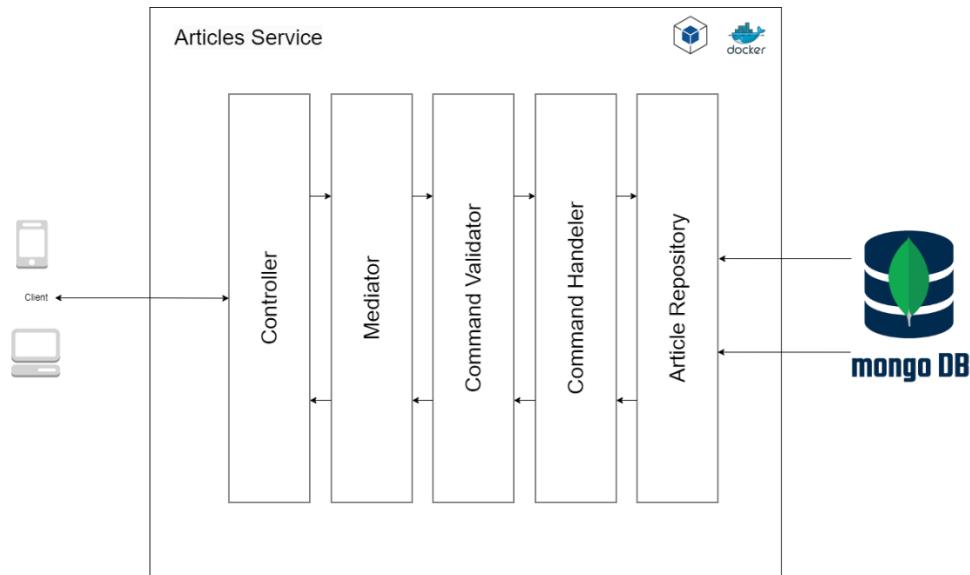


ภาพที่ 19 สถาปัตยกรรมของบริการจัดการส่วนติดต่อผู้ใช้

จากภาพที่ 19 แสดงถึงสถาปัตยกรรมของบริการจัดการส่วนติดต่อผู้ใช้ ซึ่งผู้จัดทำได้เลือกใช้เครื่องมือ Angular ในการสร้างส่วนติดต่อผู้ใช้ โดยจะเป็นตัวกลางในการรับส่งข้อมูลระหว่างผู้ใช้กับบริการอื่น ๆ

4.3.5 บริการจัดการบทความ (Article Service)

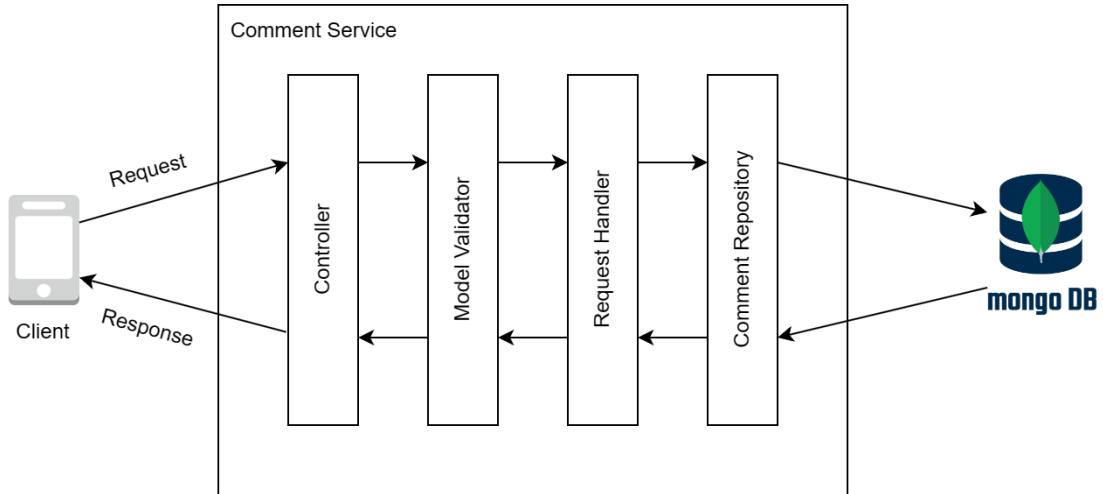
ระบบจัดการบทความเป็นบริการย่อยของระบบเบต้าบล็อกซึ่งมีหน้าที่ให้บริการจัดการบทความในระบบ เช่น การค้น การเพิ่ม แก้ไข การลบ และข้อมูลทางสถิติของบทความ



ภาพที่ 20 สถาปัตยกรรมของระบบจัดการบทความ

จากภาพที่ 20 แสดงให้เห็นถึงสถาปัตยกรรมของระบบจัดการบทความ ที่ถูกออกแบบให้เป็นชั้น เพื่อให้ ง่ายต่อการจัดการคำร้องขอ และง่ายต่อการตรวจสอบความถูกต้องของข้อมูลที่ได้รับจากผู้ใช้ [18] โดยชั้นแรกที่ พัฒนาคือ Controller เป็นตัวช่วยจัดการคำร้องขอจากผู้ใช้เพื่อดำเนินรายการให้ถูกต้องตามที่ผู้ใช้ต้องการ ซึ่งแต่ละ การดำเนินการจะถูกเรียกด้วย Command handler เพื่อทำการอ่านหรือแก้ไขข้อมูลในฐานข้อมูลผ่าน Article repository และก่อนที่คำขอจากผู้ใช้จะถูกดำเนินการ จะต้องผ่านการตรวจสอบความถูกต้องด้วย Command validator เสียก่อน เพื่อให้ง่ายต่อการจัดการ Endpoint ของ API ที่มีจำนวนมากผู้จัดทำจึงเลือกใช้ Mediator pattern เพื่อช่วยในการจัดการคำขอของผู้ใช้

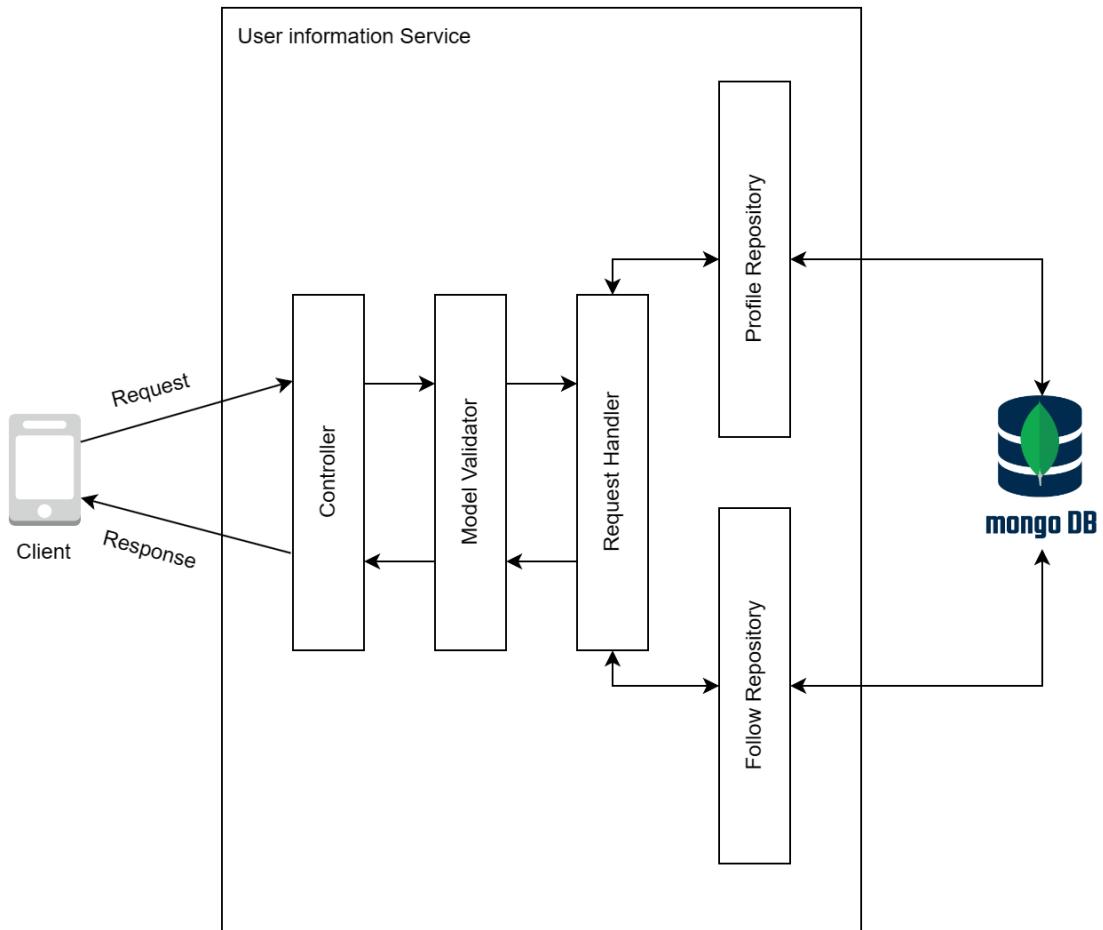
4.3.6 บริการจัดการความคิดเห็น



ภาพที่ 21 สถาปัตยกรรมของบริการจัดการความคิดเห็น

จากภาพที่ 21 แสดงให้เห็นถึงสถาปัตยกรรมของบริการจัดการความคิดเห็น ซึ่งมีหน้าที่ให้บริการในการจัดการความคิดเห็น โดยฐานข้อมูลในบริการนี้มีการเก็บความคิดเห็นแต่ละความคิดเห็นเอาไว้ ในส่วนของการจัดการความคิดเห็น มีการให้บริการในการเพิ่มความคิดเห็น และลบความคิดเห็น

4.3.7 บริการจัดการข้อมูลผู้ใช้ (User information service)

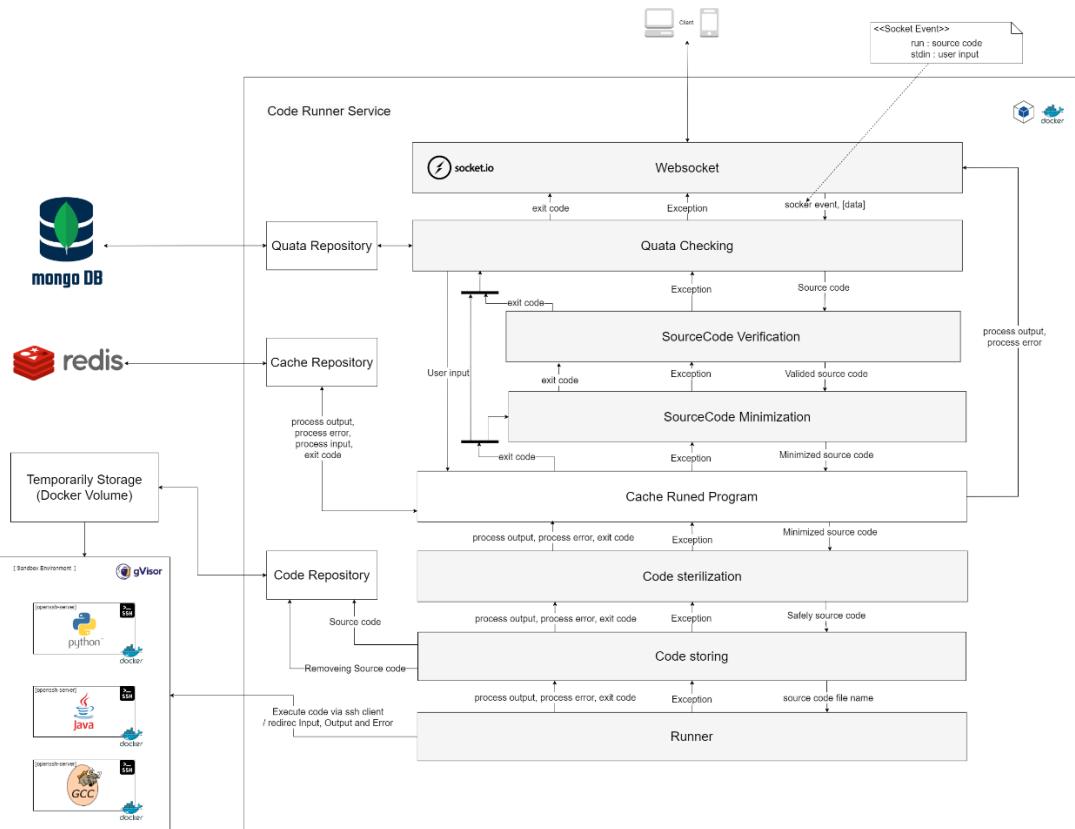


ภาพที่ 22 สถาปัตยกรรมของบริการจัดการข้อมูลผู้ใช้

จากภาพที่ 22 แสดงให้เห็นถึงสถาปัตยกรรมของบริการจัดการข้อมูลผู้ใช้ ซึ่งมีหน้าที่ในการจัดการข้อมูล โปรแกรมผู้ใช้ เช่น ชื่อที่ใช้แสดงผล และคำอธิบายตัวผู้ใช้ และจัดการการติดตามผู้ใช้ โดยจะเก็บความสัมพันธ์ลงในฐานข้อมูล เมื่อมีการติดตาม และนำความสัมพันธ์ออกจากฐานข้อมูล เมื่อมีการเลิกติดตาม ทั้งนี้ ยังสามารถขอข้อมูลจำนวนผู้ติดตาม และผู้ที่กำลังติดตามได้จากการนี้อีกด้วย

4.3.8 บริการรันโค้ด (Code runner service)

ระบบบันทึกเป็นบริการย่อยของระบบเบต้าบล็อกซึ่งมีหน้าที่ให้บริการรันโค้ดแก่ผู้ใช้พร้อมกับประมวลผลลัพธ์และข้อมูลนำเข้าจากผู้ใช้แบบเรียลไทม์



ภาพที่ 23 สถาปัตยกรรมของระบบบันทึก

จากภาพที่ 23 ระบบบันทึกจะใช้ Web socket เป็นส่วนติดต่อผู้ใช้ เพื่อให้ผู้ใช้สามารถเห็นข้อมูลส่งออกจากโปรแกรมที่ถูกส่งเข้ามาประมวลผลในระบบได้แบบเรียลไทม์ หลังจากระบบได้รับคำร้องขอจากผู้ใช้แล้ว ระบบจะทำการวิเคราะห์โค้ดที่ผู้ใช้ส่งเข้ามาในระบบ ก่อนทำการประมวลผลจริง เพื่อลดความเสี่ยงที่จะเกิดขึ้นกับระบบโดยโค้ดที่ถูกวิเคราะห์แล้วจะถูกบันทึกลงหน่วยความจำส่วนกลาง และถูกรันโดยระบบที่เป็นสภาพแวดล้อมจำลอง เพื่อลดความเสี่ยงที่อาจเกิดขึ้นกับระบบจริง ซึ่งเป็นการเพิ่มความปลอดภัยให้แก่ระบบอีกหนึ่งชั้น หากการประมวลผลเริ่มส่งข้อมูลส่งออกหรือข้อผิดพลาดแล้ว ตัวระบบบันทึกจะส่งข้อมูลเหล่านั้นกลับไปให้ผู้ใช้ประมวลผลต่อ และจะเป็นเข่นเดียวกันเมื่อผู้ใช้ส่งข้อมูลนำเข้าเข้าสู่ระบบบันทึก

บทที่ 5

สรุปผลการดำเนินงาน

5.1 สรุปการดำเนินงานโครงการ

ในการทำโครงการในครั้งนี้ ผู้จัดทำได้ทำการพัฒนาระบบในส่วนบริการการจัดการเนื้อหาเรียบร้อยแล้ว แต่ในส่วนบริการรับโค้ดยังคงพัฒนาคุณลักษณะเกี่ยวกับความมั่นคงของระบบจากการคอมไพล์โค้ดในโพสต์อยู่ในขณะนี้ และทางผู้จัดทำได้พบปัญหาระหว่างการพัฒนา ได้แก่ การตรวจสอบข้อผิดพลาดเกี่ยวกับภายสัมพันธ์ของภาษาระหว่างการเขียนโค้ดของผู้ใช้ในส่วนหน้าของระบบ

ซึ่งรายการงานทั้งหมด แสดงดังตารางที่ 18

ตารางที่ 18 รายงาน และความก้าวหน้า

Task	Done	Test
User Registration		
Develop User Registration UI	<input checked="" type="checkbox"/>	
Develop User Registration API Endpoint	<input checked="" type="checkbox"/>	
User Login		
Develop User Login UI	<input checked="" type="checkbox"/>	
Implement Token Storage Service on Client	<input checked="" type="checkbox"/>	
Develop User Login API Endpoint	<input checked="" type="checkbox"/>	
Implement Oauth of GitHub	<input checked="" type="checkbox"/>	
Implement Oauth of Facebook	<input checked="" type="checkbox"/>	
Implement Oauth of Google	<input checked="" type="checkbox"/>	

ตารางที่ 18 รายงาน และความก้าวหน้า (ต่อ)

Task	Done	Test
User Logout		
Design and Implement Logout Functionality	✓	
Article Browsing (feed)		
Design Article Card UI Component	✓	
Design Article Feed UI	✓	
Implement Service to Retrieve Articles from Backend	✓	
Develop Articles Pagination API	✓	
Establish Aggregator API for User Article Interactions (Like/Bookmark)	✓	
Article Search		
Design Article Search UI		
Develop API for Article Search by Name, Tags and Author	✓	
Article Creation		
Design Article Editor Interface	✓	
Implement Header Editing Features (H1-H4)	✓	
Implement Image Uploading and Editing	✓	
Implement Code Editor Integration (monaco)	✓	
Design Section Break Functionality	✓	
Integrate Link Embedding Features		
Develop API Endpoint for New Article Submission	✓	

ตารางที่ 18 รายงาน และความก้าวหน้า (ต่อ)

Task	Done	Test
Article Reading		
Design UI for Article Display	✓	
Implement Header Display Features (H1-H4)	✓	
Implement Image Display in Articles	✓	
Implement Code Editor Display	✓	
Design Section Break Display	✓	
Integrate Link Embed Display		
Develop API Endpoint to Retrieve Specific Article Content	✓	
Develop aggregator API to check if user are liked or saved article to bookmark	✓	
like article		
create like button and functional	✓	
create like count display and functional	✓	
create like API PATCH /article/:id/like	✓	
create unlike API PATCH /article/:id/unlike	✓	
create API to get like count of article	✓	
comment article		
create comment box (content editor)	✓	
create comment box rendering	✓	
create reply comment box rendering	✓	
create comment modifying UI	✓	

ตารางที่ 18 รายงาน และความก้าวหน้า (ต่อ)

Task	Done	Test
create comment delete button	✓	
create comment service to CRUD a comments	✓	
create API to add new comment to article	✓	
create API to get comments of one article	✓	
create API to update comment content	✓	
create API to Delete comment	✓	
bookmark article		
Design UI for Displaying Bookmarked Articles	✓	
Implement Article Bookmarking Functionality	✓	
Develop API Endpoint for Bookmarking Articles	✓	
Develop API Endpoint for Removing Article Bookmarks	✓	
Develop API Endpoint to Retrieve Bookmarked Articles	✓	
Article Modification		
create article editor UI for modifying	✓	
create Header editing H1, H2, H3, H4 for modifying	✓	
create Image editing for modifying	✓	
create Code editor editing for modifying	✓	
create Section break editing for modifying	✓	
create link embed editing for modifying		
create save button	✓	

ตารางที่ 18 รายงาน และความก้าวหน้า (ต่อ)

Task	Done	Test
create publish button	✓	
create API to update article content	✓	
create API to publish article	✓	
Article Deletion		
Design UI Prompt for Article Deletion Confirmation	✓	
Profile		
create page to display profile information that include avatar, displayname, phone, email, name, bio, website, social media, address and etc.	✓	
create UI to display all saved articles (pageable)	✓	
create UI to display all my articles (pageable) (both publish and unpublish)	✓	
create UI to show all my follower and following	✓	
create service to get profile information	✓	
create API to provide profile information	✓	
create API to provide all articles of user (pageable)	✓	
create API to provide all saved articles (pageable)	✓	
create API to provide follower and following information (pageable)	✓	
Update Profile Info		
create form to edit profile information that include avatar, displayname, phone, email, name, bio, website, social media, address and etc.	✓	
create service to update profile information	✓	

ตารางที่ 18 รายงาน และความก้าวหน้า (ต่อ)

Task	Done	Test
create API to update profile information	✓	
Run Code		
Integrate Monaco Code Editor	✓	
Design Terminal for I/O Interactions	✓	
Develop Service for Executing Code and Managing I/O	✓	
create socket server to get sourcecode from client	✓	
create socket server to transfer I/O between client and server	✓	
Ensure Security via Code Sanitization Middleware		
create service to handle child process and redirect I/O to client	✓	
Design and Implement Python Language Sandboxes (Docker)	✓	
Design and Implement Gcc Language Sandboxes (Docker)	✓	
Design and Implement Java Language Sandboxes (Docker)	✓	
Content Suggestions		
Design UI for Tag Suggestions	✓	
Develop API Endpoint for Popular Tags Based on Post Count	✓	

5.2 ข้อจำกัดของระบบ

5.2.1 ภาษาโปรแกรมที่ใช้ได้กับระบบยังมีจำนวนที่น้อย ซึ่งประกอบด้วย ภาษาซี (GCC 12.2.1 20220924) ภาษาจาวา (Microsoft-jdk-17.0.6) และ ภาษาไพธอน (Python3 13.11.6)

5.2.2 ไลบารีที่สามารถใช้ได้ในภาษาชีและภาษาจาวาเป็นเพียงไลบารีพื้นฐานเท่านั้น สำหรับภาษาไพธอน จะสามารถใช้ไลบารี numpy และ pandas ได้

5.3 ปัญหาอุปสรรค และแนวทางแก้ไข

5.3.1 ปัญหา

ส่วนหน้า ในการตรวจสอบวากยสัมพันธ์ของโปรแกรมที่ผู้ใช้เขียนขึ้นมา จำเป็นจะต้องเพิ่ม เชิร์ฟเวอร์สำหรับใช้ในการตรวจสอบวากยสัมพันธ์ ซึ่งไม่เป็นการตอบสนองต่อความต้องการในการลดการร้องขอ ข้อมูลระหว่างเชิร์ฟเวอร์กับโคลเลอนต์ ซึ่งไม่เวลาที่เพียงพอในการศึกษาการตรวจจับวากยสัมพันธ์ในส่วนหน้าด้วย ตัวเอง โดยไม่เพิ่งบริการจากภายนอก

5.3.2 แนวทางการแก้ไข

ส่วนหน้า ยอมปล่อยให้ผู้ใช้ส่งคำขอในการรันโค้ดตามปกติ แต่มีการกำหนดเวลาในการรัน โค้ดต่อหนึ่งหน่วยเวลา เพื่อป้องกันไม่ให้ผู้ใช้ส่งคำขอมากเกินกว่าที่เชิร์ฟเวอร์สามารถรับได้

5.4 ข้อเสนอแนะ

5.4.1 เนื่องจากระบบเบต้าบล็อกมีการป้องกันโค้ดที่มีความเสี่ยงเพียงเบื้องต้นเท่านั้น หากผู้อ่านได้นำ โครงการเล่นนี้ไปศึกษาและพัฒนาต่อ ทางผู้จัดทำแนะนำให้ศึกษาเรื่องการวิเคราะห์โปรแกรมแบบสถิติ เพื่อเพิ่ม ความมั่นคงให้ระบบมากยิ่งขึ้น

5.4.2 หากผู้อ่านได้นำเบต้าบล็อกไปพัฒนาต่อ ผู้อ่านสามารถเพิ่มไลบารีต่อภาษา และภาษาที่รองรับได้ โดยการสร้างคอนเทนเนอร์ที่บรรจุสภาพแวดล้อมที่ต้องการ

เอกสารอ้างอิง

- [1] Codedamn, “Why is Programming important? The importance of computer programming explained,” 2022. [ออนไลน์]. Available: <https://codedamn.com>.
- [2] Techgig, “Top 5 websites that will answer all your programming questions,” 2022. [ออนไลน์]. Available: <https://content.techgig.com>.
- [3] อ. สุคนเขต์, “ระบบการจัดการเนื้อหาของเว็บไซต์ฯ,” วารสารวิทยบริการ, เล่มที่ 1, 2552.
- [4] Wikipedia, “Blog,” 2022. [ออนไลน์]. Available: <https://en.wikipedia.org>.
- [5] J. Lewis และ M. Fowler, “Microservices,” [ออนไลน์]. Available: <https://martinfowler.com>.
- [6] iq.opengenus.org, “Different phases of Compiler,” [ออนไลน์]. Available: <https://iq.opengenus.org>.
- [7] Geeksforgeeks, “Phases of a compiler,” [ออนไลน์]. Available: <https://www.geeksforgeeks.org>.
- [8] Amazon, “Microservices,” [ออนไลน์]. Available: <https://docs.aws.amazon.com>.
- [9] “What is .NET?,” Microsoft, [ออนไลน์]. Available: <https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet>.
- [10] “Ocelot,” [ออนไลน์]. Available: <https://ocelot.readthedocs.io/>.
- [11] “Introduction to Identity on ASP.NET Core,” Microsoft, [ออนไลน์]. Available: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity>.
- [12] “What's OpenIddict?,” [ออนไลน์]. Available: <https://documentation.openiddict.com>.
- [13] “What is MongoDB?,” MongoDB, [ออนไลน์]. Available: <https://www.mongodb.com>.
- [14] “Developers bring their ideas to life with Docker,” Docker, [ออนไลน์]. Available: <https://www.docker.com/why-docker>.
- [15] “ANGULAR FEATURES,” Google, [ออนไลน์]. Available: <https://angular.io/features>.

- [16] “Editor.js,” Codex, [ออนไลน์]. Available: <https://editorjs.io>.
- [17] “Monaco - The Editor of the Web,” Microsoft, [ออนไลน์]. Available: <https://microsoft.github.io/monaco-editor>.
- [18] J. Savolainen และ V. Myllarniemi, “Layered architecture revisited — Comparison of research and practice,” ใน *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, 2009.
- [19] T. H. Laine และ J. Suhonen, “Establishing a mobile blog system in a distance education environment,” *Int. J. Mobile Learning and Organisation*, เล่มที่ 2, 2008.
- [20] ภากร คุกรินทร์รัตน์, “การออกแบบและพัฒนาเฟรมเวิร์คสำหรับระบบไมโครเซอร์วิสแบบกระจาย,” 2558.

ลงชื่อผู้ทำโครงการ

(นายอนิรุทธิ์ ชาวกล้า)

.....
(นายศักดิ์สิทธิ์ ศิริศักดา)

วันที่/...../.....

การตรวจสอบจากอาจารย์ที่ปรึกษาโครงการ

.....
.....
.....
(ลงชื่อ)

(พศ.ดร. ชิตสุรา สุ่มแล็ก)

วันที่/...../.....