



รายงานการปฏิบัติสหกิจศึกษา
การพัฒนาระบบเพื่อการให้บริการด้านการเงิน
(Fintech Services)

โดย

นาย สรวิศ นามสีธีราน

โครงการนี้เป็นส่วนหนึ่งของการปฏิบัติสหกิจศึกษาตามหลักสูตร
วิทยาศาสตรบัณฑิต สาขาวิชาการคอมพิวเตอร์
คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์
ปีการศึกษา 2566

(1)

รายงานการปฏิบัติสหกิจศึกษา
การพัฒนาระบบที่ดีของการให้บริการด้านการเงิน

โดย

นาย สรวิศ นามสีธาน

โครงการนี้เป็นส่วนหนึ่งของการปฏิบัติสหกิจศึกษาตามหลักสูตร
วิทยาศาสตรบัณฑิต สาขาวิชาบริหารคอมพิวเตอร์
คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์
ปีการศึกษา 2566

มหาวิทยาลัยธรรมศาสตร์
คณะวิทยาศาสตร์และเทคโนโลยี

รายงานการปฏิบัติสหกิจศึกษา

ของ

นาย สรวิศ นามสีฐาน

เรื่อง

การพัฒนาระบบเพื่อการให้บริการด้านการเงิน

ได้รับการตรวจสอบและอนุมัติ ให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

เมื่อ วันที่ 8 ธันวาคม พ.ศ. 2566

อาจารย์ที่ปรึกษาสหกิจศึกษา

(อ.ดร. นวฤกษ์ ชลาธักษ์)

อาจารย์ที่ปรึกษาร่วมสหกิจศึกษา

(ชื่อตำแหน่งทางวิชาการ ชื่อ ชื่อสกุลออาจารย์)

อาจารย์ที่ปรึกษาร่วมสหกิจศึกษา

(ชื่อตำแหน่งทางวิชาการ ชื่อ ชื่อสกุลออาจารย์)

พนักงานที่ปรึกษา

(Ankita Kujur)

Software Engineer

บทคัดย่อ

บริษัท อโกร์ด้า เซอร์วิสเซส จำกัด เป็นหนึ่งในบริษัทที่ทำเกี่ยวกับแพลตฟอร์มการจองการเดินทางออนไลน์ ที่เติบโตอย่างรวดเร็ว โดยมีที่กำเนิดมาจากลิงค์ปอร์ และเติบโตก้าวหน้าอย่างต่อเนื่องจนก้าวสู่ระดับสากล

ในปัจจุบันบริษัท อโกร์ด้า เซอร์วิสเซส จำกัด มีเครือข่ายที่พัฒนาไปกว่า 2 ล้านแห่งในกว่า 200 ประเทศทั่วโลก เพื่อให้นักท่องเที่ยวได้เลือกจองเที่ยวบิน ที่พัก อาหาร เมนู บ้าน ฯลฯ ที่ตรงกับงบประมาณและความต้องการในการเดินทางของนักท่องเที่ยว โดยมุ่งหวังที่จะช่วยให้ทุกคนสามารถเดินทางท่องเที่ยวได้ง่ายขึ้นในราคาน้ำเงิน พร้อมทั้งสนับสนุนกับการเดินทางมากขึ้น

จากการที่ได้เข้าปฏิบัติสหกิจศึกษาในบริษัท อโกร์ด้า เซอร์วิสเซส จำกัด ได้รับมอบหมายให้ปฏิบัติงานในแผนกการเงิน ซึ่งในการเข้าไปปฏิบัติงานนั้น ข้าพเจ้าได้รับมอบหมายให้ทำการพัฒนาระบบที่ใช้ในการให้บริการด้านการเงิน โดยพัฒนาทั้งระบบที่มีอยู่ให้มีมากยิ่งขึ้น และพัฒนาระบบใหม่ๆ เพื่อให้ฝ่ายการเงินของบริษัท อโกร์ด้า เซอร์วิสเซส จำกัด สามารถจัดการข้อมูลจำนวนมากทางด้านการเงินได้อย่างมีประสิทธิภาพมากขึ้น

คำสำคัญ: Grafana, Hadoop, Spark, Scala, Endpoint

กิตติกรรมประกาศ

การปฏิบัติสหกิจและการทำโครงการพัฒนาระบบที่ให้บริการด้านการเงิน หรือ Fintech-services นี้ทำให้ได้ประสบการณ์และได้เรียนรู้สิ่งใหม่ ๆ ที่หาจากในห้องเรียนไม่ได้ จะนำประสบการณ์และความรู้นี้ไปปรับใช้กับการทำงานในอนาคต การปฏิบัติสหกิจศึกษานี้สามารถสร้างความช่วยเหลือ คำแนะนำ จากพี่ ๆ พนักงานที่ปรึกษา ได้แก่ Ankita Kujur , พี่ที่ ภารณ์ ธนประชานนท์ และเพื่อนๆ ในทีมฝึกงานได้แก่ การัน, มด, เอ็ม, ภูมิ, ออมร และ นัท รวมถึงพนักงานในบริษัท โกลเด้า เชอร์วิสเซส จำกัด ทุกท่าน

ขอขอบพระคุณอาจารย์นวนฤทธิ์ ชา拉รักษ์ ออาจารย์ที่ปรึกษาสหกิจ ที่คอยให้คำปรึกษา คำแนะนำ ช่วยเหลือในการปฏิบัติสหกิจศึกษาตั้งแต่เริ่มต้นจนทำให้รายงานฉบับนี้สามารถสร้างสรรค์ได้ดี

สุดท้ายนี้ ขอขอบคุณครอบครัวและมายmany ที่ให้การสนับสนุนในทุกด้านและให้กำลังใจที่ดีมาเสมอ ตลอดจนผู้ที่มีส่วนเกี่ยวข้องทุกท่าน ที่ได้ให้คำแนะนำต่างๆ และการช่วยเหลือจนสามารถทำโครงการนี้ได้สำเร็จลุล่วง

นายสุริศ นามสุธรรม

สารบัญ	
	หน้า
บทคัดย่อ	3
กิตติกรรมประกาศ	4
สารบัญ	5
สารบัญตาราง	7
สารบัญภาพ	8
รายการสัญลักษณ์และคำย่อ	9
บทที่ 1 บพนฯ	1
1.1 ความเป็นมาและความสำคัญของโครงงาน	1
1.1.1 หัวข้ออย่างระดับที่ 1 (Heading 3)	Error! Bookmark not defined.
1.2 วัตถุประสงค์	2
1.3 ขอบเขตของโครงงาน	3
1.3.1	Error! Bookmark not defined.
1.4 ประโยชน์ของโครงงาน	4
บทที่ 2 วรรณกรรม งาน และเทคโนโลยีที่เกี่ยวข้อง	5
2.1 แนวคิดทฤษฎีที่เกี่ยวข้อง	5
2.1.1 หัวข้ออย่างระดับที่ 1 (Heading 3)	Error! Bookmark not defined.
2.2 ระบบงานปัจจุบัน /งานที่เกี่ยวข้อง	Error! Bookmark not defined.
2.2.1 หัวข้ออย่างระดับที่ 1 (Heading 3)	Error! Bookmark not defined.

บทที่ 3 การดำเนินงาน	27
3.1 ภาพรวมของโครงงาน	27
3.1.1 หัวข้ออย่างระดับที่ 1 (Heading 3)	Error! Bookmark not defined.
3.1.2 หัวข้ออย่างระดับที่ 1 (Heading 3)	Error! Bookmark not defined.
3.2 ขั้นตอนการดำเนินงาน	29
บทที่ 4 ผลการดำเนินงาน	Error! Bookmark not defined.
4.1 ผลการดำเนินงาน	Error! Bookmark not defined.
4.2 ข้อเสนอแนะ/ปรับปรุงในอนาคต	Error! Bookmark not defined.
บทที่ 5 วิเคราะห์และสรุปผล	67
บรรณานุกรม	77
ภาคผนวก	78
ภาคผนวก ก. ชื่อภาคผนวก	79
ภาคผนวก ข. ชื่อภาคผนวก	80
ภาคผนวก ค. ชื่อภาคผนวก	81

สารบัญตาราง

	หน้า
ตารางที่ 1.1 คำอธิบายตาราง	Error! Bookmark not defined.
ตารางที่ 4.1 การดำเนินงานที่ผ่านมาจนถึงปัจจุบัน	Error! Bookmark not defined.
ตารางที่ 4.2 แผนการดำเนินงานในอนาคต	66

(8)

สารบัญภาพ

หน้า

ภาพที่ 2.1 คำอธิบายภาพ

Error! Bookmark not defined.

ภาพที่ 2.2 คำอธิบายภาพ

Error! Bookmark not defined.

(9)

รายการสัญลักษณ์และคำย่อ

สัญลักษณ์/คำย่อ	คำเต็ม/คำจำกัดความ
FBO	Finance Backoffice
ETL	Extract, transform, and load
HDFS	Hadoop Distributed File System
AVL	Asset vs Liabilities
SP	Stored Procedures
HUE	Hadoop User Experience
พิมพ์สัญลักษณ์/คำย่อ	พิมพ์คำจำกัดความ/คำเต็ม

บทที่ 1 บทนำ

1.1 ความเป็นมาและความสำคัญของโครงการ

เนื่องจากในฝ่ายการเงินขององค์กร มีข้อมูลการเงินจำนวนมากที่ถูกสร้างขึ้นในทุกวัน และยังมีไฟล์เอกสารจำนวนมากที่ถูกสร้างขึ้น และถูกบันทึกเข้าสู่ฐานข้อมูล ในทุกๆชั่วโมง เนื่องจากข้อมูลมีจำนวนมากถูกสร้างขึ้นตลอดเวลา และเนื่องจากทาง บริษัท อโภคิ้า เชอร์วิสเซส จำกัด เป็นบริษัทที่มีผู้ใช้งานจำนวนมากในทุกๆสาขาที่จึงมีรายการธุรกรรมจำนวนมาก ซึ่งทำให้มีธุรกรรมบางรายการที่อาจจะตกหล่น หรือข้อมูลไม่ถูกต้องอยู่ และการสร้างไฟล์รายงานธุรกรรมทางการเงินที่ไม่สมบูรณ์ หรือไม่สามารถสร้างไฟล์รายงานได้ตามที่ควรจะเป็น โดยทางองค์กรณี้มีการสร้างไฟล์เป็นระบบอัตโนมัติอยู่แล้ว ทำให้อาจจะมีการตกหล่นในการตรวจสอบบางไฟล์ที่ไม่ถูกสร้างอย่างถูกต้อง

เพื่อแก้ไขปัญหาดังกล่าว ทางผู้จัดทำรายงานจึงได้พัฒนาระบบที่ช่วยเหลือฝ่ายการเงินขององค์กรณี้ให้สามารถทำงานได้สะดวกขึ้นและมีความแม่นยำ และลดข้อผิดพลาดของธุรกรรมทางการเงินโดยแบ่งออกเป็น 3 ระบบดังนี้

1.ระบบสร้างไฟล์รายงานสำหรับทรัพย์สินและหนี้สินใหม่ หรือ AVL Regenerate (asset vs liabilities) ซึ่งเป็นรายงานทางการเงินที่มีทรัพย์สิน (Asset) และ หนี้สิน (Liability) โดยทางองค์กรณี้มีระบบอัตโนมัติเพื่อช่วยเหลือในการสร้างรายงาน AVL อยู่แล้ว แต่จะทำงานตามกำหนดเวลา เช่นทุกๆวัน หรือทุกๆเดือน แต่เนื่องจากข้อมูลมีการเปลี่ยนแปลงตลอดเวลา เพื่อช่วยฝ่ายการเงินขององค์กร ทางผู้จัดทำรายงานจึงได้สร้างระบบเพื่อช่วยเหลือให้ฝ่ายการเงินสามารถกดเพื่อสร้างรายงานได้ทันทีตลอดเวลาโดยไม่ต้องรอเวลาที่ระบบอัตโนมัติจะทำงาน และยังสามารถสร้างรายงานย้อนหลังได้อีกด้วย ซึ่งระบบนี้จะช่วยแก้ปัญหาของฝ่ายการเงินขององค์กรณี้ให้สามารถเข้าถึงรายงานได้ทุกเมื่อ เมื่อต้องการนำข้อมูลไปวิเคราะห์ต่อ โดยไม่จำเป็นต้องรอตามกำหนดเวลาอัตโนมัติ

2.ระบบตรวจสอบและแสดงผลการสร้างรายงานสำคัญทางการเงิน หรือ Reports Monitoring โดยระบบตรวจสอบการสร้างรายงาน จะให้ผู้ใช้สามารถเพิ่มรายงานที่ต้องการตรวจสอบ ว่าถูกสร้างขึ้นหรือไม่ได้ โดยตัวระบบจะตรวจสอบทุกๆ 10 นาที โดยให้ผู้ใช้ใส่กำหนดการของไฟล์ว่า จะถูกสร้างเมื่อใด เช่นทุกๆวัน ทุกๆเดือนเป็นต้น และเมื่อถึงเวลานั้น ระบบ Reports Monitoring จะตรวจสอบไฟล์ว่าถูกสร้างขึ้นหรือไม่ โดยจะอ่านไฟล์จากที่อยู่ที่ผู้ใช้กรอกเอาไว้ และแจ้งเตือนไปยังผู้ใช้กรณีที่ไฟล์ไม่ถูกสร้างตามกำหนดเวลา รวมไปถึงข้อมูลจะถูกแสดงบน Grafana Dashboard ซึ่งเป็นระบบแสดงกราฟ หรือแผนภาพของข้อมูล เพื่อให้ผู้ใช้สามารถตรวจสอบสถานะของไฟล์ได้ทันที

3. ระบบตรวจสอบความถูกต้องของธุรกรรมทางการเงิน หรือ Missing Bookings โดยระบบนี้ จะดึงข้อมูลที่จำเป็นในฐานข้อมูลมา และนำมาเปรียบเทียบระหว่างผู้ให้บริการจ่ายเงิน เช่น ธนาคาร, visa หรืออื่นๆ กับฐานข้อมูลขององกรณ์ว่ามีความสอดคล้องกันหรือไม่ ถ้ามีธุรกรรมที่ไม่ สอดคล้องกัน ซึ่งหมายความว่าอาจจะเกิดข้อผิดพลาดบางอย่าง ตัวระบบตรวจสอบ จะเปรียบเทียบ ข้อมูลต่างๆ และหาเหตุผลว่าเกิดขึ้นจากอะไร เพื่อให้ฝ่ายการเงินสามารถนำไปตรวจสอบต่อได้ทันที โดยโครงงานนี้สร้างใน Fintech Services โดยระบบต่างๆ จะเชื่อมต่อกับฐานข้อมูล และ เซิฟ เวอร์ที่เก็บรายงานทางการเงินต่างๆ มีการใช้ Workflow มาเพื่อช่วยในการทำงานของระบบให้เป็น ระบบเปรียบ รวมถึงมีการใช้ HDFS หรือ Hadoop ในการทำงานอีกด้วย เพื่อใช้ในการดึงข้อมูล และ ค้นหา รวมข้อมูลจากฐานข้อมูลจำนวนมาก เพื่อให้มีประสิทธิภาพและรวดเร็วในการค้นหาข้อมูล และ ในแต่ละระบบ จะมีการแสดงผลโดยใช้ Grafana Dashboard ซึ่งเป็นระบบสำหรับแสดงกราฟ แผนภาพของข้อมูลได้อย่างระเอียด เพื่อให้ผู้ใช้สามารถตรวจสอบข้อมูลได้ทันที ซึ่งเป็นประโยชน์ สำคัญในการตัดสินใจและช่วยแก้ปัญหาที่กล่าวมาข้างต้นได้

1.2 วัตถุประสงค์

โครงงานนี้มีเป้าหมายเพื่อพัฒนา ระบบสร้างไฟล์รายงานสำหรับทรัพย์สินและหนี้สินใหม่ระบบ ตรวจสอบและแสดงผลการสร้างรายงานสำคัญทางการเงิน และ ระบบตรวจสอบความถูกต้องของ ธุรกรรมทางการเงิน โดยกำหนดวัตถุประสงค์ของโครงงานดังต่อไปนี้

1. เพื่อช่วยให้ฝ่ายการเงินสามารถสร้างรายงานทรัพย์สินและหนี้สินขององค์กรได้ ทันทีเมื่อต้องการ
2. เพื่อเพิ่มประสิทธิภาพในการสร้างรายงาน เพื่อเป็นไปตามความต้องการของฝ่าย การเงิน หรือผู้ใช้
3. เพื่อเพิ่มประสิทธิภาพในการตรวจสอบธุรกรรมทางการเงินที่มีจำนวนเยอะมาก ภายในฝ่ายการเงิน ช่วยให้ตรวจสอบได้ง่ายขึ้น
4. เพื่อลดข้อผิดพลาดในการตรวจสอบข้อมูลธุรกรรมทางการเงิน
5. เพื่อเพิ่มประสิทธิภาพในการตรวจสอบไฟล์รายงานที่ถูกสร้างขึ้นในฝ่ายการเงิน ในทุกวัน ว่ามีการดำเนินการอย่างถูกต้องหรือไม่

6. เพื่อบริการจัดการอย่างมีประสิทธิภาพของไฟล์รายงานที่ถูกสร้างขึ้น
7. เพื่อการรายงานผล และแจ้งเตือนผู้ใช้เมื่อพบข้อผิดพลาดในการทำงานของธุกรรมทางการเงิน หรือการสร้างไฟล์รายงานทางการเงิน
8. เพื่อช่วยให้ผู้ใช้สามารถตรวจสอบข้อมูลได้ทันที โดยตรวจสอบผ่านแพนก้าฟ Grafana Dashboard ได้ทันที

1.3 ขอบเขตของโครงการ

ผู้จัดทำโครงการ ได้ทำระบบเพื่อแก้ไขปัญหาโดยทางผู้จัดทำโครงการได้พัฒนาระบบทั้งหมด 3 ระบบ ได้แก่ ระบบสร้างไฟล์รายงานสำหรับทรัพย์สินและหนี้สินใหม่ ระบบตรวจสอบและแสดงผลการสร้างรายงานสำคัญทางการเงิน และระบบตรวจสอบความถูกต้องของธุกรรมทางการเงิน

ระบบสร้างไฟล์รายงานทรัพย์สินและหนี้สินใหม่ หรือ Regenerate AVL เป็นระบบที่ให้ผู้ใช้ฝ่ายการเงินขององค์กรสามารถสร้างรายงานสำคัญทางการเงินใหม่ได้เอง โดยไม่จำเป็นต้องรอให้ระบบอัตโนมัติทำงาน โดยระบบจะถูกส่งให้ทำงานผ่าน Finance Backoffice ซึ่งเป็นเวปไซต์สำหรับให้ผู้ใช้สามารถเรียกใช้ฟังชั่นต่างๆ ใน Fintech Services ได้ ซึ่งเมื่อผู้ใช้ส่งข้อมูล วันที่ที่ต้องการสร้างรายงานใหม่มา ระบบ AVL Regenerate จะตรวจสอบข้อมูล และส่งให้ Spark Job ทำงาน เพื่อสร้างรายงาน

- ระบบสามารถสร้างรายงานใหม่ได้ทันที
- ระบบสามารถเลือกวันที่ของข้อมูลเพื่อสร้างรายงานได้ตามที่ผู้ใช้ต้องการ

ระบบตรวจสอบและแสดงผลการสร้างรายงานสำคัญทางการเงิน หรือ Report Monitoring เป็นระบบเพื่อตรวจสอบไฟล์ที่ถูกสร้างขึ้น ระบบจะถูกตั้งเวลาให้ตรวจสอบทุกๆ 10 นาที โดยระบบจะตรวจสอบไฟล์ตามที่อยู่ของไฟล์ต่างๆ ซึ่งอ่านกำหนดการ และที่อยู่ของไฟล์ จากฐานข้อมูล และตรวจสอบว่าถูกสร้างขึ้นอย่างถูกต้องหรือไม่ และบันทึกข้อมูลการตรวจสอบไปยังฐานข้อมูล และนำมาแสดงผลใน Grafana Dashboard ซึ่งเป็นแพนก้าฟแสดงข้อมูล เพื่อให้ผู้ใช้สามารถตรวจสอบได้ทันที และถ้าไฟล์ไม่ถูกสร้างตามกำหนด ระบบจะส่งแจ้งเตือนไปยัง Slack ของฝ่ายที่เกี่ยวข้อง

- ระบบสามารถตั้งเวลาให้ทำงานได้ทุกๆ 10 นาที
- ระบบสามารถช่วยในการตรวจสอบไฟล์ว่าถูกสร้างตามกำหนดหรือไม่
- ระบบสามารถรายงานผลแบบ Real time ผ่าน Grafana Dashboard

- ระบบสามารถแจ้งเตือนไปยังฝ่ายที่เกี่ยวของผ่าน Slack กรณีที่ไฟล์ไม่ถูกสร้างตามกำหนด

ระบบตรวจสอบความถูกต้องของธุรกรรมทางการเงิน หรือ Missing booking เป็นระบบเพื่อตรวจสอบความถูกต้องของธุรกรรม โดยเราจะมีข้อมูลธุรกรรมทางการเงินจาก Gateway เช่น Visa, ธนาคาร และข้อมูลธุรกรรมทางการเงินทางฝั่งของ Agoda ที่เก็บข้อมูลธุรกรรมเอาไว้ โดยจะตรวจสอบจากข้อมูลทั้งสองฝั่งว่าเหมือนกันหรือไม่ ถ้าไม่เหมือนกันหมายความว่าอาจจะเป็นธุรกรรมที่มีข้อผิดพลาด โดยระบบสามารถตรวจสอบข้อมูลธุรกรรมต่างๆที่ผิดพลาด และสามารถค้นหาเหตุผล หรือข้อผิดพลาดว่าเกิดขึ้นจากอะไรได้ด้วย และแจ้งไปยังฝ่ายที่เกี่ยวข้องเพื่อให้ฝ่ายที่เกี่ยวข้องสามารถนำไปตรวจสอบและแก้ไขได้สะดวกมากยิ่งขึ้น

- ระบบสามารถตรวจสอบธุรกรรมที่ผิดพลาดได้
- ระบบสามารถหาเหตุผล ว่าเกิดข้อผิดพลาดที่ส่วนไหน
- ระบบสามารถรายงานข้อมูลให้ฝ่ายที่เกี่ยวข้องได้ทันที

1.4 ประโยชน์ของโครงงาน

ในการพัฒนาระบบทั้ง 3 ระบบ เพื่อช่วยให้ฝ่ายการเงินขององค์กรสามารถทำงานได้สะดวก และเร็วมากขึ้น โดยจะเกิดประโยชน์ต่อผู้ใช้งานระบบดังต่อไปนี้

1. ช่วยให้ผู้ใช้สามารถสร้างไฟล์รายงานได้ทันที ไม่ต้องรอเวลา หรือสร้างไฟล์รายงาน ทรัพย์สินและหนี้สินเอง
2. ช่วยลดความเสี่ยงและความผิดพลาดในการสร้างไฟล์รายงาน ถ้าฝ่ายการเงินสร้างเอง อาจจะเกิดข้อผิดพลาดได้
3. ช่วยเพิ่มประสิทธิภาพในการตรวจสอบไฟล์รายงาน ทำให้ผู้ใช้สามารถตรวจสอบไฟล์รายงานทั้งหมดได้ทันที โดยผ่านแพนก้าฟ Dashboards
4. ผู้ใช้ได้รับการแจ้งเตือนทันทีเมื่อไฟล์ไม่ถูกสร้างตามกำหนด ทำให้ผู้ใช้สามารถแก้ไข ข้อผิดพลาดได้ทันที
5. ช่วยลดเวลาผู้ใช้ในการตรวจสอบปัญหาของธุรกรรมว่าเกิดจากอะไร เนื่องจากในหนึ่ง ธุรกรรมมีข้อมูลที่ซับซ้อนจำนวนมาก ทำให้ผู้ใช้สามารถใช้งานและดูว่าผิดพลาดจากส่วนไหนได้ทันที

บทที่ 2

วรรณกรรม งาน และเทคโนโลยีที่เกี่ยวข้อง

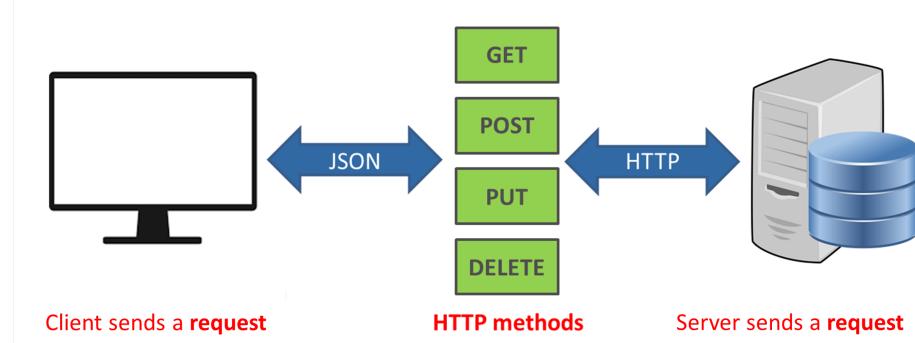
2.1 แนวคิดทฤษฎีที่เกี่ยวข้อง

2.1.1 API หรือ Application Programming interface

API คือกลไกที่ช่วยให้ส่วนประกอบซอฟต์แวร์สองส่วนคือ คลilent (Client) และ เซิฟเวอร์ (Server) สามารถสื่อสารกันได้โดยใช้ชุดคำจำกัดความและโปรโตคอล API ทำให้โปรแกรม หรือแอปพลิเคชันต่าง ๆ สามารถเรียกใช้หรือติดต่อกับซอฟต์แวร์อื่น ๆ ได้อย่างมีระบบและปลอดภัย

2.1.1.1 REST API

REST API ย่อมาจาก Representational State Transfer Application Programming Interface หรือเป็น API ที่ใช้โปรโตคอล HTTP สำหรับการสื่อสารและแลกเปลี่ยน ข้อมูลระหว่างแอปพลิเคชันที่ต่างกัน REST ถูกออกแบบให้ง่ายต่อการใช้งานและมีโครงสร้างที่ใช้ ทรัพยากรของ HTTP อย่างเต็มที่ เช่น GET, POST, PUT, DELETE เป็นต้น

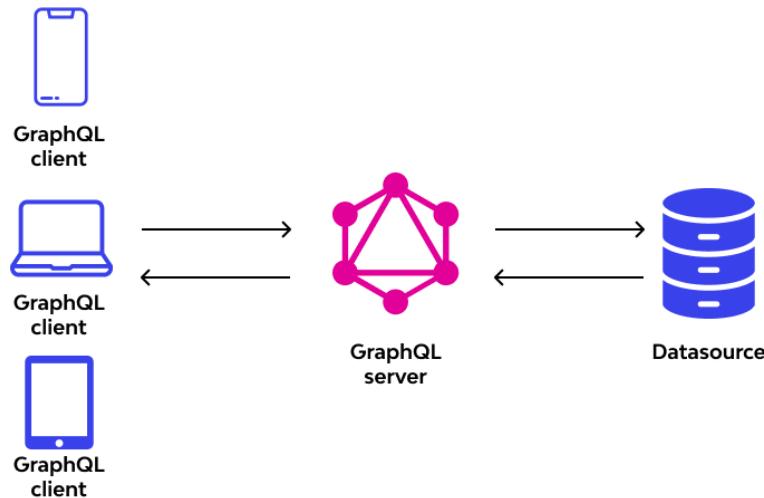


ภาพที่ 2.1 แผนภาพการทำงานของ REST API

2.1.1.2 GraphQL

GraphQL เป็นภาษาคิวเรีย (Query) และเป็น API ชนิดนึง โดยถูกพัฒนาขึ้นโดย Facebook เพื่อช่วยให้การแลกเปลี่ยนข้อมูลระหว่าง-client (Client) และ-server (Server) เป็นไปได้อย่างมีประสิทธิภาพ และยืดยุ่นมากขึ้น โดยจะมีลักษณะที่แตกต่างจาก REST API โดยผู้ใช้ สามารถร้องขอข้อมูลที่ต้องการเท่านั้นในคำขอเดียว โดยไม่ต้องรับข้อมูลทั้งหมดที่ถูกตอบคืนได้

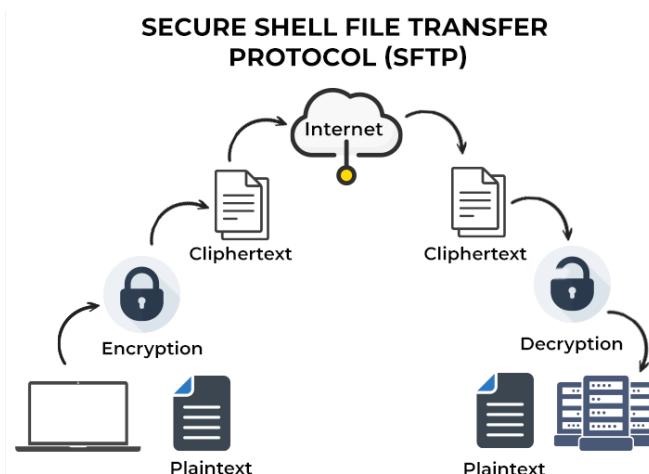
ผู้ใช้สามารถกำหนดรูปแบบข้อมูลที่ต้องการในคำขอ และเซิฟเวอร์จะคืนค่าเฉพาะข้อมูลที่ถูกขอเท่านั้น ซึ่งใน GraphQL จะมี Endpoint เดียวเท่านั้นสำหรับให้บริการ โดยผู้ใช้สามารถใช้ Endpoint นั้นในการขอข้อมูลต่างๆตามที่ผู้ใช้ หรือคลลแอนต์ (Client) ต้องการได้ผ่าน Query



ภาพที่ 2.2 แผนภาพการทำงานของ GraphQL

2.1.2 SFTP หรือ Secure File Transfer Protocol

เป็นโปรโตคอลสำหรับการเข้าถึง และถ่ายโอนไฟล์ได้อย่างปลอดภัยผ่านทางเครือข่าย โดยใช้ SSH หรือ Secure Shell เพื่อสร้างการเชื่อมต่อ กับเซิฟเวอร์ (Server) และเข้ารหัสข้อมูลระหว่างการเชื่อมต่อ ทำให้ข้อมูลที่ถูกส่งผ่านเครือข่าย มีความปลอดภัยจากการดักจับ (Sniffing) และ ป้องการข้อมูลถูกแก้ไขระหว่างส่ง อีกด้วย



ภาพที่ 2.3 แผนภาพการทำงานของ SFTP

2.1.2.1 SSH หรือ Secure Shell

SSH เป็นโปรโตคอล (Protocol) สำหรับควบคุมระยะไกล เพื่อเข้าถึงและจัดการเครื่องคอมพิวเตอร์ หรือเซิฟเวอร์ระยะไกลได้อย่างปลอดภัย เนื่องจากมีการเข้ารหัสข้อมูลในขณะที่เชื่อมต่ออยู่ และมีการตรวจสอบสิทธิ์การเข้าถึง เพื่อป้องกันการดักจับข้อมูลที่ถูกส่งผ่านเครือข่ายได้

2.1.2.2 การดักจับข้อมูล หรือ Sniffing

เป็นกระบวนการที่ผู้ไม่หวังดีพยายามดักจับข้อมูลและดูข้อมูลที่ถูกส่งผ่านเครือข่าย การดักจับสามารถทำได้โดยการใช้เครื่องมือหรือโปรแกรมที่สามารถอ่านและบันทึกข้อมูลที่ถูกส่งผ่านทางเครือข่ายได้ โดยจะสามารถอ่านข้อมูลที่ไม่ได้เข้ารหัสเอาไว้ได้ทันที

2.1.3 SMB หรือ Server Message Block

เป็นโปรโตคอล (Protocol) การแชร์ไฟล์และการเข้าถึงทรัพยากรในเครือข่าย คอมพิวเตอร์ โดยโปรโตคอลนี้มักถูกใช้ในคอมพิวเตอร์ หรือ เซิฟเวอร์ที่ทำงานบนระบบปฏิบัติการ Windows สำหรับแบ่งปันไฟล์ เครื่องพิมพ์ และทรัพยากรต่างๆภายในเครือข่าย โดยรุ่น SMB3 เป็นรุ่นที่มีความปลอดภัยและความสามารถมากที่สุด การใช้ SMB ได้แก่การเข้าถึงไฟล์แบบแชร์ในเครือข่ายที่มีระบบปฏิบัติการ Windows, การเข้าถึงทรัพยากรในเครือข่ายที่ใช้ SMB ในรูปแบบของ Samba ในระบบปฏิบัติการ Linux, และการให้บริการทรัพยากรในระบบปฏิบัติการ macOS

SMB request, response



ภาพที่ 2.4 แผนภาพการทำงานของ SMB

2.1.4 ETL หรือ Extract, Transform and Load

เป็นกระบวนการที่ใช้ในการนำเข้าข้อมูลจากต้นทาง ทำการปรับเปลี่ยนให้เป็นรูปแบบที่ต้องการเพื่อนำไปใช้ต่อ และบันทึกลงฐานข้อมูลปลายทางที่ต้องการเก็บไว้ โดยมีรายละเอียดดังนี้

2.1.4.1 Extract หรือ การดึงข้อมูล

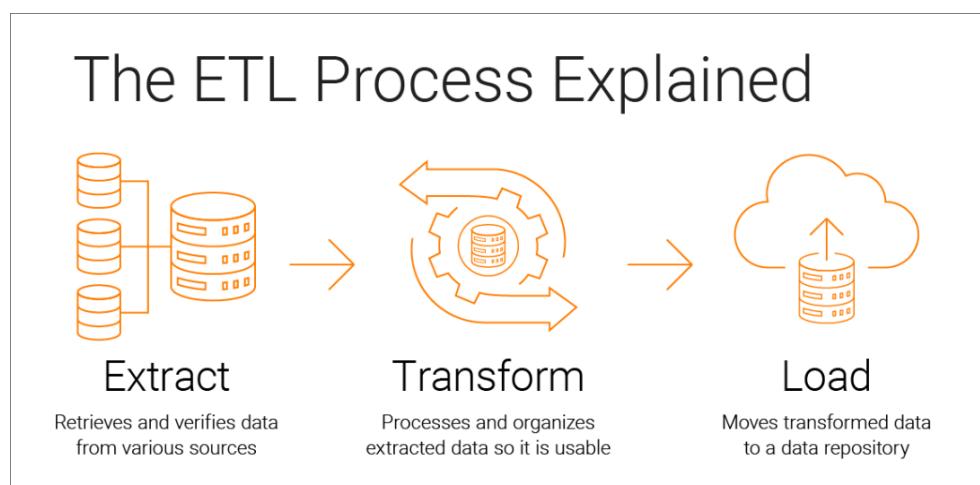
เป็นการดึงข้อมูลจากแหล่งที่เก็บข้อมูลต้นทาง เช่น ฐานข้อมูลไฟล์ Excel หรืออื่นๆ โดยการดึงข้อมูลนี้ส่วนมากจะเป็นข้อมูลที่ยังไม่ถูกเปลี่ยนแปลงมากเท่าไหร่ เพื่อสามารถนำไปใช้ต่อได้

2.1.4.2 Transform หรือ การปรับเปลี่ยน ปรับแต่ง

เป็นกระบวนการที่ใช้เพื่อปรับเปลี่ยน หรือทำความสะอาดข้อมูลที่ดึงมาจากการดึงข้อมูล เพื่อให้ได้ข้อมูลที่สามารถนำไปใช้ต่อได้ตามที่ต้องการ เช่น การคำนวณค่า การเปลี่ยนแปลงรูปแบบข้อมูล การสร้างฟิลด์ใหม่ หรือการลบข้อมูลที่ไม่ต้องการใช้ออก

2.1.4.3 Load หรือ นำเข้าข้อมูล

เป็นกระบวนการที่นำข้อมูลที่ถูกปรับเปลี่ยนแล้วนำเข้าไปไว้ระบบฐานข้อมูล หรือโครงสร้างข้อมูลที่จะนำไปใช้ต่อไป เพื่อให้สามารถนำข้อมูลไปใช้งานได้ทันที



ภาพที่ 2.5 แผนภาพการทำงานของ ETL

2.1.5 Cron Schedule

เป็นรูปแบบของเวลาที่ใช้ในระบบปฏิทิน Cron jobs ในระบบปฏิบัติการ Unix-Like เช่น Linux เพื่อกำหนดเวลาที่งานจะถูกทำงานอัตโนมัติ โดยใช้ตัวเลขและสัญลักษณ์พิเศษบุช่วงเวลา

ต่างๆ ที่ต้องการให้ทำงาน เช่น การทำงานประจำทุกๆ ชั่วโมง หรือ ทุกวันที่ 1 ของเดือนก็สามารถตั้งได้ โดยในภาษา Java มี Cron อีกรูปแบบนึง ซึ่งคือ Quartz cron เป็นไลบรารีที่ใช้เพื่อตั้งเวลาสำหรับ Java มีความยืดหยุ่นมากกว่า Unix cron เนื่องจากสามารถกำหนดได้ต่ำสุดถึงหลักวินาที โดย Unix Cron สามารถตั้งได้ถึงเพียงระดับนาทีเท่านั้น

2.1.5.1 Unix cron

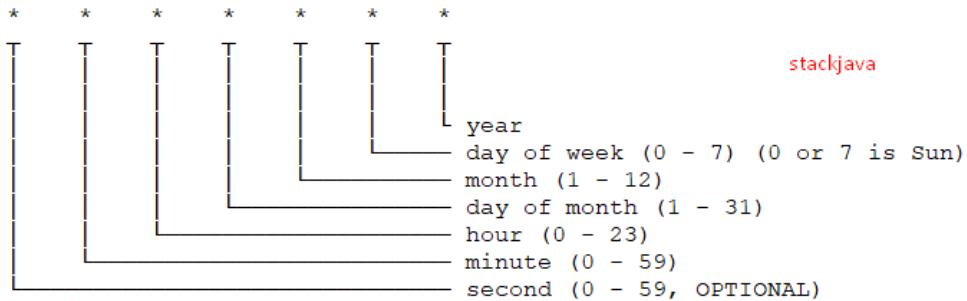
เป็นระบบการตั้งเวลาการทำงานที่ถูกระบุด้วยเครื่องหมายดอกจัน และ ตัวเลขทั้งหมด 5 หลัก เช่น * * * * * โดยตัวแรกคือหน่วย นาที ต่อมาคือ ชั่วโมง วันที่ เดือน และ วันภายในอาทิตย์ (อาทิตย์ ถึง จันทร์) ตามลำดับ โดยผู้ใช้สามารถตั้งได้โดยเปลี่ยนจาก * เป็นตัวเลขตามที่ต้องการให้ทำงาน

*	*	*	*	*	command to be executed
-	-	-	-	-	
				+----- day of week (0 - 6) (Sunday=0)	
			+----- month (1 - 12)		
		+----- day of	month (1 - 31)		
	+----- hour (0 - 23)				
+----- min (0 - 59)					

ภาพที่ 2.6 แผนภาพการตั้งค่า Unix cron

2.1.5.2 Quartz cron

เป็นไลบรารี (Library) ที่ใช้สำหรับกำหนดเวลาเมื่อกัน แต่ทำงานบนโปรแกรม Java มีความสามารถในการกำหนดและเรียกใช้ที่ยืดหยุ่นกว่า Unix cron โดยจะใช้ทั้งหมด 7 หลัก ใช้เครื่องหมายดอกจันท์ และตัวเลข และเครื่องหมายคำ窮 เช่น * * * * ? * โดยจะมีหน่วยวินาทีเพิ่มเข้ามาในหลักแรก ปีเพิ่มเข้ามาในหลักสุดท้าย และตัวเครื่องหมายคำ窮 จะอยู่ในหน่วย วันที่ หรือ วันภายในอาทิตย์ อย่างไดอย่างนึง เช่น ตั้งทุกๆวันที่ 1 วันภายในอาทิตย์จะต้องกำหนดเป็น ? เท่านั้น



ภาพที่ 2.6 แผนภาพการตั้งค่า Quartz cron

2.1.6 Unit test

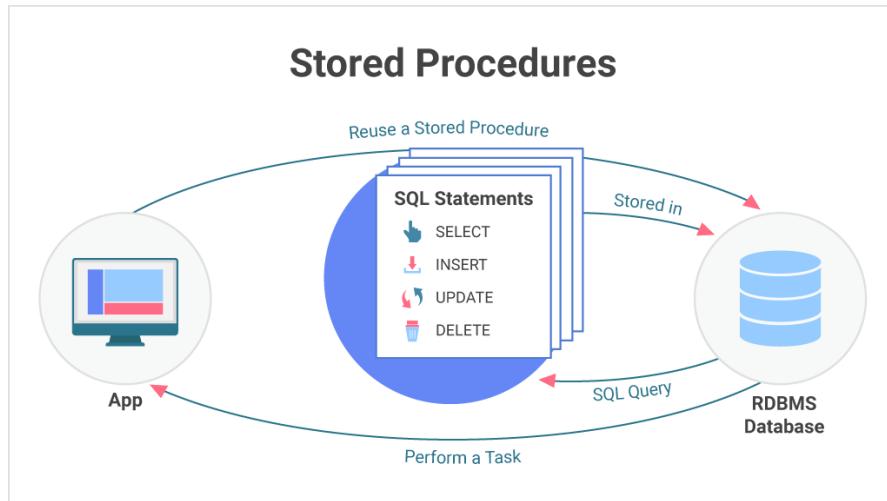
เป็นกระบวนการทดสอบซอฟต์แวร์ (Software) ที่ทำการทดสอบภาคส่วนเล็กที่สุดของระบบ หรือ ส่วนของยูนิต เพื่อตรวจสอบว่า ส่วนนั้นทำงานได้ถูกต้องหรือไม่ โดยทั่วไปแล้วยูนิต จะเป็นฟังก์ชัน (Function) ต่างๆ คลาส หรือโมดูลขนาดเล็ก โดยการทดสอบจะทำการ Mock ยูนิตส่วนอื่นๆ ที่เกี่ยวข้อง เพื่อทดสอบยูนิตส่วนนั้นให้สามารถทำงานไปตามขั้นตอนที่เราต้องการทดสอบ

2.1.7 End to End test

คือกระบวนการตรวจสอบว่าทุกๆ component หรือ ระบบย่อยในระบบของเรา ทำงานร่วมกันได้อย่างถูกต้อง โดยจะทดสอบระหว่าง Frontend และ Backend ว่าสามารถทำงานร่วมกันได้อย่างถูกต้อง โดยการทดสอบมีหลายรูปแบบ เช่นให้ผู้ทดสอบลองใช้งาน Frontend ที่จะถูกส่งการทำงานมายัง Backend และตรวจสอบว่าทำงานได้ถูกต้องหรือไม่ เป็นไปตามที่ต้องการ หรือไม่

2.1.8 Stored Procedures

เป็นชุดคำสั่ง SQL ที่จะถูกสร้างและเก็บไว้ในระบบฐานข้อมูลเอาไว้อยู่แล้ว สามารถเรียกใช้ได้เมื่อต้องการทำงานในฐานข้อมูลนั้นๆ โดยไม่ต้องเขียนคำสั่ง SQL ใหม่ทุกรอบที่ต้องการให้ทำงาน โดยสามารถเรียกผ่าน Stored Procedure ได้ทันที และ Stored Procedure ยังมีพารามิเตอร์ที่สามารถรับค่าได้ และสามารถส่งค่ากลับได้ การใช้ Stored Procedure จะช่วยให้การทำงานของ Database หรือ SQL ทำงานได้รวดเร็วมากขึ้น เนื่องจากไม่ต้องอ่านชุดคำสั่งใหม่



ภาพที่ 2.7 แผนภาพการทำงานของ Stored Procedures

2.1.9 Bulk update SQL โดยใช้ Data Table

การทำ Bulk Update หรือการอัปเดตข้อมูลจำนวนมาก ทำในระบบฐานข้อมูล หมายถึงการอัปเดตข้อมูลในปริมาณมากในรูปแบบทีละกลุ่มหรือແກວของข้อมูล โดยไม่ต้องทำการ Update ทีละคอลัมน์ โดยการใช้ Bulk Update สามารถช่วยลดการใช้ทรัพยากรและเพิ่มประสิทธิภาพของกระบวนการอัปเดตข้อมูลที่มีปริมาณมาก โดยหลักการใช้ Data table จะสร้างตารางใหม่ขึ้นมาเป็นตารางชั่วคราวก่อน และใส่ข้อมูลที่ต้องการอัปเดททั้งหมดเอาไว้ในตารางชั่วคราว หลังจากนั้น จะทำการส่งตารางใหม่เข้าไปรวมกับตารางหลัก เพื่ออัปเดตข้อมูลทั้งหมดในคราวเดียว ซึ่งจะใช้เวลาอ้อยกว่าการ Update ทีละคอลัมน์เยอะมาก เนื่องจากสามารถเชื่อมต่อกับฐานข้อมูลเพียงครั้งเดียวและอัปเดตข้อมูลทั้งหมด

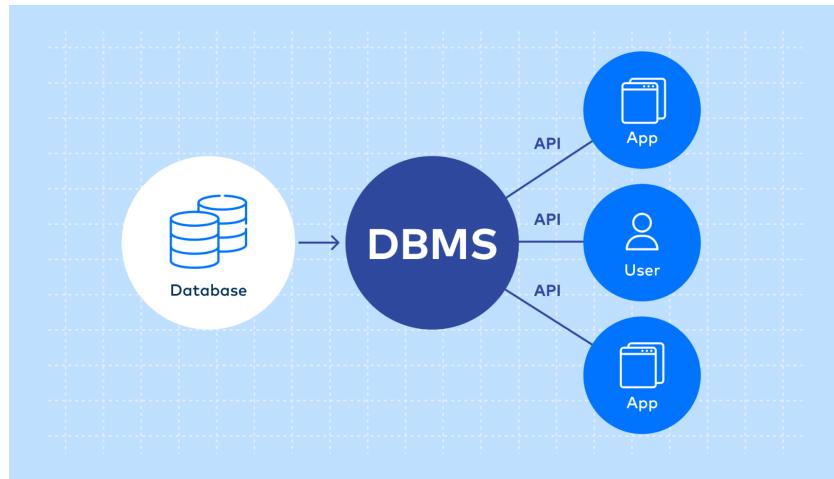
2.1.10 Database

ฐานข้อมูล เป็นcollection ของข้อมูลที่ถูกจัดเก็บอย่างเร็ว, เรียงลำดับ, และสามารถเข้าถึงได้ในระบบคอมพิวเตอร์ ข้อมูลในฐานข้อมูลมักจะถูกออกแบบให้เข้ากันได้กับโครงสร้างและระบบการจัดเก็บเพื่อให้ง่ายต่อการเข้าถึงและประมวลผล

2.1.10.1 MS SQL

MS SQL เป็นระบบฐานข้อมูลที่ถูกพัฒนาโดย Microsoft Corporation เป็นรูปแบบฐานข้อมูลแบบสัมพันธ์ (Relational Database Management System - RDBMS) ซึ่งใช้

โครงสร้างของตารางและความสัมพันธ์ระหว่างตาราง โดยใช้ภาษา SQL ในการทำงาน ซึ่งเป็นภาษามาตรฐานในการจัดการและประมวลผลข้อมูลในระบบฐานข้อมูล



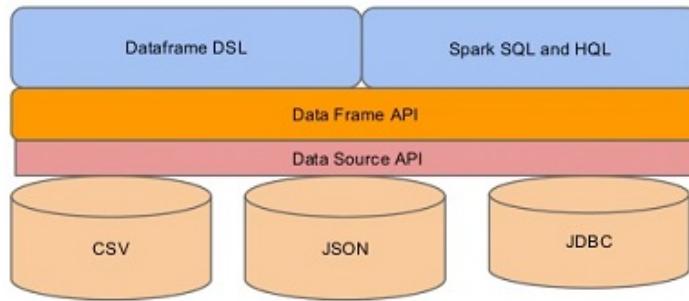
ภาพที่ 2.8 แผนภาพการทำงานของฐานข้อมูลแบบ DBMS

2.1.10.2 Spark SQL

Spark SQL เป็นฐานข้อมูลของโครงการโอเพ่นซอร์สที่พัฒนาโดย Apache Software Foundation โดยฐานข้อมูลนี้ไม่ใช่ระบบฐานข้อมูลแบบสัมพันธ์ แต่เป็นโมเดลที่ใช้ใน Apache Spark สำหรับประมวลผลข้อมูลแบบโคลัมน์ (Columnar) โดยใช้โครงสร้างข้อมูลที่เรียกว่า DataFrame

โดย Spark SQL สามารถทำงานกับข้อมูลจากหลายแหล่งได้, ไม่ว่าจะเป็นข้อมูลในรูปแบบของ Apache Hive, Apache HBase, Parquet, Avro, JSON, CSV, หรือจากข้อมูลที่ถูกจัดการด้วย Spark โดย Spark SQL สนับสนุนการใช้ภาษา SQL เมื่อเทียบกับ MS SQL Server, ทำให้ผู้ใช้สามารถใช้ความรู้ใน SQL ได้โดยไม่ต้องเรียนรู้ใหม่ แต่จะมีบางคำสั่งที่ไม่สามารถใช้ได้ เช่น LIKE, CREATE, ALTER , DROP เป็นต้น

Architecture of Spark SQL



ภาพที่ 2.9 ภาพสถาปัตยกรรมของ Spark SQL

2.1.11 Swagger Doc

เป็น REST API Documentation Tool ใช้สำหรับเก็บเอกสารข้อมูลของ REST API ทั้งหมดได้อย่างรวดเร็ว ทำให้ง่ายต่อการทดสอบและใช้งาน API ได้โดย SwaggerDoc มีหน้า UI ซึ่งเป็นอินเทอร์เฟซ (Interface) ที่สวยงามและใช้งานได้ง่าย ที่ช่วยให้ผู้พัฒนาและผู้ใช้งานสามารถเรียกดู API ที่มีและทดสอบได้โดยตรงผ่าน Swagger UI ได้ทันที

The screenshot shows the Swagger UI interface for a POST request to the endpoint `/av1/regenerateAvlReport`. The endpoint is described as "AVL Regenerate API". The "Parameters" section shows a required parameter named "body" of type "object (body)". The description for this parameter is "Specify the date of the AVL report to regenerate". An example value is provided as a JSON object: `{ "reportDate": "20230724" }`. Below the parameters, there is a "Parameter content type" dropdown set to "application/json".

ภาพที่ 2.10 ภาพตัวอย่างของ Swagger Doc UI

2.1.12 Play Framework

Play Framework เป็นเฟรมเวิร์กสำหรับการพัฒนาเว็บแอปพลิเคชัน (Web applications) และเว็บแอปพลิเคชัน Real-time ที่ใช้ภาษา Java หรือ Scala ใน การเขียนโค้ดของ แอปพลิเคชัน นอกจากนี้ยังมีภาษาที่เรียกว่า Play template language สำหรับการจัดการกับส่วน ต่าง ๆ ของ HTML และการแสดงผลที่เกี่ยวข้องกับ UI โดยมีคุณสมบัติดังนี้

2.1.12.1 Non-blocking I/O

Play Framework ใช้โครงสร้าง Non-blocking I/O ซึ่งช่วยให้แอปพลิเคชันมี ประสิทธิภาพมากขึ้น ด้วยการใช้ฟังก์ชันแบบ Callback แทนการ block การทำงานในเรเดลลัก

2.1.12.2 สนับสนุนการทำ Real-time

Play Framework มีการสนับสนุนสำหรับการทำ Real-time ผ่าน WebSocket ซึ่ง เป็นโปรโตคอลที่ช่วยให้สามารถสื่อสารแบบ full-duplex ระหว่างเซิร์ฟเวอร์และไคลเอนต์ได้

2.1.12.3 MVC Architecture

Play Framework ใช้โครงสร้าง MVC (Model-View-Controller) ที่ช่วยให้โค้ดมี การแยกแยะอย่างดีระหว่างข้อมูล (Model), การแสดงผล (View), และการควบคุม (Controller)

2.1.12.4 ระบบ Routing ที่ยืดหยุ่น

Play Framework มีระบบ Routing ที่สามารถกำหนด URL และการเชื่อมต่อ กับ Controller ได้ง่ายดาย ทำให้การจัดการเส้นทางและการเข้าถึงข้อมูลเป็นเรื่องสะดวก

2.1.12.5 การทำงานกับฐานข้อมูล

Play Framework มีการสนับสนุนการทำงานกับฐานข้อมูลแบบ asynchronous และ synchronous โดยใช้ตัวแทนของ ReactiveMongo หรือ Slick

2.1.12.6 การใช้ Dependency Injection

Play Framework ใช้ระบบ Dependency Injection ที่ช่วยให้การจัดการกับความ เชื่อมโยงของอ็อบเจกต์ (Object) และการจัดการความเข้าถึงบริการเป็นเรื่องสะดวก

2.2 เทคโนโลยีที่เกี่ยวข้อง

2.2.1 Slack

Slack คือโปรแกรมที่ส่วนใหญ่ถูกใช้งานโดยนักพัฒนาโปรแกรมภายในบริษัท ทั้งในรูปแบบแบ่งกลุ่มย่อยและการติดต่อโดยตรงระหว่างบุคคล โดยมีการสร้างกลุ่มของแต่ละทีมเพื่อสนับสนุนการติดต่อและการสื่อสาร โดยสามารถส่งข้อมูลในรูปแบบ Markdown, รูปภาพ, และมีส่วนขยายเพิ่มเติมเพื่อช่วยในการสื่อสารและการทำงาน นอกจากนี้ยังรองรับการสนทนากลุ่มผ่านอินเทอร์เน็ต ทั้งแบบเสียงและวิดีโอ และสามารถแสดงภาพหน้าจอได้ด้วย

และยังมีบอท (Bot) เพื่อช่วยเหลือผู้ใช้งานต่างๆ ได้ Bot ใน Slack เป็นบอทที่ถูกสร้างขึ้นเพื่อช่วยในการทำงานและการสื่อสารในทีมหรือองค์กร ซึ่งมีความสามารถหลากหลายขั้นอยู่ กับวัตถุประสงค์และการโปรแกรมมากของผู้สร้าง โดยสิ่งที่บอทสามารถทำได้มีดังนี้

2.2.1.1 การแจ้งเตือน (Notification)

Bot สามารถส่งข้อความแจ้งเตือนเมื่อมีเหตุการณ์ที่สำคัญเกิดขึ้น เช่น เมื่อมีข้อความใหม่ในช่องหรือเมื่อมีการอัปเดตสถานะของงาน

2.2.1.2 การจัดการ (Task Management)

บอทสามารถช่วยในการจัดการงานต่างๆ ในทีม แบ่งปันข้อมูลเกี่ยวกับงานที่ต้องทำ หรือติดตามสถานะของงานได้

2.2.1.3 การสนับสนุนการประชุม (Meeting Support)

บอทสามารถช่วยในการตั้งค่า หรือ กำหนดเวลาการประชุม และ เชื่อมต่อการประชุมออนไลน์ รวมถึงการแจ้งเตือนเวลาการประชุมผ่าน Slack ได้ทันที

2.2.1.4 การส่งข้อมูล (data Retrieval)

บอทสามารถอ่าน หรือดึงข้อมูลจากฐานข้อมูล หรือ ระบบการบริการต่างๆ ที่ต้องการได้ และดึงข้อมูลจากแหล่งข้อมูลอื่นๆ เพื่อให้ข้อมูลที่เป็นประโยชน์แก่ผู้ใช้ได้

2.2.1.5 การส่งข้อความอัตโนมัติ (Automated Messaging)

บอทสามารถส่งข้อความอัตโนมัติตามเวลาหรือเหตุการณ์ที่กำหนดไว้ หรือจะเป็นแจ้งเตือนข้อผิดพลาดของระบบต่างๆ ที่สามารถตั้งค่าให้แจ้งเตือนได้

2.2.1.6 การทำงานร่วมกับแอพพลิเคชันอื่นๆ (Integration)

บทสามารถตั้งค่าให้ทำงานร่วมกับแอพพลิเคชันอื่นๆได้ หรือบริการอื่นๆที่รองรับ เช่น อ่านกำหนดการประชุมจาก MSTeam และแสดงสถานะบน Slack เพื่อให้เพื่อนร่วมงานสามารถทราบได้ว่าสามารถคุยกับหรือไม่ หรือว่างหรือไม่

2.2.1.7 การเรียกใช้คำสั่ง (Command Execution)

บทสามารถทำงานตามคำสั่งที่กำหนดไว้ เพื่อสนับสนุนการทำงานที่ต่าง ๆ ในทีมได้ ทันที เช่นการขอความช่วยเหลือ เพื่อให้ทีมอื่นที่ต้องการความช่วยเหลือเข้ามาเรียกใช้ได้ทันที นอกจากนี้แล้วยังมีความสามารถอื่น ๆ ที่ Bot ใน Slack สามารถถูกโปรแกรมมาเพื่อตอบสนองความต้องการของทีมหรือองค์กรได้ตามที่ผู้พัฒนาต้องการ



ภาพที่ 2.11 สัญลักษณ์ของโปรแกรม Slack

2.2.2 Microsoft Teams

โปรแกรมนี้ถูกออกแบบมาสำหรับการทำงานโดยผู้ใช้ที่ไม่ใช่นักพัฒนาซึ่งเนื่องจากมีการจัดรูปแบบข้อความในลักษณะที่เข้าใจง่ายและใช้งานได้ง่ายโดยทั่วไป โดยเฉพาะในเรื่องของ User Interface (UI) ที่ถูกออกแบบให้เป็นระเบียบและเข้าใจได้อย่างง่าย นอกจากนี้ยังมีการแก้ไขบางอย่างทำให้สามารถใช้โปรแกรมได้โดยไม่ต้องมีความเชี่ยวชาญในการพัฒนาซอฟต์แวร์ โดยเฉพาะในการสร้าง User Interface ที่เป็นระเบียบและเข้าใจได้ง่ายเพื่อให้ผู้ใช้ทั่วไปสามารถนำไปใช้งานได้โดยสะดวก แต่การที่โปรแกรมถูกออกแบบมาให้ใช้งานได้ง่าย เรียบง่าย ทำให้ไม่สามารถปรับแต่งโปรแกรมได้มาก จึงใช้เพื่อการประชุม และสื่อสารเท่านั้น



ภาพที่ 2.12 สัญลักษณ์ของโปรแกรม Microsoft Teams

2.2.3 Microsoft Outlook

เป็นโปรแกรมสำหรับจัดการ รับส่งอีเมลขององกรค์ และยังสามารถนัดหมายต่างๆ กับพนักงานคนอื่นภายในบริษัทได้อีกด้วย โดยจะมาพร้อมกับปฏิทินที่สามารถ นัดหมายการประชุม หรือจองห้องประชุมภายในบริษัทได้ รวมไปถึงรับข่าวสาร และกิจกรรมต่างๆของทางองกรค์ผ่านทาง อีเมลเช่นกัน



ภาพที่ 2.13 สัญลักษณ์ของโปรแกรม Microsoft Outlook

2.2.4 Workplace from Facebook

เป็นเว็บไซต์ที่ใช้ระบบของ Facebook แต่ถูกแยกออกจากแพลตฟอร์มเดียวกันเพื่อใช้งานในองค์กรเท่านั้น โดยสามารถสร้างกลุ่ม หรือโพสข้อมูลต่างๆ ได้เหมือน Facebook เลย และจะสามารถใช้ได้เฉพาะคนในองค์กรเท่านั้น และจะเป็นข้อมูลภายในองค์กรทั้งหมด



ภาพที่ 2.14 สัญลักษณ์ของโปรแกรม Workplace from facebook

2.2.5 GitLab

GitLab คือแพลตฟอร์มการจัดการห้องที่ใช้งานร่วมกันสำหรับการพัฒนาซอฟต์แวร์ที่ใช้ระบบควบคุมเวอร์ชัน Git ซึ่งเป็นระบบที่ช่วยในการติดตามการเปลี่ยนแปลงในโค้ดของโปรเจคและช่วยให้ทีมพัฒนาสามารถทำงานร่วมกันได้โดยมีประสิทธิภาพสูง ๆ โดย GitLab มีคุณสมบัติหลากหลายดังนี้

2.2.5.1 ระบบควบคุมเวอร์ชัน (Version Control System)

GitLab ใช้ Git เป็นระบบควบคุมเวอร์ชันหลัก เพื่อติดตามการเปลี่ยนแปลงในโค้ด และทราบถึงการพัฒนาที่เกิดขึ้น และยังสามารถแยกการพัฒนาเป็นหลาย Branch ก่อนที่จะรวมกันได้ เพื่อให้ง่ายต่อการพัฒนาและทดสอบฟังชันใหม่ๆ ที่ถูกเพิ่มเข้ามา และยังสามารถย้อนเวอร์ชันกลับได้ทันที เมื่อเกิดเหตุการที่ต้องย้อนเวอร์ชัน

2.2.5.2 ระบบติดตามปัญหา (Issue Tracking)

ใน GitLab สามารถสร้างปัญหา (issue) เพื่อติดตามและจัดการกับงานที่ต้องทำ ปัญหานี้สามารถใช้เพื่อรายงานข้อบกพร่อง ขอความช่วยเหลือ หรือกิจกรรมอื่น ๆ ในโปรเจคได้

2.2.5.3 ระบบการสร้าง (CI/CD)

GitLab มีระบบ CI/CD ซึ่งช่วยในการทดสอบ, สร้าง, และปล่อยโปรแกรมอัตโนมัติ โดยนำมาใช้เพื่อทดสอบระบบก่อนจะเข้าสู่ระบบจริง และตรวจสอบฟอร์แมตของโค้ด ก่อนที่จะรวมไฟล์ได้ และยังช่วยในการ Deploy ระบบขึ้นเซิฟเวอร์ได้ด้วย

2.2.5.4 การจัดการทรัพยากร (Resource Management)

GitLab มีคุณสมบัติสำหรับการจัดการทรัพยากรต่าง ๆ ในโปรเจค เช่น ตารางงาน, การตั้งค่า, และการจัดการผู้ใช้ ได้อย่างละเอียด สามารถตั้งตำแหน่งของแต่ละคนได้อย่างละเอียด เพื่อควบคุมการเพิ่มฟังชั่น ที่ต้องผ่านการตรวจสอบก่อน



ภาพที่ 2.15 สัญลักษณ์ของโปรแกรม GitLab

2.2.6 Jira Software

Jira เป็นบริการการจัดการงานภายใต้ทีมที่ออกแบบมาในรูปแบบของกระดาน (board) เพื่อให้ทีมสามารถติดตามและจัดการงานที่ต้องทำได้ รองรับการกำหนดระยะเวลาที่คาดว่าจะใช้, รายละเอียดต่าง ๆ และสนับสนุนการทำงานแบบ SCRUM

Jira สามารถเชื่อมต่อกับ GitLab เพื่อให้ทีมสามารถติดตามและบันทึกความคืบหน้าได้ด้วย การเชื่อมต่อกับ GitLab ช่วยในการซิงค์ข้อมูลที่เกี่ยวข้องกับโปรเจค, การจัดการแก้ไขข้อบก (issues), และการติดตามการเปลี่ยนแปลงในโค้ดได้ทันที

ดังนั้น, Jira ไม่เพียงแค่เป็นเครื่องมือสำหรับการจัดการงาน แต่ยังเสริมสร้างความสามารถในการทำงานร่วมกับวิธีการพัฒนาซอฟต์แวร์แบบ SCRUM และการทำงานร่วมกับ GitLab เพื่อการบริหารจัดการโปรเจคที่มีประสิทธิภาพ



ภาพที่ 2.16 สัญลักษณ์ของโปรแกรม Jira

2.2.7 Confluence

เป็นระบบการจัดการเนื้อหา (Content Management System) ที่พัฒนาขึ้นโดย บริษัท Atlassian ซึ่งเป็นบริษัทที่พัฒนาซอฟต์แวร์สำหรับการจัดการโปรเจกและการทำงานร่วมกัน ในองค์กร Confluence ถูกออกแบบมาเพื่อให้ทีมงานสามารถสร้าง เก็บรวบรวม และแบ่งปันเนื้อหา ที่สำคัญภายในองค์กรได้โดยง่าย โดยนำมาใช้เพื่อเขียนข้อมูลของระบบที่ถูกสร้างขึ้น เพื่อให้ผู้ใช้ และผู้ที่จะพัฒนาต่อสามารถอ่าน และเข้าในการทำงานของระบบได้ทันที



ภาพที่ 2.17 สัญลักษณ์ของโปรแกรม Confluence

2.2.8 Grafana Dashboard

เป็นเครื่องมือที่ใช้สร้างและแสดงผลข้อมูลต่าง ๆ ในรูปแบบของแดชบอร์ด (Dashboard) อย่างสวยงามและมีประสิทธิภาพ Grafana ถูกออกแบบมาเพื่อการวิเคราะห์และการแสดงผลข้อมูลที่มาจากการแสวงหา เช่น ฐานข้อมูล, บริการเว็บ, หรือระบบตรวจสอบ เพื่อให้ผู้ใช้สามารถตรวจสอบข้อมูลผ่าน Dashboard ได้ทันที



ภาพที่ 2.18 สัญลักษณ์ของโปรแกรม Grafana Dashboard

2.2.9 Hadoop Distributed File System (HDFS)

เป็นระบบฟังก์ชันของ Apache Hadoop, ซึ่งถูกออกแบบมาเพื่อจัดการและจัดเก็บข้อมูลขนาดใหญ่ที่กระจายอยู่ในหลายเครื่องเซิร์ฟเวอร์ (distributed storage) HDFS เป็นส่วนหนึ่งของโครงการ Apache Hadoop ที่ถูกใช้ในการประมวลผลข้อมูลขนาดใหญ่โดยมีหลายๆ เครื่องเซิร์ฟเวอร์ทำงานร่วมกันซึ่งทำให้สามารถประมวลผลข้อมูลที่มีขนาดใหญ่มากๆ ได้อย่างมีประสิทธิภาพ คุณลักษณะที่สำคัญของ HDFS ได้แก่

2.2.9.1 การกระจายข้อมูล (Distributed Data)

HDFS จัดเก็บข้อมูลอย่างกระจายในหลายๆ เครื่องเซิร์ฟเวอร์ เพื่อเพิ่มประสิทธิภาพในการอ่านและเขียนข้อมูล

2.2.9.2 ความคงทนและความเสถียร

HDFS มีความสามารถในการจัดการกับข้อผิดพลาดที่เป็นไปได้ในระบบหลายๆ เครื่อง เช่น การกู้คืนข้อมูลจากการล้มเหลวของเครื่องเซิร์ฟเวอร์.

2.2.9.3 บล็อกข้อมูล (Block Storage)

ข้อมูลใน HDFS ถูกแบ่งเป็นบล็อกขนาดใหญ่ และถูกเก็บไว้ในหลายๆ เครื่องเซิร์ฟเวอร์ เพื่อให้สามารถทำงานได้อย่างมีประสิทธิภาพ

2.2.9.4 การโಯกย้ายข้อมูล (Replication)

HDFS มีการโยกย้ายข้อมูล (replication) โดยทำสำเนาข้อมูลไปยังหลายๆ เครื่องเซิร์ฟเวอร์ เพื่อความคงทนต่อความล้มเหลวของเครื่องเซิร์ฟเวอร์

2.2.9.5 ชุดคำสั่ง (Command Line Interface)

นอกจากการใช้งานผ่าน API, HDFS ยังมีชุดคำสั่ง (Command Line Interface) ที่ให้ผู้ใช้สามารถจัดการข้อมูลได้ และยังสามารถทำ ETL บน Hadoop ได้อีกด้วย



ภาพที่ 2.19 สัญลักษณ์ของโปรแกรม Hadoop (HDFS)

2.2.10 HUE (Hadoop User Experience)

เป็นเครื่องมือที่ออกแบบมาเพื่ออำนวยความสะดวกในการใช้งาน Apache Hadoop และส่วนประกอบต่าง ๆ ของโครงการที่เกี่ยวข้อง โดยพัฒนาโดย Cloudera. Hue นำเสนออินเทอร์เฟซกราฟิกที่เข้าถึง Hadoop อย่างง่ายดายสำหรับผู้ใช้ทั่วไปโดยไม่ต้องมีความรู้ลึกเกี่ยวกับการเขียนคำสั่งหรือการทำงานทางเทคนิค โดยทำงานผ่าน Web browser เพื่อให้ผู้ใช้สามารถเข้าไปเพื่อกำหนดค่า ควบคุม หรือตรวจสอบข้อมูลใน HDFS ผ่าน Web browser ได้ทันที



ภาพที่ 2.20 สัญลักษณ์ของโปรแกรม HUE

2.2.11 Vault Software

Vault Software เป็นระบบจัดการความปลอดภัย (security) ที่ใช้เก็บและจัดการข้อมูลรับรองต่าง ๆ อย่างปลอดภัย เช่น รหัสผ่าน, รูปแบบการเข้ารหัส, คีย์, และข้อมูลอื่น ๆ ที่เกี่ยวข้องกับความปลอดภัยของระบบ โดยสามารถจัดการรหัสผ่านอย่างปลอดภัย ทำให้ผู้ใช้สามารถสร้าง จัดการ และเรียกใช้รหัสผ่านได้อย่างมีประสิทธิภาพ และความปลอดภัยสูง Vault ยังสามารถจัดการและจัดเก็บคีย์ต่างๆ ที่ใช้ในกระบวนการเข้ารหัสและถอดรหัสข้อมูลได้ด้วย และยังมีระบบควบคุมสิทธิ์ของผู้ใช้แต่ละคนว่าสามารถเข้าถึงข้อมูลส่วนไหนได้บ้าง



ภาพที่ 2.21 สัญลักษณ์ของโปรแกรม Vault

2.2.12 Quilliuup

เป็นโปรแกรมที่ช่วยตรวจสอบคุณภาพของข้อมูลในฐานข้อมูลว่าเป็นไปตามที่กำหนดหรือไม่ เช่น ตรวจสอบว่าข้อมูลในฐานข้อมูลถูกอัปเดตตรงตามกำหนดหรือไม่ มีข้อมูลครบหรือไม่ และยังสามารถแจ้งเตือนไปยังอีเมลของผู้ใช้ได้ เมื่อเกิดข้อผิดพลาดของข้อมูลบนฐานข้อมูล



ภาพที่ 2.22 สัญลักษณ์ของโปรแกรม quilliuup

2.2.13 Visual Studio Code

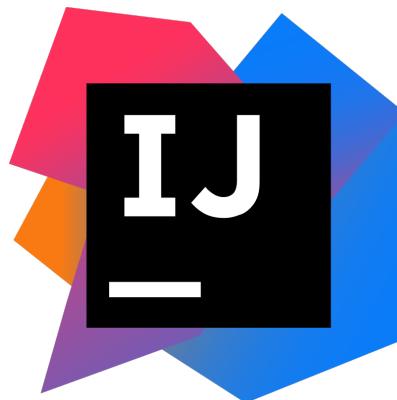
เป็น IDE หรือ Code Editor ที่เน้นให้สามารถใช้งานได้ง่าย สะดวกรวดเร็ว เหมาะกับการทำงานแก้ไขไฟล์ ไม่เหมาะกับการทำโปรเจคใหญ่ๆ โดยผู้ใช้สามารถติดตั้งส่วนเสริมเพิ่มได้ เพื่อตอบสนองความต้องการของผู้ใช้ ช่วยให้การพัฒนาโค้ดเป็นไปได้อย่างมีประสิทธิภาพ



ภาพที่ 2.23 สัญลักษณ์ของโปรแกรม Visual Studio Code

2.2.14 IntelliJ IDEA

IntelliJ IDEA เป็นโปรแกรมแก้ไขรหัสซอร์สแบบแบบพัฒนาบนแพลตฟอร์ม Java ที่ถูกพัฒนาโดย JetBrains ซึ่งเป็นบริษัทซอฟต์แวร์ที่มีชื่อเสียงในการพัฒนาซอฟต์แวร์ IntelliJ IDEA มีคุณสมบัติที่มากมายที่ช่วยให้นักพัฒนาสามารถทำงานได้สะดวกและมีประสิทธิภาพมากขึ้น รองรับ Scala และส่วนเสริมอื่นๆ ที่สามารถเลือก用เพิ่มได้



ภาพที่ 2.24 สัญลักษณ์ของโปรแกรม IntelliJ IDEA

2.2.15 DataGrip

เป็นโปรแกรมจัดการฐานข้อมูล โดยรองรับฐานข้อมูลเกือบทุกยี่ห้อที่ถูกใช้อยู่ในปัจจุบัน ช่วยให้เขียนคำสั่ง SQL, ดำเนินการข้อมูล และทำงานกับฐานข้อมูลต่างๆได้อย่างมีประสิทธิภาพ และช่วยลดเวลาในการเชื่อมต่อฐานข้อมูลหลายๆที่ด้วย เนื่องจากโปรแกรมสามารถเชื่อมต่อหลายฐานข้อมูลพร้อมกันได้

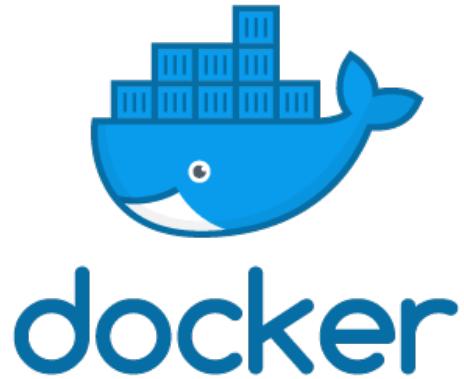


ภาพที่ 2.25 สัญลักษณ์ของโปรแกรม DataGrip

2.2.16 Docker

เป็นแพลตฟอร์มที่ช่วยให้เราสามารถสร้าง ทดสอบ และติดตั้งแอปพลิเคชันใช้งานได้อย่างรวดเร็ว Docker จะบรรจุซอฟต์แวร์ลงในหน่วยที่เป็นมาตรฐานเรียกว่า คอนเทนเนอร์ ซึ่งจะมีทุกสิ่งที่ซอฟต์แวร์ต้องใช้ในการเรียกใช้งาน รวมทั้งไลบรารี เครื่องมือสำหรับระบบ โค้ด และรันไทม์ เมื่อใช้ Docker คุณจะสามารถติดตั้งใช้งานและปรับขนาดแอปพลิเคชันให้เหมาะสมกับทุกสภาพแวดล้อมและทราบว่าโค้ดของคุณจะเรียกใช้ได้อย่างรวดเร็ว

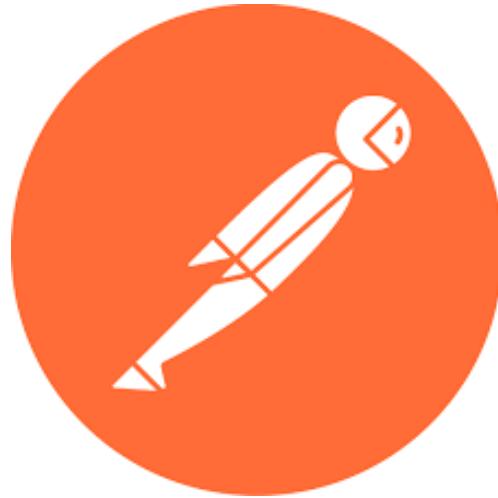
Docker ทำงานโดยการช่วยสร้างวิธีมาตรฐานในการเรียกใช้โค้ด Docker เป็นระบบปฏิบัติการสำหรับคอนเทนเนอร์ คอนเทนเนอร์จะจำลองระบบปฏิบัติการของเซิร์ฟเวอร์ ซึ่งคล้ายคลึงกับวิธีการที่เครื่องเสมือนจำลอง (ลดความจำเป็นในการจัดการโดยตรง) haar ด้วย Docker ได้รับการติดตั้งลงบนแต่ละเซิร์ฟเวอร์และสร้างคำสั่งง่ายๆ ที่คุณสามารถใช้ในการสร้าง เริ่มต้น หรือหยุดคอนเทนเนอร์



ภาพที่ 2.26 สัญลักษณ์ของโปรแกรม Docker

2.2.17 Postman

เป็นเครื่องมือที่ใช้ในการทดสอบและพัฒนา API ซึ่งเป็นวิธีที่แตกต่างกันที่แอปพลิเคชันสามารถสื่อสารกันได้ Postman ถูกออกแบบมาเพื่อช่วยในการทดสอบ API ว่าสามารถทำงานได้อย่างถูกต้อง สามารถส่ง API ได้หลายแบบ สามารถทดสอบข้อมูลเกี่ยวกับ API ได้ทุกรูปแบบ เช่น HTTP, GraphQL, gRPC ทำให้ผู้พัฒนาสามารถทดสอบระบบได้อย่างมีประสิทธิภาพ



ภาพที่ 2.27 สัญลักษณ์ของโปรแกรม Postman

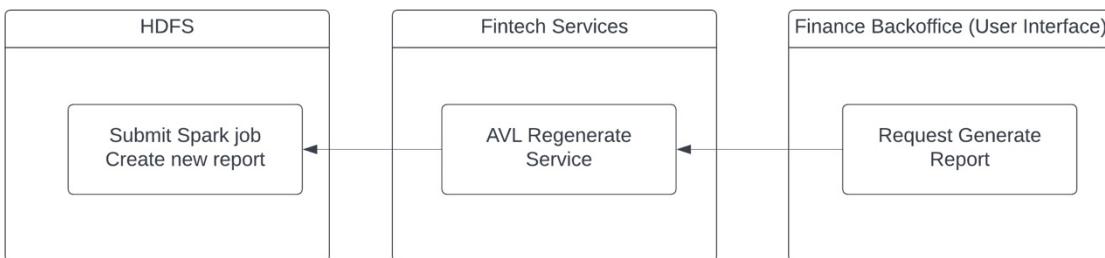
บทที่ 3 การดำเนินงาน

3.1 ภาพรวมของโครงการ

ทางผู้จัดทำโครงการได้พัฒนาระบบ เพื่อแก้ไขปัญหาด้านธุรกรรมทางการเงิน การสร้างไฟล์รายงานทางการเงิน และการตรวจสอบไฟล์ที่ถูกสร้างอัตโนมัติว่าถูกสร้างตามกำหนดการ หรือไม่ เพื่อช่วยให้ฝ่ายการเงินขององค์กรสามารถนำไปใช้งาน เพื่อเพิ่มประสิทธิภาพการทำงานของฝ่ายการเงิน โดยทางผู้จัดทำโครงการได้ทำระบบ แบ่งออกเป็นทั้งหมด 3 ระบบ โดยระบบแรก คือ

3.1.1 ระบบสร้างไฟล์รายงานสำหรับทรัพย์สินและหนี้สินใหม่

โดยปกติไฟล์รายงานสำหรับข้อมูลทรัพย์สินและหนี้สิน จะถูกสร้างโดยอัตโนมัติตามกำหนดการ แต่เนื่องจากทางองค์กรมีข้อมูลที่เปลี่ยนแปลงตลอดเวลา ผู้ใช้งาน หรือฝ่ายการเงินจะต้องการระบบที่สามารถสร้างไฟล์รายงานได้เองทันที โดยไม่ต้องรอระบบอัตโนมัติ และสามารถสร้างไฟล์รายงานย้อนหลังได้ ทางผู้จัดทำโครงการจึงพัฒนาระบบสำหรับการสร้างไฟล์รายงานขึ้นเพื่อให้ฝ่ายการเงินขององค์กรสามารถใช้งานเพื่อสร้างไฟล์รายงานได้ทันที

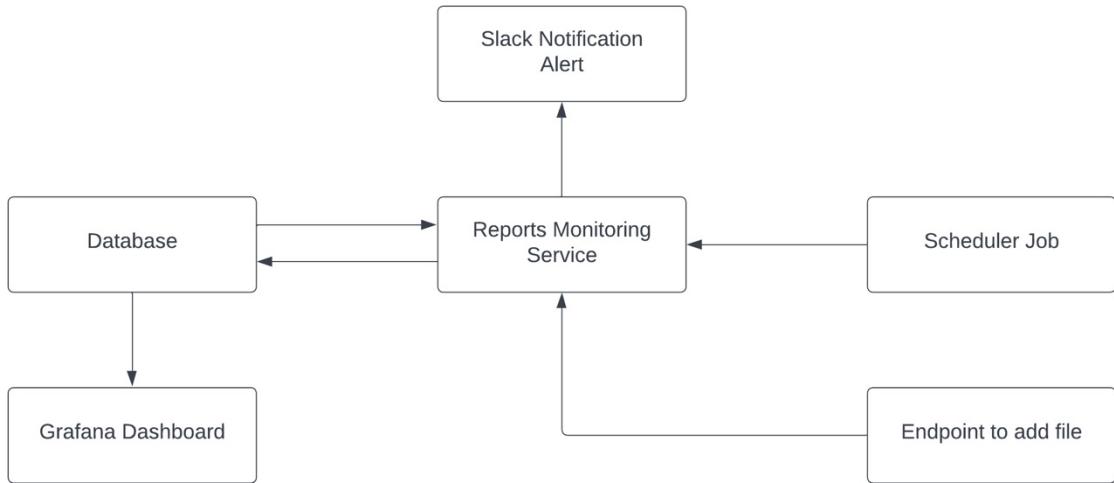


ภาพที่ 3.1 แผนผังการทำงานของระบบสร้างรายงานทรัพย์สินและหนี้สินใหม่

3.1.2 ระบบตรวจสอบและแสดงผลการสร้างรายงานสำคัญทางการเงิน

จากที่กล่าวมาข้างต้นในข้อ 3.1.1 โดยปกติรายงานจำนวนมากในฝ่ายการเงินจะถูกสร้างโดยอัตโนมัติตามกำหนดการของแต่ละรายงาน ซึ่งมีรายงานจำนวนมากที่ถูกสร้างขึ้นในแต่ละวัน ผู้จัดทำโครงการจึงได้พัฒนาระบบนี้เพื่อใช้สำหรับตรวจสอบรายงานที่กำลังจะถูกสร้างในวันนั้นๆ ว่า ถูกสร้างได้ตามกำหนดการหรือไม่ หรือมีข้อผิดพลาดในการสร้างไฟล์หรือไม่ และสามารถแจ้งเตือนไปยังฝ่ายที่เกี่ยวข้องได้ทันที ถ้าไฟล์ไม่ถูกสร้างตามกำหนดการ และฝ่ายที่เกี่ยวข้องยังสามารถดูสถานะ

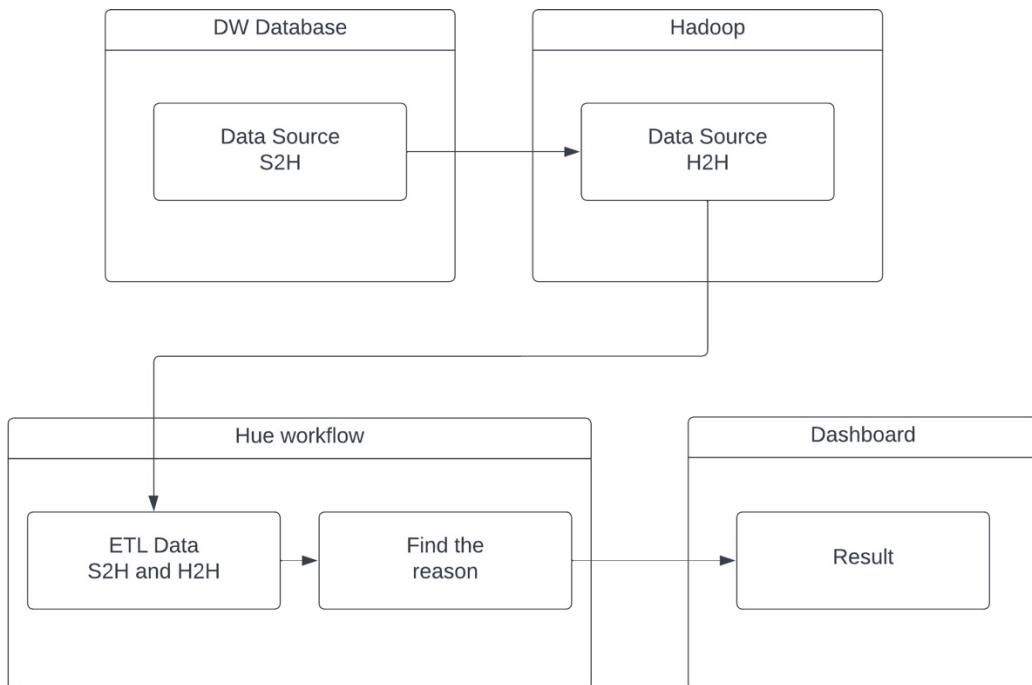
ไฟล์ทุกรายงานได้ผ่าน Grafana Dashboard ด้วย ซึ่งเป็นแดชบอร์ดแผนภูมิแสดงข้อมูลสถานการณ์สร้างไฟล์ของทุกๆไฟล์รายงาน



ภาพที่ 3.2 แผนภาพการทำงานของระบบตรวจสอบการสร้างไฟล์รายงาน

3.1.3 ระบบตรวจสอบความถูกต้องของธุกรรมทางการเงิน

โดยเนื่องจากองค์กรของเรามีการทำธุกรรมจำนวนมากในแต่ละวัน ทำให้เราตรวจสอบได้ยากว่ามีธุกรรมไหนที่มีข้อผิดพลาดบ้าง และเกิดขึ้นจากอะไร หากผู้จัดทำโครงงานจึงพัฒนาระบบนี้เพื่อช่วยค้นหาธุกรรมที่มีความผิดพลาด และ ระบบนี้ยังสามารถหาเหตุผลของข้อผิดพลาดนั้นได้ด้วย ว่าเกิดจากอะไร เพื่อให้ฝ่ายการเงินสามารถนำข้อมูลที่ได้เปรียบเทียบต่อได้ทันท่วงที



ภาพที่ 3.3 แผนภาพการทำงานของระบบตรวจสอบความถูกต้องธุกรรมทางการเงิน

3.2 ขั้นตอนการดำเนินงาน

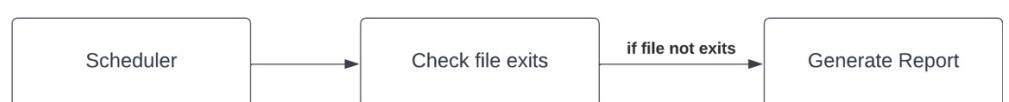
3.2.1 ระบบสร้างไฟล์รายงานสำหรับทรัพย์สินและหนี้สินใหม่

3.2.1.1 เก็บข้อมูล และ วิเคราะห์ความต้องการของผู้ใช้งาน

ผู้จัดทำโครงการได้รับความต้องการจากผู้ใช้ ว่าต้องการระบบที่สามารถสร้างไฟล์รายงานได้เองตลอดเวลาที่ผู้ใช้ต้องการ โดยไม่ต้องรอระบบอัตโนมัติในการสร้างไฟล์รายงาน หรือสามารถสร้างไฟล์รายงานใหม่ได้ภายในวันเดียวจากการสร้างไฟล์แบบอัตโนมัติจะทำงานเพียงวันละครั้ง ผู้จัดทำโครงการจึงมีแนวคิดที่จะพัฒนาระบบที่สามารถสร้างไฟล์รายงานขึ้นมาได้ทันที โดยไม่ต้องรอระบบอัตโนมัติในการสร้างไฟล์รายงาน

3.2.1.2 วิเคราะห์ขอบเขต และ ออกแบบระบบเพื่อนำมาแก้ปัญหา

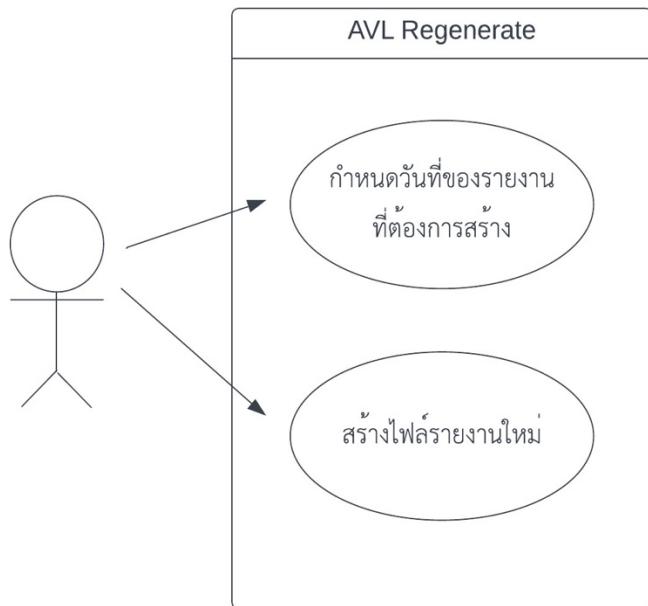
เริ่มจากศึกษาระบบที่มีอยู่ ซึ่งคือระบบอัตโนมัติที่สร้างไฟล์รายงานตามกำหนด โดยจะทำงานดังภาพ



ภาพที่ 3.4 แผนภาพการทำงานของระบบอัตโนมัติ

จากภาพที่ 3.4 แสดงแผนภาพของระบบอัตโนมัติ ระบบอัตโนมัติ จะถูกเรียกโดยการตั้งเวลา หรือ Scheduler และตรวจสอบไฟล์ก่อนที่จะสร้างเสมอ เพื่อกันไม่ให้มีไฟล์ที่ซ้ำกันได้ เช่น ถ้าไฟล์ของวันนี้ถูกสร้างขึ้นแล้ว แม้ต่อระบบ อัตโนมัติถูกสั่งให้ทำงานอีกครั้ง รายงานก็จะไม่ถูกสร้างใหม่

ผู้จัดทำโครงงานจึงได้ออกแบบระบบใหม่ เพื่อให้ผู้ใช้สามารถใช้ งานการสร้างไฟล์รายงานด้วยตัวเองได้ โดยผู้ใช้สามารถใช้หน้า UI ที่ผู้จัดทำ โครงงานพัฒนาขึ้นเพื่อสร้างไฟล์รายงานได้ทันที และในส่วนของ Service ผู้จัดทำ วางแผนที่จะลบการตรวจสอบไฟล์ออก เพื่อให้ผู้ใช้สามารถสร้างรายงานได้ใหม่ทุก เมื่อ



ภาพที่ 3.5 แผนภาพกรณีใช้งานระบบสร้างไฟล์รายงานใหม่

จากภาพที่ 3.5 แสดงแผนผังการใช้งานข้อมูลใช้ ซึ่งจากการวิเคราะห์ขอบเขตและความต้องการของผู้ใช้ โดยผู้ใช้จะต้องเลือกวันของข้อมูลรายงานที่ต้องการสร้างก่อน และจึงสามารถกดสร้างไฟล์รายงานใหม่ได้

ตารางที่ 3.1 แสดงรายละเอียด Actor ใน Use case diagrams

Actor	Description
พนักงาน	พนักงานฝ่ายการเงินที่ต้องการสร้างไฟล์รายงานใหม่

ตารางที่ 3.2 แสดงรายละเอียดกรณีใช้งาน ต่างๆใน Use Case Diagrams

No.	Use Case Name	Actor	Description
UC-01	เลือกวันที่ของข้อมูลที่ต้องการสร้างไฟล์รายงาน	พนักงาน	พนักงานเลือกวันที่ของข้อมูลที่ต้องการจะนำไปสร้างเป็นไฟล์รายงานใหม่ได้
UC-02	สร้างไฟล์รายงานใหม่	พนักงาน	พนักงานกดคลิกเพื่อให้ระบบสร้างไฟล์รายงานใหม่

ตารางที่ 3.3 รายละเอียดกรณีการใช้งาน Use Case UC-01

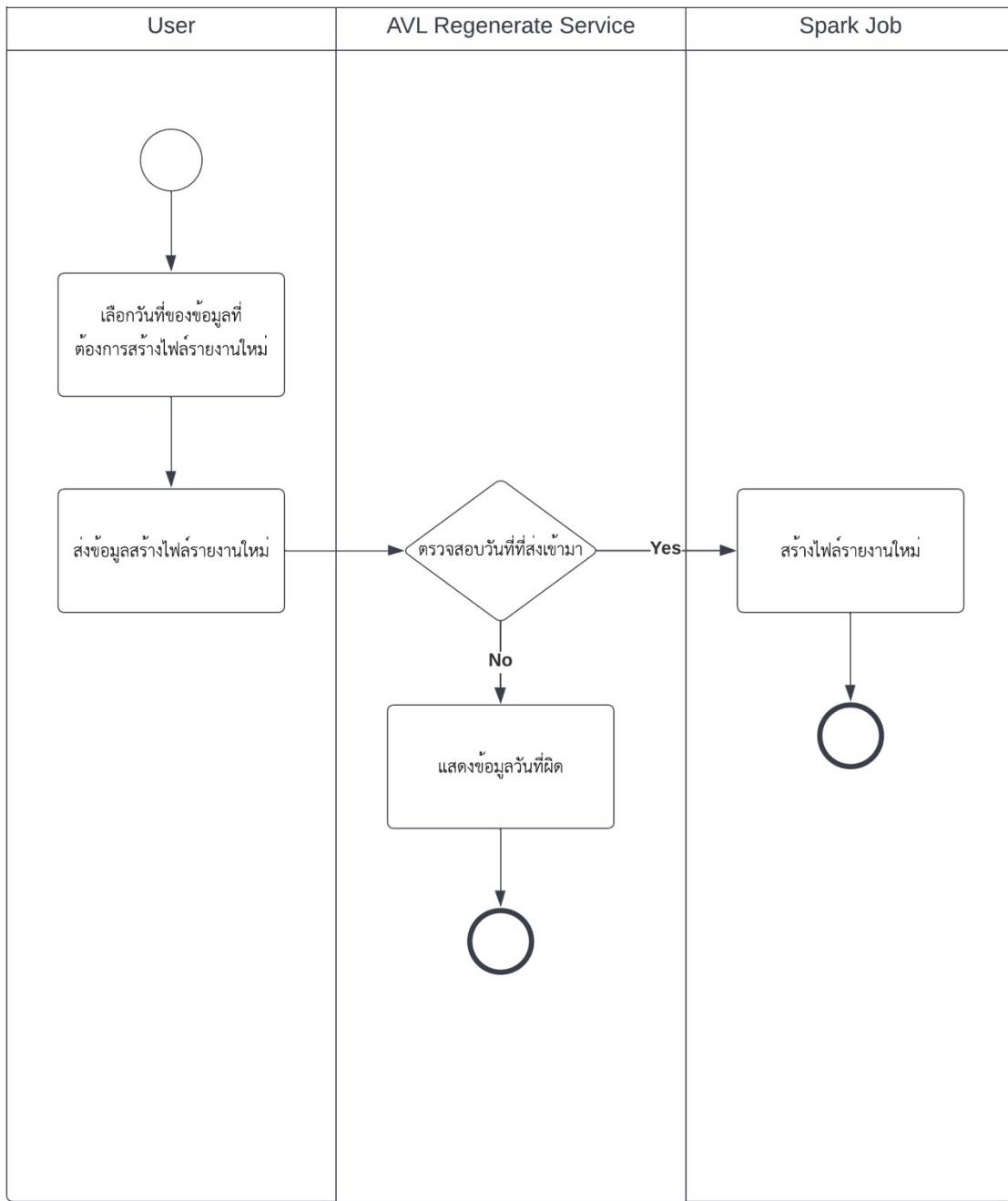
รหัสยูสเคส (Use Case ID)	UC-01
ชื่อยูสเคส (Use Case Name)	เลือกวันที่ของข้อมูลที่ต้องการสร้างไฟล์รายงาน
ผู้ใช้งาน (Actor)	พนักงาน
คำอธิบาย (Description)	พนักงานสามารถเลือกวันที่ของข้อมูลที่ต้องการจะนำไปสร้างเป็นไฟล์รายงานใหม่ได้
เงื่อนไขก่อนหน้า (Pre-condition)	ผู้ใช้จะต้องเข้าสู่ระบบ
เงื่อนไขภายหลัง (Post-condition)	-
กระแสหลัก (Basic Flow)	1. พนักงานเข้าสู่ระบบ 2. พนักงานเข้าสู่หน้าของ AVL Regenerate 3. พนักงานเลือกวันที่ที่ต้องการจะนำไปสร้างเป็นไฟล์รายงานใหม่
กระแสรอง (Alternative Flow)	-

ตารางที่ 3.4 รายละเอียดกรณีการใช้งาน Use Case UC-02

รหัสยูสเคส (Use Case ID)	UC-02
ชื่อยูสเคส (Use Case Name)	สร้างไฟล์รายงานใหม่

ผู้ใช้งาน (Actor)	พนักงาน
คำอธิบาย (Description)	พนักงานกดคลิกเพื่อให้ระบบสร้างไฟล์รายงานใหม่
เงื่อนไขก่อนหน้า (Pre-condition)	ผู้ใช้จะต้องเลือกวันที่ของข้อมูลที่จะนำมาสร้างเป็นไฟล์รายงานใหม่
เงื่อนไขภายหลัง (Post-condition)	ระบบแสดงข้อความผลลัพธ์การสร้างไฟล์รายงานใหม่
กระแสหลัก (Basic Flow)	<ol style="list-style-type: none"> พนักงานเลือกวันที่ที่ต้องการจะนำไปสร้างเป็นไฟล์รายงานใหม่ พนักงานกดปุ่มเพื่อสร้างไฟล์รายงานใหม่ ระบบแสดงข้อความ สร้างไฟล์รายงานใหม่สำเร็จ
กระแสรอง (Alternative Flow)	<ol style="list-style-type: none"> เมื่อสร้างไฟล์ผิดพลาด จะมีข้อความแสดงว่า Error และข้อมูลข้อผิดพลาดว่าเกิดจากอะไรด้านบนขวา

แผนภาพแสดงขั้นตอนการทำงาน (Activity Diagram)

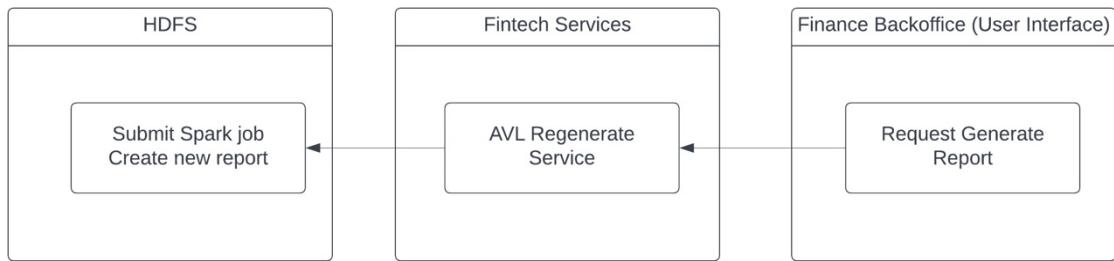


ภาพที่ 3.6 ภาพแสดง Activity Diagram สำหรับการสร้างไฟล์ใหม่

3.2.1.3 ขั้นตอนการ implement

โปรเจกนี้ถูกออกแบบมาเพื่อให้การทำงานเป็นไปอย่างมีประสิทธิภาพและมีประสิทธิภาพทั้งในด้าน User Interface (UI), Backend, และการประมวลผลข้อมูลด้วย Spark job

บน Hadoop Distributed File System (HDFS) โดยแบ่งเป็น 3 ส่วนในการ implement จากที่กล่าวมาข้างต้น



ภาพที่ 3.7 แผนภาพแสดงโครงสร้างของระบบสร้างไฟล์รายงาน

เนื่องจากผู้จัดทำโครงการได้เริ่มพัฒนาระบบจาก Backend เป็นส่วนแรก และเพื่อจ่ายต่อ การทดสอบฟังก์ชันการทำงานของบริการในฝั่ง Backend จึงเริ่มจากการพัฒนาระบบทัว Service ที่อยู่บนฝั่ง Backend หรือ Fintech Service สำหรับสร้างไฟล์เป็นอันดับแรก โดยพัฒนาด้วยภาษา Scala ที่ทำงานบน JVM หรือ Java Virtual Machine

(1) การสร้าง Stored Procedure

ผู้จัดทำโครงการเริ่มต้นการสร้าง Stored Procedure สำหรับค้นหาข้อมูลบนฐานข้อมูล เพื่อให้เกิดประสิทธิภาพสูงสุด เมื่อใช้ Stored Procedure ตัวฐานข้อมูลจะไม่จำเป็นต้องอ่าน Query ทุกรั้งที่เรียกใช้ซ้ำๆ โดยสร้าง Stored Procedure เพื่อใช้สำหรับดึงข้อมูลสถานะของรายงานจากฐานข้อมูล

```

CREATE PROCEDURE [dbo].[fetch_avl_status]
    @datadate AS int
AS
BEGIN
    SET NOCOUNT ON;
    SELECT
        avl_status_data
    FROM avl_status_table
    WHERE datadate = @datadate
END
  
```

ภาพที่ 3.8 ภาพโครงสร้าง Stored Procedure สำหรับดึงข้อมูลสถานะรายงาน

โดยผู้ใช้สามารถส่งวันที่ ที่ต้องการข้อมูลสถานะของรายงานได้ เพื่อตรวจสอบสถานะข้อมูลได้ในวันที่ต้องการได้ทันที

(2) Backend

โครงสร้างของระบบ Backend จะประกอบด้วย ส่วนของ Endpoint ที่เอาไว้ให้ฝั่ง client หรือ frontend เรียกใช้ได้ โดยเขียนด้วย Play framework บนภาษา Scala

Name	Description
body * required	Specify the date of the AVL report to regenerate
Example Value	Model
<pre>{ "reportDate": "20230724" }</pre>	
Parameter content type	
application/json	

ภาพที่ 3.9 SwaggerDoc ของ Endpoint ระบบสำหรับสร้างไฟล์รายงานใหม่

จากการที่ 3.9 ผู้จัดทำโครงงานได้สร้าง Endpoint สำหรับเรียกใช้เพื่อสร้างไฟล์รายงานใหม่ โดยใช้ method POST ผ่าน REST API เพื่อให้ผู้ใช้งานสามารถใช้งานได้ง่าย โดยที่ผู้จัดทำโครงงานเลือกใช้ Method POST เนื่องจากเป็นการส่งคำขอเพื่อสร้างไฟล์รายงานใหม่

โดยใน Endpoint นี้จะรับค่าพารามิเตอร์เป็น reportDate หรือวันที่ของข้อมูล ที่ผู้ใช้ต้องการทำไปสร้างเป็นไฟล์รายงานใหม่ โดยใช้รูปแบบ yyyyMMdd ส่งผ่าน body ที่เป็นข้อมูล JSON และผู้จัดทำโครงงานได้ใช้ SwaggerDoc ซึ่งเป็นเครื่องมือที่ช่วยในการเขียนและเพิ่มเติมเอกสารข้อมูล API ได้อย่างรวดเร็ว และยังสามารถให้ผู้พัฒนาทดสอบเรียกใช้งาน Endpoint หรือ API ที่ถูกสร้างขึ้นมาใหม่ผ่านหน้าเว็บของ SwaggerDoc ได้ทันทีดังภาพที่ 3.8

ระบบ Backend จะมีโครงสร้าง ถูกแบ่งออกเป็น 3 ส่วนคือ Repository, Service และ Controller โดยเริ่มออกแบบจาก Repository

(g) Repository

เป็นส่วนที่เอาไว้สำหรับเข้ามต่อ กับระบบฐานข้อมูล เพื่อดึงข้อมูลที่จำเป็น โดยระบบการสร้างไฟล์รายงานใหม่จะมีฟังก์ชันใน Repository ดังนี้

- fetchAVLStatus เป็นฟังชันที่จะเชื่อมต่อเข้าสู่ฐานข้อมูล และเรียกใช้ Stored Procedure ที่ถูกสร้างไว้ชื่อ dbo.fetch_avl_status เพื่อใช้ในการดึงข้อมูลสถานะของรายงานในฐานข้อมูลว่าวันที่ผู้ใช้ส่งมาเพื่อสร้างไฟล์รายงานใหม่เมื่อข้อมูลหรือไม่

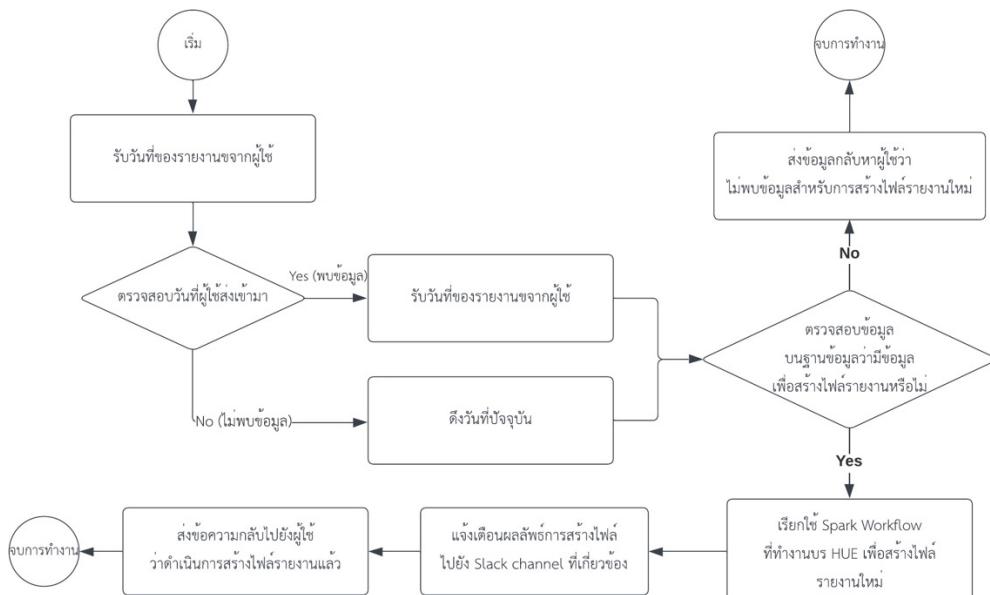
(ข) Service

ทำหน้าที่ในการให้บริการในการทำงานต่างๆ แต่ละเซอร์วิสจะมีหน้าที่เฉพาะจง เช่น การจัดการข้อมูล หรือการประมวลผลข้อมูล โดยภายในระบบจะมี Service ดังนี้

- fetchAVLStatus เป็นเซอร์วิส หรือฟังชันที่ใช้จะเรียกฟังชันใน Repository ในการดึงข้อมูลสถานะจากฐานข้อมูล เพื่อนำมาตรวจสอบสถานะข้อมูลรายงานในฐานข้อมูลว่ามีข้อมูลเพื่อนำมาสร้างไฟล์รายงานใหม่หรือไม่
- avlSlackNotifierService เป็นเซอร์วิส สำหรับส่งแจ้งเตือนไปยังผู้ใช้ที่เกี่ยวข้อง ผ่านโปรแกรม Slack โดยใช้ Bot ใน การส่งข้อความแจ้งเตือน ซึ่งสามารถกำหนดข้อมูลเนื้อหาของข้อความ และ แฟลตแลนที่ต้องการจะส่งได้
-

(ค) Controller

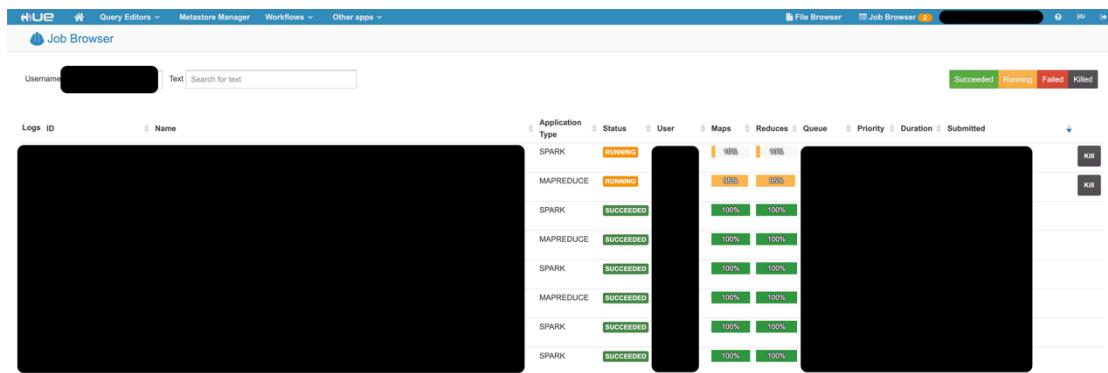
ทำหน้าที่ในการควบคุม Flow ทั้งหมด ตั้งแต่ถูกเรียก Endpoint เข้ามา และนำไปประมวลผลตามที่ผู้พัฒนาต้องการ แล้วนำข้อมูลที่ได้ ส่งกลับไปยังผู้ใช้ หรือส่วนต่างๆ ของแอปพลิเคชัน ช่วยให้เราสามารถจัดการกระบวนการทำงาน และสื่อสารระหว่างส่วนต่างๆ ของระบบได้อย่างมีประสิทธิภาพ



ภาพที่ 3.10 ภาพ Flow การทำงานของ Controller ระบบสร้างไฟล์รายงานใหม่

ระบบ Spark Job ที่ทำงานบน HDFS Workflow เมื่อ Controller ของ Service ที่เขียนบน Scala ได้เรียกใช้ Spark Workflow โดยจะทำงานบน HDFS จะเริ่มทำงานโดยดึงข้อมูลจากฐานข้อมูลที่เกี่ยวข้องนำมาสร้างเป็นไฟล์รายงานใหม่ และส่งไฟล์รายงานใหม่จะบันทึกเข้าไปในเซิฟเวอร์ที่อยู่ และแจ้งเตือนไปยังผู้ใช้งานทางอีเมล

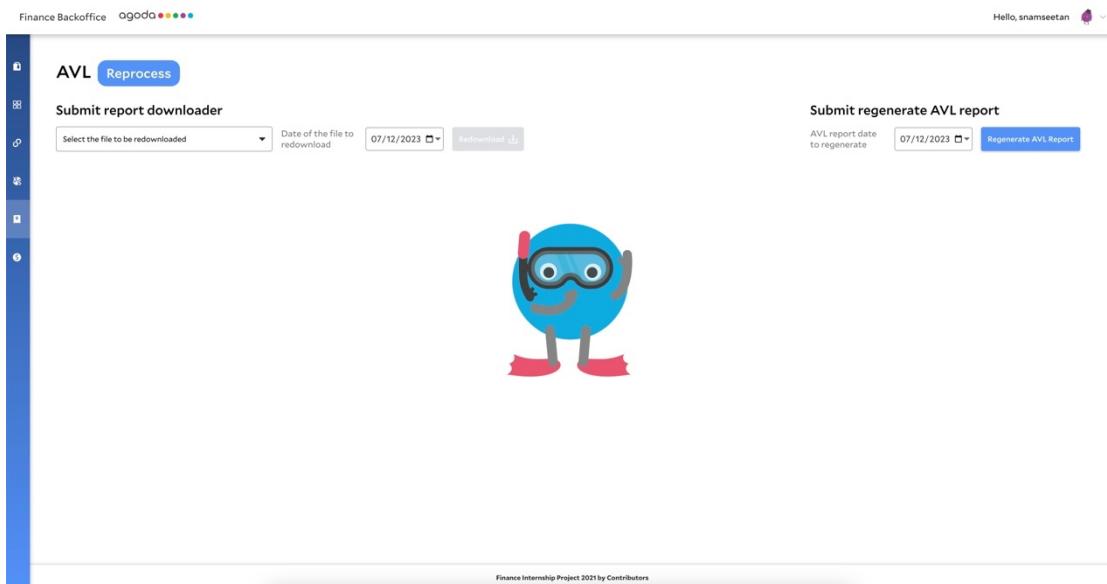
โดยผู้ใช้สามารถตรวจสอบการทำงานของ Workflow ว่าเสร็จสิ้นหรือยังได้ผ่านหน้าเว็บไซต์ของ HUE ซึ่งเป็นแอปพลิเคชัน User Interface สำหรับใช้ในระบบ HDFS



ภาพที่ 3.11 หน้า UI บน HUE เพื่อตรวจสอบสถานะการทำงานของ Workflow

(3) Frontend

ระบบ Frontend หรือ User Interface เพื่อให้ผู้ใช้สามารถสั่งงานการทำงานของระบบ เพื่อที่จะสร้างไฟล์รายงานใหม่ได้ โดยผู้จัดทำโครงงานได้พัฒนาต่อจากเว็บไซต์ที่มีอยู่แล้ว นั่นคือ Finance Backoffice เป็นเว็บไซต์ Frontend ที่ถูกพัฒนาด้วย React ทางผู้จัดทำโครงงานได้เพิ่มหน้าใหม่เข้าไปซึ่งว่า Reprocess AVL โดยภายในหน้าเพจนั้นจะมีสองระบบ คือระบบสำหรับดาวโหลดไฟล์ และ ระบบการสร้างไฟล์รายงานใหม่ ซึ่งผู้จัดทำโครงงานได้พัฒนาในส่วนของระบบการสร้างไฟล์รายงานใหม่



ภาพที่ 3.12 ภาพ Finance Backoffice หน้าสำหรับให้ผู้ใช้งานเพื่อสร้างไฟล์รายงานใหม่

จากการที่ 3.12 ผู้จัดทำโครงการได้ทำในส่วนด้านขวา สำหรับสร้างไฟล์รายงานใหม่ ซึ่งจะมีช่องให้ผู้ใช้สามารถเลือกวันที่ของข้อมูลรายงานที่ต้องการจะสร้างใหม่ได้ และกดปุ่มเพื่อสร้างไฟล์ โดยผู้ใช้ไม่จำเป็นต้องเลือกวันที่ก็ได้ โดยถ้าผู้ใช้ไม่เลือกวัน ระบบจะดำเนินการสร้างไฟล์รายงานด้วยข้อมูลของวันที่ปัจจุบันที่ผู้ใช้ทำการเรียกใช้งานระบบ

3.2.1.4 การทดสอบระบบ

การทดสอบระบบ ของระบบการสร้างไฟล์รายงานใหม่ จะแบ่งออกเป็น 2 ส่วนคือ การทดสอบระบบในระดับยูนิต หรือ Unit test และการทดสอบแบบ End to End Test

(1) Unit test

ผู้จัดทำโครงการต้องทำ Unit test ให้ได้มากกว่า 80% เป็นอย่างน้อย เพื่อให้มั่นใจได้ว่า ระบบสามารถทำงานได้อย่างถูกต้อง โดยการทำ Unit test เป็นการทดสอบระบบระดับต่ำสุด หรือในระดับยูนิตนั้นเอง ในส่วนนี้จะหมายถึงการทดสอบการทำงานในแต่ละฟังก์ชันต่างๆ ซึ่งภายในบางฟังก์ชัน จะมีการเรียกใช้งานฟังก์ชันอื่นร่วมด้วย จึงจำเป็นต้องใช้การ Mock ฟังก์ชันที่เกี่ยวข้องให้เป็นไปตาม Flow ของการทดสอบนั้นๆ ที่เราต้องการ

โดยจะ Mock ฟังก์ชันต่างๆ ด้วย Mockito Framework ซึ่งช่วยให้ผู้พัฒนาระบบสามารถดำเนินการทดสอบในแต่ละยูนิตของระบบได้โดยไม่กระทบต่อส่วนอื่นๆ ของระบบจริง

```

val mockAvlService = mock[AvlService]

when(mockAvlService.fetchAvlStatus(any[Int])).thenReturn(mockData)
val avlService = new AvlService(mockAvlService)

avlService.fetchAvlStatus(1) should be(mockData)

```

ภาพที่ 3.13 ภาพตัวอย่างการทดสอบยูนิตเทส ด้วยการใช้ Mockito ใน การ Mock พังก์ชัน

จากภาพที่ 3.13 เราจะทดสอบการทำงานของ avlService ซึ่งภายในพังก์ชันนั้น จะต้องเรียกใช้ตัวพังก์ชัน fetchAvlStatus ซึ่ง ตัว Mockito ทำให้เราสามารถตั้งได้ว่าพังก์ชันที่เกี่ยวข้องจะตอบกลับ หรือส่งข้อมูลกลับมาอย่างไร โดยไม่จำเป็นต้องรันพังก์ชันจริงๆ ส่งผลให้ผู้พัฒนาสามารถทดสอบในแต่ละยูนิตได้ทันที และเป็นไปได้ครบตามทุกๆ สถานการณ์ที่ควรจะเป็น และมีการทดสอบข้อมูลจำลอง โดยการจำลองข้อมูลฐานข้อมูลผ่าน Docker เพื่อทดสอบบนเครื่องตนเองก่อนที่จะนำไปทดสอบบนเซิฟเวอร์ QA ต่อไป

(2) End to End Test

การทดสอบแบบ End to End Test ผู้พัฒนาจะทำการทดสอบระหว่างโปรแกรมหลายโปรแกรมที่เชื่อมต่อกัน โดยทางผู้จัดทำโครงการได้ทดสอบการทำงานระหว่าง Frontend, Backend และ SparkJob โดยการทดสอบนี้เป็นการทดสอบบนเซิฟเวอร์ QA ที่สร้างมาเพื่อใช้สำหรับทดสอบการทำงานของระบบโดยเฉพาะ โดยผู้จัดทำโครงการได้ทดสอบระบบด้วยการเรียกใช้งานผ่าน Finance Backoffice (Frontend) ที่เป็น User Interface ซึ่งเมื่อเรียกใช้แล้ว การทำงานจะถูกส่งต่อไปยัง Backend และ Spark Job ตามลำดับ และตรวจสอบการทำงานว่าถูกต้องครบถ้วน หรือไม่

3.2.2 ระบบตรวจสอบและแสดงผลการสร้างรายงานสำคัญทางการเงิน

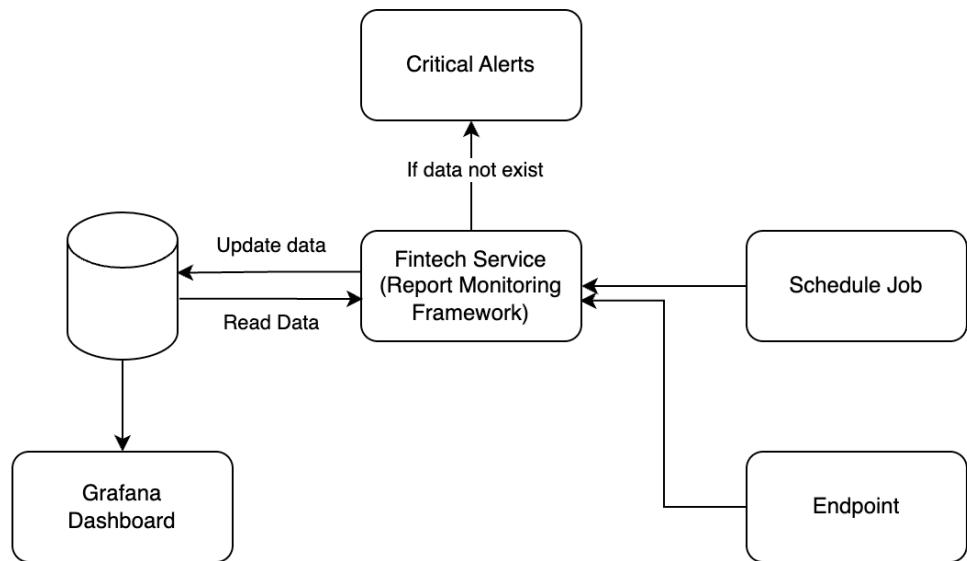
3.2.2.1 เก็บข้อมูล และ วิเคราะห์ความต้องการของผู้ใช้งาน

เนื่องจากในฝ่ายการเงินขององค์กร มีข้อมูลจำนวนมากที่ถูกเพิ่มเข้า
แก้ไข หรือลบออกอยู่ตลอดเวลาในทุกวัน ทำให้มีระบบอัตโนมัติที่ใช้สำหรับสร้างไฟล์รายงาน
ทางการเงินจำนวนมาก ซึ่งส่งผลให้มีการสร้างไฟล์รายงานทางการเงินจำนวนมากในทุกวัน ซึ่งผู้ใช้มี
สามารถตรวจสอบได้ทั้งหมด หรือทันท่วงที่ถ้าหากเกินข้อผิดพลาดในการสร้างไฟล์รายงานทางการ
เงินในบางส่วน เนื่องจากการตรวจสอบทั้งหมดเป็นไปได้ยาก เนื่องจากมีการสร้างไฟล์จำนวนมาก
พร้อมๆกัน และถูกจัดเก็บไว้ในแต่ละส่วน หรือฐานข้อมูลที่ต่างกัน

ส่งผลให้ผู้ใช้ ต้องใช้เวลานานมากเพื่อค้นหา และตรวจสอบไฟล์
รายงานที่ถูกสร้างขึ้นมาใหม่ ว่าถูกสร้างตามระบบอัตโนมัติที่กำหนดเอาไว้ได้อย่างถูกต้องหรือไม่ ถ้า
หากว่ามีไฟล์ที่ไม่ถูกสร้างตามกำหนดการ และทราบช้า อาจจะส่งผลกระทบในหลายส่วนได้
เนื่องจากเป็นรายงานสำคัญทางการเงิน

3.2.2.2 วิเคราะห์ขอบเขต และ ออกแบบระบบเพื่อนำมาแก้ปัญหา

ผู้จัดทำได้เห็นปัญหาของผู้ใช้ว่าไม่สามารถตรวจสอบว่าไฟล์ถูกสร้าง
แล้วหรือไม่ในทุกวัน ได้อย่างนีประสิทธิภาพ จึงออกแบบระบบอัตโนมัติที่จะตรวจสอบไฟล์ ตาม
กำหนดการของแต่ละไฟล์รายงานที่จะถูกสร้างโดยอัตโนมัติ โดยระบบจะเช็คไฟล์ทุกๆ 10 นาทีของ
ทุกวัน โดยจะออกแบบระบบให้ทำงานอัตโนมัติโดยอ่านข้อมูลชื่อไฟล์ สถานที่อยู่ของไฟล์ และ
กำหนดการที่ไฟล์จะถูกสร้าง จากฐานข้อมูล และตรวจสอบทุกๆครั้งเมื่อถึงกำหนดการสร้างไฟล์
รายงาน ซึ่งจำเป็นต้องสร้างฐานข้อมูลใหม่เพื่อให้ผู้ใช้ที่ต้องการตรวจสอบไฟล์ต่างๆที่เกี่ยวข้อง นำไป
เพื่อเพิ่มข้อมูลของการสร้างไฟล์อัตโนมัติ เพื่อใช้ในการตรวจสอบต่อไป



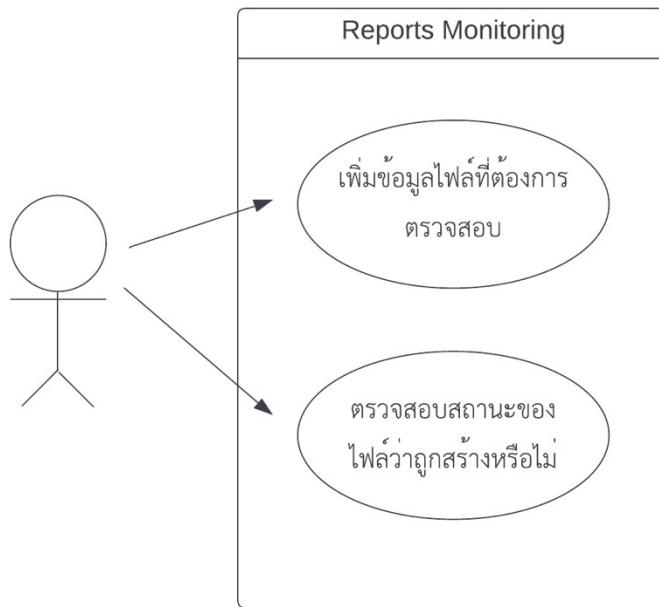
ภาพที่ 3.13 ภาพรวมการทำงานโดยรวมของระบบตรวจสอบไฟล์

จากภาพที่ 3.13 ระบบจะถูกเรียกใช้สำหรับตรวจสอบไฟล์ผ่าน Schedule Job โดยจะทำงานทุกๆ 10 นาที การทำงานตัวระบบจะอ่านข้อมูลไฟล์ที่ต้องตรวจสอบจากฐานข้อมูล และตรวจสอบว่าไฟล์นั้นถูกสร้างขึ้นหรือไม่ และบันทึกสถานะของไฟล์นั้นๆ กลับเข้าฐานข้อมูล ถ้าเกิดข้อมูลไม่ถูกสร้างขึ้นเกินเวลาที่กำหนด ตัวระบบจะส่งแจ้งเตือนไปยังผู้ใช้ที่เกี่ยวข้องกับไฟล์นั้นๆ และตัว Grafana Dashboard จะอ่านข้อมูลจากฐานข้อมูล และแสดงข้อมูลของสถานะไฟล์ทั้งหมดเพื่อให้ผู้ใช้สามารถตรวจสอบได้ทันที

โดยผู้ที่ต้องการใช้งานระบบ สามารถเพิ่มชื่อไฟล์ และข้อมูลของไฟล์ที่จะตรวจสอบเข้าไปในฐานข้อมูลของระบบได้ทันที โดยเพิ่มผ่าน API ซึ่งระบบทำสำหรับผู้พัฒนาคนอื่นในส่วนของการสร้างไฟล์ สามารถเพิ่มข้อมูลของการสร้างไฟล์ได้ผ่าน API เพื่อนำไปใช้เพื่อตรวจสอบสถานะการสร้างไฟล์ ดังนั้นตัวระบบจึงไม่มีหน้า User Interface สำหรับการเพิ่มข้อมูลของการสร้างไฟล์อัตโนมัติ แต่สามารถเพิ่มด้วย UI ของ Swagger Docs ได้เลย

ระบบการตรวจสอบไฟล์ต้องเชื่อมต่อไปยัง Server ต่างๆ ที่เก็บไฟล์ที่ถูกสร้างเอาไว้ เพื่อตรวจสอบว่าไฟล์ได้ถูกสร้างตามกำหนดโดย yogurt ต้องหรือไม่ จึงต้องมี ผู้ใช้และรหัสผ่าน ในการเข้าสู่ Server ต่างๆ โดยจะเก็บ ชื่อผู้ใช้เอาไว้ในฐานข้อมูลเลย แต่รหัสผ่านจะถูกเก็บไว้ใน Vault ซึ่งเป็นระบบความปลอดภัยสูง สำหรับเก็บรหัสผ่าน หรือ Token ต่างๆ เพื่อความปลอดภัย และ จะแสดงผลของสถานะไฟล์ไปยัง Grafana Dashboard เพื่อให้ผู้ใช้ดูได้ง่ายและรวดเร็ว รวมถึงมีการแจ้ง

เตือนไปยังแพลตฟอร์ม Slack เพื่อให้ผู้ใช้ได้ทราบทันที เมื่อเกิดข้อผิดพลาดในการสร้างไฟล์รายงาน เกิดขึ้น



ภาพที่ 3.14 แผนภาพกรณีใช้งานระบบตรวจสอบและแสดงผลการสร้างรายงานสำคัญ

จากภาพที่ 3.14 แสดงแผนผังการใช้งานของผู้ใช้ ซึ่งจากการวิเคราะห์ขอบเขตและความต้องการของผู้ใช้ โดยระบบจะสามารถให้ผู้ใช้ เพิ่มข้อมูลของไฟล์ที่ต้องการให้ตรวจสอบได้ และ ผู้ใช้สามารถดูสถานะของไฟล์ว่าถูกสร้างตามกำหนดการหรือไม่ได้ทันทีผ่าน Grafana Dashboard และ เมื่อไฟล์ไม่ถูกสร้างตามกำหนด จะแจ้งเตือนไปยังผู้ใช้ผ่าน Slack Application ทันที เพื่อให้ผู้ใช้เข้ามาตรวจสอบได้อย่างทันทีและมีประสิทธิภาพ

ตารางที่ 3.5 แสดงรายละเอียด Actor ใน Use case diagrams

Actor	Description
พนักงาน	พนักงานที่ต้องการตรวจสอบสถานะการสร้างไฟล์

ตารางที่ 3.6 แสดงรายละเอียดกรณีใช้งาน ต่างๆใน Use Case Diagrams

No.	Use Case Name	Actor	Description

UC-01	เพิ่มข้อมูลของไฟล์ที่ต้องการตรวจสอบ	พนักงาน	พนักงานนำเข้าข้อมูลที่เกี่ยวข้อง เพื่อให้ระบบสามารถตรวจสอบสถานะของไฟล์ได้
UC-02	ตรวจสอบสถานะของไฟล์	พนักงาน	ตรวจสอบสถานะของไฟล์ ว่าถูกสร้างตามกำหนดหรือไม่ ผ่าน Dashboard

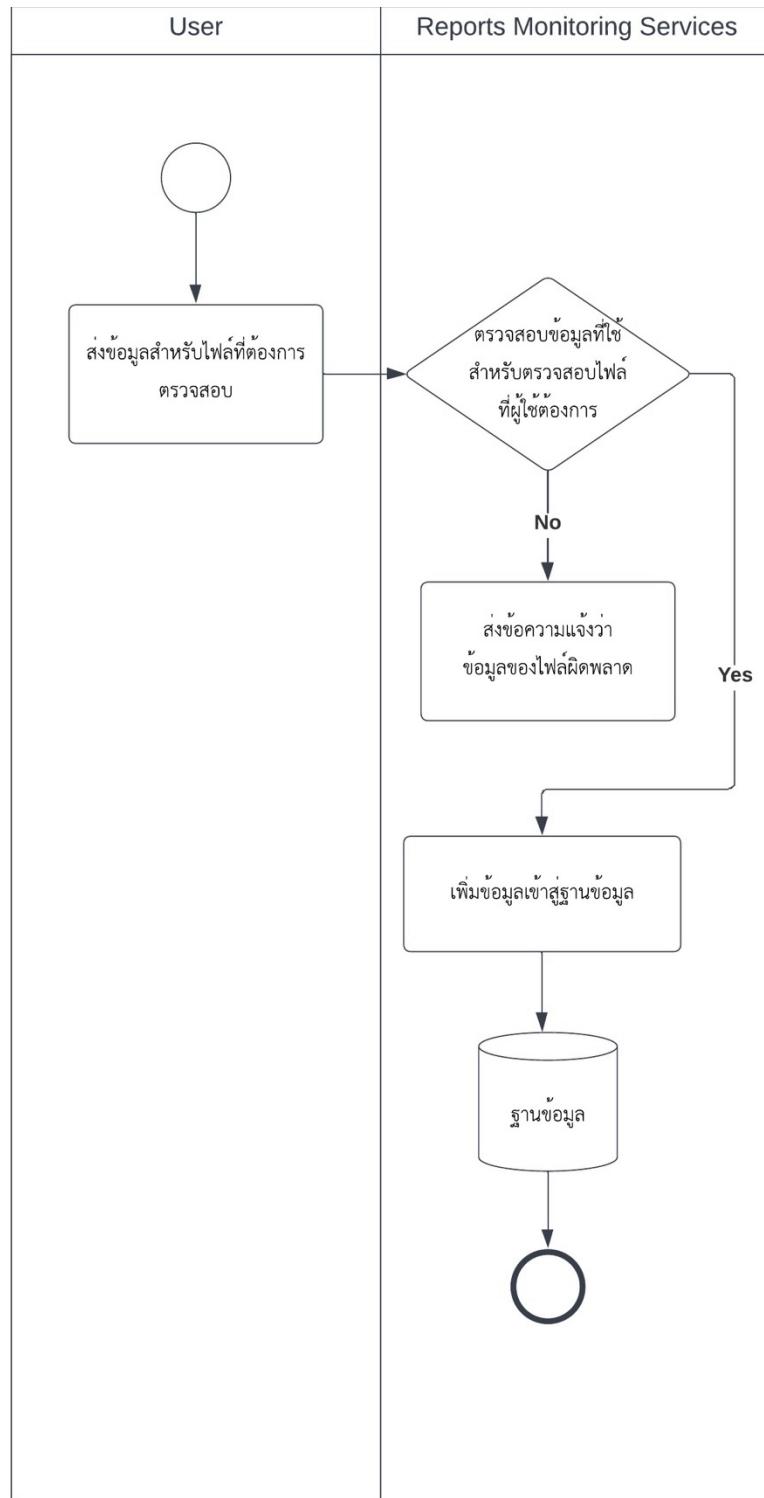
ตารางที่ 3.7 รายละเอียดกรณีใช้งาน Use Case UC-01

รหัสยูสเคส (Use Case ID)	UC-01
ชื่อยูสเคส (Use Case Name)	เพิ่มข้อมูลของไฟล์ที่ต้องการตรวจสอบ
ผู้ใช้งาน (Actor)	พนักงาน
คำอธิบาย (Description)	พนักงานนำเข้าข้อมูลที่เกี่ยวข้อง เพื่อให้ระบบสามารถตรวจสอบสถานะของไฟล์ได้
เงื่อนไขก่อนหน้า (Pre-condition)	-
เงื่อนไขภายหลัง (Post-condition)	ระบบตรวจสอบข้อมูลที่ผู้ใช้ต้องการเพิ่มเข้าสู่ฐานข้อมูล
กระแสหลัก (Basic Flow)	<ol style="list-style-type: none"> พนักงานเข้าสู่หน้า UI ของ Swagger Docs เพิ่มข้อมูลของไฟล์ที่จะให้ตรวจสอบ ใน Endpoint ที่ใช้สำหรับเพิ่มข้อมูลเพื่อตรวจสอบไฟล์ กดส่งข้อมูลไปยัง Endpoint
กระแสรอง (Alternative Flow)	<ol style="list-style-type: none"> กรณีที่ข้อมูลผิด หรือ ไม่สามารถเพิ่มข้อมูลได้ ระบบจะแสดงข้อความเพื่อบอกผู้ใช้ว่าเกิดข้อผิดพลาดในการเพิ่มข้อมูล แสดงการแจ้งเตือนที่จอภาพ

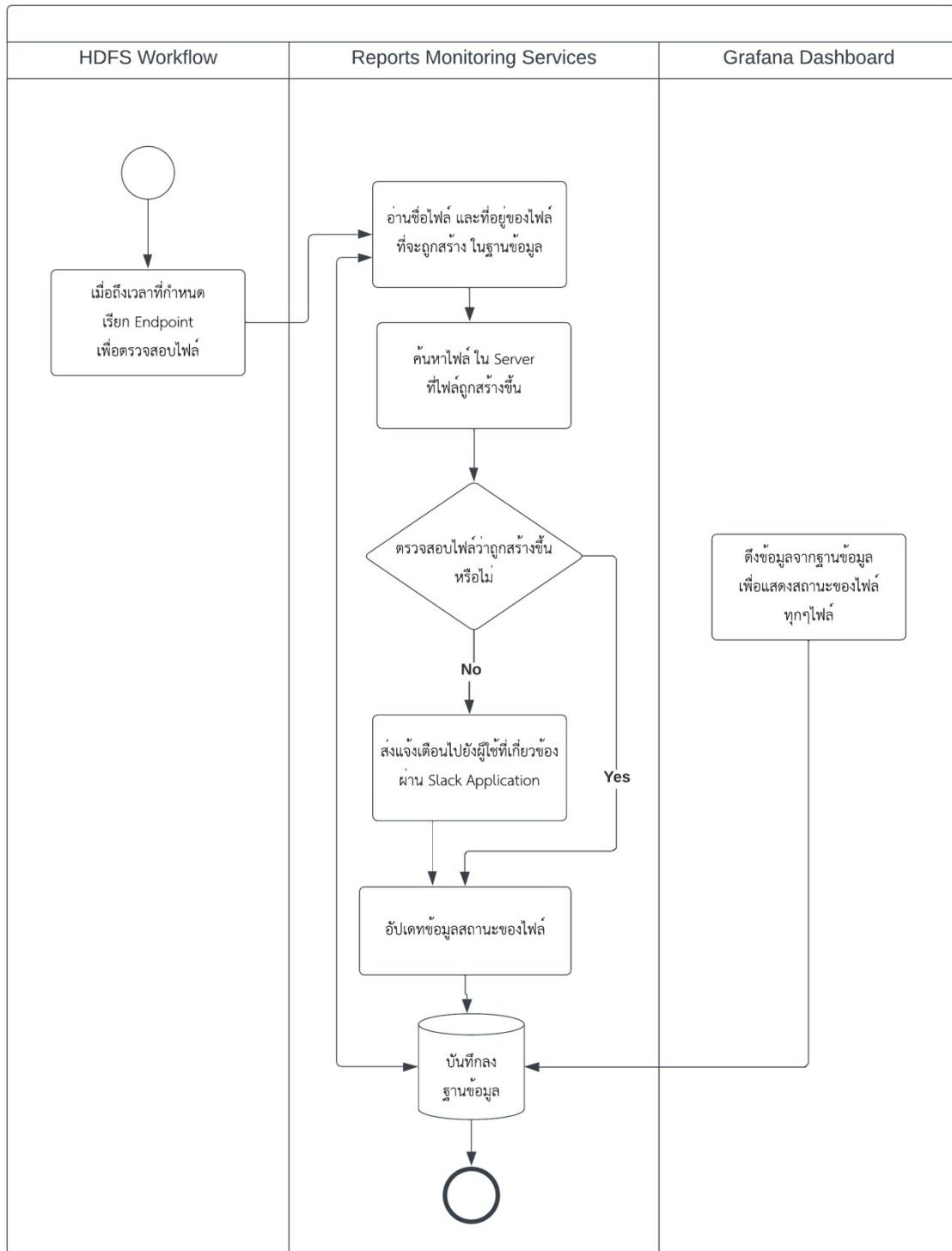
ตารางที่ 3.8 รายละเอียดกรณีใช้งาน Use Case UC-02

รหัสยูสเคส (Use Case ID)	UC-02
ชื่อยูสเคส (Use Case Name)	ตรวจสอบสถานะของไฟล์
ผู้ใช้งาน (Actor)	พนักงาน
คำอธิบาย (Description)	ตรวจสอบสถานะของไฟล์ ว่าถูกสร้างตามกำหนดหรือไม่ ผ่าน Dashboard
เงื่อนไขก่อนหน้า (Pre-condition)	ผู้ใช้ต้องเพิ่มข้อมูลสำหรับไฟล์ ที่ต้องการให้ตรวจสอบสถานะ
เงื่อนไขภายหลัง (Post-condition)	-
กระแสหลัก (Basic Flow)	<ol style="list-style-type: none"> พนักงานเข้าสู่หน้า Grafana ของระบบ ตรวจสอบไฟล์รายงาน ค้นหาไฟล์ และตรวจสอบสถานะของไฟล์
กระแสรอง (Alternative Flow)	-

แผนภาพแสดงขั้นตอนการทำงาน (Activity Diagram)



ภาพที่ 3.15 ภาพแสดง Activity Diagram สำหรับเพิ่มข้อมูลไฟล์เพื่อตรวจสอบ



ภาพที่ 3.16 ภาพแสดง Activity Diagram การตรวจสอบไฟล์

3.2.2.3 ขั้นตอนการ implement

ระบบนี้ถูกออกแบบให้ทำงานอัตโนมัติโดยตรวจสอบไฟล์ทุกวันที่ถูกสร้างขึ้น โดยจะทำงานในส่วนของการตรวจสอบสถานะการเมื่อยื่นของไฟล์ทุกๆ 10 นาที โดยตั้ง Schedule Job ผ่าน HDFS Workflow มตั้งแต่เวลา ให้ไปเรียกใช้ Endpoint สำหรับตรวจสอบไฟล์ทุกๆ 10 นาที โดยใช้ Cron Schedule ในการตั้งค่า ซึ่งผู้ดูแลโครงงาน ได้ตั้งให้ตรวจสอบไฟล์ทุกๆ นาทีที่ 10 โมง เวลา 08.00 ถึง 20.00 ของทุกวัน สาเหตุที่ไม่ตรวจสอบตลอด 24 ชั่วโมง เนื่องจากเวลาทำงานขององค์กรจะอยู่ในช่วง 08.00 ถึง 20.00 เท่านั้น ทำให้มีความจำเป็นที่จะให้ทำงานตลอด เพื่อลดการใช้ทรัพยากรที่ไม่จำเป็นด้วย

(1) การสร้างฐานข้อมูล

การพัฒนาระบบ เริ่มจากออกแบบ และสร้างโครงสร้างของฐานข้อมูล โดยออกแบบฐานข้อมูลโดยมีคอลัมน์ทั้งหมด 17 คอลัมน์

```
CREATE TABLE [dbo].[finance_report_monitoring]
(
    [report_monitoring_id] [int] IDENTITY (1,1) NOT FOR REPLICATION NOT NULL,
    [file_name] [varchar](100) NOT NULL,
    [file_location] [varchar](100) NOT NULL,
    [is_erp] [bit] NOT NULL,
    [team_owner] [varchar](50) NOT NULL,
    [user_info] [varchar](50) NOT NULL,
    [offset] [smallint] NOT NULL,
    [cron_schedule] [varchar](30) NOT NULL,
    [cutoff_time] [time](0) NOT NULL,
    [source_type] [varchar](10) NOT NULL,
    [file_generation_date] [datetime] NOT NULL,
    [is_generated] [bit] NOT NULL,
    [rec_created_when] [datetime] NOT NULL,
    [rec_created_by] [uniqueidentifier] NOT NULL,
    [rec_status] [smallint] NOT NULL,
    [rec_modified_when] [datetime] NULL,
    [rec_modified_by] [uniqueidentifier] NULL
)
GO
```

ภาพที่ 3.17 ภาพคำสั่งการสร้างฐานข้อมูลของระบบตรวจสอบไฟล์รายงาน

จากภาพที่ 3.16 เป็นการสร้างฐานข้อมูลเพื่อใช้เป็นข้อมูลในการตรวจสอบไฟล์รายงานที่ถูกสร้างขึ้นโดยอัตโนมัติ โดยการทำงานของแต่ละคอลัมน์บนฐานข้อมูลมีดังนี้

- report_monitoring_id เป็นชนิดข้อมูล Int โดยใช้เป็น primary key สำหรับตารางนี้ โดยจะถูกสร้างโดยอัตโนมัติ เมื่อเพิ่มข้อมูลเข้าสู่ตาราง
- file_name ชนิด varchar 100 ใช้สำหรับเก็บชื่อไฟล์ของรายงานที่ต้องการตรวจสอบ

- file_location ชนิด varchar 100 ใช้สำหรับเก็บที่อยู่ของไฟล์รายงานนั้นๆ ว่าถูกสร้างอยู่ในโฟเดอร์ใด
- is_erp ชนิด bit หรือ Boolean ใช้สำหรับเก็บค่าว่าเป็นไฟล์ที่มีความสำคัญสูงหรือไม่โดยจะเก็บค่า true หรือ false
- team_owner ชนิด varchar 50 ใช้สำหรับเก็บชื่อของทีมที่เกี่ยวข้อง เพื่อนำไปใช้สำหรับการแจ้งเตือนผ่าน Slack Application
- user_info ชนิด varchar 50 ใช้สำหรับเก็บชื่อผู้ใช้ ที่ใช้สำหรับเข้าสู่ระบบของเชิฟเวอร์ที่ถูกเก็บไฟล์เอาไว้ โดยรหัสผ่านจะถูกดึงมาจาก Vault อีกที
- offset ชนิด smallint ใช้สำหรับกำหนดวัน Delay เนื่องจากจะมีบางไฟล์ที่จะสร้างรายงานของเมื่อวาน โดยถ้าตั้ง offset เป็น 1 ระบบตรวจสอบไฟล์ จะทำการตรวจสอบไฟล์ของเมื่อวานเท่านั้น ถ้าเป็น 0 คือจะตรวจสอบไฟล์ของวันที่ปัจุบัน
- cron_schedule ชนิด varchar 30 ใช้สำหรับเก็บ cron schedule ของไฟล์รายงานว่าจะถูกสร้างขึ้นเมื่อใด โดยใช้ชนิดของ unix cron
- cutoff_time ชนิด time 0 ใช้สำหรับเก็บเวลาที่ยอมรับได้ไฟล์ไม่ควรสร้างข้ากวันนั้น เมื่อกินเวลา cutoff แล้วไฟล์รายงานยังไม่ถูกสร้างขึ้น ระบบจะทำการแจ้งเตือนไปยังผู้ใช้ และแจ้งเตือนใน Slack Application ทันที
- source_type ชนิด varchar 10 ใช้สำหรับเก็บ Protocol ที่ใช้สำหรับเชื่อมต่อเชิฟเวอร์ที่เก็บไฟล์ โดยจะมี 2 ชนิดคือ sftp และ sbm
- file_generation_date ชนิด datetime ใช้สำหรับเก็บวันที่และเวลาของไฟล์รายงานที่จะถูกสร้างขึ้น โดยจะแปลงจาก cron schedule เป็นวันที่ในทุกๆรอบของการสร้างไฟล์
- is_generated ชนิด bit หรือ Boolean ใช้สำหรับเก็บสถานะของไฟล์รายงานว่าถูกสร้างขึ้นได้ตามกำหนดหรือไม่ โดยถ้าถูกสร้างขึ้นแล้วค่าจะถูกเปลี่ยนเป็น True ถ้ายังไม่ถูกสร้างค่าจะเป็น False
- rec_created_when ชนิด datetime ใช้สำหรับเก็บวันที่และเวลาที่เพิ่มข้อมูลเข้าสู่ตารางนี้
- rec_created_by ชนิด uniqueidentifier หรือ UUID ใช้สำหรับเก็บ UUID ของผู้ที่เพิ่มข้อมูลเข้าสู่ตาราง
- rec_modified_when ชนิด datetime ใช้สำหรับเก็บวันที่และเวลาล่าสุดที่อัปเดต record นั้นๆ
- rec_modified_by ชนิด uniqueidentifier หรือ UUID ใช้สำหรับเก็บ UUID ของผู้ที่แก้ไขหรืออัปเดตข้อมูลใน record นั้นๆล่าสุด

(2) การสร้าง Stored Procedure และ Data Table

เมื่อสร้างตารางสำหรับเก็บข้อมูลเรียบร้อยแล้ว จึงสร้าง Stored Procedure เพื่อให้สามารถเรียกใช้งาน Query SQL ได้ทันที และมีประสิทธิภาพ โดยไม่ต้องส่งคำสั่ง SQL ไปใหม่ หลายครั้ง โดยมี Stored Procedure ทั้งหมด 3 ตัว และมี 1 Data Type เพื่อใช้สำหรับ Bulk Update เนื่องจากการตรวจสอบไฟล์ มีไฟล์จำนวนมากที่ต้องตรวจสอบและอัปเดตสถานะของไฟล์ ตลอดเวลา จึงพัฒนาการ Bulk Update หรือการอัปเดตจำนวนมาก โดยใช้ Data table เข้ามาแทน การวนลูปเพื่ออัปเดตข้อมูลในแต่ละไฟล์ โดยมีดังนี้

(ก) dbo.fetch_all_reports_monitoring

ใช้สำหรับดึงข้อมูลรายงานทั้งหมดที่ถูกสร้างขึ้น โดยจะรับ datetime เข้ามา และจะค้นหาข้อมูลที่ไฟล์ควรจะถูกสร้างวันนี้ หรือย้อนหลัง

```

CREATE PROCEDURE [dbo].[fetch_all_reports_monitoring_v1]
    @today_date AS datetime
AS
BEGIN
    SET NOCOUNT ON;
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

    ;WITH T AS
    (
        SELECT report_monitoring_id
        FROM dbo.finance_report_monitoring
        WHERE file_generation_date <= @today_date
    )

    SELECT T.report_monitoring_id
        ,F.file_name
        ,F.file_location
        ,F.is_erp
        ,F.team_owner
        ,F.user_info
        ,F.offset
        ,F.cron_schedule
        ,F.cutoff_time
        ,F.source_type
        ,F.file_generation_date
        ,F.is_generated
        ,F.rec_created_when
        ,F.rec_created_by
        ,F.rec_status
        ,F.rec_modified_when
        ,F.rec_modified_by
    FROM T
        INNER JOIN dbo.finance_report_monitoring AS F
        ON T.report_monitoring_id = F.report_monitoring_id
END
GO

```

ภาพที่ 3.18 ภาพ Stored Procedure dbo.fetch_all_reports_monitoring

ใน Stored Procedure นี้จะเห็นว่ามีการดึง report_monitoring_id อย่างเดียวมาก่อน และนำ id ที่ได้มาใช้ดึงข้อมูลของทุกๆ คอลัมน์อีกที ที่ต้องทำแบบนี้เนื่องจาก report_monitoring_id เป็น key index ซึ่งเมื่อค้นหาด้วย key index จะช่วยเพิ่มประสิทธิภาพ และความเร็วในการค้นหาข้อมูลของฐานข้อมูลได้ ทำให้ประหยัดทรัพยากรน์ และเวลาในการค้นหาข้อมูลจำนวนมาก

(ข) dbo.upsert_reports_monitoring

เป็น Stored Procedure ที่ใช้สำหรับเพิ่มข้อมูลเข้ามาในตารางใหม่ และยังสามารถใช้ในการอัปเดทข้อมูลที่มีอยู่แล้วได้อีกด้วย จึงใช้คำว่า Upsert โดยข้อแตกต่างของ Insert และ Upsert คือ Upsert หมายถึงการ Insert และสามารถ Update ได้ด้วย

```

CREATE PROCEDURE [dbo].[upsert_reports_monitoring_v1]
    ... @file_name varchar(100),
    ... @file_location varchar(100),
    ... @is_erp bit,
    ... @team_owner varchar(50),
    ... @user_info varchar(50),
    ... @offset smallint,
    ... @cron_schedule varchar(30),
    ... @cutoff_time time(0),
    ... @source_type varchar(10),
    ... @file_generation_date datetime,
    ... @rec_status smallint
AS
BEGIN
    ... SET NOCOUNT ON
    ... SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    ... IF NOT EXISTS (SELECT 1 FROM [dbo].[finance_report_monitoring] WHERE file_name = @file_name)
BEGIN
    INSERT INTO [dbo].[finance_report_monitoring]
    (file_name,
    file_location,
    is_erp,
    team_owner,
    user_info,
    offset,
    cron_schedule,
    cutoff_time,
    source_type,
    file_generation_date,
    is_generated,
    rec_created_when,
    rec_created_by,
    rec_status,
    rec_modified_when,
    rec_modified_by)
    VALUES (@file_name,
    ... @file_location,
    ... @is_erp,
    ... @team_owner,
    ... @user_info,
    ... @offset,
    ... @cron_schedule,
    ... @cutoff_time,
    ... @source_type,
    ... @file_generation_date,
    ... 0,
    ... GETDATE(),
    ... '00000000-0000-0000-0000-000000000027',
    ... @rec_status,
    ... NULL,
    ... NULL
    ... )
END
ELSE

```

ການພິຈາລະນາ Stored Procedure dbo.upsert_reports_monitoring

```

BEGIN
WITH cte AS (
SELECT [report_monitoring_id]
FROM [dbo].[finance_report_monitoring]
WHERE file_name = @file_name
)

UPDATE f
SET file_location = @file_location,
is_erp = @is_erp,
team_owner = @team_owner,
user_info = @user_info,
offset = @offset,
cron_schedule = @cron_schedule,
cutoff_time = @cutoff_time,
source_type = @source_type,
file_generation_date = @file_generation_date,
rec_status = @rec_status,
rec_modified_when = GETDATE(),
rec_modified_by = '00000000-0000-0000-0000-000000000027'
FROM [dbo].[finance_report_monitoring] f
INNER JOIN cte ON cte.[report_monitoring_id]=f.[report_monitoring_id]
END
END
GO

```

ภาพที่ 3.20 ภาพ Stored Procedure dbo.upsert_reports_monitoring (ต่อ)

จากภาพที่ 3.18 และ 3.19 ผู้ใช้จะต้องส่งข้อมูลเพื่อเพิ่มไปยังฐานข้อมูล หรือ อัปเดทข้อมูล โดยการอัปเดทข้อมูล จะยึดกับชื่อของไฟล์รายงาน ถ้าชื่อไฟล์ที่เพิ่มเข้ามามีอยู่ในฐานข้อมูลอยู่แล้ว แต่ข้อมูลอื่นๆมีการเปลี่ยนแปลง Stored Procedure นี้จะทำการอัปเดทข้อมูลที่ถูกเปลี่ยนแปลงไปยัง Recode นั้นแทนการเพิ่มข้อมูลลงใหม่

โดยรับข้อมูลตามมิต่อรดังนี้

- file_name ชื่อไฟล์ของรายงานที่ต้องการตรวจสอบ
- file_location ที่อยู่ของไฟล์รายงานนั้นๆ ว่าถูกสร้างอยู่ในไฟล์เดอร์ใด
- is_erp ค่าว่าเป็นไฟล์ที่มีความสำคัญสูงหรือไม่
- team_owner ชื่อทีมที่เกี่ยวข้อง
- user_info ชื่อผู้ใช้สำหรับเข้าสู่ Server ที่เก็บไฟล์อยู่ เพื่อตรวจสอบไฟล์
- offset กำหนดวัน Delay ของไฟล์ โดยถ้าเป็น 1 จะตรวจสอบไฟล์ของเมื่อวานแทน
- cron_schedule เก็บกำหนดการที่ไฟล์รายงานจะถูกสร้างขึ้น
- cutoff_time เวลาที่ยอมรับได้ ในการรอไฟล์รายงานสร้าง
- source_type รูปแบบการเชื่อมต่อไปยัง Server (smb, sftp)

- file_generation_date วันที่ไฟล์จะถูกสร้างขึ้น (ระบบจะสร้างขึ้น ผู้ใช้มีจำเป็นต้องใส่ เอง)
- rec_status กำหนดค่ายังให้ระบบตรวจสอบไฟล์นี้อยู่หรือไม่

(ค) dbo.update_reports_monitoring_batch

เป็น Stored Procedure ที่ใช้สำหรับอัปเดtex้อมูลจำนวนมาก โดยจะใช้รวม กับ Data table ในการทำงาน โดยจะสร้างตารางใหม่ที่เป็น Temp ขึ้นมา และใส่ ข้อมูลที่ต้องการอัปเดททั้งหมดลงในตารางใหม่ที่สร้างขึ้น จากนั้นนำข้อมูลทั้งหมดใน ตารางชั่วคราว ไปรวมกับตารางจริง โดยใช้ INNER JOIN จะทำให้มีต้องอัปเดท ข้อมูลทีละ Record เพื่อเพิ่มประสิทธิภาพ และความเร็วในการอัปเดtex้อมูลบน ฐานข้อมูล

```
CREATE TYPE dbo.update_reports_monitoring_v1 AS TABLE (
    report_monitoring_id int NOT NULL,
    file_generation_date datetime NOT NULL,
    is_generated bit NOT NULL,
    rec_modified_when datetime NULL,
    rec_modified_by uniqueidentifier NULL
)
GO
```

ภาพที่ 3.21 ภาพ Data Table dbo.update_reports_monitoring

จากภาพที่ 3.19 เป็นการสร้างตารางชั่วคราว (Temp table) เพื่อใช้สำหรับใส่ ข้อมูลทั้งหมดที่ต้องการอัปเดทเข้าไป เพื่อนำไปรวมกับตารางหลักอีกที

```

CREATE PROCEDURE [dbo].[update_reports_monitoring_batch_v1]
    --@report_monitoring_data AS dbo.update_reports_monitoring_v1 READONLY
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
    BEGIN TRANSACTION
        BEGIN
            UPDATE dbo.finance_report_monitoring
            SET file_generation_date = T.file_generation_date,
                is_generated = T.is_generated,
                rec_modified_when = T.rec_modified_when,
                rec_modified_by = T.rec_modified_by
            FROM dbo.finance_report_monitoring AS F
            INNER JOIN @report_monitoring_data AS T
            ON F.report_monitoring_id = T.report_monitoring_id
        END
    COMMIT TRANSACTION
    SELECT report_monitoring_id,
        file_name,
        file_location,
        is_erp,
        team_owner,
        user_info,
        offset,
        cron_schedule,
        cutoff_time,
        source_type,
        file_generation_date,
        is_generated,
        rec_created_when,
        rec_created_by,
        rec_status,
        rec_modified_when,
        rec_modified_by
    FROM dbo.finance_report_monitoring
    WHERE report_monitoring_id IN (SELECT report_monitoring_id FROM @report_monitoring_data)
END
GO

```

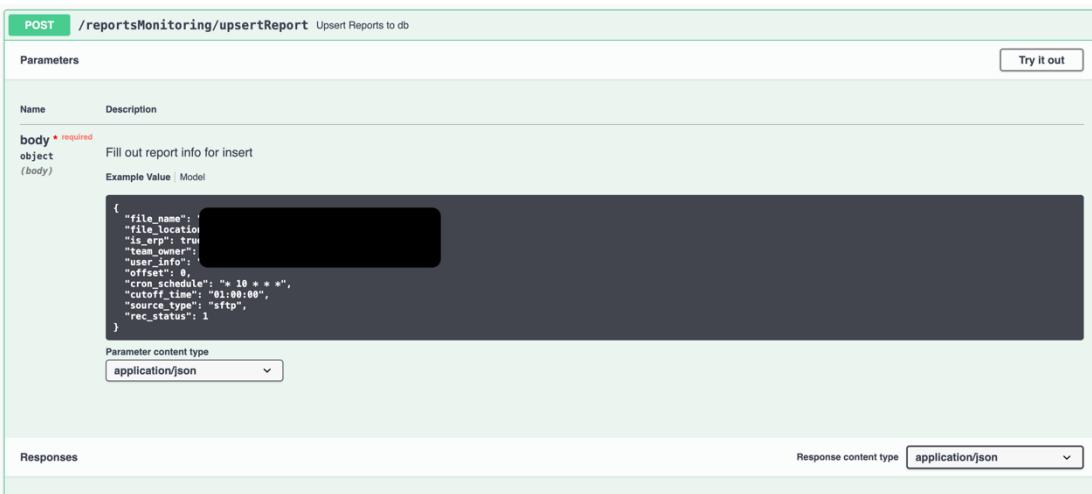
ภาพที่ 3.22 ภาพ Stored Procedure dbo.update_reports_monitoring_batch

ใช้สำหรับอัปเดทข้อมูลในตาราง โดยเอาข้อมูลที่ต้องการอัปเดททั้งหมดไปใส่ไว้ในตารางชั่วคราวก่อน ดังภาพที่ 3.20 จากนั้นส่งตารางชั่วคราว ส่งมา�ัง Stored Procedure นี้ และใช้ Inno join เพื่อร่วมข้อมูลจากตารางชั่วคราวเข้าสู่ตารางหลัก ทำให้สามารถอัปเดทข้อมูลจำนวนหลาย record ได้ในการสั่งงานฐานข้อมูลเพียงครั้งเดียว ช่วยให้ระบบมีประสิทธิภาพและรวดเร็ว

(3) Backend

โครงสร้างของระบบ Backend จะถูกพัฒนาด้วย play framework บนภาษา Scala และโครงสร้างจะถูกแบ่งออกเป็น 3 ส่วนคือ Repository, Service และ Controller เพื่อแยกการทำงานในแต่ละส่วน และมี Endpoint สำหรับให้ผู้ใช้สามารถเรียกใช้ระบบบันได

โดยในระบบบันไดมี Endpoint ทั้งหมด 2 Endpoint เพื่อใช้สำหรับเรียกการทำงานเพื่อตรวจสอบไฟล์รายงานว่าถูกสร้างได้ตามกำหนดหรือไม่ และ ใช้สำหรับเพิ่มหรือแก้ไขข้อมูลใหม่เข้าสู่ฐานข้อมูลของระบบนี้ โดยการเพิ่มข้อมูลเข้าสู่ฐานข้อมูล ผู้ใช้สามารถเพิ่มข้อมูลผ่าน Swagger Doc Tool ได้ทันที



ภาพที่ 3.23 ภาพ Swagger Doc ของ Endpoint สำหรับเพิ่ม หรือ แก้ไขข้อมูลในระบบ

จากภาพที่ 2.23 เป็น Endpoint สำหรับเพิ่มข้อมูลของไฟล์รายงานที่ต้องการให้ระบบตรวจสอบ โดยสามารถเพิ่มข้อมูลใหม่ และ แก้ไขข้อมูลได้ผ่าน Endpoint นี้ได้ทันที โดยใช้เป็น method POST ผ่าน REST API เพื่อให้ผู้ใช้งานสามารถใช้งานได้ง่ายผ่าน Swagger Doc ได้ทันที และใช้ POST เนื่องจากเป็นการเพิ่มข้อมูลเข้าไปในระบบฐานข้อมูล

POST /reportsMonitoring/generateReportsMonitoring Update Status of Reports Monitoring

Parameters

No parameters

Responses

Code	Description
200	successful operation

ภาพที่ 3.24 ภาพ Swagger Doc ของ Endpoint สำหรับให้ระบบอัตโนมัติเรียกใช้ เพื่อตรวจสอบไฟล์รายงาน

จากภาพ 3.24 เป็น Endpoint ที่ไม่ได้รับพารามิเตอร์ใดๆ ใช้เพื่อให้ระบบทำการตรวจสอบไฟล์เท่านั้น โดยทำงานร่วมกับ Schedule ใน HDFS Workflow เพื่อส่งคำสั่งมาเรียกใช้ Endpoint นี้ทุกๆ 10 นาทีตั้งแต่เวลา 08.00 ถึง 20.00 เพื่อสามารถตรวจสอบไฟล์ได้ทันท่วงที และมีประสิทธิภาพ

(ก) Repository

ใช้สำหรับเชื่อมต่อ กับระบบฐานข้อมูล เพื่อดึงข้อมูล และบันทึกข้อมูลต่างๆ ไปยังฐานข้อมูล โดยระบบตรวจสอบไฟล์รายงานจะมีฟังก์ชันดังนี้

- fetchAllReportsMonitoring ใช้สำหรับเรียกใช้ Stored Procedure เพื่อดึงข้อมูล ทั้งหมดของไฟล์รายงาน เพื่อนำไปตรวจสอบต่อไป โดยจะเรียกใช้ Stored Procedure ชื่อว่า dbo.fetch_reports_monitoring
- updateReportsMonitoring ใช้สำหรับอัปเดทข้อมูลต่างๆ เช่นสถานะของไฟล์ปัจจุบัน โดยจะส่งข้อมูลจาก Data table dbo.update_reports_monitoring ไปยัง Stored Procedure dbo.update_reports_monitoring_batch เพื่ออัปเดทข้อมูลจำนวนมาก
- upsertReport ใช้สำหรับเพิ่ม หรือแก้ไขข้อมูลของไฟล์รายงานที่ต้องการตรวจสอบเข้าไปยังฐานข้อมูลโดยจะเรียกใช้ Stored Procedure dbo.upsert_reports_monitoring

(ข) Service

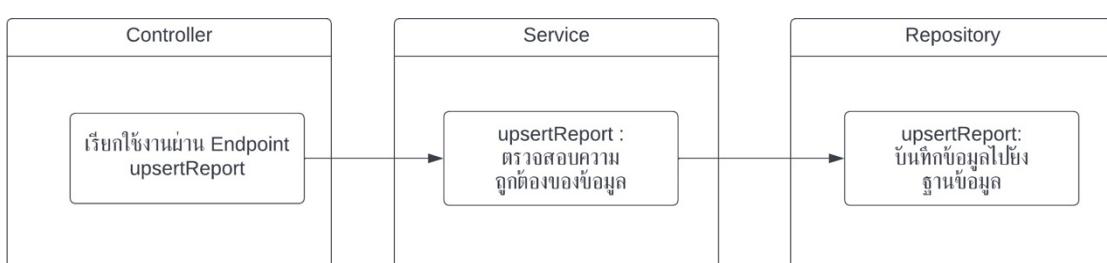
ใช้สำหรับให้บริการฟังก์ชันการทำงานในส่วนต่างๆ ของระบบ โดยแต่ละเซอร์วิสจะมีหน้าที่เฉพาะโดยเฉพาะ โดยจะมีฟังก์ชันดังนี้

- fetchAllReportsMonitoring พังก์ชันนี้จะไปเรียกใช้ฟังก์ชันใน Repository เพื่อดึงข้อมูลของไฟล์รายงาน เพื่อนำไปตรวจสอบต่อ
- updateReportsMonitoring พังก์ชันนี้จะเรียกใช้ฟังก์ชันที่เชื่อมเดียวกันใน Repository เพื่อใช้สำหรับการอัปเดตสถานะของแต่ละไฟล์
- upsertReport พังก์ชันนี้จะเรียกใช้ฟังก์ชันที่เชื่อมเดียวกันใน Repository เพื่อใช้สำหรับเพิ่ม หรือแก้ไขข้อมูลของไฟล์รายงาน
- convertUnixToQuartzCron พังก์ชันนี้ใช้สำหรับแปลง unix cron ให้เป็น quartz cron เพื่อให้สามารถใช้ไลบ์แลรี่ของ Java ในการตรวจสอบเวลาการทำงานของ cron ได้
- convertCronJobToDate พังก์ชันนี้ใช้สำหรับหารวบ หรือเวลาถัดไปที่จะเกิดการทำงาน เป็นการแปลง Quartz cron เป็นวัน เวลาถัดไปที่จะทำงาน
- fileExits ใช้เพื่อเชื่อมต่อไปยัง Server ต่างๆตามที่อยู่ของแต่ละไฟล์เพื่อตรวจสอบว่าไฟล์ได้ถูกสร้างตามกำหนดหรือไม่
- getReportsMonitoringPassword ใช้สำหรับดึงรหัสผ่านที่ใช้เชื่อมต่อกับ Server ต่างๆ ของแต่ละผู้ใช้ โดยจะเชื่อมต่อไปยัง Vault ที่เก็บรหัสผ่านเอาไว้

(ค) Controller

หน้าที่ควบคุมการทำงานทั้งหมด ในการเรียกใช้งานเซอร์วิสต่างๆ โดยเมื่อผู้ใช้เรียก Endpoint เข้ามา ตัว Controller จะนำไปประมวลผลตามที่ผู้พัฒนาได้กำหนดไว้ในแต่ละ Endpoint ช่วยให้สามารถจัดการกระบวนการทำงาน ในส่วนต่างๆของระบบไฟล์อย่างมีประสิทธิภาพ โดยมีการทำงานตามจำนวน Endpoint ที่มีคือ 2 การทำงานดังนี้

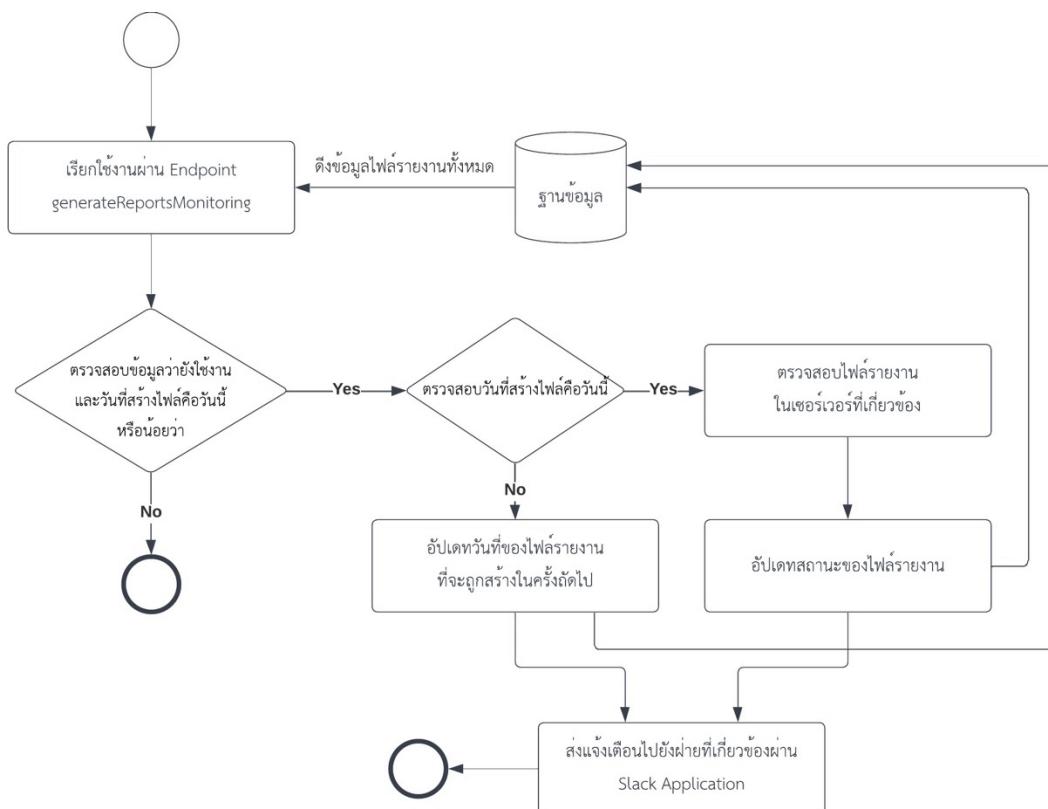
- upsertReport
เมื่อผู้ใช้เรียกใช้ upsertReport ตัว Controller จะรับค่าข้อมูลต่างๆจากผู้ใช้ และส่งไปยัง Service upsertReport เพื่อตรวจสอบข้อมูลว่าถูกต้องหรือไม่ ถ้าข้อมูลถูกต้องจะถูกส่งต่อไปยัง Repository upsertReport เพื่อนำไปเพิ่มหรือแก้ไขข้อมูลในฐานข้อมูลต่อไป



ภาพที่ 3.25 แสดงการทำงานของ Endpoint upsertReport

- generateReportsMonitoring

เมื่อถูกเรียกใช้ Endpoint นี้จะดึงข้อมูลทั้งหมดจากฐานข้อมูลที่ไฟล์ควรจะถูกสร้างวันนี้ หรือน้อยกว่าวันนี้มาทั้งหมด จากนั้นจะแยกเป็น 2 ส่วนคือ ถ้าไฟล์ควรจะถูกสร้างวันนี้ หรือถึงกำหนดการสร้างไฟล์รายงานแล้ว ระบบจะเข้าไปตรวจสอบไฟล์ภายในเซิฟเวอร์ ต่างๆเพื่อตรวจสอบว่าไฟล์รายงานถูกสร้างตามกำหนดหรือไม่ และแก้ไขข้อมูลสถานะในฐานข้อมูล และกรณีที่วันที่ของไฟล์ที่ควรถูกสร้าง น้อยกว่าวันนี้ จะทำการแปลง Cron schedule ให้เป็นวันที่ในรอบต่อไป ว่าจะถูกสร้างครั้งถัดไปในวันไหน และบันทึกไปยังฐานข้อมูล เพื่อใช้ในการตรวจสอบต่อไป



ภาพที่ 3.25 แสดงการทำงานของ Endpoint generateReportsMonitoring

(4) HDFS Workflow Scheduler

The screenshot shows the HDFS Workflow Scheduler interface. On the left, there are sections for COORDINATOR and SUBMITTER, both showing '[SCHEDULED][34568][reports-monitoring] scheduler'. Below these are sections for STATUS (RUNNING), PROGRESS (100%), and FREQUENCY (*/10 1-13 * * *). On the right, the main area is titled 'Coordinator [SCHEDULED][34568][reports-monitoring] scheduler' and contains a table of scheduled tasks:

Day	Warning message
113-07 Dec 2023 20:50:00	-
112-07 Dec 2023 20:40:00	-
111-07 Dec 2023 20:30:00	-
110-07 Dec 2023 20:20:00	-
109-07 Dec 2023 20:10:00	-
108-07 Dec 2023 20:00:00	-
107-07 Dec 2023 19:50:00	-
106-07 Dec 2023 19:40:00	-

ภาพที่ 3.25 Scheduler สำหรับเรียก Endpoint เพื่อตรวจสอบไฟล์

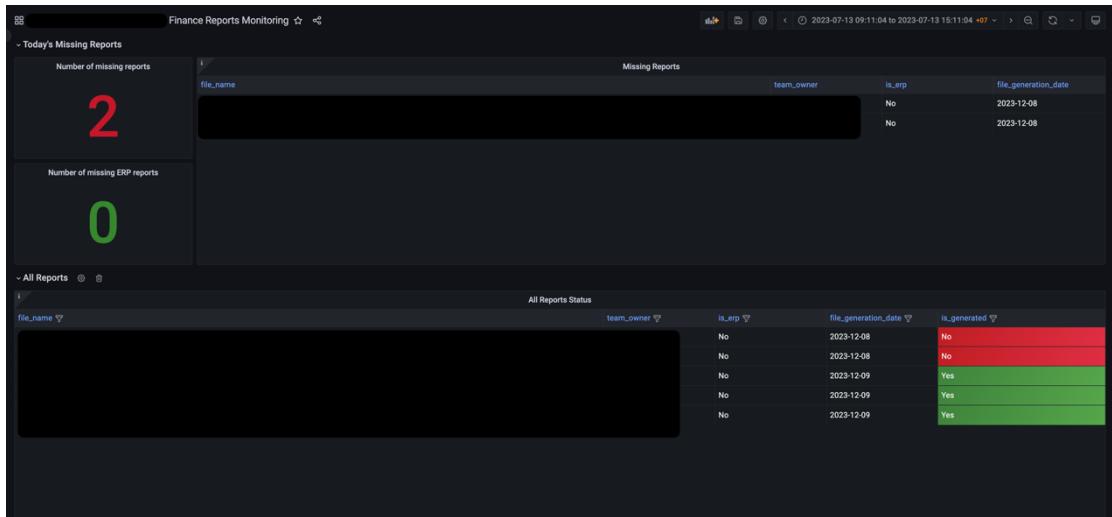
จากราฟ 3.25 เป็นการตั้งค่า HDFS Workflow ให้เรียกใช้ Endpoint ของระบบทุกๆ 10 นาที ตั้งแต่ 08.00 ถึง 20.00 โดยตั้ง Cron schedule ดังนี้ */10 1-13 * * * โดย */10 หมายความว่าทุกๆ 10 นาที ถ้ามาเป็นชั่วโมง โดยตั้งเป็น 1-13 คือเวลาของ UTC เมื่อแปลงเป็นเวลาของประเทศไทย +7 จะได้เวลาที่ต้องการคือ 08.00 ถึง 20.00 พอดี และ * * * ที่เหลือคือ วันที่ เดือน และวันในอาทิตย์ ใส่เป็น * เนื่องจาก จะให้ทำงานในทุกวัน เดือน โดยเมื่อถึงเวลาด้วย Scheduler จะเรียกใช้ Shell Script เพื่อเรียกใช้ Endpoint ของระบบหลัก

```
#!/bin/bash
echo "Call Report generated"
curl -X POST "http://localhost:9000/reportsMonitoring/generateReportMonitoring"
echo "Call Report generated successfully"
```

ภาพที่ 3.26 ตัวอย่าง Shell Script เพื่อเรียกใช้ Endpoint สำหรับตรวจสอบไฟล์

(5) Grafana Dashboard

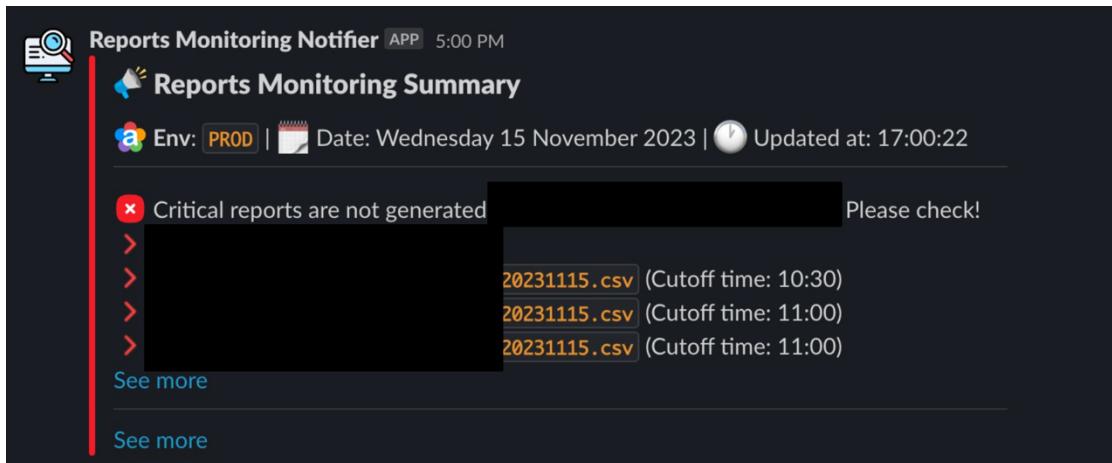
ระบบ Grafana Dashboard จะเชื่อมต่อไปยังฐานข้อมูลที่เกี่ยวข้อง เพื่อแสดงข้อมูลที่จำเป็นผ่าน Dashboard ได้ทันที และยังสามารถสร้างเป็นแผนภูมิ หรือกราฟ ตามที่ผู้ใช้ต้องการใช้งานได้อีกด้วย โดยในระบบนี้ ผู้พัฒนาได้สร้าง Dashboard เพื่อให้ผู้ใช้สามารถตรวจสอบสถานะของไฟล์ทั้งหมดได้ และแสดงข้อมูลไฟล์ที่ไม่ถูกอัปเดตตามกำหนดด้วย



ภาพที่ 3.27 Grafana Dashboard แสดงสถานะของไฟล์รายงานทั้งหมด

(6) Slack bot Notification

ผู้จัดทำโครงการได้พัฒนาระบบแจ้งเตือนไปยัง Slack เมื่อมีไฟล์ที่ไม่สามารถสร้างได้ตามกำหนด หรือมีข้อผิดพลาดของการสร้างไฟล์รายงาน จะถูกแจ้งเตือนไปยัง Slack ของผู้ที่เกี่ยวข้องทันที โดยจะอ่านข้อมูลจากฐานข้อมูล และส่งไปเป็นข้อความผ่าน Slack ทันที



ภาพที่ 3.28 ข้อความการแจ้งเตือนไฟล์ไม่ถูกสร้าง ผ่าน Slack Application

3.2.2.4 การทดสอบระบบ

การทดสอบระบบของระบบการตรวจสอบไฟล์รายงานคัญ จะทดสอบด้วย Unit test และ End to end Test ผ่าน Workflow ที่เป็นการทำงานอัตโนมัติ

(1) Unit test

การทดสอบ Unit test ต้องครอบคลุมโค้ดมากกว่า 80% เป็นอย่างน้อย เพื่อให้มั่นใจว่าทุกๆฟังก์ชันทำงานได้ถูกต้อง โดยจะทดสอบทุกๆฟังก์ชันที่ใช้งาน โดยถ้ามีฟังก์ชันอื่นๆที่เกี่ยวข้อง จะใช้การ Mock เพื่อให้ทดสอบให้เป็นไปตาม Flow ของการทดสอบนั้นๆ โดยใช้การ Mock ด้วย Mockito Framework ใน Scala ซึ่งช่วยให้ผู้พัฒนาระบบสามารถดำเนินการทดสอบในแต่ละยูนิตของระบบได้โดยไม่กระทบกับระบบอื่นๆ

```
val mockInsertRepository = mock[FinanceInsertReportsRepository]
val insertReportsService = new DefaultFinanceInsertReportsService(mockInsertRepository)

val result = insertReportsService.insert(
    "filename",
    "Special/UpcMigration",
    true,
    "Fintech-Service",
    "agoda",
    2,
    "*.*.*.*",
    "01:00:00",
    "source-type",
    file_generation_date = DateTimeFormat.forPattern("yyyy-MM-dd").parseDateTime(LocalDate.now.toString()),
    2
)

verify(mockInsertRepository, times(1)).insertReport(
    "filename",
    "Special/UpcMigration",
    true,
    "Fintech-Service",
    "agoda",
    2,
    "*.*.*.*",
    "01:00:00",
    "source-type",
    file_generation_date = DateTimeFormat.forPattern("yyyy-MM-dd").parseDateTime(LocalDate.now.toString()),
    2
)

result.shouldEqual CommonUpdateStatus(isSuccess = true, msg = "Successfully upsert report to db")
```

ภาพที่ 3.29 ตัวอย่างการทดสอบยูนิต ด้วยการใช้ Mockito ในการทดสอบฟังก์ชันเพิ่มข้อมูล

(2) End to end test

ในการทดสอบ End to end test จะเป็นการทดสอบการเรียกใช้ Endpoint ผ่าน HDFS Workflow ที่เป็นโปรแกรมอัตโนมัติ เพื่อให้มั่นใจว่า สามารถเรียกใช้งานได้อย่างถูกต้อง เนื่องจากระบบนี้ จะตรวจสอบไฟล์ตลอดเวลา จึงต้องถูกเรียกใช้ผ่านโปรแกรมอัตโนมัติตลอดเวลา โดยทดสอบระบบบนเซิฟเวอร์ QA ที่สร้างมาเพื่อใช้ทดสอบระบบโดยเฉพาะ

3.2.3 ระบบตรวจสอบความถูกต้องของธุกรรมทางการเงิน

3.2.3.1 เก็บข้อมูล และ วิเคราะห์ความต้องการของผู้ใช้งาน

เนื่องจากในฝ่ายการเงินขององค์กรพบความผิดปกติของธุกรรมทางการเงินบางส่วน ซึ่งในองค์กรมีธุกรรมทางการเงินจำนวนมากที่ถูกเพิ่มเข้ามาใน 1 วัน ทำให้ฝ่ายการเงินเป็นเรื่องที่ยากจะตรวจสอบข้อมูลของธุกรรมทางการเงินได้ทันเวลา และมีประสิทธิภาพ

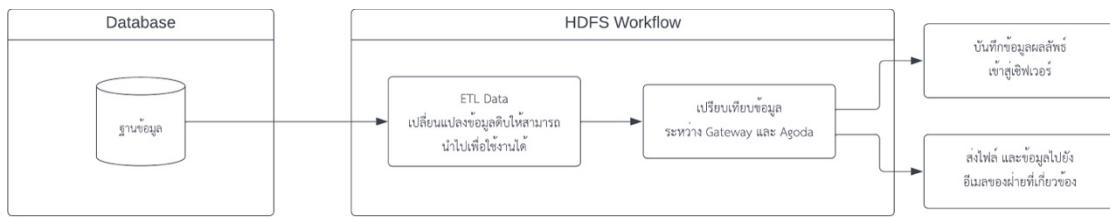
ดังนั้นเพื่อแก้ไขปัญหาในการตรวจสอบธุกรรมทางการเงิน ผู้จัดทำโครงการจึงได้คิด พัฒนาระบบตรวจสอบความถูกต้องของธุกรรมทางการเงิน หรือ Missing booking เพื่อช่วยเหลือ ฝ่ายการเงินในการค้นหาธุกรรมทางการเงินที่มีความผิดปกติ และช่วยหาสาเหตุของข้อผิดพลาด เป็นต้นได้อีกด้วย เพื่อช่วยลดเวลา และลดสโคปของการตรวจสอบธุรกรรมนั้นๆ ของฝ่ายการเงิน นั้นเอง สามารถนำข้อมูลที่ได้ไปตรวจสอบต่อและค้นหาสาเหตุที่แท้จริงต่อไปได้ทันที

3.2.3.2 วิเคราะห์ขอบเขต และ ออกแบบระบบเพื่อนำมาแก้ปัญหา

ผู้จัดทำได้เห็นปัญหาของผู้ใช้ หรือฝ่ายการเงินขององค์กร ว่าไม่สามารถตรวจสอบ ข้อมูลธุกรรมที่ผิดพลาดได้ทันเวลา เนื่องจากมีข้อมูลธุกรรมจำนวนมากถูกเพิ่มเข้ามาในทุกวัน จึง คิดพัฒนาระบบที่ช่วยตรวจสอบธุกรรมที่ผิดปกติ และหาสาเหตุของข้อผิดพลาดของธุรกรรมนั้นๆ ด้วย ว่าเกิดขึ้นจากสาเหตุอะไรได้บ้าง เพื่อช่วยลดสโคปการทำงานของผู้ใช้ และเพื่อให้ผู้ใช้สามารถ นำไปตรวจสอบเพิ่มเติมได้อย่างมีประสิทธิภาพ

โดยการออกแบบระบบ จะดึงข้อมูลธุกรรมทางการเงินจากผู้ให้บริการธุรกรรม ทางการเงิน หรือ Gateway เช่น Visa, Mastercard หรือ ธนาคารต่างๆ เพื่อนำมาเบรี่ยบเทียบกับ ข้อมูลของผู้ใช้ Agoda ที่เก็บบันทึกเอาไว้ตอนทำธุกรรมทางการเงิน โดยในเบื้องต้นในการหาว่า ธุรกรรมไหนมีความผิดปกติ สามารถหาได้โดยถ้าข้อมูลจากผู้ใช้ไม่ตรงกับข้อมูลของ Agoda ไม่ตรงกัน นั่นหมายความว่ามีความผิดปกติเกิดขึ้น

เมื่อพบความผิดปกติของธุกรรมแล้ว จะดำเนินการค้นหาสาเหตุต่อว่าเกิดจากอะไร โดยจะดึงข้อมูลที่เกี่ยวข้องกับธุรกรรมนั้นๆ ทั้งหมดมา โดยเนื่องจากมีข้อมูลจำนวนมากจึงต้องทำ ETL หรือ Extract, Transform and Load ข้อมูลก่อนจะนำมาใช้งาน เพื่อให้มีประสิทธิภาพสูงสุด และทั้ง ETL และการเบรี่ยบเทียบข้อมูลจะถูกทำงานบน HDFS ผ่าน Spark SQL เนื่องจากข้อมูลจำนวนมาก จึงจำเป็นต้องทำงานบน HDFS เพื่อให้มีประสิทธิภาพการทำงานสูงสุด



ภาพที่ 3.30 การทำงานของระบบตรวจสอบความถูกต้องของธุรกรรมทางการเงิน

3.2.3.3 ขั้นตอนการ Implement

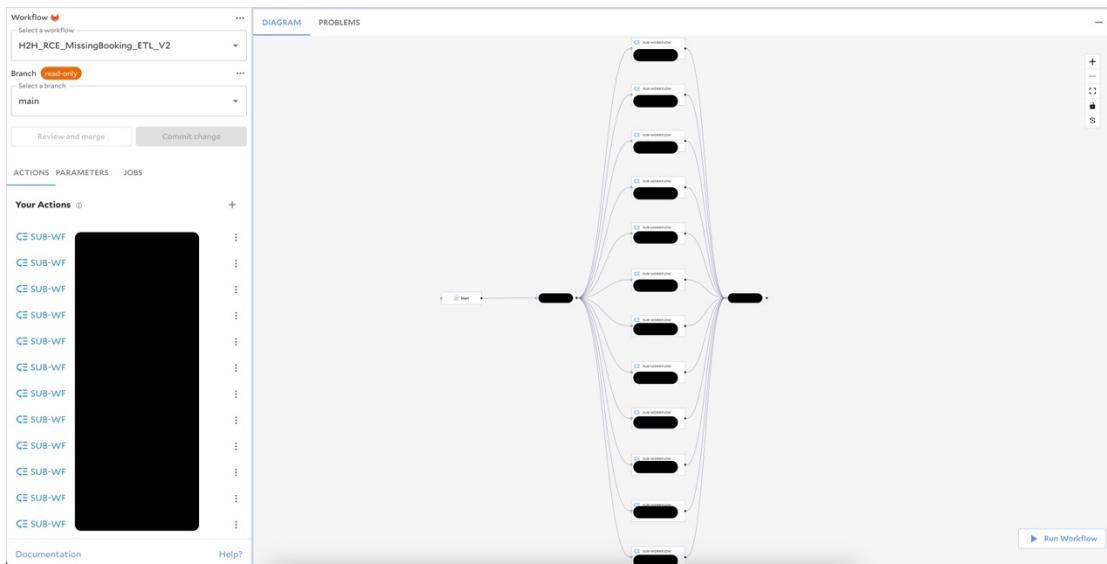
ระบบนี้ถูกออกแบบให้ทำงานอัตโนมัติเพื่อตรวจสอบข้อผิดพลาดของธุรกรรมทางการเงิน โดยจะทำงานในทุกๆเดือน โดยแบ่งเป็น 2 ส่วนคือ ส่วนของระบบหลัก ที่จะใช้ข้อมูลในฐานข้อมูลของ ทั้งฝั่ง Gateway และ Agoda มาเปรียบเทียบกัน เพื่อหาสาเหตุของข้อผิดพลาด และอีกส่วนนึงเป็นส่วนที่จะถูกทำงานก่อน นั่นคือการ ETL ข้อมูลก่อนนำมาใช้จริง

การ ETL เป็นกระบวนการที่จะรวมข้อมูลจากหลายแหล่งเข้าด้วยกันเพื่อนำไปใช้งานต่อไป เป็นการเตรียมข้อมูลก่อนจะนำมายังเคราะห์หาสาเหตุของธุรกรรมที่ผิดปกติ เพื่อช่วยให้ได้ข้อมูลครบถ้วนและถูกต้อง

(1) สร้าง ETL Workflow

การสร้าง ETL Workflow ถูกสร้างขึ้นให้ทำงานบน HDFS เนื่องจากมีข้อมูลจากฐานข้อมูลจำนวนมาก และหลากหลายตารางที่ต้องนำข้อมูลมารวมกัน โดยการทำ ETL เป็นการดึงข้อมูลที่เกี่ยวข้องทั้งหมดจากตารางที่เกี่ยวข้อง มาเก็บไว้ในตารางที่จะนำไปทำการค้นหาข้อผิดพลาดของธุรกรรมทางการเงินต่อไป

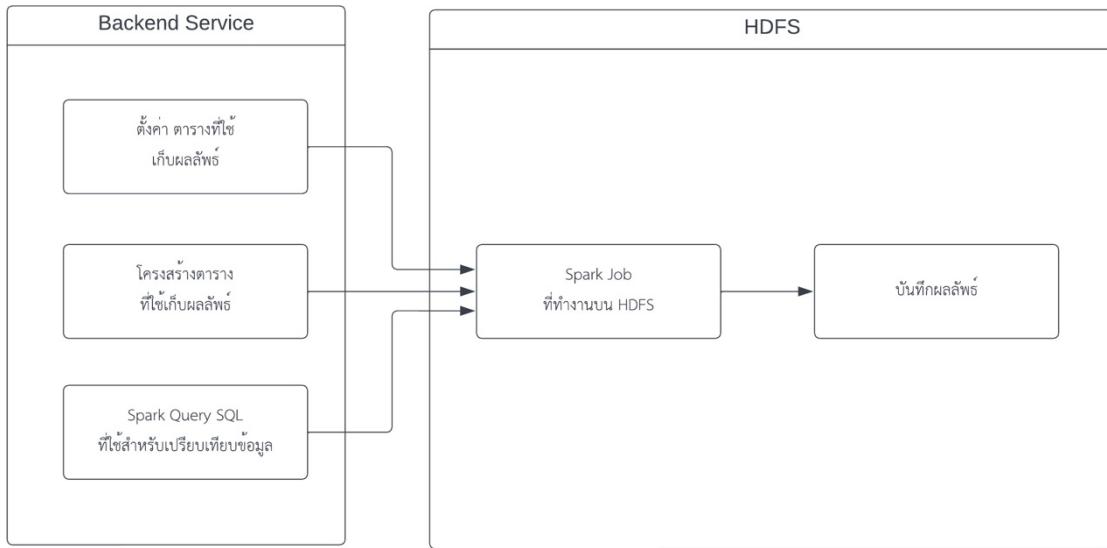
โดยเริ่มจากการตรวจสอบฐานข้อมูลทั้งหมดที่เกี่ยวข้อง ในระบบนี้มีการดึงข้อมูลจากตารางทั้งหมด 12 ตาราง และนำมาเก็บไว้ในตารางที่จะนำไปใช้ในการค้นหาสาเหตุต่อไป



ภาพที่ 3.31 ภาพ Workflow สำหรับการทำ ETL เพื่อรับรวมข้อมูลจากหลายๆ ตาราง

(2) สร้าง Backend Services

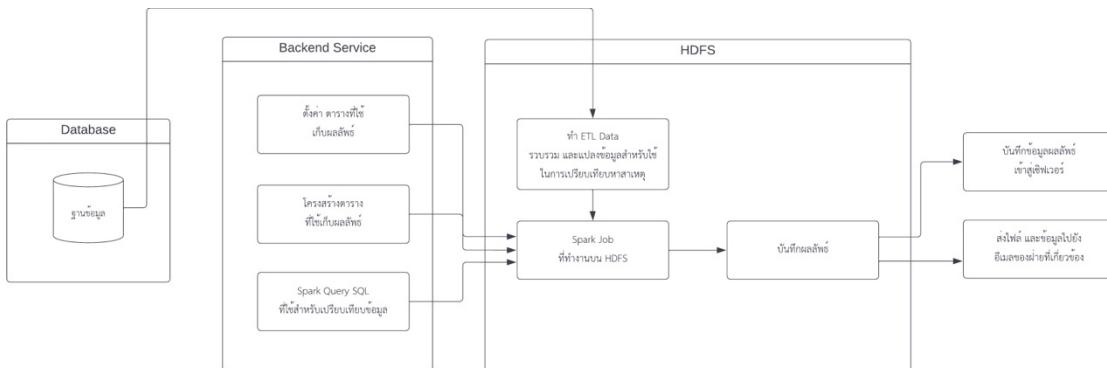
การพัฒนาระบบ เพื่อค้นหาธุรกรรมที่ผิดพลาด และหาสาเหตุ จะถูกพัฒนาด้วย Spark และใช้ Spark SQL ในการรวมข้อมูล และเปรียบเทียบข้อมูล รวมถึงการหาสาเหตุของข้อผิดพลาดของธุรกรรมนั้นๆ อีกด้วย โดยผู้พัฒนาได้เขียน Spark SQL Query จำนวนมาก ในการรวบรวมหาสาเหตุของธุรกรรมที่ผิดพลาด แบ่งเป็นหลายภาษา เนื่องจาก Scala ถูกออกแบบมาเพื่อให้สามารถเขียนภาษา SQL ทั้งหมดที่จำเป็นเอาไว้ใน Scala รวมถึงการตั้งค่าต่างๆ ฐานข้อมูล อีเมลของฝ่ายที่เกี่ยวข้อง ต่างๆ จะถูกตั้งค่าทั้งหมดในภาษา Scala และในระบบที่เขียนด้วย Scala จะส่งการตั้งค่าทั้งหมดไปยัง Spark Job เพื่อให้ทำงานตามที่ตั้งค่าเอาไว้ และเมื่อ Submit Spark Job แล้ว ตัว Spark จะถูกทำงานอยู่บน HDFS เพื่อให้ได้ประสิทธิภาพการทำงานที่ดีที่สุด



ภาพที่ 3.32 แผนภาพการทำงานของ Backend Service

(3) Workflow ของระบบห้องหมุด เมื่อนำมารวมกัน

ใน Workflow ของระบบนี้จะมี 3 ส่วนหลักๆ คือการดึงข้อมูลจากฐานข้อมูลและนำไปทำ ETL จากนั้นนำผลลัพธ์ของ ETL ไปทำงานบน Spark เพื่อค้นหาธุกรรมที่ผิดพลาด และสาเหตุของธุกรรมที่ผิดพลาดว่าเกิดจากอะไร จากนั้นจะบันทึกข้อมูลลงตารางผลลัพธ์ และบันทึกข้อมูลผลลัพธ์เป็นไฟล์ เก็บไปยังเซิฟเวอร์ รวมถึงส่งให้ผู้ที่เกี่ยวข้องผ่านทางอีเมล



ภาพที่ 3.33 แผนภาพการทำงานของระบบตรวจสอบธุกรรมห้องหมุด

2.3.3.4 การทดสอบระบบ

การทดสอบระบบในระบบนี้จะเป็นการทดสอบแบบ End to end Test เท่านั้น เนื่องจากเป็นระบบที่นำข้อมูลมาเปรียบเทียบเพื่อหาสาเหตุ การทดสอบระบบจึงเป็นการทดสอบการ

ทำงานจริง และตรวจสอบผลลัพธ์ที่ได้ โดยจะทดสอบบนเซิฟเวอร์ QA เพื่อให้มีข้อมูลที่มีคุณภาพเข้าสู่ฐานข้อมูลจริง โดยการทดสอบเป็นการส่ง Spark Job ไปทำงานบนเซิฟเวอร์ QA และตรวจสอบผลลัพธ์ที่ได้ ว่าเป็นไปตามที่ควรจะเป็นหรือไม่

บทที่ 4

4.1 แผนการดำเนินงาน

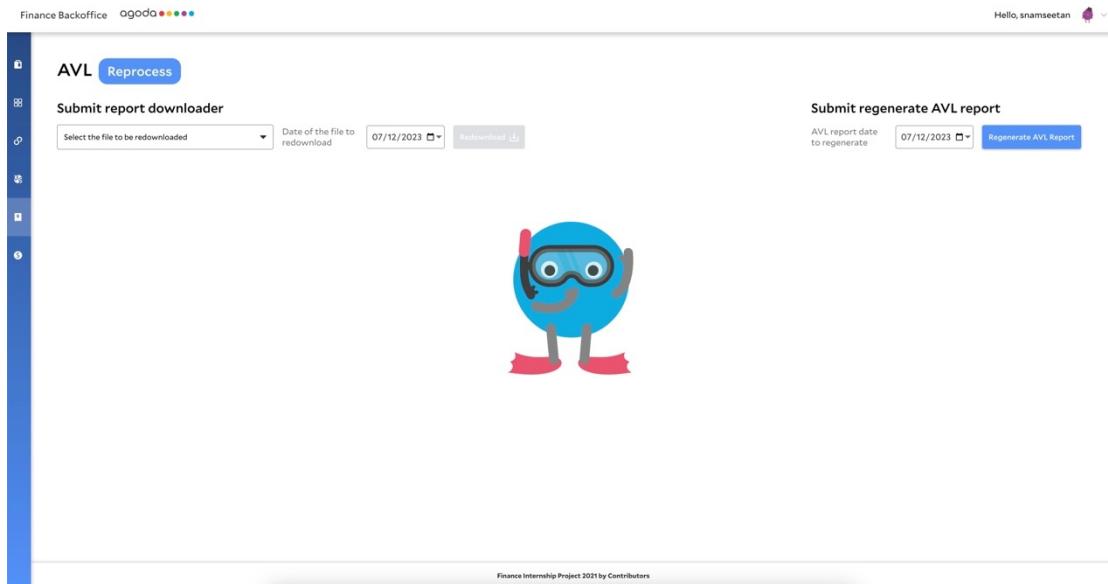
ขั้นตอน	มิถุนายน				กรกฎาคม				สิงหาคม				กันยายน				ตุลาคม				พฤศจิกายน				ธันวาคม				
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	
12. วิเคราะห์ความต้องการและออกแบบระบบตรวจสอบความถูกต้องของธุรกรรมทางการเงิน									1	2	3	4																	
13. พัฒนาระบบตรวจสอบความถูกต้องของธุรกรรมทางการเงิน										1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
14. ทดสอบระบบตรวจสอบความถูกต้องของธุรกรรมทางการเงิน											1	2	3	4															
15. เปิดใช้งานระบบตรวจสอบความถูกต้องของธุรกรรมทางการเงิน											1	2	3	4											1	2	3	4	

4.2 ผลการดำเนินงาน

จากการทำสหกิจศึกษาผู้จัดทำได้พัฒนาระบบทั้งหมด 3 ระบบโดยทุกระบบในตอนนี้ถูกใช้งานบนระบบจริงและทำงานได้อย่างถูกต้องเรียบร้อย โดยแบ่งตามแต่ละระบบได้ดังนี้

4.2.1 ระบบสร้างไฟล์รายงานสำหรับทรัพยากรและหนี้สินใหม่

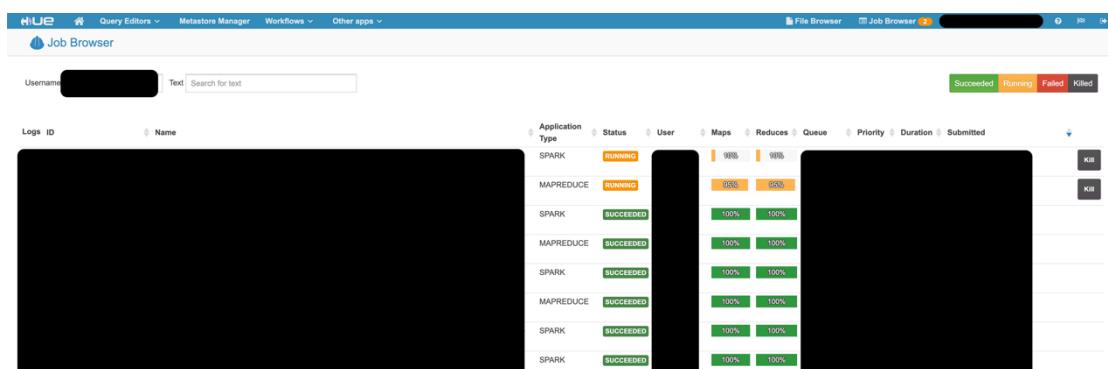
ผลลัพธ์ที่ได้จากการดำเนินงานคือ ผู้จัดทำโครงงานได้พัฒนาระบบที่ต้องสนองแก้ไขปัญหาของผู้ใช้เดียวอย่างมีประสิทธิภาพ ทั้งในส่วนของระบบหลังบ้าน หรือ Backend ที่ทำงานได้เป็นอย่างดี และระบบหน้าบ้าน User Interface หรือ Frontend โดยผู้ใช้สามารถเข้าใจการทำงานได้ง่าย และสามารถใช้งานการสร้างไฟล์รายงานสำหรับทรัพยากรและหนี้สินใหม่ได้ทันที



ภาพที่ 4.1 หน้า Frontend สำหรับผู้ใช้งานสั่งทำงานการสร้างไฟล์รายงานใหม่

จากภาพที่ 4.1 เป็นหน้า UI ที่ถูกเพิ่มเข้าไปใน Finance Backoffice ซึ่งเป็นเว็บที่ให้บริการต่างๆเกี่ยวกับระบบทางการเงิน

โดยผู้พัฒนาได้เพิ่มระบบ Regenerate AVL report เข้าไป เพื่อให้ผู้ใช้งานที่ต้องการสร้างไฟล์รายงานใหม่ เพียงแค่เลือกวันที่ของรายงานที่ต้องการ และกด Regenerate เท่านั้น ระบบจะส่งข้อมูลวันที่ต้องการสร้างไฟล์ใหม่ไปยังระบบหลังบ้าน เพื่อทำการสร้างไฟล์รายงานใหม่ ทันที โดยได้รับผลตอบรับของผู้ใช้งานเป็นอย่างดี ช่วยแก้ปัญหาของผู้ใช้ที่ไม่สามารถสร้างไฟล์รายงานเองใหม่ได้ด้วยตนเอง และตอบโจทย์การสร้างไฟล์รายงานเก่าๆที่อาจจะถูกลบออกไปแล้ว ก็สามารถทำได้



ภาพที่ 4.2 หน้าแสดงสถานะการทำงานสร้างไฟล์ใหม่บน HDFS Workflow

ในส่วนของระบบการทำงานในการสร้างไฟล์รายงานใหม่ สามารถทำงานได้อย่าง ถูกต้อง โดยส่งข้อมูลรายงานที่เกี่ยวข้องทั้งหมดไปยัง Spark และทำงานบน HDFS Workflow เพื่อ ดำเนินการสร้างไฟล์รายงานใหม่ และส่งไปยังเซิฟเวอร์ที่เก็บไฟล์ได้ทันที

โดยจากภาพที่ 4.2 เป็นหน้าเว็บไซต์ที่สามารถตรวจสอบการทำงานของการสร้างไฟล์รายงานใหม่ได้ทันที และยังสามารถดูข้อมูลการทำงานของระบบได้ทั้งหมด หากเกินข้อผิดพลาด เกิดขึ้น ผู้พัฒนาสามารถเข้ามาตรวจสอบและหาสาเหตุได้ผ่านย Hue UI ได้ทันที ซึ่งช่วยให้ผู้พัฒนา คนต่อๆไป หรือผู้ที่จะนำไปพัฒนาต่อ สามารถตรวจสอบและแก้ไขได้อย่างมีประสิทธิภาพ

ในส่วนของการทดสอบระบบ ก่อนที่จะใช้งานระบบ มีการทดสอบ Unittest ใน ทุกๆโปรเจคที่ถูกพัฒนาเสมอ โดยกำหนดให้ครอบคลุมได้ทั้งหมดไม่ต่ำกว่า 80% ของระบบ โดยการ ทดสอบของระบบนี้ผู้จัดทำโครงงานได้ทดสอบระบบไปทั้งหมด 95% โดยทดสอบในทุกๆส่วน และ ทุกๆฟังก์ชันที่มีอยู่ภายในระบบทั้งหมด โดยมีการใช้ Docker ในจำลองฐานข้อมูลบน Local เพื่อ ทดสอบด้วย ส่วนของการทดสอบระบบที่ยากที่สุดคือการทดสอบสร้างไฟล์รายงาน เนื่องจากไฟล์จะ ถูกสร้างบน Spark ซึ่งต้องทดสอบด้วยการทำงานจริงเท่านั้น ทำให้ต้องทดสอบหลายครั้งจนได้ ผลลัพธ์ที่ถูกต้องตามต้องการ

4.2.2 ระบบตรวจสอบและแสดงผลการสร้างรายงานสำคัญทางการเงิน

ผลลัพธ์ที่ได้จากการพัฒนาระบบตรวจสอบและแสดงผลการสร้างรายงานสำคัญ ทางการเงิน แก้ปัญหาของผู้ใช้งาน ให้สามารถช่วยตรวจสอบไฟล์ และแจ้งเตือนได้อย่างมี ประสิทธิภาพ โดยผู้ใช้งานระบบการตรวจสอบและแสดงผลการสร้างรายงานสำคัญทางการเงิน สามารถตรวจสอบไฟล์ทั้งหมดได้ใน Grafana Dashboard ได้ทันที และยังมีการแจ้งเตือนไปยัง Slack ของทีมที่เกี่ยวข้องอีกด้วย

และในส่วนของการเพิ่มข้อมูลเข้าสู่ระบบ เพื่อให้ระบบตรวจสอบ โดยใช้ผ่าน Swagger Doc UI นัnek เพียงพอสำหรับผู้ใช้งาน เนื่องจากเป็นระบบที่เรียบง่าย และทำงานได้ทันที โดยผู้ใช้เพิ่มข้อมูลของไฟล์รายงานเพียงครั้งเดียว ระบบก็จะตรวจสอบการสร้างรายงานไปตลอดใน ทุกๆครั้งที่มีการสร้างไฟล์ใหม่เกิดขึ้น

The screenshot shows the Swagger UI interface for a POST request to the endpoint `/reportsMonitoring/upsertReport`. The request is described as "Upsert Reports to db". The "Parameters" section includes a required parameter `body` which is a JSON object representing a report. The JSON structure for `body` is as follows:

```
{
  "file_name": "string",
  "file_location": "string",
  "is_erp": true,
  "team_owner": "string",
  "user_id": "string",
  "offset": 0,
  "cron_schedule": "* * * * *",
  "cutoff_time": "01:00:00",
  "source_type": "ftp",
  "rec_status": 1
}
```

The "Parameter content type" is set to `application/json`.

The "Responses" section indicates a response content type of `application/json`.

ภาพที่ 4.3 หน้า UI Swagger Doc สำหรับให้ผู้ใช้เพิ่มข้อมูลของไฟล์รายงานที่ต้องการตรวจสอบ

จากภาพที่ 4.3 ผู้ใช้สามารถเข้ามาที่หน้า UI และกรอกข้อมูลทั้งหมดที่จำเป็นในการเพิ่มลงฐานข้อมูลของระบบ และสามารถเพิ่มเข้าสู่ระบบตรวจสอบได้ทันที และยังสามารถแก้ไขข้อมูลได้อีกด้วย ด้วยการใส่ชื่อไฟล์เดิม แต่เปลี่ยนข้อมูลอื่นๆ เข้าไป ระบบจะทำการอัปเดทข้อมูลรายละเอียดอื่นๆ ของไฟล์นั้นๆ แทนการเพิ่มเข้าไปใหม่

team_owner	is_erp	file_generation_date	is_generated
No	No	2023-12-08	No
No	No	2023-12-08	No
No	No	2023-12-09	Yes
No	No	2023-12-09	Yes
No	No	2023-12-09	Yes

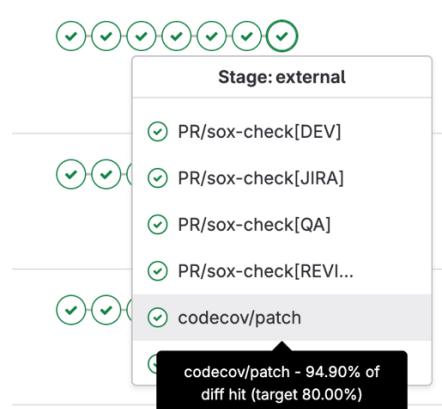
ภาพที่ 4.4 หน้า Grafana Dashboard แสดงสถานะของไฟล์ทั้งหมดที่ถูกตรวจสอบ

ในส่วนของการแสดงผลลัพธ์ผ่าน Grafana Dashboard ช่วยแก้ปัญหาของผู้ใช้งานได้เป็นอย่างดี เนื่องจากผู้ใช้สามารถเข้ามาดูไฟล์ทั้งหมด หรือไฟล์ที่ต้องการก็สามารถเลือกได้ ผ่านทาง Grafana Dashboard ได้ทันที ซึ่งเป็นการอัปเดทข้อมูลแบบ Realtime ตลอดเวลา ซึ่งช่วยให้ผู้ใช้งานสามารถตรวจสอบไฟล์ได้ทันที และมีประสิทธิภาพสูงสุด



ภาพที่ 4.5 ข้อความแจ้งเตือนกรณีไฟล์ไม่ถูกสร้างอย่างถูกต้อง ใน Slack Application

ในส่วนของระบบการแจ้งเตือนไปยัง Slack Application สามารถทำงานได้อย่างถูกต้องและมีประสิทธิภาพ ซึ่งช่วยแก้ปัญหาของผู้ใช้งานได้ดีมาก เพราะผู้ใช้ไม่จำเป็นต้องคอยตรวจสอบด้วยตนเองตลอดเวลา จึงแก้ปัญหาส่วนนี้ได้ดี เมื่อมีไฟล์ที่ไม่ถูกสร้างตามกำหนด ก็จะแจ้งเตือนไปยัง Slack ของผู้ที่เกี่ยวข้องได้ทันที ช่วยแก้ปัญหาของผู้ใช้งานได้เป็นอย่างดี



ภาพที่ 4.6 ผลการรัน Unit test ของระบบตรวจสอบและแสดงผลการสร้างรายงาน

ในการทดสอบระบบตรวจสอบและแสดงผลการสร้างรายงานสำคัญทางการเงิน มีการทดสอบ Unit test ในทุกๆ ฟังก์ชัน โดยมีการกำหนดการทดสอบขั้นต่ำไม่น้อยกว่า 80% โดยระบบนี้ผู้จัดทำได้เขียนการทดสอบ Unit test ไปทั้งหมด 94.90% ดังภาพที่ 4.6 โดยมีการทดสอบในทุกๆ ส่วนของระบบอย่างครบถ้วน และใช้ Docker ในการจัดองข้อมูลในการทดสอบข้อมูลด้วย

นอกจากนี้ยังมีการทดสอบแบบ End to End test ที่เป็นการทดสอบการทำงานจริงบนเซิฟเวอร์ QA หรือเซิฟเวอร์ที่ใช้สำหรับทดสอบ ซึ่งระบบสามารถทำงานได้ตามที่ผู้พัฒนาต้องการ

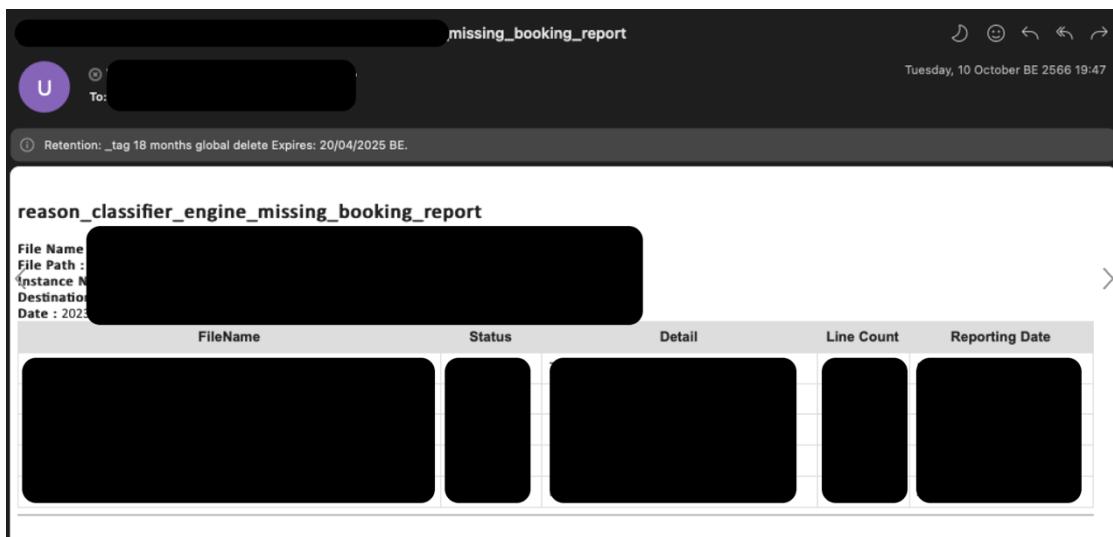
ในส่วนการทดสอบนี้ จะต้องมีการตรวจสอบไฟล์จริงๆบนเซิฟเวอร์ด้วย โดยผู้พัฒนาได้สร้างไฟล์จำลองเพื่อใช้ในการทดสอบระบบนี้

4.2.3 ระบบตรวจสอบความถูกต้องของธุกรรมทางการเงิน

ผลลัพธ์ที่ได้จากการพัฒนาระบบตรวจสอบความถูกต้องของธุกรรมทางการเงินนี้สามารถช่วยแก้ปัญหาของผู้ใช้งานได้เป็นอย่างดี และประหยัดเวลาในการเปรียบเทียบข้อมูลเป็นจำนวนมาก และผลลัพธ์ที่ได้จากการทำงานของระบบมีความถูกต้องตามความต้องการของผู้ใช้เป็นอย่างดี โดยระบบจะถูกทำงานในทุกๆเดือน เดือนละครึ่งตามกำหนดเวลาที่ตั้งไว้ใน HDFS Workflow และเมื่อทำงานเสร็จ จะส่งแจ้งเตือน และไฟล์ผลลัพธ์ไปยังฝ่ายที่เกี่ยวข้องผ่าน Email ซึ่งช่วยแก้ปัญหาทั้งหมดของผู้ใช้งานได้อย่างมีประสิทธิภาพ



ภาพที่ 4.7 Email ส่งไฟล์ผลลัพธ์ไปยังผู้ใช้งานระบบ

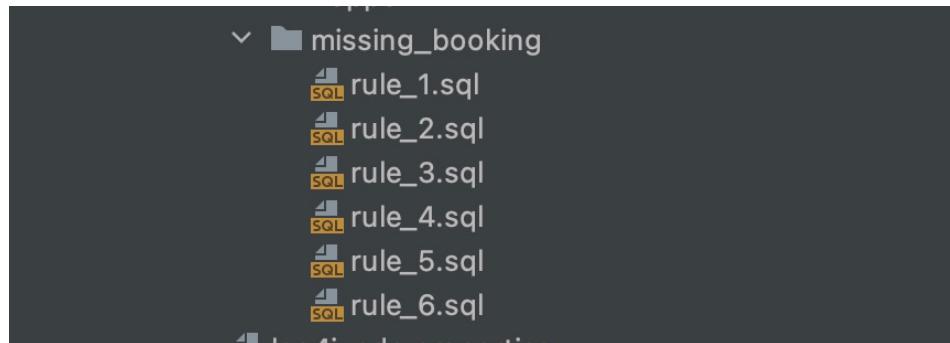


ภาพที่ 4.8 Email ที่แจ้งเตือน และรายงานเมื่อระบบทำงานเสร็จล้วน

จากการที่ 4.7 และ 4.8 เมื่อระบบทำงานตามกำหนดเวลาที่ตั้งไว้เสร็จสิ้น ระบบจะส่งข้อมูลการทำงาน และไฟล์รายงานผลลัพธ์ของการทำงานไปให้ผู้ใช้งานทันที โดยผู้ใช้สามารถนำข้อมูลผลลัพธ์ไปหาสาเหตุเพิ่มเติม หรือแก้ปัญหาต่อไปได้อย่างมีประสิทธิภาพ ผลลัพธ์จากระบบนี้ช่วยให้ฝ่ายการเงินประหยัดเวลา และ ทำงานในการตรวจสอบความผิดปกติของธุรกรรมทางการเงินได้อย่างมีประสิทธิภาพ

การทดสอบระบบนี้จะทดสอบเฉพาะส่วนของ End to end test เนื่องจาก การทำงานหลักๆ ของระบบนี้จะเป็นการดึงข้อมูลต่างๆ เข้ามาเปรียบเทียบกันด้วย Spark SQL การทดสอบนี้จะถูกทดสอบบนเซิฟเวอร์ QA และข้อมูลจำลอง เนื่องจากการทดสอบบนเครื่อง-ton เอง เป็นไปได้ยาก เพราะมีข้อมูลจำนวนมากที่ต้องทดสอบ และตารางจำนวนมากที่จะนำมาใช้งานเพื่อหาสาเหตุต่างๆ ของธุรกรรมที่ผิดปกติ โดยจะทดสอบโดยการสั่งให้ระบบทำงานบน HDFS ในเซิฟเวอร์ที่ใช้สำหรับทดสอบโดยเฉพาะ หรือ QA นั้นเอง และตรวจสอบผลลัพธ์ว่าถูกต้องตามความต้องการของผู้ใช้หรือไม่

ความยากในการทดสอบนี้คือ เนื่องจากมีการดึงข้อมูลมาเป็นจำนวนมาก และจากหลายตาราง จึงทำให้ทดสอบได้ยาก ทางผู้พัฒนาจึงได้แยกการรวมข้อมูลเป็นหลายส่วน และแบ่งทดสอบ Spark SQL ในการรวม และคนหาที่ลับส่วนไป เพื่อให้สะดวกและง่ายต่อการทดสอบ และการพัฒนาระบบด้วย



ภาพที่ 4.9 การแบ่ง Spark SQL Query เพื่อพัฒนาระบบ

จากการที่ 4.9 ผู้พัฒนาได้แบ่งคำสั่ง SQL ออกเป็น 6 ส่วน โดยสามารถทดสอบในที่ลับส่วนได้ ซึ่งเมื่อทดสอบทั้งหมดแล้ว การทำงานของระบบจะเริ่มทำงานตั้งแต่ส่วนที่ 1 จนถึง 6 ตามลำดับ

4.3 ข้อเสนอแนะ/ปรับปรุงในอนาคต

ในการไปปฏิบัติศึกษาครั้งนี้สิ่งที่ผู้จัดทำมีข้อที่ควรปรับปรุงมากที่สุดคือการใช้ภาษาใน การสื่อสาร เนื่องจากผู้จัดทำโครงการได้ไปปฏิบัติศึกษาที่บริษัท Agoda ซึ่งเป็นบริษัท

ต่างประเทศที่มีพนักงานชาวต่างชาตินากกว่า 50 ชาติภายในองค์กร ซึ่งทำให้ภาษาหลักที่ต้องใช้คือภาษาอังกฤษ สำหรับสื่อสารข้อมูลทั้งหมดในการทำงาน ซึ่งคิดว่าผู้จัดทำยังต้องศึกษาการใช้ภาษาเพิ่มเติมเพื่อใช้สำหรับทางลือ หรือสื่อสารได้มากยิ่งขึ้น

แต่ในส่วนของการช่วยเหลือของทีม หรือทีมอื่นๆ บริษัทมีประณีตที่ดีมากทุกคนสามารถถ่ายทอดได้ไม่แบบตัวแทนง ซึ่งผู้จัดทำโครงงานนี้ได้พูดคุย และปรึกษากับเพื่อนในหลายๆ ทีมภายในองค์กรจำนวนมาก ซึ่งช่วยให้การปฏิบัติศึกษาครั้งนี้ผ่านไปได้ด้วยดีจากความช่วยเหลือจากทุกๆ คนในองค์กรไม่ใช่เพียงแค่ภายในทีมเท่านั้น

ในส่วนของเทคนิคที่ใช้ในการทำงานส่วนมากเป็นการใช้เทคนิคขั้นสูง หรือมีประสิทธิภาพอยู่แล้ว ทางผู้จัดทำคิดว่าเหมาะสมสมกับการทำงานที่ต้องการความปลอดภัยสูง และมีประสิทธิภาพอยู่แล้ว แต่เนื่องจากมีหลายส่วนที่ไม่เคยศึกษามาก่อน ทำให้ต้องใช้เวลาในการศึกษาอุปกรณ์เครื่องมือต่างๆ และอยากรับรู้ในการเขียนโค้ดให้มีประสิทธิภาพมากยิ่งขึ้น เนื่องจากยังมีบางส่วนที่ทำให้ทำงานได้เร็วมากกว่าเดิม แต่เนื่องจากเวลาการทำงานที่จำกัด จึงพัฒนาให้ดีที่สุดในตอนนี้ก่อน

การทดสอบระบบของทุกๆ ระบบ มีมาตรฐานขั้นต่ำเสมอ ซึ่งเป็นสิ่งที่ดีในการพัฒนาระบบ เพื่อให้มั่นใจได้ว่าระบบสามารถทำงานได้อย่างถูกต้องตามที่ผู้พัฒนาต้องการ และมีประสิทธิภาพ

บทที่ 5

สรุปผลการปฏิบัติสหกิจ

ตลอดการทำงานใน Agoda ผู้จัดทำได้รับมอบหมายให้ทำงานต่างๆ ที่แตกต่างกันในหลายๆ ส่วน และมีงานที่ต้องใช้ระบบที่ไม่คุ้นเคย มีการเรียนเพิ่มเติมตลอดการทำงาน และการสอบถามกับทีมต่างๆ ในการช่วยเหลือการพัฒนาระบบ ซึ่งหลายทีมในองค์กรได้เข้ามาช่วยในการออกแบบระบบ หรือการแก้ไขระบบที่ลูกพัฒนาขึ้น ซึ่งเป็นวัฒนธรรมขององค์กรที่ดีมาก เราสามารถพูดคุยกับทุกๆ คน ภายในองค์กรได้อย่างอิสระ โดยไม่ว่าจะเป็นตำแหน่งสูงหรือต่ำกว่า และมีการใช้ระบบ Scrum ใน การปฏิบัติงาน แบ่งการทำงานเป็น Sprint โดยจะแบ่งเป็นทุกๆ 2 อาทิตย์ มีการทำ Standup Meeting ในทุกๆ เช้าเพื่อให้ทุกคนมาอัปเดตงานที่ทำอยู่ และหารือในเรื่องต่างๆ ซึ่งทำให้ผู้ปฏิบัติสหกิจได้พัฒนาทักษะหลากหลายด้านทั้ง Soft Skill และ Hard Skill มีการเรียนรู้การแก้ปัญหาต่างๆ จากพี่ๆ ในองค์กรเสมอ

และในส่วนของการพัฒนาระบบทั้ง 3 ระบบ ได้มีการปรึกษา หารือ และพูดคุยกับผู้ใช้ เพื่อแก้ไขปัญหาเสมอ รวมถึงหลังจากที่พัฒนาระบบเสร็จแล้ว ก็มีผลตอบลัพธ์จากผู้ใช้งานจริงเสมอ ทำให้เราสามารถต่อยอด หรือพัฒนาระบบนั้นๆ ให้มีความสามารถ และทำงานได้มีประสิทธิภาพได้มากยิ่งขึ้น

การใช้เครื่องมือ หรือภาษาในการพัฒนาระบบทั้งหมด ผู้ปฏิบัติสหกิจศึกษาไม่เคยทำงานเกี่ยวกับเครื่องมือที่เกี่ยวข้องมาก่อนเลย หรือแม้แต่ภาษา Scala และภาษา Spark ที่ใช้พัฒนาระบบก็ไม่เคยทำงานมาก่อน ในช่วงแรกที่เข้ามา หรือได้รับงานใหม่ๆ มา ก็สามารถมีเวลาเพื่อเรียนรู้การใช้เครื่องมือต่างๆ หรือภาษาต่างๆ ได้ก่อน เพื่อสามารถนำไปพัฒนาระบบได้อย่างมีประสิทธิภาพ และภายในองค์กร มีหลักสูตรให้เรียนมากมาย ทั้งการใช้งานระบบต่างๆ เขียนภาษาต่างๆ หรือแม้กระทั่งการพัฒนา Soft Skill ต่างๆ อีกด้วย ซึ่งพนักงานทุกคนสามารถสมัครเข้าไปเรียนคอสออนไลน์ได้ ตลอดเรื่องที่สนใจได้ฟรีตลอดเวลา

สิ่งที่ผู้ปฏิบัติสหกิจศึกษาได้รับครั้นนี้มีหลากหลายส่วน และความรู้เยอะมาก ไม่ว่าจะเป็นการทำงานรูปแบบ Scrum ที่ใช้ในการทำงานจริงๆ การติดต่อผู้ใช้งานเพื่อขอความคิดเห็น การหารือในการออกแบบระบบในแต่ละทีมที่เกี่ยวข้องกับระบบ รวมไปถึงการใช้เครื่องมือใหม่ๆ ที่ไม่เคยเรียนในห้องเรียนมาก่อน การใช้ SQL ให้มีประสิทธิภาพสูง และประหยัดเวลา การออกแบบระบบให้รองรับกับข้อมูลจำนวนมาก รวมไปถึงพัฒนาการทำงานร่วมกับคนอื่นเยอะมาก เนื่องจากต้องพูดคุยกับหลายๆ ทีมตลอดเวลาที่พัฒนาระบบ ทั้งหมดนี้เป็นประสบการณ์การทำงานที่ดีมาก และเป็นประโยชน์ กับผู้ปฏิบัติสหกิจเป็นอย่างมาก รวมถึงได้รับความรู้ใหม่ๆ การแก้ปัญหาต่างๆ อีกด้วย

บรรณานุกรม

(รายการอ้างอิง)

ภาคผนวก

ภาคผนวก ก. ชื่อภาคผนวก

ภาคผนวก ข. ชื่อภาคผนวก

เริ่มพิมพ์เนื้อหา

ภาคผนวก ค. ชื่อภาคผนวก

เริ่มพิมพ์เนื้อหา

