การประยุกต์ใช้ไมโครฟรอนต์เอนส์กับการปรับโครงสร้างใหม่ชิงเกิลเพจแอปพลิเคชัน

น.ส.ณัฐพร นพปฎล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2563

Application of Microfrontends to Reengineering Single Page Application

Miss Nattaporn Noppadol

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Software Engineering
Department of Computer Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2020

หัวข้อวิทยานิพนธ์	การประยุกต์ใช้ไมโครฟรอนต์เอนส์กับการปรับโครงสร้างใหม่ซิงเกิลเพจแอปพลิเคชัน
โดย	น.ส.ณัฐพร นพปฎล
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.สุกรี สินธุภิญโญ)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์)

..... กรรมการภายนอกมหาวิทยาลัย
(อาจารย์ ดร.ภาสกร อภิรักษ์วรพินิต)

ณัฐพร นพปฎล : การประยุกต์ใช้ไมโครฟรอนต์เอนส์กับการปรับโครงสร้างใหม่ซิงเกิลเพจแอปพลิเคชัน. (Application of Microfrontends to Reengineering Single Page Application) อ.ที่ปรึกษาหลัก : รศ. ดร.ญาใจ ลิ่มปิยะกรณ์

สถาปัตยกรรมระบบซอฟต์แวร์ปกติมักแบ่งแยกเป็นส่วนหน้าและส่วนหลัง โดยที่แบ็กเอนด์มีหน้าที่ในการประมวลผลข้อมูลทางฝั่งเซิร์ฟเวอร์ ในขณะที่ฟรอนต์เอนด์รับผิดชอบการโต้ตอบระหว่างไคลเอนต์และระบบ บรรดาแนวทางสถาปัตยกรรมแบ็กเอนด์ในปัจจุบัน ไมโครเซอร์วิสเป็นทางเลือกหนึ่งที่เหมาะสมที่สุดสำหรับระบบที่สามารถขยายได้ ในขณะเดียวกัน แอปพลิเคชันฝั่งไคลเอนต์ก็เติบโตขึ้นตามขนาดและความซับซ้อนเช่นกัน แนวคิดไมโครฟรอนต์เอนส์ได้ปรากฏขึ้นเป็นวิวัฒนาการทางตรรกของสถาปัตยกรรมฝั่งฟรอนต์เอนด์ของเว็บแอปพลิเคชัน คล้ายคลึงกับไมโครเซอร์วิส แนวคิดทั้งสองมีประโยชน์ต่อการพัฒนาระบบพร้อมกัน นอกเหนือจากสมรรถนะที่เพิ่มขึ้นอันเป็นผลมาจากการแบ่งแยกแอปพลิเคชันขนาดใหญ่ออกเป็นส่วนเล็กๆ วิทยานิพนธ์นี้นำเสนอการประยุกต์ใช้ไมโครฟรอนต์เอนส์สำหรับการพัฒนาเว็บไซต์โปรแกรมสืบค้นทางกฎหมายเป็นกรณีศึกษา เนื่องจากโปรแกรมสืบค้นโดยปกติทั่วไปจัดเป็นโครงการซอฟต์แวร์ขนาดใหญ่ การพัฒนาเป็นซิงเกิลเพจแอปพลิเคชันจึงมีแนวโน้มที่จะไม่สามารถรองรับงานที่มีปริมาณมากขึ้น ระบบขยายตัวไม่ได้ดี และค่าใช้จ่ายบำรุงรักษาสูง การออกแบบฝั่งไคลแอนต์บนพื้นฐานไมโครฟรอนต์เอนด์ผนวกกับเทคโนโลยีแบ็กเอนด์แบบไมโครเซอร์วิสได้ถูกนำเสนอในงานวิจัยนี้ สำหรับการประเมินผล ความต้องการใหม่ของหน้าเว็บ *รายละเอียดเอกสาร* ได้ถูกพัฒนาขึ้นด้วยแองกูลาร์และเพิ่มเข้าไปในระบบปัจจุบันที่เป็นซิงเกิลเพจแอปพลิเคชัน เพื่อเปรียบเทียบกับการพัฒนาฟีเจอร์ดังกล่าวด้วยไมโครฟรอนต์เอนด์ ตัววัดทั้งสามที่ถูกเลือกสำหรับการประเมินสมรรถนะประกอบด้วย ความขึ้นต่อกันของคอมโพเนนต์ เวลาการพัฒนา และเวลาการทดสอบ ผลลัพธ์ค่าการวัดรายงานค่าตัววัดทั้งสามตัวที่ลดลงเมื่อพัฒนาด้วยไมโครฟรอนต์เอนด์ อย่างไรก็ตาม ทีมงานมีความรู้สึกว่าต้องใช้ความพยายามมากขึ้นกับการพัฒนาแบบใหม่ด้วยไมโครฟรอนต์เอนด์

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อนิสิต .....

ปีการศึกษา 2563 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6270085221 : MAJOR SOFTWARE ENGINEERING

KEYWORD: Micro-frontends, Microservices, Architecture, Web development

Nattaporn Noppadol : Application of Microfrontends to Reengineering Single Page Application. Advisor: Assoc. Prof. Yachai Limpiyakorn, Ph.D.

Software systems are traditionally separated into front and rear architecture. The backend is responsible for data processing on server side, while the frontend accounts for interaction between clients and a system. Among several today backend architectural approaches, Microservices is an alternative most suitable for scalable systems. Meanwhile, the client-side applications are also growing with size and complexity. The concept of Micro-frontends has recently emerged as logical evolution of architecture for frontend side of web applications. Similar to Microservices, both concepts benefit concurrent development in addition to greater performance that result from partitioning large applications into smaller parts. This thesis presents an application of Micro-frontends to Thai legal search engine web development as a case study. Since search engines are typically large-scale software projects, single page applications tend to bloat up, not well-scaled, and costly maintain. A design of the client-side based on Micro-frontends combined with the backend technology, Microservices, is introduced in this research. For evaluation, the new requirement of Document Detail webpage is implemented with Angular and added to the current system considered as single page application. The new feature is also implemented with Micro-frontends for comparison. The three measures: component dependency, development time, and test time are selected for performance evaluation. The results report that the measurement values of component dependency, development time, and test time all decrease with Micro-frontends implementation. However, the team feel that they spent more effort with the novel Micro-frontend development.

Field of Study:	Software Engineering	Student's Signature .....
Academic Year:	2020	Advisor's Signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจสำเร็จลุล่วงไปได้หากไม่ได้รับความอนุเคราะห์จากรองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์ ที่กรุณาเสียสละเวลาอันมีค่ารับเป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งได้คอยช่วยเหลือ ให้คำปรึกษา ตรวจสอบ ให้คำแนะนำ และให้ความรู้ในการเขียนวิทยานิพนธ์นี้จนสำเร็จและผ่านพ้นไปได้ด้วยดี ข้าพเจ้าขอกราบระลึกถึงพระคุณของรองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์ ไว้ ณ โอกาสนี้ ตลอดจนขอขอบพระคุณ ผศ. ดร.สุกรี สินธุภิญโญ และอ. ดร.ภาสกร อภิรักษ์วรพินิต ที่กรุณาเสียสละเวลาอันมีค่ารับเป็นประธาน และกรรมการสอบวิทยานิพนธ์ฉบับนี้

ขอขอบพระคุณมารดาและครอบครัวของข้าพเจ้าที่คอยให้ความช่วยเหลือ เป็นกำลังใจที่ดีเสมอ รวมถึงขอขอบคุณนายทวีศักดิ์ ชูศรี ขอบคุณที่สละเวลาให้ความช่วยเหลือทั้งในด้านข้อมูล ให้คำปรึกษาในการจัดทำวิทยานิพนธ์ฉบับนี้ และให้กำลังใจที่ดีกับข้าพเจ้าเสมอมา

ขอบพระคุณเพื่อนร่วมเรียนปริญญาโท และมิตรสหาย ที่คอยให้กำลังใจ ให้การสนับสนุนและความช่วยเหลือในด้านต่างๆ

สุดท้ายนี้ข้าพเจ้าขอขอบพระคุณผู้ที่เกี่ยวข้องท่านอื่นๆที่ไม่ได้กล่าวมาในข้างต้น ซึ่งมีผลให้วิทยานิพนธ์นี้สำเร็จลุล่วงไปได้ด้วยดี ข้าพเจ้าหวังเป็นอย่างยิ่งว่า วิทยานิพนธ์ฉบับนี้จะสามารถสร้างคุณประโยชน์ให้แก่ผู้ที่สนใจศึกษาได้ไม่มากก็น้อย และหากวิทยานิพนธ์ฉบับนี้มีข้อผิดพลาดประการใดต้องขออภัยมา ณ ทีนี้ ด้วย

ณัฐพร นพปฎล

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

ในหลายปีที่ผ่านมามีการนำเสนอสถาปัตยกรรมไมโครเซอร์วิส เพื่อแก้ปัญหาความซับซ้อน และเพิ่มความสามารถในการขยาย (Scalability) ของแบ็กเอนด์ (Backend) และโครงสร้างพื้นฐาน (Infrastructure) ของระบบ ซึ่งเป็นประโยชน์อย่างมาก จึงมีการนำเทคนิคนี้มาใช้กับงานพัฒนาซอฟต์แวร์ส่วนหน้า (Front end) เรียกว่า ไมโครฟรอนต์เอนส์ (Micro Frontends) ทั้งนี้ มีการกล่าวถึง ไมโครฟรอนต์เอนส์ครั้งแรกในปี 2016 โดย ThoughtWorks Technology Radar [1] ใจความโดยรวมได้กล่าวถึงปัญหาความซับซ้อนในการทำงานของแต่ละทีม และปัญหาในการสร้างเว็บแอปพลิเคชันขนาดใหญ่ที่พัฒนาฟรอนต์เอนด์ด้วยสถาปัตยกรรมแบบโมโนลิทิก (Monolithic Architecture) ที่มีขนาดใหญ่และซับซ้อน จึงมีการนำเสนอแนวคิดไมโครฟรอนต์เอนส์ ขึ้นมาด้วยสมมติฐานว่า หากแบ่งแต่ละทีมให้ทำงานแยกกันโดยมีองค์ประกอบตั้งแต่ฟรอนต์เอนด์จนถึงแบ็กเอนด์ แบบตั้งแต่ต้นจนจบ (end-to-end) น่าจะลดความซับซ้อนและปัญหาลงได้ ต่อมา Michael Geers [2, 3] ได้มีการขยายแนวคิดและนำเสนอหลักการไมโครฟรอนต์เอนส์ โดยได้มีการชี้ให้เห็นถึงปัญหาของฟรอนต์เอนด์แบบโมโนลิทิก ซึ่งการแก้ปัญหานั้น ทำได้ด้วยการนำไมโครฟรอนต์เอนส์เข้ามาประยุกต์ใช้ อีกทั้งยังมีขั้นตอนแนวคิดและวิธีเริ่มต้นเพื่อเป็นแนวปฏิบัติ (Guideline) สำหรับผู้เริ่มต้นในการพัฒนาด้วยแนวคิดแบบไมโครฟรอนต์เอนส์ หลังจากนั้นได้มีการศึกษาอย่างแพร่หลาย ยกตัวอย่างเช่น ในช่วงต้นปี 2019 Ö Zafer [4] ได้ทำการเผยแพร่บทความถึงความเข้าใจและแนวคิดของการพัฒนาเว็บแอปพลิเคชันด้วยไมโครฟรอนต์เอนส์โดยขยายแนวคิดจากสิ่งที่ Geers ได้เผยแพร่ไว้และได้นำเสนอปัญหาของการพัฒนาด้วยกระบวนการนี้ รวมถึงแนวทางในการแก้ไขของปัญหา ต่อมาในปีเดียวกัน Yang และคณะ [5] ได้นำเสนอแนวคิดในการพัฒนาระบบการจัดการข้อมูล (Content Management System) ด้วยสถาปัตยกรรมแบบไมโครฟรอนต์เอนส์ (Micro Frontends Architecture) โดยพัฒนาฟรอนต์เอนด์ด้วย Mooa Framework ซึ่งเป็นหนึ่งใน Micro Frontend Framework ที่ได้รับความนิยม หลังจากนั้น Pavlenko และคณะ [6] ได้ทำการศึกษารวบรวมงานวิจัยพร้อมทั้งแสดงถึงข้อดี-ข้อเสียและแนวทางแก้ไข รวมถึงแนวทางปฏิบัติของการนำไมโครฟรอนต์เอนส์มาใช้ร่วมกันในการพัฒนาเว็บแอปพลิเคชัน

งานวิจัยนี้ ผู้วิจัยได้นำเสนอการประยุกต์ใช้แนวคิดไมโครฟรอนต์เอนส์สำหรับการออกแบบโครงสร้างและพัฒนาซอฟต์แวร์ฝั่งไคลแอนต์ (Client-side) ของเว็บไซต์โปรแกรมสืบค้นทางกฎหมาย

(Legal Search Engine web site) ซึ่งมีระบบขนาดกลาง เดิมพัฒนาขึ้นด้วยเทคโนโลยีเก่าอย่าง Single Page Application (SPA) เพื่อเป็นแนวทางในการประยุกต์ใช้ปรับโครงสร้างใหม่ (Reengineering) ในการพัฒนาเว็บแอปพลิเคชันส่วนหน้าด้วยสถาปัตยกรรมไมโครฟรอนต์เอนส์

1.2 วัตถุประสงค์

เพื่อวิเคราะห์ข้อดีข้อด้อยของระบบกรณีศึกษาที่ใช้ส่วนฟรอนต์เอนด์เดิมพัฒนาแบบซิงเกิลเพจแอปพลิเคชัน เปรียบเทียบกับระบบกรณีศึกษาที่ปรับโครงสร้างใหม่ของส่วนฟรอนต์เอนด์ด้วยสถาปัตยกรรมไมโครฟรอนต์เอนส์

1.3 ขอบเขตการดำเนินงาน

1. ปรับโครงสร้างและพัฒนาเฉพาะซอฟต์แวร์ส่วนหน้าของ Lawphin ด้วยแนวคิดไมโครฟรอนต์เอนส์ที่นำเสนอในงานวิจัยนี้
2. พัฒนา Document Service ตามความต้องการใหม่ เฉพาะซอฟต์แวร์ส่วนหน้าของ Lawphin ด้วย Angular version 1 ตามรูปแบบโครงสร้างดั้งเดิมแบบโมโนลิทิก
3. พัฒนา Document Service ตามความต้องการใหม่ เฉพาะซอฟต์แวร์ส่วนหน้าของ Lawphin ด้วยแนวคิดไมโครฟรอนต์เอนส์
4. ทดสอบระบบเว็บแอปพลิเคชันที่พัฒนาขึ้นในงานวิจัยนี้บนเบราว์เซอร์กูเกิลโครม โดยเชื่อมต่อไปยังระบบหลังบ้านไมโครเซอร์วิสเดิมที่มีอยู่แล้วด้วย REST API
5. เครื่องมือที่ใช้ คือ Vue JS, Angular Js และ React JS
6. ประเมินงานวิจัยด้วยการทดสอบแบบถดถอย และทดสอบความเร็วในการโหลดข้อมูลของทั้งสถาปัตยกรรมแบบโมโนลิทิกและสถาปัตยกรรมแบบไมโครฟรอนต์เอนส์เพื่อวิเคราะห์เปรียบเทียบ ข้อจำกัด ข้อดีและข้อเสีย

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาค้นคว้าทฤษฎีและงานวิจัยที่เกี่ยวข้อง
2. ออกแบบกรอบงานในการเปลี่ยนฟรอนต์เอนส์เป็นไมโครฟรอนต์เอนส์
3. พัฒนาระบบตัวอย่าง
4. ทดสอบกระบวนการในการเปลี่ยนเป็นสถาปัตยกรรมไมโครฟรอนต์เอนส์
5. วิเคราะห์ข้อมูลและสรุปผลที่ได้จากการทดลอง
6. เผยแพร่งานวิจัย
7. จัดทำเอกสารวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ได้แนวทางการประยุกต์ใช้ไมโครฟรอนต์เอนส์สำหรับการพัฒนาซอฟต์แวร์ส่วนหน้าของซิงเกิลเพจเว็บแอปพลิเคชัน

1.6 ลำดับการจัดเรียงเนื้อหาในวิทยานิพนธ์

เนื้อหาในวิทยานิพนธ์แบ่งออกเป็น 6 บท ได้แก่ บทที่ 1 บทนำ อธิบายถึงที่มาและความสำคัญของปัญหา วัตถุประสงค์ของงานวิจัย ขอบเขตงานวิจัย ประโยชน์ที่คาดว่าจะได้รับ และผลงานที่ได้รับการตีพิมพ์จากวิทยานิพนธ์ บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง บทที่ 3 แนวคิดและวิธีการวิจัย บทที่ 4 การออกแบบและพัฒนาเครื่องมือ และบทที่ 5 การทดสอบและการวิเคราะห์ผล

1.7 ผลงานที่ได้รับการตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์ฉบับนี้ได้รับการตีพิมพ์บทความทางวิชาการ

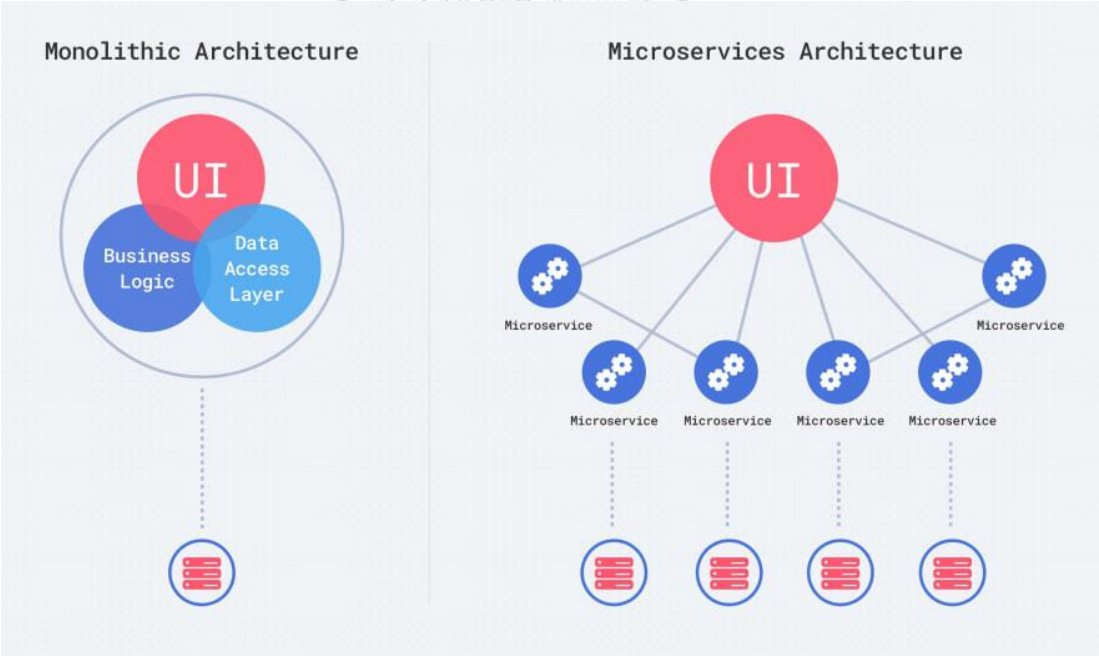
Noppadol, N. & Limpiyakorn, Y. (2021). APPLICATION OF MICRO-FRONTENDS TO LEGAL SEARCH ENGINE WEB DEVELOPMENT. iCaste 9th International Conference on IT Convergence and Security 2021, Virtual Conference.

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1. ทฤษฎีที่เกี่ยวข้อง

2.1.1. ไมโครเซอร์วิส (Microservice)

ไมโครเซอร์วิส หรือ สถาปัตยกรรมไมโครเซอร์วิส (Microservice Architecture) เป็นสถาปัตยกรรมในการออกแบบซอฟต์แวร์ที่ถูกออกแบบมาเพื่อแก้ไขข้อเสียของสถาปัตยกรรมโมโนลิทิก โดยแนวคิดของการออกแบบไมโครเซอร์วิสนั้นจะขึ้นอยู่กับแบบจำลองการปรับขนาดที่เรียกว่าสเกลคิวบ์ [7] แต่ละบริการสามารถปรับขนาดได้ ส่งผลให้แอปพลิเคชันมีการปรับขนาดอย่างมีประสิทธิภาพมากขึ้น โดยแยกการทำงานของเซอร์วิส โดยสามารถแยกตามบริการหรือตามฟังก์ชันการทำงานของซอฟต์แวร์ ดังภาพที่ 1 ซึ่งแตกต่างจากสถาปัตยกรรมโมโนลิทิก ที่ระบบและฟังก์ชันการทำงานทุกอย่างจะรวมอยู่ในกลุ่มก้อนเดียวกัน ใช้ฐานข้อมูลร่วมกัน



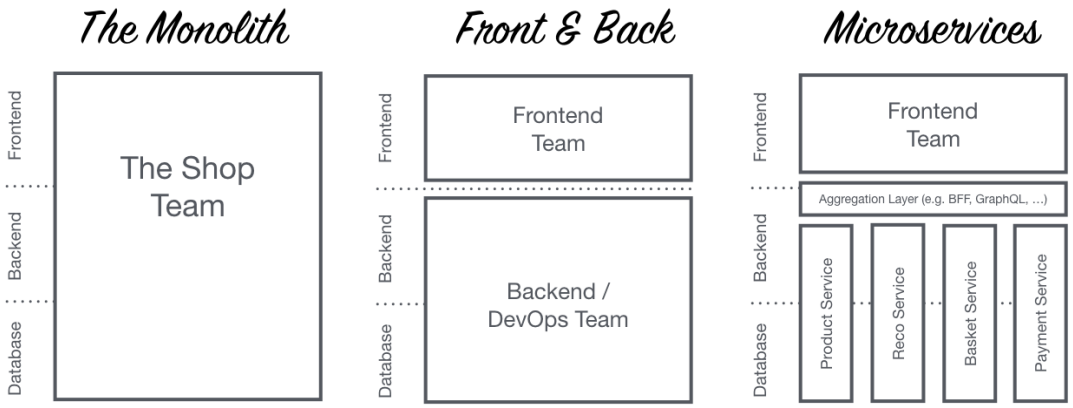
ภาพที่ 1 เปรียบเทียบการออกแบบระหว่างโมโนลิทิกกับไมโครเซอร์วิส [8]

สถาปัตยกรรมไมโครเซอร์วิสสามารถแก้ไขปัญหาในเรื่องของความซับซ้อนของระบบขนาดใหญ่ได้ดี เนื่องจากมีการแบ่งบริการออกมาเป็นส่วนย่อยๆ ทำให้การพัฒนาสามารถปรับใช้งานได้ทันที ในแต่ละบริการมีกระบวนการที่ไม่ซ้ำกันและสื่อสารผ่านกลไกที่กำหนดไว้อย่างดีและมีน้ำหนักเบา ซึ่งมักเป็น HTTP Resource API [9] และในแต่ละบริการจะมีฐานข้อมูลเป็นของตัวเอง เพื่อไม่ให้เกิด

การกระทบกับฐานข้อมูลของบริการชุดอื่นๆ การแยกบริการออกเป็นส่วนย่อยๆนั้น ทำให้ความเร็วในการพัฒนาซอฟต์แวร์ง่ายขึ้นและเร็วขึ้นจากการที่สามารถแบ่งทีมในการพัฒนาแต่ละส่วนได้ โดยที่แต่ละทีมสามารถเลือกภาษาหรือเครื่องมือที่จะนำมาใช้ในการพัฒนาซอฟต์แวร์ได้อย่างอิสระ ในส่วนที่เกี่ยวกับฐานข้อมูลนั้นไม่จำเป็นต้องใช้ฐานข้อมูลเดียวกันเนื่องจากบริการหนึ่งเป็นเจ้าของข้อมูลบางส่วนและสามารถสร้างฟังก์ชัน ทดสอบระบบและให้ผู้ใช้งานใช้ได้ โดยไม่มีผลกระทบใด ๆ กับฟังก์ชันที่มีอยู่

2.1.2 ไมโครฟรอนต์เอนส์ (Micro Frontends)

เกิดจากการพัฒนาทคนิคในการพัฒนาฟรอนต์เอนด์ เพื่อปรับปรุงประสิทธิภาพของทีมในการพัฒนา ฟรอนต์เอนส์โดยได้มีการนำแนวคิดของไมโครเซอร์วิสจากฝั่งแบ็กเอนด์เพื่อให้สามารถแยกการพัฒนาจากชิ้นใหญ่รวมกันหรือที่เรียกว่า โมโนลิท (Monolith) (ภาพที่ 2) ให้แบ่งออกมาเป็นชิ้นส่วนเล็กๆ ย่อยลงมาได้ในระดับคอมโพเนนต์ ซึ่งเป็นการทำงานที่แยกออกจากกันแบบคนละ layer อย่างชัดเจน เพื่อให้ง่ายต่อการจัดการและการบำรุงรักษามากกว่าการพัฒนาในรูปแบบเดิม

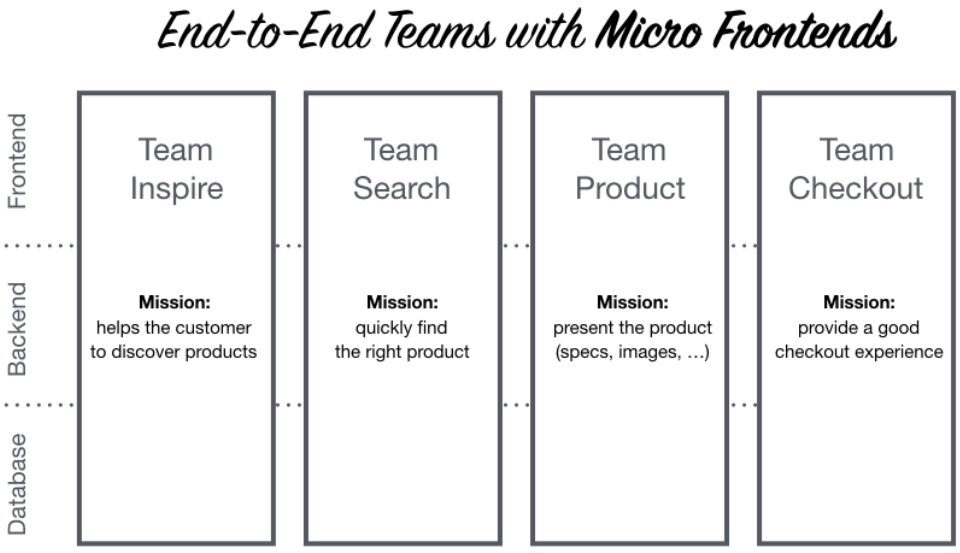


ภาพที่ 2 รูปแบบการพัฒนาแบบโมโนลิท [3]

โครไมฟรอนต์เอนส์ เป็นคำที่ถูกกำหนดขึ้นโดย ThoughtWorks Technology Radar [1] ได้ให้นิยามและแนวคิดเกี่ยวกับไมโครฟรอนต์เอนส์ โดยเป็นการประยุกต์แนวคิดการพัฒนาไมโครเซอร์วิสทางฝั่งฟรอนต์เอนด์ ปัจจุบันไมโครฟรอนต์เอนส์ได้ถูกนำไปใช้โดยเริ่มต้นจากการเปลี่ยนมุมมองการพัฒนา ซึ่งมองว่า ฟีเจอร์เล็กๆหนึ่งฟีเจอร์ เป็นงานหนึ่งชิ้นที่ทำงานหนึ่งอย่างโดยหนึ่งทีมพัฒนา ดังภาพที่ 3 จะเห็นได้ว่าได้มีการแบ่งออกเป็นทีมย่อยตามเซอร์วิสที่สร้างขึ้น ทำให้กรณีฟรอนต์เอนด์ต้องการเรียกใช้งาน ในแต่ละชั้นจะไม่ผูกกัน มีความเป็นอิสระต่อกัน ในการวางโครงสร้างแบบไมโครฟรอนต์เอนส์นี้จะช่วยทำให้เกิดความคล่องตัวของแต่ละทีมในการพัฒนามากขึ้น

ความแตกต่างที่เห็นได้ชัดระหว่างรูปแบบการพัฒนาตามภาพที่ 3 กับรูปแบบการพัฒนาแบบแรกในตัวอย่างภาพที่ 2 คือ การแบ่งทีมพัฒนา มีการแบ่งทีมพัฒนาตามบริการ (Service) ที่แตกต่างกัน โดยแต่ละทีมจะมีเป้าหมายที่แตกต่างกันไปตามเซอร์วิสที่ต่างกัน ซึ่งในแต่ละทีมพัฒนาจะมีสิทธิ์และความคล่องตัวในการดูแลพัฒนาฟีเจอร์ของผลิตภัณฑ์ได้ตั้งแต่ต้นจนจบงาน ทำให้ทีมได้เห็นทุกชั้นในการทำงานตั้งแต่ฟรอนต์เอนด์ถึงแบ็กเอนด์ มีส่วนร่วมและรับทราบผลลัพธ์จากการพัฒนาที่ชัดเจน

แนวคิดหลักในการพัฒนาไมโครฟรอนต์เอนส์ คือ ไม่มีความจำเป็นในการกำหนดกรอบงานฟรอนต์เอนด์ (Frontend framework) ในการพัฒนา เพียงแต่ละทีมแยกกันพัฒนาคอมโพเนนต์ และให้คอมโพเนนต์ของแต่ละทีมนั้นสามารถติดต่อ เชื่อมโยงและทำงานร่มกันได้ ส่งผลให้แต่ละทีมมีอิสระในการเลือกกรอบงาน (Framework) หรือเครื่องมือที่ใช้ในการนำมาพัฒนา ซึ่ง Michael Geers [2, 3] ได้กำหนดแนวคิดหลักในการพัฒนาไมโครฟรอนต์เอนส์ ดังนี้



ภาพที่ 3 รูปแบบการพัฒนาแบบไมโครฟรอนต์เอนส์ [3]

- ความเป็นอิสระในด้านเทคโนโลยี ทีมควรตัดสินใจในการเลือก เทคโนโลยีของตนโดยไม่มีผลกระทบใด ๆ จากทีมอื่น ๆ
- การแยกรหัส ไม่ควรแชร์รันไทม์ (Runtime) และเว็บแอปพลิเคชัน ควรมีอยู่ในตัวเอง
- คำนำหน้าทีม ควรใส่คำนำหน้าชิ้นส่วนทั้งหมดที่ไม่สามารถแยก ได้เพื่อเป็นการหลีกเลี่ยงการชนกันระหว่างทีมอื่น

- ค่ากำหนด API ของเบราว์เซอร์ (Browsers) ดั้งเดิม ควรใช้ API ของเบราว์เซอร์เริ่มต้นสำหรับการพัฒนาวิธีการสื่อสารระหว่างแอปพลิเคชันที่กำหนดเอง
- ความยืดหยุ่น คุณลักษณะนี้ควรมีประโยชน์แม้ในกรณีที่จาวาสคริปต์ล้มเหลวหรือไม่ทำงาน

จากแนวคิดของ Michael Geers [2, 3] ในการพัฒนาไมโครฟรอนต์เอนส์นั้น ในปัจจุบันมีไม่กี่วิธีในการพัฒนา ซึ่งวิธีที่เห็นชัดคือการใช้ Web components เนื่องจาก web components นั้นเป็นพื้นฐานของเบราว์เซอร์ (Native browser) สามารถใช้งานได้กับบราวเซอร์ทุกตัว และมีเครื่องมือที่ใช้สำหรับสร้างมากมาย เช่น React, Vue, Angular นั้นสามารถใช้เว็บคอมโพเนนต์ในการคอมไพล์ได้เช่นกัน ดังนั้นหากเราต้องการใช้การพัฒนาแบบไมโครฟรอนต์เอนส์ เราจำเป็นต้องปรับคอมโพเนนต์ทั้งหมดให้กลายเป็นรูปแบบของเว็บคอมโพเนนต์ก่อน

2.1.2. เว็บคอมโพเนนต์ (Web Components)

เว็บคอมโพเนนต์เป็นมาตรฐานในการสร้างเว็บไซต์ที่ถูกสร้างขึ้นโดยทีม W3C ในปี 2014 [10] ซึ่งถือเป็นฟีเจอร์ใหม่ของเว็บเบราว์เซอร์ โดยหลักการทำงานของ Web Components นั้นจะสามารถสร้างหรือปรับแต่งโครงสร้างของภาษา HTML เพื่อนำมาใช้ในเว็บแอปพลิเคชัน (Web Application) ได้ โดยที่ Web Components มีองค์ประกอบ 4 อย่าง ประกอบด้วย

- **แผ่นแบบ HTML (HTML Template)** เป็นส่วนที่ทำการประกาศภาษากำกับเพิ่ม (Markup Language) ของคอมโพเนนต์ (Components) โดยมักจะทำการห่อหุ้มโครงสร้างของภาษา HTML ต่างๆ ด้วยแผ่นแบบ tag ซึ่งมีรูปแบบการเขียนดังนี้ “<template>” ตัวอย่างเช่น เพแผ่นแบบที่ประกอบด้วย h3 และ div ดังภาพที่ 4

```
<template>
    <h3>Hello</h3>
    <div class="red">This is sample template.</div>
</template>
```

ภาพที่ 4 ตัวอย่างการใช้แผ่นแบบ HTML tag

- **Shadow DOM** คือ DOM (Document Object Model) ประเภทหนึ่งที่ถูกซ่อนไว้จากการแสดงผล (Render) ของเว็บไซต์ในตอนเริ่มต้นของการทำงาน แต่จะถูกแสดงผลในภายหลังด้วยชุดคำสั่งของจาวาสคริปต์ (+Script) ทำให้ไม่ได้

รับผลกระทบจากการแสดงผลของ CSS (Cascading Style Sheets) และจาวาสคริปต์ของหน้าเว็บไซต์นั้นๆ

- **Custom Elements** เป็นการสร้าง HTML element ขึ้นมาใหม่ เพื่อให้สามารถเรียกใช้งานได้โดยง่ายและสามารถนำไปใช้ซ้ำได้ในส่วนอื่นๆ ของเว็บไซต์ได้ โดยมีข้อกำหนดคือ HTML element นั้นต้องแบ่งคำด้วยเครื่องหมาย “-” โดยคณะผู้วิจัยจะใช้วิธีนี้ในการรวบรวมไมโครฟรอนต์เอนส์ของแต่ละทีมนำมาแสดงผลร่วมกัน
- **HTML Imports** คือ ความสามารถในการนำไฟล์ HTML ที่ถูกสร้างด้วย HTML Template, Shadow DOM หรือ Custom Elements ไปใช้ต่อในไฟล์ HTML สำหรับหน้าเว็บไซต์อื่นๆเช่น เมื่อสร้างคอมโพเนนต์ (Components) ชื่อ <foo-bar> ไว้ในไฟล์ชื่อ foo-bar.html จะสามารถเรียกใช้ไฟล์ foo-bar.html ได้โดยการนำเข้า (Import) ดังภาพที่ 5

```
<head>
    <link rel="import" href="foob-bar.html">
</head>
<body>
    <foob-bar></foob-bar>
</body>
```

ภาพที่ 5 การนำเข้าไฟล์ HTML

2.1.3 ซิงเกิลเพจแอปพลิเคชัน (Single Page Application:SPA)

เป็นเทคนิคในการออกแบบโครงสร้างเว็บแอปพลิเคชัน โดยจะมีตัวกลางหนึ่งตัวมาเป็นจุดศูนย์กลางในการเรียกใช้งานส่วนต่างๆ เพื่อนำมาแสดงผลของตัวเว็บแอปพลิเคชัน ซึ่งรูปแบบการทำงานลักษณะนี้ จะช่วยลดขั้นตอนในการเรียกใช้งานหน้าเว็บแอปพลิเคชัน เพราะสามารถนำแสดงผลข้อมูลได้โดยที่ไม่ต้องทำการรีเฟชร (Refresh) เว็บแอปพชิเคชันใหม่ เนื่องจากกระบวนการแสดงผลจะเกิดขึ้นที่ฝั่งไคลเอนต์ทั้งหมด

- **ข้อดีของซิงเกิลเพจแอปพลิเคชัน**
  1) หน้าเว็บจะทำงานเร็วมากเพราะโหลด Assets ต่างๆครั้งเดียว เนื่องจากการเปลี่ยน URL แต่ละครั้งจากผู้ใช้งาน ระบบไม่จำเป็นที่จะต้องไปทำงานที่แบ็กเอนด์ แต่ระบบจะทำงานที่ ไคลเอนต์เพื่อทำการแสดงผลบนหน้าเว็บแอปพลิเคชัน

2) ไม่กินทรัพยากร เนื่องจากใช้ Bandwidth น้อย เพราะระบบจะไม่ได้ส่ง HTML ทั้งหมดไปแบ็กเอนด์ ในการเปลี่ยน URL แต่ละครั้ง แต่จะทำการเรียก API แทน เพื่อให้ได้ผลลัพธ์ที่เพียงพอต่อการแสดงผลบนหน้าเว็บแอปพลิเคชัน.

- **ข้อเสียของซิงเกิลเพจแอปพลิเคชัน**
  1) เบราว์เซอร์ทำงานหนักขึ้นเพราะการประมวลผลเกิดขึ้นที่ไคลเอนต์ ซึ่งอาจจะเกิดปัญหาเรื่องของประสิทธิภาพในการใช้งาน การโหลดเว็บแอปพลิเคชันครั้งแรกอาจใช้เวลานาน
  2) ผู้พัฒนาต้องแยกแบ็กเอนด์ออกจากฟรอนต์เอนด์ เพื่อการขยายที่ดี
  3) ผู้พัฒนาต้องตรวจสอบความปลอดภัยของข้อมูลให้ดี เนื่องเป็นการทำงานเพียงหน้าเดียว

2.2. งานวิจัยที่เกี่ยวข้อง

2.2.1. Research and Application of Micro Frontends [5]

งานวิจัยนี้ได้นำเสนอการนำแนวคิดของไมโครเซอร์วิสมาใช้ในการพัฒนาฟรอนต์เอนส์ โดยมีจุดประสงค์ในการออกแบบเพื่อพัฒนาระบบการจัดการข้อมูล (Content Management System) ให้เป็นไมโครฟรอนต์เอนส์ จากเดิมที่ใช้เทคโนโลยี Single Page Applicaiton (SPA) ในการพัฒนาเว็บไซต์ ซึ่งมีการเรียกใช้ REST API ในการดึงข้อมูลจากแบ็กเอนด์ ต่อมาได้พบว่า การพัฒนาด้วย SPA นั้นส่งผลให้การขยายเว็บไซต์เพื่อบำรุงรักษาในอนาคตเป็นเรื่องยาก โดยเฉพาะเมื่อเว็บไซต์นั้นมีฟรอนต์เอนด์ที่ซับซ้อนและเรียกใช้งานแบ็กเอนด์แบบไมโครเซอร์วิสอยู่ ซึ่งในด้านแบ็กเอนด์นั้นสถาปัตยกรรมไมโครเซอร์วิสช่วยให้เกิดความเชื่องโยงกันในแต่ละบริการ ทำให้เว็บไซต์พัฒนาทดสอบ และปรับใช้ได้โดยอิสระ เมื่อนำแนวคิดนี้มาปรับใช้ในส่วนของฟรอนต์เอนด์ ทำให้เกิดการแบ่งทีมในการพัฒนา โดยแต่ละทีมจะมีเป้าหมายในการพัฒนาที่แตกต่างกัน ซึ่งแต่ละทีมต่างก็พัฒนาตั้งแต่แบ็กเอนด์ไปจนถึงฟรอนต์เอนส์ และได้เลือกใช้ Mooa Framework ในการพัฒนา ซึ่งเป็นกรอบงาน (Framework) ที่ให้บริการสำหรับ Angular และเหมาะสมกับการพัฒนาฟรอนต์เอนด์ ทำให้เว็บไซต์สามารถเรียกใช้ Angular ได้หลายที่ในเวลาเดียวกัน อย่างไรก็ตาม Mooa Framework ไม่สามารถพัฒนาในแทคโนโลยีที่แตกต่างกันได้ ทำให้ต้องพิจารณาการใช้จาวาสคริปต์เพื่อหลีกเลี่ยงความขัดแย้ง ในการพัฒนา CSS

2.2.2. Micro-frontends: application of microservices to web front-ends [6]

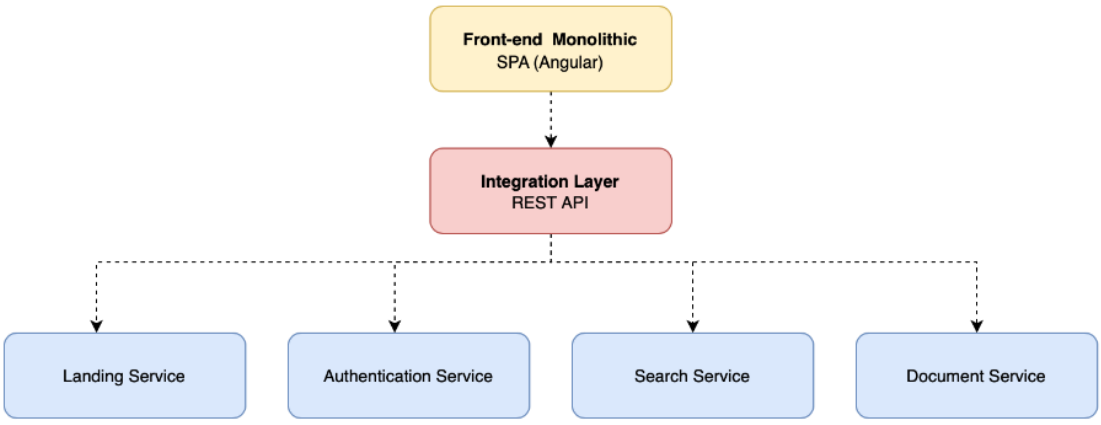
คณะผู้วิจัยได้ทำการศึกษาสำรวจแนวคิดของไมโครฟรอนต์เอนส์ในการพัฒนาเว็บแอปพลิเคชัน โดยได้มีการนำเสนอกรณีศึกษาเกี่ยวกับการนำแนวคิดไปใช้ในการพัฒนาจริง และมีการวิเคราะห์หา

ประเด็นที่เกิดขึ้นระหว่างการพัฒนา โดยตัวอย่างของกรณีศึกษา คือการพัฒนาเว็บแอปพลิเคชันชื่อ Education Hub System ซึ่งเป็นระบบที่ทำการรวบรวมบทเรียนออนไลน์จากหลายๆ แหล่งเพื่อมาให้ ผู้ใช้ ได้ใช้งานและค้นหาบทเรียนออนไลน์เพียงที่เดียว สำหรับภาพรวมในการพัฒนานั้น คณะผู้วิจัยได้ทำการออกแบบระบบด้วย Multi-Layer Architect โดยแบ่งออกเป็น Present Layer, Business Logic Layer, Data Layer ซึ่งจะนำไมโครฟรอนต์เอนส์มาใช้ใน Present Layer โดยได้ทำการแบ่งไมโครฟรอนต์เอนส์ออกเป็น 4 ไมโครฟรอนต์เอนส์ประกอบด้วย Admin Frontend, Account Frontend, Search Frontend, Data Frontend ภายใต้แนวคิดในการแบ่ง “แอปพลิเคชันฟรอนต์เอนด์ขนาดเล็กที่เป็นอิสระซึ่งประกอบเป็นแอปพลิเคชันที่ใหญ่กว่า” และไมโครฟรอนต์เอนส์ทั้งหมดสื่อสารกับ Business Logic Layer โดยใช้จุดเข้าเดียว (Single entry point) คือ API เกตเวย์ สำหรับการพัฒนาไมโครฟรอนต์เอนส์ทั้ง 4 ไมโครฟรอนต์เอนส์นั้นทางคณะผู้วิจัยได้เลือกใช้ React ซึ่งเป็นหนึ่งใน Library แสดงผลที่นิยมใช้ในปัจจุบัน เพื่อทำหน้าที่ในการจัดการเรื่องตรรกะต่างๆ ในการแสดงผล สำหรับการจัดการ Routing ของแต่ละไมโครฟรอนต์เอนส์นั้น ได้เลือกใช้ React Router เพื่อเป็นการใช้งานร่วมกันกับ React ได้อย่างมีประสิทธิภาพ และสุดท้ายทางคณะผู้วิจัยได้ใช้ SingleSPA ซึ่งเป็นตัวควบคุมไมโครฟรอนต์เอนส์ เช่นการเลือกว่าจะนำไมโครฟรอนต์เอนส์ตัวไหนมาแสดงผล เป็นต้น

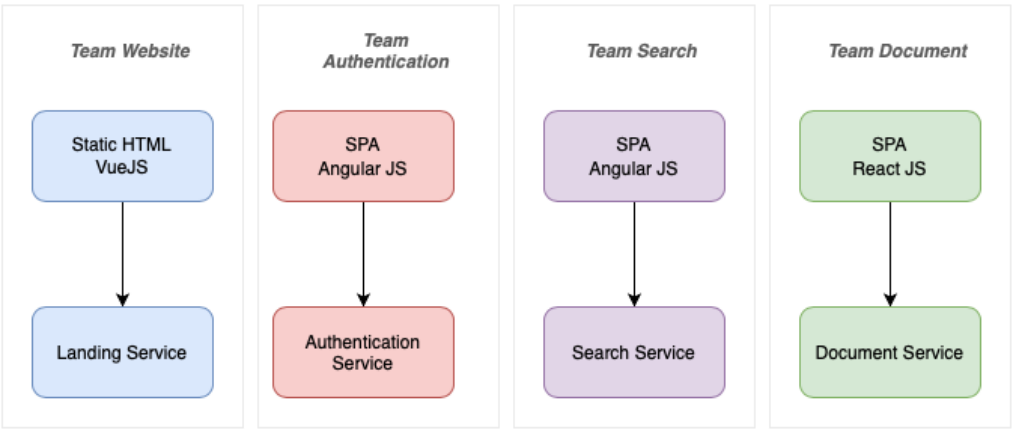
ภายหลังจากการพัฒนากรณีศึกษาทางคณะผู้วิจัยก็ได้ค้นพบว่า สำหรับโครงการขนาดเล็กหรือโครงการที่มีนักพัฒนาจำนวนน้อย สถาปัตยกรรมประเภทนี้ไม่เหมาะสม เนื่องจากความพยายามของนักพัฒนาส่วนใหญ่ใช้ไปกับ การสนับสนุนสถาปัตยกรรมแทนการพัฒนาคุณลักษณะและ เวลาโดยรวมของการพัฒนาเพิ่มขึ้น ยิ่งไปกว่านั้นความซับซ้อนที่เพิ่มขึ้นไม่สามารถจัดการโดยคนกลุ่มเล็กๆ แต่อย่างไรก็ดีหากเป็นการพัฒนาโปรแกรมที่มีขนาดกลางถึงใหญ่ น่าจะช่วยลดความซ้ำซ้นในการพัฒนาและสื่อสารของแต่ละโมดูล ได้ในระยะยาว

บทที่ 3
แนวคิดและวิธีการวิจัย

งานวิจัยนี้ได้นำเสนอการนำเทคโนโลยีไมโครฟรอนต์เอนส์มาใช้กับระบบค้นหาคำพิพากษาชื่อว่า Lawphin โดยเดิมทีนั้นส่วนแบ็กเอนด์ของ Lawphin ได้ถูกพัฒนาขึ้นด้วยแนวคิดแบบไมโครเซอร์วิส แต่ยังคงใช้ฟรอนต์เอนด์แบบโมโนลิทิกในรูปแบบของ Single Page Applicaiton (SPA) ดังภาพที่ 6 ซึ่ง SPA เป็นการใช้ภาษาจาวาสคริปต์ในการดึงข้อมูลมาแสดง โดยที่ผู้ใช้งานระบบเมื่อเข้าใช้งานระบบจะทำการดึงข้อมูลที่หน้าแรกแสดงดัชนี (Index) ของระบบ จากนั้นเมื่อผู้ใช้ระบบได้กระทำการบางอย่างกับระบบ เช่น การกดปุ่มเพื่อไปยังอีกหน้าหนึ่ง จาวาสคริปต์จะทำการดึงข้อมูลเนื้อหาเฉพาะจุดที่เกิดการเปลี่ยนแปลง อย่างไรก็ตามเทคโนโลยีดังกล่าวค่อนข้างมีข้อจำกัดในเรื่องการขยาย (scalability) เนื่องจากซอร์สโค้ดจะถูกรวบรวมไว้ที่เดียวกัน เมื่อมีการเปลี่ยนแปลงหรือเพิ่มฟีเจอร์ จะต้องทำการทดสอบทุกโมดูลที่มีในระบบ ซึ่งอาจมีผลกระทบต่อโมดูลอื่นได้ อีกทั้งในระบบเดิมมีการใช้กรอบงาน (Framework) ที่ค่อนข้างเก่าในการแสดงผล โดยกรอบงานที่ใช้นั้นคือ Angular version 1 ซึ่งในปัจจุบัน Angular ได้ถูกอัปเกรดไปจนถึง version 11 จึงเป็นที่มาในการพัฒนาและปรับปรุง โดยได้มีการนำสถาปัตยกรรมแบบไมโครฟรอนต์เอนส์มาประยุกต์ใช้ในการพัฒนาครั้งนี้ ในเบื้องต้น ผู้วิจัยขอยกตัวอย่างการใช้บริการ (Service) เป็นหลักในการแยกทีมเพื่อทำการแยกโมดูลของฟรอนต์เอนด์ในแต่ละส่วนออกมา ดังภาพที่ 7 ข้อดีคือ แต่ละทีมสามารถทำการสร้าง แก้ไข และทดสอบได้ตั้งแต่ฟรอนต์เอนด์ไปจนถึงแบ็กเอนด์ อีกทั้งในแต่ละทีมยังสามารถทำงานได้แบบอิสระต่อกัน กล่าวคือ แต่ละทีมไม่จำเป็นต้องเลือกใช้กรอบงาน หรือเทคโนโลยีเดียวกันในการพัฒนาผลิตภัณฑ์งาน (work product) ซึ่งเป็นแนวคิดที่ถูกขยายโดย Geers [2, 3] ตัวอย่างผลการแบ่งทีมในงานวิจัยที่นำเสนอจะได้ 4 ทีมหลัก ดังภาพที่ 7



ภาพที่ 6 โครงสร้างเดิมของเว็บแอปพลิเคชัน Lawphin



ภาพที่ 7 การออกแบบโครงสร้างเว็บแอปพลิเคชัน Lawphin ด้วยสถาปัตยกรรมไมโครฟรอนต์เอนส์

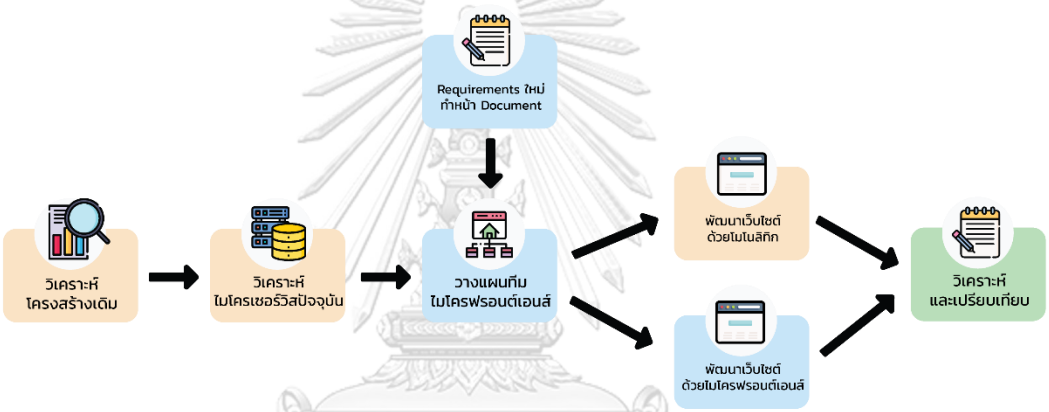
ทีมหลักต่างๆ ในภาพที่ 7 ประกอบด้วย

- ทีม Website เป็นทีมที่รับผิดชอบในการสร้าง Landing page หรือ หน้าแรกของเว็บแอปพลิเคชัน ซึ่งมีเป้าหมายเป็นหน้าแสดงข้อมูลโดยรวมของเว็บแอปพลิเคชัน
- ทีม Authentication เป็นทีมที่มีหน้าที่ในการสร้างคอมโพเนนต์สำหรับการเข้าสู่ระบบ (Login) โดยคอมโพเนนต์นี้ จะถูกเรียกใช้งานจากทุกหน้า
- ทีม Search เป็นทีมที่มีหน้าที่พัฒนาคอมโพเนนต์สำหรับการค้นหา (Search component) โดยคอมโพเนนต์นี้มีการเรียกใช้ข้อมูลจากไมโครเซอร์วิสที่เป็นข้อมูลหลัก (Master Data)
- ถ้าหากผู้ใช้กรอกข้อมูลสำหรับการค้นหาและผู้ใช้ได้ทำการกดค้นหาสำเร็จ คอมโพเนนต์ Main layout จะทำการเปลี่ยนข้อมูลไปแสดงผลการค้นหาข้อมูล (Search result)
- ทีม Document เป็นทีมที่มีหน้าที่ในการแสดงผลของข้อมูลฉบับเต็มโดยละเอียด

งานวิจัยนี้มุ่งเน้นการปรับโครงสร้างใหม่ส่วนฟรอนต์เอนด์ระบบเดิมที่ได้ปรับโครงสร้างส่วนแบ็กเอนด์ด้วยสถาปัตยกรรมไมโครเซอร์วิสแล้ว ดังนั้น ผู้วิจัยจึงแบ่งทีมโดยยึดหลักของไมโครเซอร์วิสเดิม โดยทีมจะเป็นตัวกำหนดโครงสร้างไมโครฟรอนต์เอนส์ เพื่อให้ในแต่ละทีมสามารถพัฒนาและทดสอบซอฟต์แวร์ได้ตั้งแต่ฟรอนต์เอนด์ไปจนถึงแบ็กเอนด์ หรือแบบ end-to-end โดยทั้ง 4 ทีม สามารถเลือกใช้เครื่องมือในการพัฒนาได้อย่างอิสระตามความเหมาะสมของบริการนั้นๆ

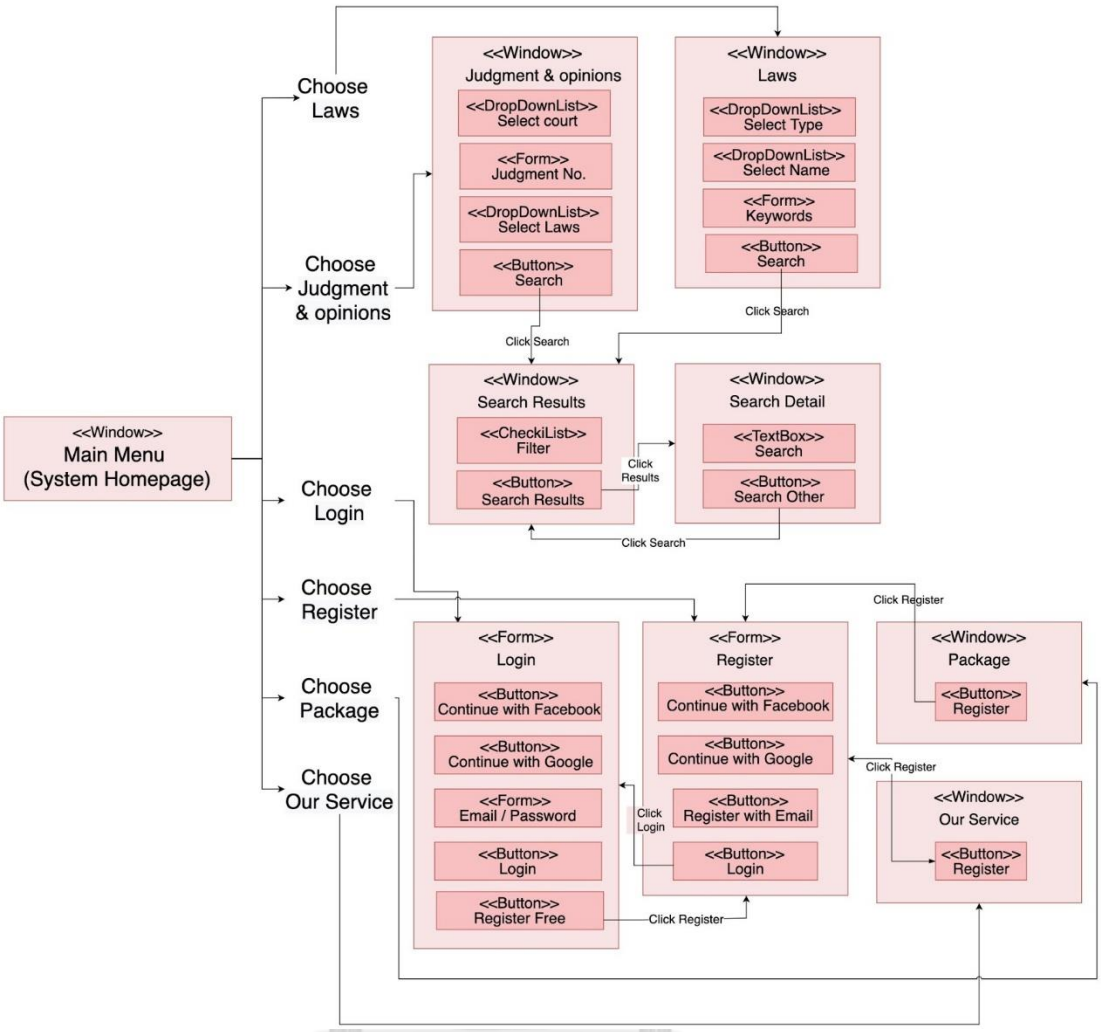
หลังจากการแบ่งทีมพัฒนาไมโครฟรอนต์เอนส์ ทางผู้วิจัยจะมีการเพิ่มความต้องการ (Requirements) ใหม่ในการพัฒนาหน้า Document ดังภาพที่ 8 เนื่องจากเดิม Document

Service เป็นเพียงแค่การดึงข้อมูลมาแสดงจากเว็บไซต์ภายนอก โดยทางผู้วิจัยจะใช้ความต้องการในส่วนนี้นำไปพัฒนาซอฟต์แวร์ส่วนหน้าทั้งในระบบเดิม ซึ่งเป็นสถาปัตยกรรมแบบโมโนลิทิก และในระบบใหม่ที่ปรับโครงสร้างเป็นสถาปัตยกรรมไมโครฟรอนต์เอนส์ ทั้งนี้ ในระบบเดิมนั้น ทางผู้วิจัยจะพัฒนาโดยใช้ Angular version 1 เช่นเดิม ส่วนในระบบใหม่จะถูกปรับโครงสร้างและพัฒนาแบบไมโครฟรอนต์เอนส์โดยใช้แบ็กเอนด์ไมโครเซอร์วิสดั้งเดิม จากนั้น ผู้วิจัยจะทำการทดสอบ ด้วยการทดสอบแบบถดถอย (Regression Test) และ ทดสอบความเร็วในการโหลดข้อมูล (Loading Time) เพื่อเปรียบเทียบสมรรถนะของระบบเดิมที่พัฒนาฟีเจอร์ส่วนหน้าเพิ่มด้วยสถาปัตยกรรมแบบโมโนลิทิกกับระบบที่พัฒนาฟีเจอร์ส่วนหน้าเพิ่มด้วยสถาปัตยกรรมแบบไมโครฟรอนต์เอนส์ และวิเคราะห์เปรียบเทียบ ข้อจำกัด ข้อดีและข้อเสีย ของโครงสร้างระบบทั้งสองแบบ



ภาพที่ 8 ขั้นตอนการทำงานวิจัย

Windows Navigation Diagram (WND) ดังภาพที่ 9 การออกแบบส่วนต่อประสานผู้ใช้ในการพัฒนาเว็บแอปพลิเคชันของทุกทีม โดยมีการแสดงตัวกำหนดการกระทำต่างๆ ของปุ่มเพื่อเป็นการเรียกใช้หรือปรับเปลี่ยนคอมโพเนนต์ ตัวอย่างเช่น เมื่อผู้ใช้กดปุ่ม Judgment or Opinions ระบบจะทำการนำผู้ใช้ไปสู่หน้าสืบค้น เมื่อผู้ใช้กรอกข้อมูลสำหรับการค้นหาและทำการกดปุ่มค้นหา ระบบจะทำการแสดงคอมโพเนนต์ของการค้นหามาแสดง เป็นต้น



ภาพที่ 9 การออกแบบโครงสร้างส่วนต่อประสานผู้ใช้ของเว็บแอปพลิเคชัน Lawphin

บทที่ 4
การออกแบบและพัฒนาเครื่องมือ

4.1 สถาปัตยกรรมของระบบ

ในการพัฒนาเว็บแอปพลิเคชันแบบเดิมนั้น ถูกพัฒนาขึ้นด้วยแนวคิดแบบไมโครเซอร์วิส แต่ยังคงใช้ฟรอนต์เอนด์แบบโมโนลิทิก ซึ่งการออกแบบโครงสร้างแบ็กเอนด์ในรูปแบบสถาปัตยกรรมไมโครเซอร์วิสนั้น สามารถรองรับการ ขยายได้ค่อนข้างดีจากการใช้งานมาตลอดหลายปี แต่เนื่องจากการที่ทีมผู้พัฒนาต้องทำการเพิ่มฟีเจอร์ใหม่ส่งผลให้ค่อนข้างยาก ที่จะทำการพัฒนาฟรอนต์เอนด์ เนื่องจากยังคงใช้การพัฒนาด้วย Angular js version 1 ซึ่งเป็นกรอบงานรุ่นเก่าที่ผู้พัฒนาส่วนมากเลิกใช้และไม่มีการสนับสนุน สำหรับหลายๆ Library ดังนั้นทางทีมผู้พัฒนาจึงเปลี่ยนการพัฒนามาเป็นการพัฒนาโดยใช้การออกแบบสถาปัตยกรรมแบบไมโครฟรอนต์เอนส์แทนโดยยึด โครงสร้างการทำงานของ สถาปัตยกรรมไมโครเซอร์วิสแบบดั้งเดิมไว้ โดยมีรูปแบบสถาปัตยกรรม ระบบใหม่ (System Architect) ดัง รูปสถาปัตยกรรมใหม่ภาพที่ 7 จากภาพทางทีมผู้พัฒนานั้นสามารถที่จะพัฒนาหลายฟีเจอร์ เช่น การแสดงผลการค้นหา ซึ่งผู้พัฒนาได้ทำการพัฒนาระบบด้วยการแบ่งทีมออกมาตามฟีเจอร์ต่างๆ ที่ได้แบ่งจัดโครงสร้างดังภาพ ซึ่งในแต่ละทีมพัฒนาจะประกอบด้วยผู้พัฒนาอย่างต่ำสองตำแหน่ง ดังนี้

- ผู้พัฒนาฟรอนต์เอนด์ รับผิดชอบในการพัฒนาระบบในส่วนหน้าของระบบที่ทีมนั้นๆ ได้รับผิดชอบ
- ผู้พัฒนาแบ็กเอนด์ ทำหน้าที่เตรียมโครงสร้างสำหรับการติดต่อส่วนประสานผู้ใช้และการสื่อสารระหว่าง แบ็กเอนด์แบบไมโครเซอร์วิสที่ได้พัฒนาไว้ในระบบดั้งเดิม และต้องทำการอัปเกรด ส่วนต่อประสานผู้ใช้ในบางส่วน เพื่อให้เหมาะกับการทำงานกับสถาปัตยกรรมไมโครเซอร์วิสในโมดูลนั้นๆ

4.2 เครื่องมือในการพัฒนา

4.2.1. Single SPA JS คือกรอบงานในการผสานการทำงานของสถาปัตยกรรมไมโครฟรอนต์เอนส์ หลายๆ ตัวเข้าด้วยกันให้ทำงานได้แบบไร้ร่อยต่อ (Seamless)

4.2.2. Vue JS คือ กรอบงานฟรอนต์เอนด์ที่ใช้ในการพัฒนาในส่วนไมโครฟรอนต์เอนส์ของหน้า Landing เนื่องจากเป็นกรอบงานที่เหมาะกับงานไม่ซับซ้อนและเหมาะกับงานที่สนับสนุนการทำงานของ เว็บแอปพลิเคชันแบบ Static

4.2.3. **Angular** คือ กรอบงานฟรอนต์เอนด์ที่ใช้พัฒนาเป็นหลักในวิทยานิพนธ์นี้เนื่องจากโครงสร้างดั้งเดิมนั้นได้ถูกพัฒนาขึ้นด้วย Angular JS ทางทีมผู้พัฒนาจึงตัดสินใจว่าในการพัฒนาโมดูลที่มีความซับซ้อน ทางทีมจะทำการพัฒนาด้วย Angular JS

4.2.4. **CSS** คือ ภาษาโปรแกรม (Programming Language) รูปแบบหนึ่ง ที่ใช้ในการพัฒนาเพื่อเพิ่มความสวยงานของเว็บแอปพลิเคชัน

4.2.5. **Docker** คือ Containerized Service ซึ่งทำหน้าที่ในการจัดการเรื่องรันไทม์

4.2.6. **MongoDB** คือ พื้นที่จัดเก็บฐานข้อมูล (Database Storage) ของระบบ ใช้ในการเก็บข้อมูลเอกสาร ที่ใช้ภายในเว็บแอปพลิเคชัน

4.2.7. **Elasticsearch** เป็นตัวหลักในการพัฒนาโปรแกรมค้นหา (Search Engine) ของระบบ

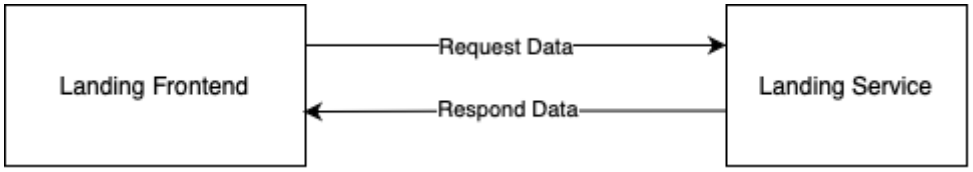
4.2.8. **Visual studio code** คือ เครื่องมือที่ใช้ในการเขียนโค้ด (Code Editor)

4.3 แนวคิดในการออกแบบและแบ่งทีมในการพัฒนา

ในงานวิจัยนี้ทางทีมผู้พัฒนาได้มุ่งเน้นในการพัฒนาและปรับโครงสร้างระบบ จากระบบเก่าที่ถูกพัฒนาขึ้นมาด้วยสถาปัตยกรรมฟรอนต์เอนด์แบบโมโนลิทิก สู่การพัฒนาและปรับโครงสร้างให้เป็นรูปแบบของสถาปัตยกรรมไมโครฟรอนต์เอนส์ ซึ่งกระบวนการสำคัญในการพัฒนา คือ การแบ่งฟีเจอร์และการแบ่งทีมในการพัฒนาเพื่อให้มีการทำงานที่เป็นอิสระต่อกันมากที่สุด ซึ่งทางทีมผู้พัฒนาได้ทำการแบ่งฟีเจอร์ดังนี้

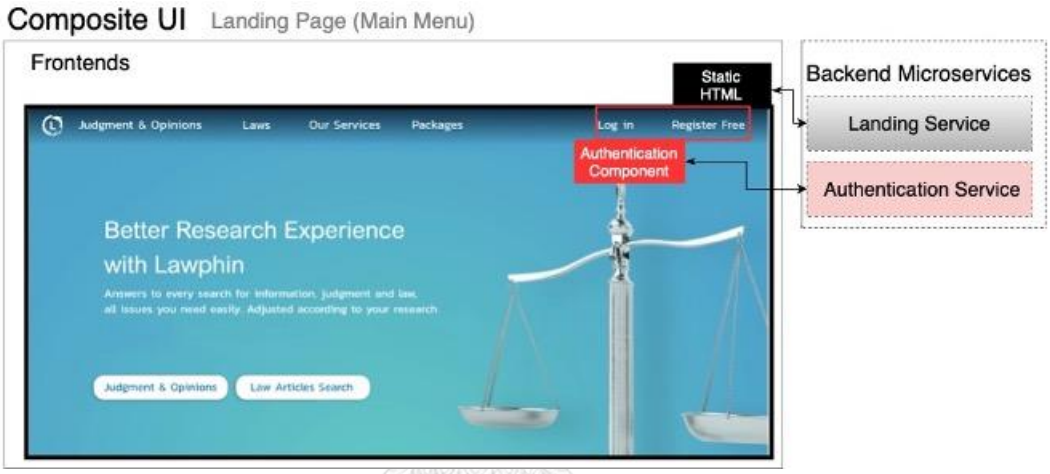
4.3.1. ทีม Website

เป็นทีมที่ดูแลการนำเสนอข้อมูลที่อยู่ในรูปแบบ Static เว็บไซต์ โดยใน ฟีเจอร์นี้ ทางทีมผู้พัฒนาจะทำการใช้ภาษา HTML, ภาษาจาวาสคริปต์ และ CSS โดยตรง เพื่อทำการนำเสนอข้อมูลที่อยู่ในรูปแบบ Static ภายในเว็บไซต์ กล่าวคือ เป็นข้อมูลที่ไม่ไม่มีการเปลี่ยนแปลงบ่อย โดยส่วนที่มีการเรียกข้อมูลจาก ระบบไมโครเซอร์วิสหรือแม้กระทั่งข้อมูลการตรวจสอบผู้ใช้ (User Tracking) จะใช้ VueJs เป็นหลักในการสื่อสาร ดังภาพที่ 10



ภาพที่ 10 การดึงข้อมูลและส่งไปยัง landing service

เมื่อได้โค้ด Static HTML ที่พร้อมนำไปใช้เรียบร้อย จากภาพที่ 10 และภาพที่ 11 ทางทีม Website จะดำเนินการทำการทดสอบพฤติกรรม (Behavior Testing) และ ทดสอบในรูปแบบ end-to-end ตั้งแต่ฟรอนด์เอนด์แบบ Static ไปจนถึงแบ็กเอนด์แบบไมโครเซอร์วิส เมื่อทำการทดสอบเสร็จแล้ว ทีม Website จะทำการรวบรวม โค้ดที่สร้างเสร็จนั้น Integrate เข้ากับกรอบงานซิงเกิลเพจแอปพลิเคชัน ในขั้นตอนนี้เนื่องจาก Landing page ส่วนใหญ่เป็นเพียงการนำเอาข้อมูลรูปแบบ Static มาทำการแสดงผล ทีม Website จึงต้องทำการห่อหุ้ม (Pack) แอปพลิเคชัน ทั้งหมดในรันไทม์เดียวกันสำหรับงานด้านฟรอนต์เอนด์

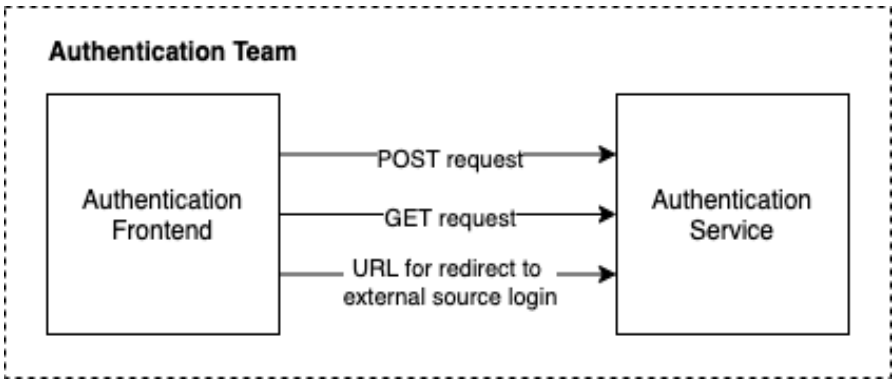


ภาพที่ 11 หน้าจอการพัฒนาของทีมพัฒนา Website

4.3.2. ทีม Authentication

เป็นทีมที่รับผิดชอบในการพัฒนาหน้าจอสำหรับการเข้าใช้งาน (Login) และการยืนยันตัวตนของผู้ใช้ เพื่อให้ผู้ใช้งานสามารถเข้าใช้งานระบบของ Lawphin

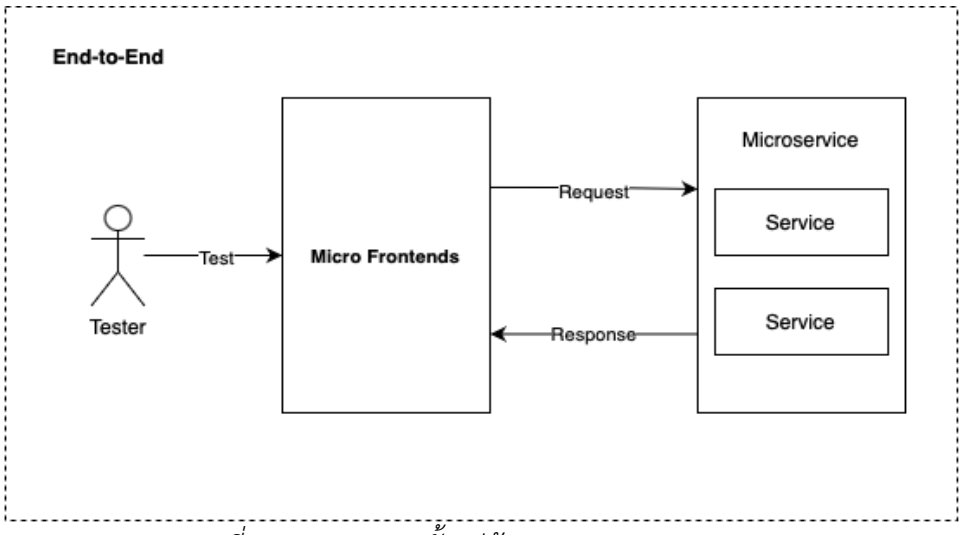
สำหรับการพัฒนาไมโครฟรอนต์เอนส์ในส่วนของทีม Authentication Service ได้นำไมโครเซอร์วิสของ Authentication แบบเดิมที่ถูกสร้างไว้แล้ว ซึ่งมีองค์ประกอบสำคัญที่ใช้ในการติดต่อสื่อสาร (Communication) ผ่านทาง HTTP Protocol ดังภาพที่ 12 จะมีทั้งหมด 2 องค์ประกอบ คือ Payload และ Request path โดย Payload จะเป็นส่วนที่กำหนดว่าฟรอนต์เอนด์ต้องส่งพารามิเตอร์ จำพวกชื่อผู้ใช้งาน (Username) และ รหัสผ่าน (Password) ในกรณีของการเข้าสู่ระบบด้วยผู้ใช้ภายใน (Internal User) ซึ่งหากเป็นการเข้าสู่ระบบภายนอก External Login API จะทำการ Redirect ไปยังที่มาของการเข้าสู่ระบบนั้นๆ



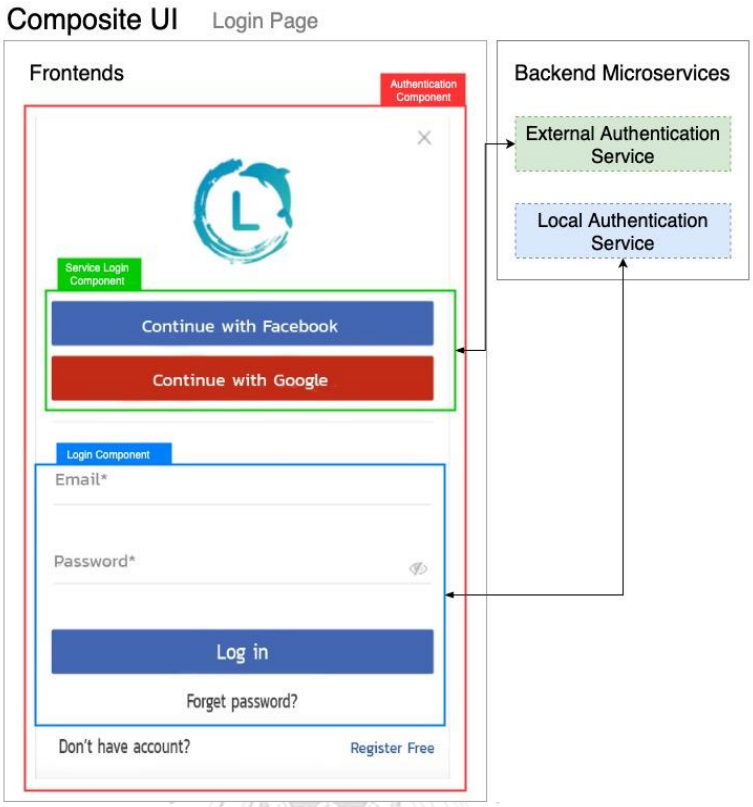
ภาพที่ 12 การติดต่อสื่อสารของทีม Authentication

ในส่วนของการพัฒนาระบบ ทีม Authentication ได้ทำการสร้างแบบฟอร์มระบบและองค์ประกอบต่างๆ ด้วยกรอบงาน Angular ซึ่งการสร้างองค์ประกอบต่างๆ ที่สามารถนำกลับมาใช้ใหม่ได้นั้น ทางทีมพัฒนาได้ใช้วิธีการสร้าง HTML คอมโพเนนต์ ดังที่ได้กล่าวมาข้างต้น เมื่อคอมโพเนนต์ทั้งหมดได้ถูกสร้างขึ้นจะถูกนำมารวมกันเป็นซิงเกิลเพจแอปพลิเคชัน

แนวปฏิบัติที่เป็นเลิศ (best practice) ระหว่างการพัฒนาซอฟต์แวร์ จะมีการทดสอบระบบในรูปแบบ Unit Test ในแต่ละฟังก์ชัน เพื่อให้มั่นใจว่าส่วนต่อประสานผู้ใช้ ในส่วนที่ทีมพัฒนารับผิดชอบสามารถทำงานได้ถูกต้อง และจะทำการทดสอบระบบในรูปแบบของ Functional Testing เมื่อทำการรวมคอมโพเนนต์ต่างๆ เข้าด้วยกัน จากนั้นทางทีม Authentication จะทำการทดสอบอีกครั้งตั้งแต่ต้นจนจบ (end-to-end) เฉพาะในส่วนความรับผิดชอบของทีมเท่านั้น กล่าวคือจะทำการทดสอบตั้งแต่ส่วนต่อประสานผู้ใช้ไปจนถึงแบ็กเอนด์ดั้งเดิมที่มีอยู่แล้ว ดังภาพที่ 13



ภาพที่ 13 การทดสอบตั้งแต่ต้นจนจบ (end-to-end)



ภาพที่ 14 หน้าจอแอปพลิเคชันภายหลังการพัฒนา

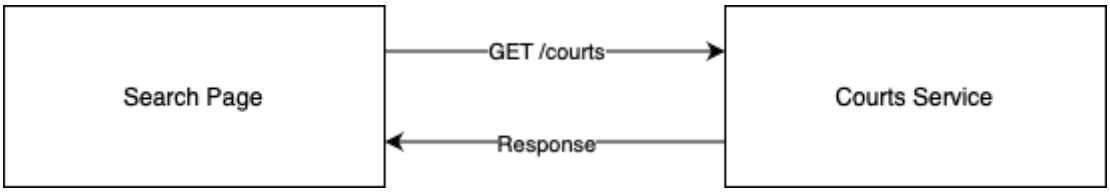
4.3.3. ทีม Search

ในส่วนของทีม Search จะมีหน้าที่ในการนำข้อมูลที่เป็นข้อมูลหลัก (Master Data Service) มาแสดงเพื่อให้ผู้ใช้ได้ทำการเลือก ข้อมูลเกณฑ์ในการค้นหาก่อน โดยมีการเรียกใช้บริการหลัก 2 บริการ ประกอบด้วย

- บริการ Court (Court service) เป็นหนึ่งในไมโครเซอร์วิสที่มีสำหรับดูแลการนำเสนอข้อมูลของศาลในประเทศไทย เพื่อนำมาเป็นข้อมูลให้ผู้ใช้สามารถเลือกข้อมูลในการค้นหาได้
- บริการ Relevant Law (Relevant Law service) เป็นไมโครเซอร์วิสที่ถูกสร้างขึ้นเพื่อดูแลข้อมูลของกฎหมายต่างๆ ในประเทศไทย รวมถึงข้อมูลมาตราต่างๆ ของแต่ละกฎหมายที่เกี่ยวข้อง

ทั้งนี้ ไมโครเซอร์วิสทั้งหมดนั้นเป็น Rest API ทีมผู้พัฒนาสามารถทำการ สร้าง อ่าน อัปเดต และลบข้อมูลได้ในทุกบริการ แต่เนื่องจากเป็นส่วนต่อประสานของผู้ใช้ จึงถูกจำกัดให้ทำได้แค่อ่าน

เท่านั้น ตัวอย่างเช่น ในกรณีของข้อมูลของบริการ Court จะทำได้เพียงค้นหาและทำจัดทำรายชื่อของศาล ซึ่งบริการที่ได้กล่าวมาข้างต้นนั้น อธิบายได้ด้วยภาพที่ 15 และภาพที่ 16



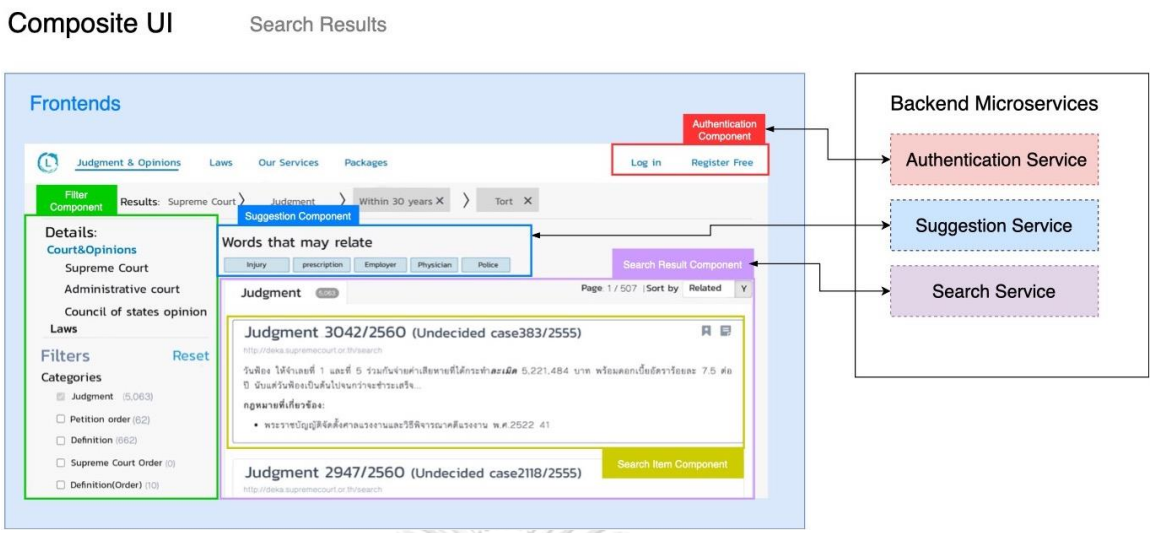
ภาพที่ 15 การติดต่อสื่อสารของทีม Search Page ไปยัง courts

ภาพที่ 16 การติดต่อสื่อสารของทีม Search Page ไปยัง relevant law

การพัฒนาของฟรอนต์เอนด์ นั้นจะแยกการพัฒนาออกเป็นคอมโพเนนต์เช่นเดียวกันเพื่อความง่ายในการนำมากลับมาใช้ใหม่ เพื่อสามารถนำคอมโพเนนต์เดิมที่พัฒนาไปแล้วนำกลับมาใช้กับส่วนอื่นๆ ในอนาคตหากมีการอัปเกรดหรือการนำไปใช้เพิ่มเติมในส่วนอื่น เช่นเดียวกับโมดูลบริการ Authentication ทุกอย่างที่เป็นคอมโพเนนต์จะถูกสร้างด้วยคอมโพเนนต์ HTML แต่เราจะไม่มีการใช้งานร่วมกันของรันไทม์ และโค้ดของบริการ Authentication แต่จะเป็นการแยกกันทำงานเพื่อหลีกเลี่ยงปัญหาของการใช้ Library คนละรุ่นกันรวมถึงการอัปเกรดหรือการพัฒนาในอนาคต คอมโพเนนต์ที่สำคัญและมีการสื่อสารไปยังไมโครเซอร์วิสมี ดังนี้

- Dropdown ของการเลือกศาล ซึ่งมีการเรียกข้อมูลจากบริการ Court
- Dropdown ของการเลือกบริการ Relevant Law ซึ่งจะมีเหตุการณ์ต่อเนื่องกันเมื่อเลือกกฎหมายที่เกี่ยวข้องแล้วนั้นผู้ใช้จะสามารถเลือกมาตราที่เกี่ยวข้องได้ซึ่งการเรียกข้อมูลครั้งที่สองจะทำการคัดกรองเฉพาะมาตราที่เกี่ยวข้องของกฎหมายนั้นๆ

  เมื่อทีมพัฒนาได้ทำการสร้างคอมโพเนนต์ ต่างๆ เสร็จจะทำการนำทุกคอมโพเนนต์มารวมกันเป็นซิงเกิลเพจแอปพลิเคชัน เพื่อรอการทดสอบ แก้ไข และนำไปรวมกันกับเค้าโครงหลัก (Main Layout) อีกครั้ง



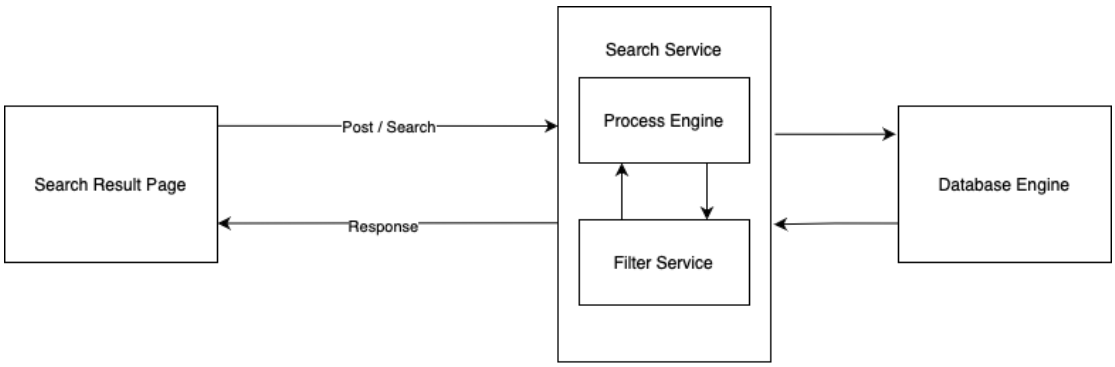
ภาพที่ 17 หน้าจอการค้นหา

จากภาพที่ 17 เมื่อระบบทุกอย่างได้ถูกพัฒนาและทดสอบเสร็จสิ้น ทางทีมพัฒนาจะทำการนำซิงเกิลเพจแอปพลิเคชัน มาทดสอบอีกครั้งตั้งแต่ต้นจนจบในรูปแบบ end-to-end โดยเริ่มจากการทดสอบความถูกต้องขององค์ประกอบ, ส่วนต่อประสานผู้ใช้ รวมถึงทดสอบการทำงานของส่วนต่อการสื่อสารระหว่างกรอบงาน Angular ซิงเกิลเพจแอปพลิเคชันที่เป็นไมโครฟรอนต์เอนส์และไมโครเซอร์วิสต่างๆ ที่ทีมผู้พัฒนาได้กล่าวถึงในข้างต้น

4.3.4. ทีม Search Results

เมื่อผู้ใช้งานเลือกเกณฑ์ในการค้นหาเรียบร้อยแล้ว จะทำการส่งข้อมูลทั้งหมดมายังฟรอนต์เอนด์ Search Result เมื่อได้พารามิเตอร์ที่ต้องการ ทีมพัฒนาจะมีหน้าที่นำข้อมูลทั้งหมดส่งไปค้นหาซึ่งจะทำการค้นหาจากไมโครเซอร์วิสต่างๆ ซึ่งทีมผู้พัฒนาได้เตรียมไว้เป็นไมโครเซอร์วิส ดังนี้

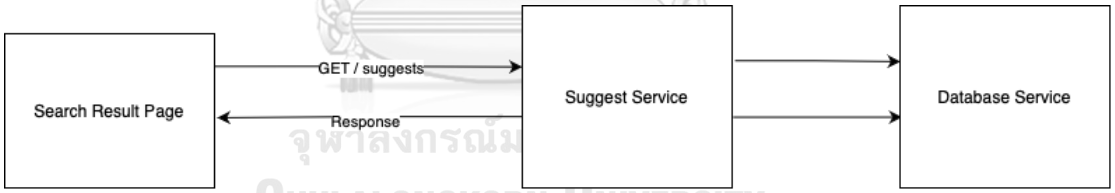
- **บริการ Search (Search Service)** คือ บริการที่ได้รับค่าจากการเลือกของผู้ใช้เพื่อใช้ในการนำไปค้นหาในระบบฐานข้อมูลของ Lawphin เมื่อทำการค้นหนาเสร็จแล้วนั้นข้อมูบทั้งหมดจะถูกนำมาประมวณผลที่ Compute Engine เพื่อปรับเปลี่ยนลำดับผลการค้นหาและ Compute Engine เองก็จะส่งผลการค้นหาไปสร้าง Filter Options สำหรับ Filter Component ในภาพที่ 18 และเมื่อประมวณผลเสร็จแล้วทุกอย่างจะถูก Response กลับไปให้ Micro Frontend นำไปแสดงผลต่อไป ซึ่งกระบวนการทั้งหมดเป็นไปดังภาพที่ 18



ภาพที่ 18 การสื่อสารและการประมวลผลหาค่าคำสำคัญ

- **บริการ Suggestion (Suggestion Service)** คือ บริการที่รับค่าจากการเลือกของผู้ใช้เพื่อใช้ในการนำไปหาคำสำคัญ (Keyword) Suggestion จากระบบฐานข้อมูล โดยใช้วิธีการ นำคำที่เกี่ยวข้องทั้งหมดที่ทางระบบได้เตรียมไว้ทำการจับคู่ กับผลการค้นหา เช่นเดียวกันกับบริการอื่นๆ

ภายในระบบทางผู้พัฒนาได้กำหนดแนวทางในการพัฒนาของบริการไว้ให้สามารถเรียกใช้ด้วย GET Method ได้เพียงอย่างเดียว เนื่องจากเป็นการเรียกข้อมูลจากไมโครฟรอนต์เอนส์เพื่อนำไปใช้ ดังภาพที่ 19



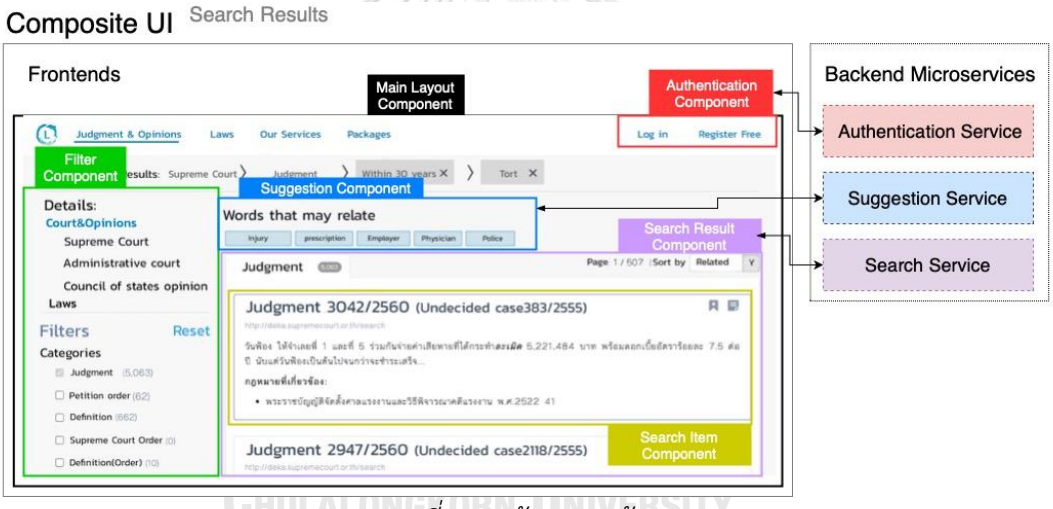
ภาพที่ 19 แผนภาพขั้นตอนการสอบถามเพื่อให้ได้คำสำคัญต่างๆ ที่เกี่ยวกับผลการค้นหา

ในส่วนของการพัฒนาไมโครฟรอนต์เอนส์ ทีมผู้พัฒนาจะใช้กรอบงาน Angular เช่นเดียวกับไมโครฟรอนต์เอนส์ในส่วนของทีม Search แต่ภายในหน้านี้จะมีความซับซ้อนค่อนข้างสูงกว่า เช่น การแสดงผลการค้นหาด้วยการแสดงแบบยืดหยุ่นของแต่ละผลการค้นหา ดังนั้น ทางทีมผู้พัฒนาจะทำการแบ่งคอมโพเนนต์ของหน้าดังกล่าวออกเป็นส่วนๆ ดังนี้

- คอมโพเนนต์ Suggestion จะทำการนำผลการวิเคราะห์ Suggestion Keyword จาก บริการ Suggestion

- คอมโพเนนต์ Filter จะนำผลจากการประมวณผลของบริการ Search นำมาแสดงผลเพื่อให้ผู้ใช้สามารถเลือกเพื่อทำการเพิ่มความเฉพาะเจาะจงของการค้นหาเพื่อที่จะทำให้ผลการค้นหาลดลง
- คอมโพเนนต์ Search Result คือ คอมโพเนนต์หลัก ที่จะทำการแสดงผลของการค้นหาทั้งหมดรวมถึงจากการคัดกรองของผู้ใช้ ซึ่งผลการค้นหาที่ได้จะต้องแสดงผลแบบซ้ำๆ กัน จึงมีการสร้างคอมโพเนนต์เพิ่มขึ้นมา คือ
  - คอมโพเนนต์ Search item ซึ่งจะนำผลการค้นหาที่ได้มาแสดงผล จากภาพที่ 20 จะทำการแสดงผลของคำพิพากษา

เมื่อแต่ละคอมโพเนนต์ได้ถูกพัฒนาและทำการทดสอบครบเสร็จสิ้น ทางทีมผู้พัฒนาจึงทำการนำคอมโพเนนต์ทั้งหมดมารวมกันเป็นเพื่อสร้างซิงเกิลเพจแอปพลิเคชันต่อไป

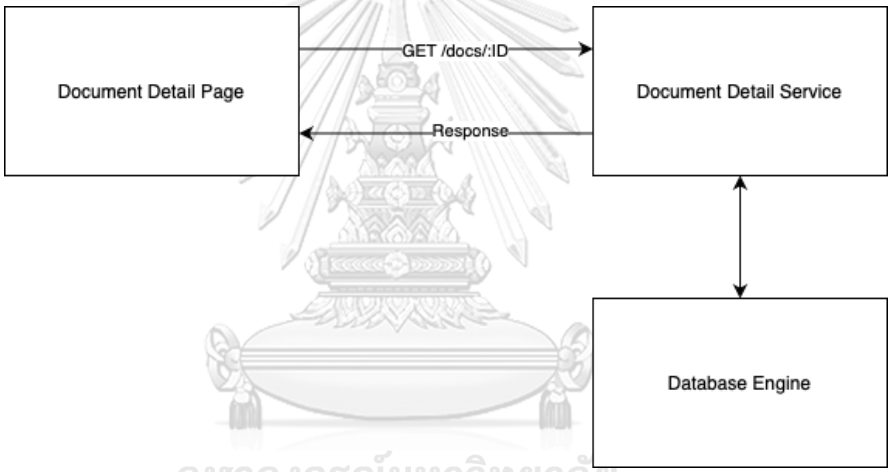


ภาพที่ 20 หน้าจอการค้นหา

จากภาพที่ 20 เมื่อทีมผู้พัฒนาได้ทำการสร้างซิงเกิลเพจแอปพลิเคชัน ของคอมโพเนนต์ Search Result เสร็จ ทางทีมผู้พัฒนาก็จะมีการทดสอบตั้งแต่ต้นจนจบในรูปแบบ end-to-end เช่นเดียวกันกับไมโครฟรอนต์เอนส์ อื่นๆ โดยจะทำการทดสอบการแสดงผลของแต่ละคอมโพเนนต์ เช่น คอมโพเนนต์ Search Item จะต้องสามารถแสดงผลการค้นหาให้ได้ในหลากหลายรูปแบบ เพราะอาจจะมีข้อมูลที่ขาดหายจากระบบฐานข้อมูล ดังนั้นจึงมีความจำเป็นต้องสามารถแสดงผลได้อย่างถูกต้อง เมื่อทำการทดสอบและแก้ไขเสร็จจะเข้าสู่กระบวนการนำไปรวมกับคอมโพเนนต์ Main Layout โดยใช้ซิงเกิลเพจแอปพลิเคชัน ที่กล่าวมาข้างต้นอีกครั้ง

4.3.5. ทีม Document

เมื่อผู้ใช้ทำการเลือกผลการค้นหาที่ตนพึงพอใจ ผู้ใช้คาดหวังว่าจะสามารถอ่านคำพิพากษาหรือรายละเอียดของผลการค้นหาได้ สำหรับในส่วนของฟีเจอร์การค้นหานี้ถือเป็นฟีเจอร์ใหม่ที่ทางทีมได้พัฒนาขึ้นมา เนื่องจากก่อนหน้านี้บริการ Document Detail จะถูกสร้างขึ้นเพื่อที่จะทำการเชื่อมต่อไปยังหน้าเอกสารของเว็บไซต์สำนักงานศาลฎีกาโดยตรง ดังนั้น ทีมผู้พัฒนาทีมนี้จึงต้องพัฒนาตั้งแต่แบ็กเอนด์ไมโครเซอร์วิสของหน้า Document Detail ซึ่งหลักการทำงานของบริการนี้ คือ ผู้ใช้ที่เลือกผลการค้นหามาเรียบร้อย จะถูกนำพามายังหน้า Document Detail ด้วย URL ที่มี Document ID ซึ่งเป็นค่าเฉพาะ (Unique) ที่ทางระบบได้เตรียมไว้ให้ในหน้า Search Result เมื่อได้ Document ID มา ตัวบริการจะทำการนำข้อมูลไปค้นหาในระบบฐานข้อมูลของเว็บไซต์เพื่อให้ได้มาซึ่งข้อมูลทั้งหมดที่ใช้ในการแสดงผล สามารถแสดงการหลักการทำงานได้ตามภาพที่ 21

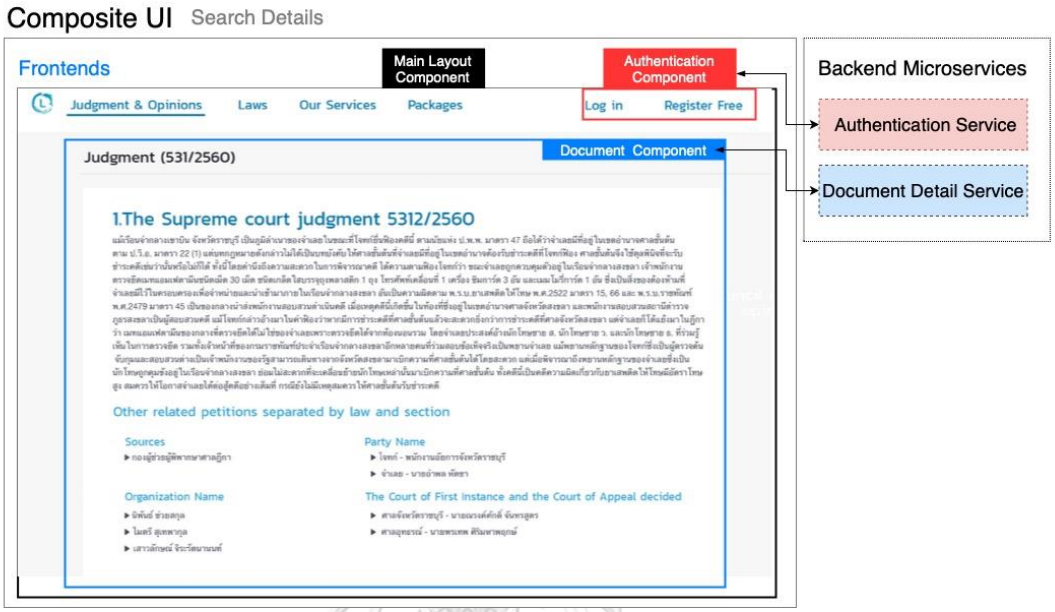


ภาพที่ 21 การดึงข้อมูล Document Detail Page

สำหรับการพัฒนาไมโครฟรอนต์เอนส์ของทีม Document นั้น จะเน้นการนำเสนอข้อมูลที่เป็นข้อความและข้อมูลคงที่ (Static Data) ดังนั้น ทางทีมผู้พัฒนาจึงตัดสินใจใช้ React Library เพื่อช่วยในการพัฒนาในส่วนของจาวาสคริปต์และส่วนการติดต่อไปยัง ไมโครเซอร์วิส เนื่องจาก React เป็น Library ที่มีขนาดค่อนข้างเล็กจะช่วยให้สามารถทำการประมวลผลและโหลดคอมโพเนนต์ได้เร็วยิ่งขึ้น โดยการแบ่งคอมโพเนนต์ของทีมนี้ จะมีคอมโพเนนต์หลักๆ เพียงคอมโพเนนต์ Document Detail ซึ่งทำการร้องขอข้อมูลไปยังบริการ Document Detail และนำข้อมูลที่ได้มาแสดงผลในคอมโพเนนต์ ตัวอย่างดังภาพที่ 22

ภายหลังกระบวนการพัฒนาเสร็จสิ้นลง ทางทีมผู้พัฒนาจะทำการทดสอบตั้งแต่ต้นจนจบในรูปแบบ end-to-end ตั้งแต่ในส่วนของ ฟรอนต์เอนด์ Document Detail ไปจนถึงบริการของ

Document Detail โดยการทดสอบจะเริ่มตั้งแต่การรับ Document ID มาจากการเลือกในหน้า Search Result จากนั้นจะทำการนำ Document ID ส่งไปให้บริการ Document Detail เพื่อทำการค้นหาข้อมูลมาให้ไมโครฟรอนต์เอนส์ทำการแสดงผลการค้นหา ตัวอย่างหน้าจอแสดงผลลัพธ์การค้นหาดังภาพที่ 22



ภาพที่ 22 หน้าจอผลการค้นหา

4.4 การรวมไมโครฟรอนต์เอนส์

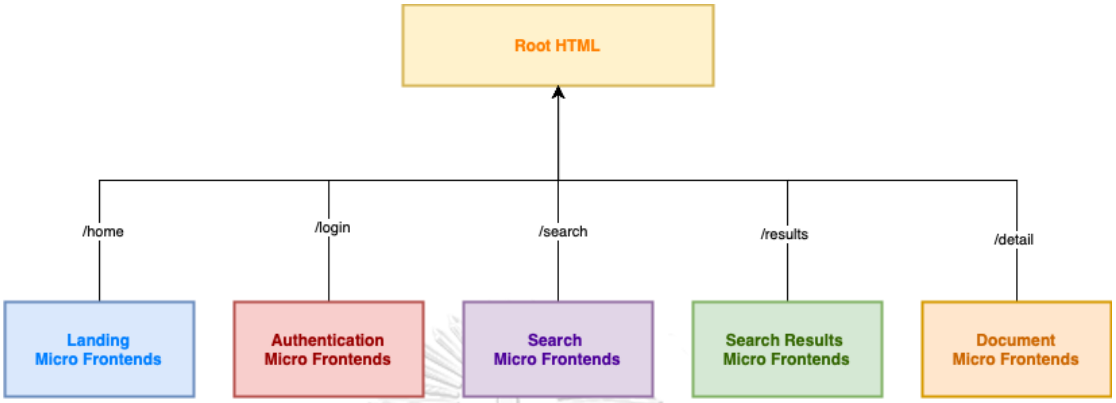
เมื่อผู้พัฒนาทุกทีมได้ทำการพัฒนาไมโครฟรอนต์เอนส์เสร็จแล้วนั้นจะต้องทำการนำไมโครฟรอนต์เอนส์มารวมกันด้วยกรอบงานซิงเกิลเพจแอปพลิเคชัน ซึ่งจะเป็นเครื่องมือที่ช่วยในการจัดการเรื่องการแสดงผลของแต่ละไมโครฟรอนต์เอนส์ และทำการตรวจสอบหากผู้ใช้งานเข้าใช้ด้วย URL path นั้น จะต้องพาผู้ใช้ไปเจอกับหน้าจอของไมโครฟรอนต์เอนส์ตามที่ระบุมาจาก URL path นั้น โดยภายในซิงเกิลเพจแอปพลิเคชัน ประกอบด้วยสองส่วนหลักคือ

- Root HTML ทำหน้าที่เป็นตัวแสดงผลของไมโครฟรอนต์เอนส์ต่างๆ
- จาวาสคริปต์ เป็นตัวการในการจัดการตั้งค่า route ต่างๆ

จากระบบไมโครฟรอนต์เอนส์ในงานวิจัยนี้นั้นสามารถนำเสนอในรูปภาพรวม ได้ดังภาพที่ 23 โดยที่ Root HTML จะแสดงผล path ดังนี้

- เมื่อผู้ใช้เข้าด้วย path /home จะแสดงไมโครฟรอนต์เอนส์ Landing
- เมื่อผู้ใช้เข้าด้วย path /login จะแสดงไมโครฟรอนต์เอนส์ Authentication
- เมื่อผู้ใช้เข้าด้วย path /search จะแสดงไมโครฟรอนต์เอนส์ Search

- เมื่อผู้ใช้เข้าด้วย path /results จะแสดงไมโครฟรอนต์เอนส์ Search Results
- เมื่อผู้ใช้เข้าด้วย path /detail จะแสดงไมโครฟรอนต์เอนส์ Document



ภาพที่ 23 การแสดงไมโครฟรอนต์เอนส์ผ่าน Root HTML

บทที่ 5
การทดสอบและการวิเคราะห์ผล

5.1 วัตถุประสงค์ของการทดสอบ (Purpose of the evaluation)

วิทยานิพนธ์ฉบับนี้ได้นำเสนอแนวทางการพัฒนาเว็บแอปพลิเคชันจากการประยุกต์ใช้สถาปัตยกรรมไมโครฟรอนต์เอนส์สำหรับการพัฒนาส่วนฟรอนต์เอนด์ โดยใช้ระบบสืบค้นข้อมูลทางกฎหมายไทยเป็นกรณีศึกษา การประเมินผลแนวทางที่นำเสนอในงานวิจัยนี้ได้ทำการทดสอบโดยพัฒนาเพิ่มฟีเจอร์ใหม่คือหน้า Document Detail เพื่อเปรียบเทียบสมรรถนะของระบบเดิมซึ่งใช้เทคโนโลยี Single Page Application (SPA) ในการพัฒนาเว็บไซต์ส่วนฟรอนต์เอนด์กับระบบที่ปรับโครงสร้างใหม่ของส่วนฟรอนต์เอนด์ด้วยสถาปัตยกรรมไมโครฟรอนต์เอนส์ ด้วยการวัดค่าต่างๆ จากการทดสอบระบบ ดังรายละเอียดอธิบายในหัวข้อต่อไป

5.2 ตัววัดสมรรถนะ (Performance Measures)

ในงานวิจัยนี้ ได้ทำการทดลองพัฒนาเพิ่มเติมความต้องการใหม่ของระบบกรณีศึกษา คือ หน้าจอแสดง Document Detail และเก็บค่าข้อมูลตัววัดมิติต่างๆ 3 ค่า ประกอบด้วย

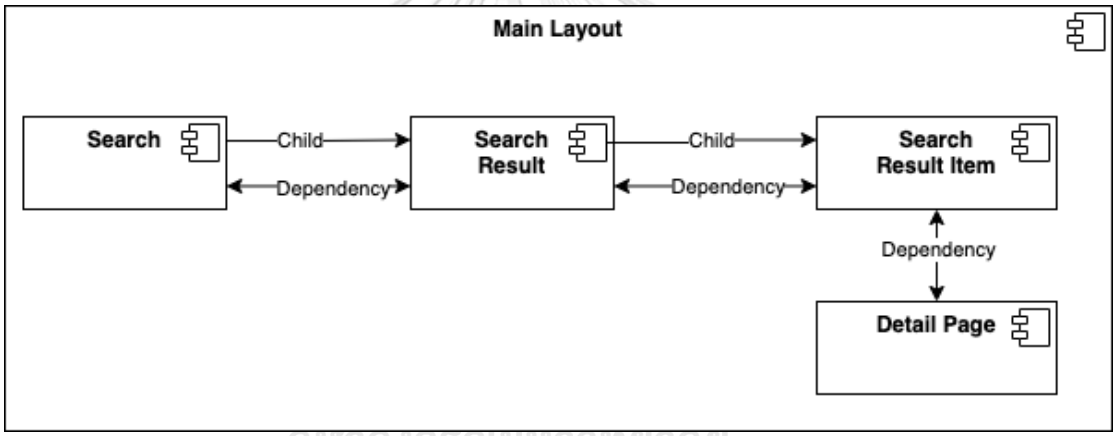
- **Component Dependency** คือ ความเป็นอิสระต่อกัน ในส่วนนี้จะใช้สำหรับการคำนวณเพื่อหาความซับซ้อนของการพัฒนา ยิ่งมีค่า Component Dependency มากขึ้นเท่าไหร่จะสะท้อนถึงความซับซ้อนของการพัฒนามากขึ้นเท่านั้น
- **Development Time (Man-hour)** คือ เวลาที่ใช้ในการพัฒนาฟีเจอร์ใหม่ เนื่องจากเป็นการทดสอบบนทีมพัฒนาจริง ในส่วนของฟรอนต์เอนด์แบบโมโนลิทิกจากการพัฒนาในอดีตจึงเป็นเพียงแค่การประมาณเวลา (Time Estimate) และจะใช้การประเมินเวลาจริงสำหรับการพัฒนาแบบไมโครฟรอนต์เอนส์ เนื่องจากในการเสนอแนวคิดในการใช้ไมโครฟรอนต์เอนส์นั้นมีการพัฒนาที่ไม่ขึ้นตรงต่อโค้ด หรือแม้กระทั่งกรอบงานของแต่ละทีม ดังนั้น ทางผู้วิจัยจึงมีสมมติฐานว่าไมโครฟรอนต์เอนส์ จะสามารถช่วยลดเวลาในการพัฒนาได้ไม่มากก็น้อย
- **Test Time (Man-hour)** คือ เวลาทั้งหมดที่ใช้ไปในการทดสอบ เนื่องจากไมโครฟรอนต์เอนส์มีการทดสอบที่แยกจากกันแบบ end-to-end ดังที่กล่าวมาข้างต้น ทางทีมพัฒนาจึงใช้วิธีการบันทึกการทดสอบรวมถึงบันทึกความผิดพลาดของการพัฒนาในขณะพัฒนาเพื่อสะท้อนถึงเวลาที่ใช้ในการทดสอบ

นอกจากนี้ได้ทำการสำรวจความคิดเห็น (Survey) ของทีมงานถึงความยากในการร่วมกันพัฒนาและการสื่อสาร ค่าการประเมินแบ่งเป็นระดับ ง่าย ปานกลาง ยาก

5.3 วิธีการวัด (Measure Method)

5.3.1 Component Dependency

ในการพัฒนาส่วนของโมโนลิทิกเนื่องจากยังคงใช้การพัฒนาด้วย Angular js version 1 ซึ่งเป็นกรอบงานรุ่นเก่า เมื่อมีแก้ไขเพิ่มเติมทางทีมผู้พัฒนาจำเป็นต้องแก้ไขไปยัง Parent ของคอมโพเนนต์ เนื่องจากรูปแบบการพัฒนาแบบเดิมไม่ได้พัฒนาด้วยพื้นฐานของคอมโพเนนต์เป็นหลัก ทำให้ต้องทำการแก้ไขย้อนขึ้นไปยัง Parent ของคอมโพเนนต์ เพื่อให้ระบบทั้งหมดยังคงทำงานได้ซึ่งการเพิ่มเติมหน้า Document Detail ในฝั่งของโมโนลิทิกนั้นเกิดผลกระทบระหว่าง 5 คอมโพเนนต์ ดังภาพที่ 24



ภาพที่ 24 Component Dependency ของโมโนลิทิก

กรณีพัฒนาเพิ่มคอมโพเนนต์ Document Detail ด้วยสถาปัตยกรรมเดิมโมโนลิทิก รายการคอมโพเนนต์ที่ได้รับผลกระทบ ประกอบด้วย

- คอมโพเนนต์ Detail Page เนื่องจากการสร้างหน้า Document Detail เพิ่มเข้ามาในระบบจึงเกิดความ Dependency เกิดขึ้น
- คอมโพเนนต์ Search Result Item เนื่องจากในแต่ละการ์ดของผลลัพธ์ในการค้นหาจะต้องสามารถเชื่อมต่อไปยังหน้า Detail Page ที่เป็นคอมโพเนนต์ที่เพิ่มเข้ามาใหม่
- คอมโพเนนต์ Search Result เนื่องจากเป็นการพัฒนาในรูปแบบเก่า ทำให้เมื่อมีการแก้ไขในคอมโพเนนต์ Search Result Item ผู้พัฒนาจำเป็นต้องแก้ไขไปยัง Parent ด้วย

- คอมโพเนนต์ Search เนื่องจากการเรียกข้อมูลจากไมโครเซอร์วิสจะอยู่ที่ระดับ Page Level
  จึงมีการแก้ไขให้คอมโพเนนต์ Search ส่งข้อมูลกลับมาตามลำดับ
- คอมโพเนนต์ Main Layout เนื่องจากมีการแก้ไข Search Result ทำให้ผู้พัฒนาต้องทำการตั้งค่าให้สามารถเข้าถึง URL/detail ในส่วนของ Detail Page ได้

ในทางตรงกันข้ามไมโครฟรอนต์เอนส์มีการพัฒนาแบบโมดูล ซึ่งในแต่ละทีมมีความอิสระต่อกัน การที่ผู้พัฒนาได้ทำการเพิ่ม Detail Page มาแล้วนั้น การที่ผู้ใช้ต้องการมายังหน้า Detail Page ได้นั้นทางทีมผู้พัฒนาจำเป็นต้องแก้ไขหน้า Search Result Page ซึ่งได้พัฒนาคอมโพเนนต์ Search Result Item ไว้ ดังนั้นจึงต้องทำการแก้ไขเพียงคอมโพเนนต์เดียว โดยการเพิ่มการนำทางไปยัง URL ของ Detail Page ดังภาพที่ 25

ภาพที่ 25 Component Dependency ของไมโครฟรอนต์เอนส์

จากที่กล่าวมาข้างต้นทำให้ในส่วนของไมโครฟรอนต์เอนส์จะมี Component Dependency เกิดขึ้นเพียงแค่ 2 คอมโพเนนต์เท่านั้นสำหรับการเพิ่มหน้า Detail Page ซึ่งคือคอมโพเนนต์ดังต่อไปนี้

- คอมโพเนนต์ Search Result Item เนื่องจากต้องมีการแก้ไขในการใส่ URL ในส่วนของ Document ID เพิ่มเข้ามา
- คอมโพเนนต์ Detail Page เนื่องจากการสร้างหน้า Document Detail เพิ่มเข้ามา ในระบบจึงเกิดความ Dependency เกิดขึ้น

5.3.2 Development Time (Man-hour)

- เวลาที่ใช้ในการพัฒนาในส่วนของโมโนลิทิก ดังตารางที่ 1

ตารางที่ 1 Development Time ของการพัฒนาแบบโมโนลิทิก

ทีม	จำนวนคนในทีม	จำนวนชั่วโมงการทำงานต่อวัน	จำนวนวันที่ใช้ในการทำงาน	รวมเวลาทั้งหมด
ทีม Webstie	3	3	15	135
ทีม Authentication	2	3	13	78
ทีม Seach	3	3	20	180
ทีม Document	3	3	21	189
			รวมทั้งหมด	**582**

- เวลาที่ใช้ในการพัฒนาในส่วนของไมโครฟรอนต์เอนส์ ดังตารางที่ 2

ตารางที่ 2 Development Time ของการพัฒนาแบบไมโครฟรอนต์เอนส์

ทีม	จำนวนคนในทีม	จำนวนชั่วโมงการทำงานต่อวัน	จำนวนวันที่ใช้ในการทำงาน	รวมเวลาทั้งหมด
ทีม Webstie	3	3	13	117
ทีม Authentication	2	3	10	60
ทีม Seach	3	3	17	153
ทีม Document	3	3	17	153
			รวมทั้งหมด	**483**

5.3.3 Test Time (Man-hour)

ในการทดสอบได้แบ่งการทดสอบออกเป็น 2 ส่วนหลัก คือในส่วนของการพัฒนาแบบโมโนลิทิก และการพัฒนาแบบไมโครฟรอนต์เอนส์ โดยทางผู้วิจัยได้มีการจัดทำแบบทดสอบ (Test Case) เพื่อทำการทดสอบทั้ง Behavior Testing ในรูปบบ end-to-end และการทำ Integration Testing ในส่วนของเซอร์วิสจากตารางที่ 3 ผลของการทดสอบในส่วนของฟีเจอร์ที่มีอยู่แล้วในการพัฒนาแบบโมโนลิทิก ซึ่งประกอบด้วย Landing Page, Authentication Page, Search Page และ Search

Result Page จะใช้วิธีการประมาณค่า (Estimate) จากที่เคยทดสอบในอดีต แต่ในส่วนของฟีเจอร์ใหม่ Detail Page จะใช้การวัดค่าจริงจากการพัฒนา ซึ่งจะเป็นการวัดค่าจริงเช่นเดียวกับแบบไมโครเซอร์วิส

ตารางที่ 3 เปรียบเทียบจำนวนชั่วโมงการทดแบบทดสอบระหว่างการพัฒนาแบบโมโนลิทิกและการพัฒนาแบบไมโครฟรอนต์เอนส์

แบบทดสอบ (Test Case)	**โมโนลิทิก** (ชั่วโมง)	**ไมโครฟรอนต์เอนส์** (ชั่วโมง)
ทีม Website		
Landing Page สามารถแสดงเนื้อหาได้	0.5	0.5
Landing Page สามารถส่ง User Behavior Tracking ได้	1.5	1
Landing Page สามารถแสดงข้อมูล Testimonial จาก Landing Service ได้	1	1
ทีม Authentication		
สามารถทำการ Authentication ด้วย Local Login ได้	1	0.5
สามารถทำการ Authentication ด้วย Facebook Login ได้	1	0.5
สามารถทำการ Authentication ด้วย Google Login ได้	1	0.5
สามารถทำการกู้คืนรหัสผ่านด้วย Local Login	1.5	1
ทีม Search		
สามารถรับค่า Search Criteria จาก Court Service	3	3
สามารถรับค่า Search Criteria จาก Legal Rule Service	2	2
สามารถรับค่าและเลือก Search Criteria ได้	2	1
สามารถส่งข้อมูล Search Criteria ได้	2	1
ทีม Search result		
สามารถรับค่า Search Result จาก Search Service โดยใช้ Criteria ได้	3	3
สามารถรับค่า Keywords จาก Search Service โดยใช้	3	3

Criteria ได้		
สามารถแสดง Search Result Items	2	1
สามารถแสดงส่วนของการคัดกรองและสามารถคัดกรองได้	2	1
สามารถแสดงในส่วนของ Keywords ได้	2	0.5
สามารถคลิก Search Result เพื่อไปยังหน้า Detail Page ได้	1	1
ทีม Document		
สามารถรับค่า Document ด้วย ID จาก Detail Service	3	3
สามารถแสดงหน้า Document Detail ได้	4	2
ผลรวม	**39.5**	**26.5**

ตารางที่ 4 สรุปเปรียบเทียบผลค่าตัววัดทั้งสามระหว่างการพัฒนาแบบโมโนลิทิกและไมโครฟรอนต์เอนส์

ตารางที่ 4 ผลสรุปค่าตัววัดระหว่างการพัฒนาฟีเจอร์ใหม่แบบโมโนลิทิกและไมโครฟรอนต์เอนส์

รายละเอียดที่ใช้ในการเปรียบเทียบ	**ฟรอนต์เอนด์แบบโมโนลิทิก**	**ไมโครฟรอนต์เอนส์**
Component Dependency	5	2
Development Time (Man-hour)	483	582
Test Time (Man-hour)	39.5	26.5

การวิเคราะห์ประเมินผลค่าตัววัด มีดังนี้

- **Component Dependency** ทีมผู้พัฒนาได้ทำการนับจำนวนคอมโพเนนต์ที่เกี่ยวข้องในการเพิ่มหน้า Document Detail เป็นอีก 1 ฟีเจอร์ ซึ่งหากพัฒนาด้วยฟรอนต์เอนด์แบบโมโนลิทิก ทางผู้พัฒนาจำเป็นต้องสร้างคอมโพเนนต์ใหม่ขึ้นมาเป็น Document View สำหรับนำมาผสานเข้ากับส่วนต่อประสานผู้ใช้เดิม ซึ่งการจัดทำพัฒนาคอมโพเนนต์ Document View ขึ้นมาใหม่ จะต้องทำการแก้ไขคอมโพเนนต์จำนวน 2 ตัว โดยเริ่มจากคอมโพเนนต์ Search Result Items เพื่อเพิ่มการ Routing มายังหน้า Document Detail แต่คอมโพเนนต์นี้ถูกพัฒนาด้วย Angular เวอร์ชันเก่า ซึ่งการสร้างเว็บคอมโพเนนต์จะยังคงมีประสิทธิภาพที่ด้อยกว่าการพัฒนาด้วย Angular เวอร์ชันใหม่

ส่งผลให้ทางทีมพัฒนาจำเป็นต้องทำการแก้ไขคอมโพเนนต์ Parent ของ Search Results Items ด้วย กล่าวคือต้องทำการแก้ไขคอมโพเนนต์ Search Result เพื่อให้สามารถทำการส่งค่า Document ID ไปยัง Search Result Item ได้ ในทางตรงกันข้าม หากพัฒนาด้วยไมโครฟรอนต์เอนส์ ทางทีมผู้พัฒนาจะต้องพัฒนาเพิ่มเพียงสร้างไมโครฟรอนต์เอนส์ เพิ่มขึ้นมาใหม่และเพียงพัฒนาเพิ่ม Routing URL เพื่อให้เชื่อมมาสู่หน้าไมโครฟรอนต์เอนส์ที่คอมโพเนนต์ Search Result Item

- **Development Time** เป็นการพิจารณาจากการทำ Sprint Planning ในอดีตและการวัดจากประสิทธิภาพในการพัฒนา เนื่องจากไมโครฟรอนต์เอนส์ที่ทางทีมผู้พัฒนาสามารถสร้างแต่ละไมโครฟรอนต์เอนส์แบบขนานกันได้เลย โดยไม่ต้องรอหน้าอื่นเสร็จก่อน แต่จะต้องใช้เวลาเพิ่มในการนำไมโครฟรอนต์เอนส์แต่ละตัวมารวมกัน ซึ่งทางผู้วิจัยได้ให้ความเห็นว่า หากเป็นการพัฒนาในระบบที่สามารถแบ่งคอมโพเนนต์และทำงานพร้อมกันได้รวมถึงเปรียบเทียบด้วยการพัฒนาระบบตั้งแต่เริ่มต้นจัดทำไมโครฟรอนต์เอนส์จะใช้เวลาในการพัฒนาน้อยกว่าพัฒนาด้วยโมโนลิทิกแบบมีนัยสำคัญ
- **Test Time** เนื่องจากไมโครฟรอนต์เอนส์สามารถทำการทดสอบได้แบบ end-to-end ตามไมโครฟรอนต์เอนส์ที่แต่ละทีมรับผิดชอบ ซึ่งจะลดความผิดพลาดในการทดสอบครั้งสุดท้ายก่อนการใช้งานจริงได้ค่อนข้างมาก

5.3.4 สำรวจความคิดเห็น (Survey)

ทางผู้วิจัยได้จัดทำแบบสอบถามเพื่อสำรวจความคิดเห็นของทีมผู้พัฒนา เพื่อวัดผลเปรียบเทียบระหว่างการพัฒนาแบบโมโนลิทิกและการพัฒนาในรูปแบบไมโครฟรอนต์เอนส์ โดยผลสำรวจความคิดเห็นระดับความยากง่ายในด้านต่างๆ ดังตารางที่ 5

ตารางที่ 5 ผลการสำรวจ

	การพัฒนาแบบโมโนลิทิก	การพัฒนาแบบไมโครฟรอนต์เอนส์
มีความรู้ความสามารถเหมาะสมกับงานที่ได้รับมอบหมาย	มาก	ปานกลาง
ประเมินความยาก-ง่ายของการพัฒนางานในส่วนนี้	ง่าย	ปานกลาง
งานที่รับผิดชอบส่งเสริมให้พัฒนาตนเอง มีความรู้	น้อย	มาก

ความสามารถเพิ่มขึ้น		
ท่านสามารถแลกเปลี่ยนความคิดเห็นและให้ข้อเสนอแนะในการปฏิบัติงานกับเพื่อนร่วมงานได้	น้อย	ปานกลาง

จากการสำรวจและสอบถามซึ่งทุกทีมพัฒนาให้ผลที่ค่อนข้างตรงกัน คือ ไม่คุ้นเคยกับการพัฒนาโครงการในรูปแบบไมโครฟรอนต์เอนส์ แต่วิธีการพัฒนาไม่ได้ยากเกินไป เมื่อเปรียบเทียบกันกับการพัฒนาแบบแบบโมโนลิทิก ซึ่งจะค่อนข้างง่ายกว่าในการพัฒนา เพียงแต่จะค่อนข้างซับซ้อนกว่าในการปรับแก้ไขหรือเพิ่มเติมฟีเจอร์ใหม่ เนื่องจากใช้เวลาในการทดสอบและพัฒนามากกว่าการพัฒนาแบบไมโครฟรอนต์เอนส์ ซึ่งการพัฒนาแบบไมโครฟรอนต์เอนส์ช่วยให้ทีมผู้พัฒนาได้ฝึกฝนความสามารถเพิ่มเติมมากขึ้น

บรรณานุกรม

1. ThoughtWorks. *Technology Radar: Micro frontends*. 2016 Jan, 2021]; Available from: https://www.thoughtworks.com/radar/techniques/micro-frontends.
2. Geers, M., *Micro Frontends-Extending the microservice idea to frontend development.* Accessed on, 2017.
3. Geers, M., *Micro Frontends in Action*. 2020: Manning Publications.
4. Zafer, Ö. *Understanding Micro-frontends*. Jan 2021]; Available from: https://hackernoon.com/understanding-micro-frontends-b1c11585a297.
5. Yang, C., C. Liu, and Z. Su. *Research and application of micro frontends*. in *IOP Conference Series: Materials Science and Engineering*. 2019. IOP Publishing.
6. Pavlenko, A., et al., *Micro-frontends: application of microservices to web front-ends.* Journal of Internet Services and Information Security (JISIS), 2020. **10**(2): p. 49-66.
7. Abbott, M.L. and M.T. Fisher, *The art of scalability: Scalable web architecture, processes, and organizations for the modern enterprise*. 2015: Addison-Wesley Professional.
8. A., B. *Microservices vs. Monolith Architecture*. Jan 2021]; Available from: https://dev.to/alex_barashkov/microservices-vs-monolith-architecture-4l1m.
9. Malavalli, D. and S. Sathappan. *Scalable microservice based architecture for enabling DMTF profiles*. in *2015 11th International Conference on Network and Service Management (CNSM)*. 2015. IEEE.
10. W3C. *w3c/webcomponents*. 2019 Jan 2021]; Available from: https://github.com/WICG/webcomponents.

ประวัติผู้เขียน

ชื่อ-สกุล	Nattaporn Noppadol
วัน เดือน ปี เกิด	29 May 1991
สถานที่เกิด	Bangkok
ที่อยู่ปัจจุบัน	111/46 M.10 Samrong, Prapa Daeng, Samut Prakan 10130