

CP Capstone Project Proposal
ข้อเสนอโครงการรบบยอวิศวกรรมคอมพิวเตอร์ 1

เรื่อง
แพลตฟอร์มเปิดอินเทอร์เน็ตของสรรพสิ่งสำหรับระบบสุขภาพของประเทศไทยพร้อมระบบ
ปัญญาประดิษฐ์
Open Healthcare IoT Platform with AI

นายณัฐภัทร	จารุชัยสิทธิกุล	รหัสนิสิต 6230177821
นางสาวณิชากร	ชัยพจน	รหัสนิสิต 6231322621
นางสาวมารีนญา	ตะโจปะรัง	รหัสนิสิต 6231352421
นางสาวศิวกาญจน์	จิตต์วโรดม	รหัสนิสิต 6231363321

อาจารย์ที่ปรึกษา
รศ.ดร. กุสธิดา โรจน์วิบูลย์ชัย

รายงานนี้เป็นส่วนหนึ่งของวิชา 2110488 โครงการรบบยอวิศวกรรมคอมพิวเตอร์
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ประจำปีการศึกษา 2563

บทคัดย่อ

ในปัจจุบันเทคโนโลยีถูกพัฒนาอย่างต่อเนื่องอย่างรวดเร็ว ทุกหน่วยงานมีการปรับใช้เทคโนโลยีเข้ากับระบบการทำงานของตัวเอง หน่วยงานทางการแพทย์ก็เช่นกัน มีการใช้อุปกรณ์ IoT ในการตรวจวัดข้อมูลสุขภาพของผู้ป่วย เช่น เซนเซอร์วัดความดัน เซนเซอร์วัดอัตราการเต้นของหัวใจ เซนเซอร์วัดปริมาณออกซิเจนในเลือด เป็นต้น แต่ยังมีขาดระบบกลางในการจัดเก็บและจัดการข้อมูลเหล่านั้น อีกทั้งอุปกรณ์แต่ละประเภทยังมีรูปแบบการส่งข้อมูลที่ต่างกัน ทำให้ต้องใช้ซอฟต์แวร์เฉพาะของแต่ละอุปกรณ์ในการจัดการข้อมูล ซึ่งมีค่าใช้จ่ายค่อนข้างสูง นอกจากนี้ยังส่งผลให้ผู้ใช้งานมีประสบการณ์การใช้งานที่ไม่น่าพอใจเท่าที่ควรอีกด้วย โครงการนี้นำเสนอการออกแบบพัฒนาแพลตฟอร์มสำหรับข้อมูลด้านสุขภาพจากอุปกรณ์ IoT เพื่อเป็นศูนย์กลางข้อมูลด้านสุขภาพ ลดค่าใช้จ่ายในการพัฒนาและดูแลระบบ IoT รวมทั้งอำนวยความสะดวกให้กับบุคลากรทางการแพทย์และผู้ป่วย โดยระบบแรก คือ ระบบการส่งข้อมูลจากอุปกรณ์เข้าสู่แพลตฟอร์มเพื่อเป็นสร้างศูนย์กลางที่รวมข้อมูลด้านสุขภาพที่รองรับอุปกรณ์ทุกประเภท ระบบที่สองคือการจัดเก็บข้อมูลในแพลตฟอร์มเพื่อจัดเก็บข้อมูลเป็นสัดส่วนเพื่อส่งข้อมูลย้อนหลังหรือข้อมูล real-time ให้กับผู้ใช้งานและนำข้อมูลไปวิเคราะห์เพื่อใช้ประโยชน์ต่อไป ระบบที่สามคือระบบติดต่อผู้ใช้งาน เพื่อตอบสนองต่อผู้ใช้งานในแต่ละประเภท และระบบที่สี่คือระบบแอดมินเพื่อดูแลจัดการอุปกรณ์และข้อมูลในระบบ โดยปัจจุบันได้ดำเนินการสร้างระบบที่รองรับการนำข้อมูลเข้าโดยผ่าน MQTT protocol Kafka server เพื่อบันทึกข้อมูลไปยัง database ระบบส่งข้อมูลย้อนหลังและข้อมูลแบบ real-time ให้แก่ผู้ใช้งาน และระบบแอดมินเพื่อจัดการกับอุปกรณ์การแพทย์และสิทธิของผู้ใช้งานในการเข้าถึงข้อมูลของอุปกรณ์แล้วเสร็จเรียบร้อยแล้ว

Abstract

Nowadays, technology is constantly evolving rapidly. Every department should adapt the technology to their own system. The medical departments do as well. IoT devices are used to measure health data such as pressure sensors, heart rate sensor, Blood Oxygen Sensor, etc. At present, it still lacks a central system to store and manage those data. Moreover, each type of device has a different data transmission protocol, so they require a specific software for each device to control. The software has a relatively high cost, and it also results in an unsatisfying user experience. This project proposes the design and development of a platform for IoT healthcare as a health information hub. It reduces the cost of developing and administering IoT systems, as well as facilitating both doctors and patients. The project divides the entire system into 4 subsystems, those are a data receiving system, a data management system, a user interaction system, and an admin system to manage devices and data in the system. Now, the data receiving system by using MQTT protocol Kafka server, sending data to users, and the admin system have already implemented.

สารบัญ

1. บทนำ.....	1
1.1. ที่มาและความสำคัญ.....	1
1.2. วัตถุประสงค์ของโครงการ.....	1
1.3. ขอบเขตของโครงการและผู้รับผิดชอบในแต่ละส่วน.....	2
1.4. ประโยชน์ที่คาดว่าจะได้รับ.....	5
2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง.....	6
2.1. ทฤษฎีที่เกี่ยวข้อง.....	6
3. วิธีการที่นำเสนอ/แนวทางที่คิดว่าจะทำ.....	8
3.1. สถาปัตยกรรมของระบบ.....	8
3.2. ระบบภายใน.....	8
4. ผลการวิเคราะห์.....	13
4.1. ภาษาที่เลือกใช้.....	13
4.2. ระบบฐานข้อมูลที่เลือกใช้.....	13
4.3. Library ในการให้ Web Service ติดต่อกับ Kafka server.....	14
4.4. เครื่องมือในการจัดการ Logs ของ web service.....	14
4.5. การทำ Data Lake.....	14
4.6. การทำ Data Warehouse.....	14
4.7. การทำ Dashboard.....	15
4.8. การเชื่อมต่อ MQTT กับ Kafka.....	15
5. ผลกระทบทางสังคมของโครงการ.....	16
6. บทสรุปและงานที่จะทำต่อไป.....	16
7. เอกสารอ้างอิง.....	17

สารบัญรูปภาพ

รูปที่ 1 Gantt chart.....	4
รูปที่ 2 สถาปัตยกรรมของระบบ.....	8
รูปที่ 3 การส่งข้อมูลของอุปกรณ์ IoT.....	9
รูปที่ 4 ระบบการจัดการข้อมูล	10
รูปที่ 5 การส่งข้อมูลจาก GCS ไป BigQuery.....	11
รูปที่ 6 ระบบการติดต่อกับผู้ใช้งานและระบบแอดมิน	13
รูปที่ 7 GUI ของ BigQuery	15
รูปที่ 8 งานที่ทำแล้วเสร็จในปัจจุบัน.....	16
รูปที่ 9 งานที่จะทำต่อในอนาคต.....	17

สารบัญตาราง

ตารางที่ 1 แสดงผู้รับผิดชอบงานในแต่ละส่วน.....	3
--	---

1. บทนำ

1.1. ที่มาและความสำคัญ

Internet of Things (IoT) หรือ "อินเทอร์เน็ตในทุกสิ่ง" คือ การที่อุปกรณ์หรือสิ่งต่าง ๆ ได้ถูกเชื่อมต่อกับอินเทอร์เน็ต ทำให้มนุษย์สามารถรับข้อมูลและสั่งการควบคุมการใช้งานอุปกรณ์ต่างๆ ผ่านทางเครือข่ายอินเทอร์เน็ต เช่น การรับข้อมูลจากอุปกรณ์การแพทย์ของผู้ป่วยติดเตียง และส่งสัญญาณไปยังอุปกรณ์ผู้ดูแล

เทคโนโลยีอินเทอร์เน็ตประสานสรรพสิ่งจำนวนของอุปกรณ์ที่เชื่อมต่อระบบพื้นฐานสำหรับอินเทอร์เน็ตประสานสรรพสิ่ง (IoT platform) จะมีกว่าหลายพันล้านอุปกรณ์ อย่างไรก็ตามในปัจจุบัน การทำแพลตฟอร์ม IoT เพื่อรองรับอุปกรณ์การแพทย์นั้นมีค่าใช้จ่ายที่สูง โดยเซิร์ฟเวอร์ขนาดเล็กมีราคาเริ่มที่ 10,000 บาท [1] ต่อเดือนซึ่งโรงพยาบาลในประเทศไทยมีมากกว่า 1,356 โรงพยาบาล [2] ทำให้มีค่าใช้จ่ายต่อเดือนประมาณ 13,560,000 บาท นอกจากนี้การใช้งานอาจจะต้องใช้งานกับซอฟต์แวร์เฉพาะจากผู้ผลิต ซึ่งมีค่าใช้จ่ายเพิ่ม และอาจสร้างประสบการณ์การใช้งานที่ไม่ดีแก่ผู้ใช้ อีกทั้งยังไม่มีกระบวนการรวบรวมข้อมูลที่ดีเพื่อนำเอาข้อมูลไปใช้ประโยชน์ เช่น การสร้างโมเดลปัญญาประดิษฐ์

จากปัญหาดังกล่าวข้างต้นจึงควรเตรียมระบบ IoT Platform ไว้ เพื่อช่วยโรงพยาบาลในประเทศไทยในการลดค่าใช้จ่ายและความเสี่ยงจากการพึ่งพาซอฟต์แวร์ดังกล่าวของต่างประเทศ และเพิ่มทางเลือกในประเทศให้เข้าถึงระบบ IoT Platform ได้มากขึ้น เพิ่มประสิทธิภาพการใช้ทรัพยากร โทรคมนาคม ในประเทศ และขยายโอกาสสร้างนวัตกรรมดิจิทัลได้มากขึ้น

1.2. วัตถุประสงค์ของโครงการ

- 1.2.1. เพื่อติดตามอาการของผู้ป่วยในห้องฉุกเฉิน ห้อง ICU และรพพยาบาลได้แบบอัตโนมัติ
- 1.2.2. เพื่อติดตามผู้ป่วยที่ได้รับการรักษาต่อเนื่องในรูปแบบของ Self Care หรือ Home Care ได้
- 1.2.3. เพื่อให้บริการผ่านทาง Telemedicine โดยที่แพทย์สามารถใช้ข้อมูลจากอุปกรณ์ IoT ประกอบการวินิจฉัย
- 1.2.4. เพื่อสร้างศูนย์รวมข้อมูลจากอุปกรณ์ทางการแพทย์ IoT จากทั่วประเทศ และเป็นแหล่งข้อมูลในการประมวลผลเชิงสถิติและพัฒนาระบบปัญญาประดิษฐ์ที่ช่วยในการวินิจฉัยโรค
- 1.2.5. เพื่อพัฒนาระบบสาธารณสุข โดยการแนะนำและให้การดูแลแบบ Personalization
- 1.2.6. เพื่อสร้างมาตรฐานการส่งข้อมูลที่เปิดให้นักพัฒนาสามารถนำไปใช้ในการพัฒนาอุปกรณ์ IoT ได้อย่างอิสระ
- 1.2.7. เพื่อพัฒนาส่วนเชื่อมต่อกับอุปกรณ์ทางการแพทย์ที่มีใช้อยู่เดิมที่อาจเป็น IoT หรือไม่เป็น IoT ให้สามารถใช้งานกับแพลตฟอร์มได้โดยตรง

- 1.2.8. เพื่อลดการพึ่งพาอุปกรณ์ทางการแพทย์และซอฟต์แวร์จากต่างประเทศที่มีราคาสูงและมีข้อจำกัดในการใช้งาน
 - 1.2.9. เพื่อลดต้นทุนการพัฒนาและดูแลระบบ IoT ของผู้ให้บริการด้านสุขภาพโดยใช้การรวมศูนย์และเปิดให้ทุกฝ่ายสามารถใช้งานได้โดยอาจมีหรือไม่มีค่าใช้จ่าย
 - 1.2.10. เพื่อส่งเสริมการใช้งาน Blockchain-based Personal Health Record เช่น HealthTAG [3] ในการให้สิทธิ์การเข้าถึงข้อมูลระหว่างโรงพยาบาลและหน่วยงานที่เกี่ยวข้องโดยมีการยินยอมจากเจ้าของข้อมูลซึ่งเป็นผู้รับบริการด้านสุขภาพ
 - 1.2.11. เพื่อส่งเสริมโครงการ Smart ER, Smart EMS และ Smart ICU ให้สามารถใช้ประโยชน์จากข้อมูลจากอุปกรณ์ทางการแพทย์ให้มีประสิทธิภาพสูงสุด
- 1.3. ขอบเขตของโครงการและผู้รับผิดชอบในแต่ละส่วน
 - 1.3.1. ขอบเขตของโครงการ

โครงการนี้พัฒนาระบบเพื่อส่งเสริมการทำแพลตฟอร์มรวบรวมข้อมูลจากอุปกรณ์ IoT ทางทางการแพทย์เพื่อรวบรวมข้อมูลให้อยู่ที่ส่วนกลางทำให้สามารถนำใช้ประโยชน์ได้ง่ายขึ้น ทั้งนี้ขอบเขตของแอปพลิเคชันที่จะพัฒนาขึ้นนั้นมีทั้งหมด 4 ระบบงาน ดังนี้

 - 1.3.1.1. ระบบการส่งข้อมูลของอุปกรณ์การแพทย์
 - 1.3.1.1.1. รองรับข้อมูลที่ส่งผ่าน MQTT protocol
 - 1.3.1.1.2. รองรับการส่งข้อมูลของ Kafka
 - 1.3.1.2. ระบบการจัดการข้อมูล
 - 1.3.1.2.1. การบันทึกข้อมูลลงฐานข้อมูลได้แก่ InfluxDB และ MongoDB
 - 1.3.1.2.2. การบันทึกข้อมูลลง Data Lake และการทำ Data Warehouse
 - 1.3.1.2.3. การทำ Time Series Database Clustering
 - 1.3.1.3. ระบบการติดต่อผู้ใช้งาน
 - 1.3.1.3.1. การลงทะเบียนผู้ใช้งาน
 - 1.3.1.3.2. การทำ Dashboard สำหรับผู้ใช้โดยใช้ Data Studio
 - 1.3.1.3.3. การส่งข้อมูล IoT ไปให้ผู้ใช้งาน
 - 1.3.1.4. ระบบแอดมิน
 - 1.3.1.4.1. การลงทะเบียนอุปกรณ์ IoT
 - 1.3.1.4.2. การจัดการสิทธิของผู้ใช้งานในการเข้าถึงข้อมูล

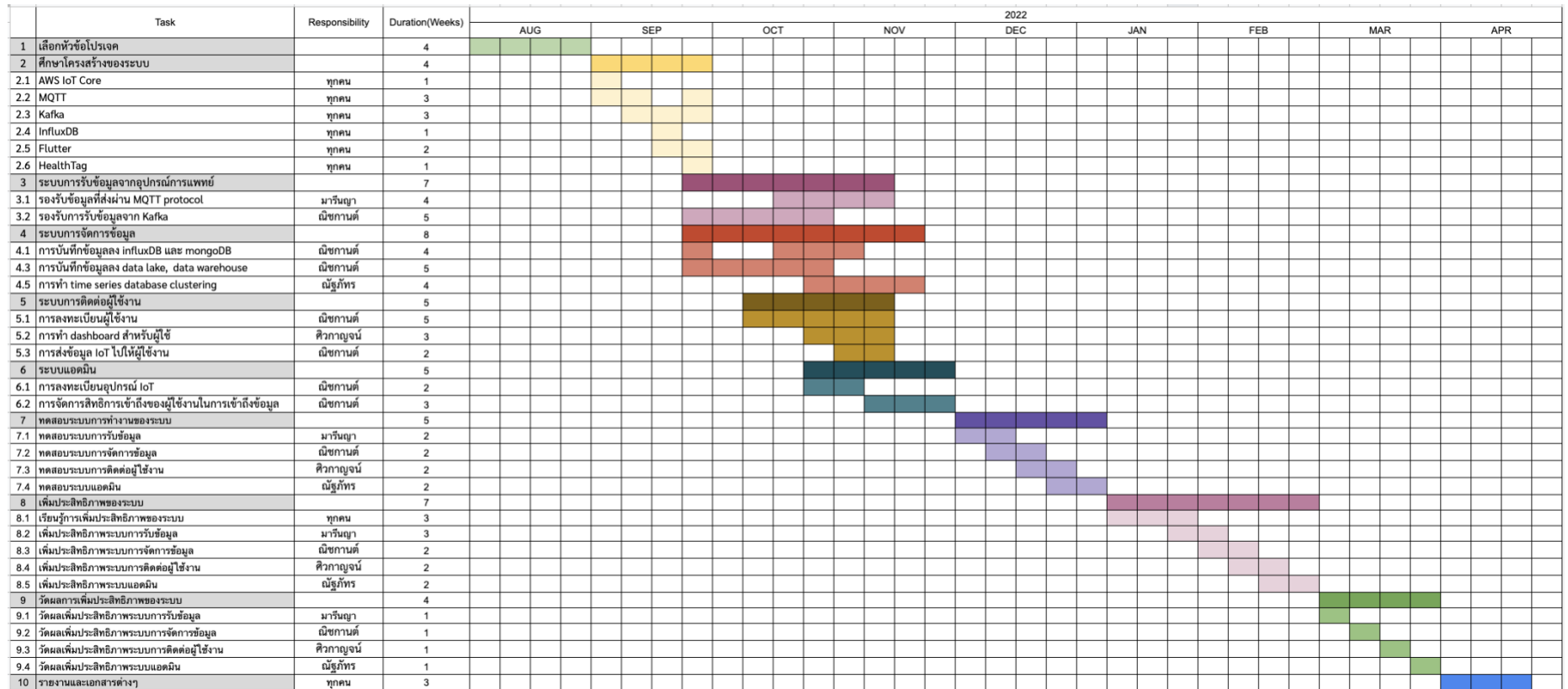
1.3.2. ผู้รับผิดชอบในแต่ละส่วน

ตารางที่ 1 แสดงผู้รับผิดชอบงานในแต่ละส่วน

ชื่อ - สกุล	งานที่รับผิดชอบ
นายณัฐภัทร จารุชัยสิทธิกุล	- การทำ Time Series Database Clustering
นางสาวณิกานต์ ชัยพจนา	<ul style="list-style-type: none">- รองรับการส่งข้อมูลของ Kafka server- การบันทึกข้อมูลลงฐานข้อมูล- การบันทึกข้อมูลลง Data Lake- การลงทะเบียนอุปกรณ์ IoT- การลงทะเบียนผู้ใช้งาน- การส่งข้อมูล IoT ไปให้ผู้ใช้งาน- การจัดการสิทธิของผู้ใช้งานในการเข้าถึงข้อมูล
นางสาวมารีนญา ตะโจปะรัง	- รองรับข้อมูลที่ส่งผ่าน MQTT protocol
นางสาวศิวกาญจน์ จิตต์วโรดม	<ul style="list-style-type: none">- การทำ Data Warehouse- การทำ Dashboard สำหรับผู้ใช้

1.5 แผนการดำเนินงาน

คณะผู้จัดทำได้วางแผนการทำงานตลอดระยะเวลา 9 เดือน คือ ตั้งแต่เดือนสิงหาคม 2565 ถึงเดือนเมษายน 2566 โดยแบ่งออกงานเป็น 10 ส่วนหลัก ๆ ดังรูปที่ 1



รูปที่ 1 Gantt chart

1.4. ประโยชน์ที่คาดว่าจะได้รับ

1.4.1. บุคลากรทางการแพทย์

- 1.4.1.1. ได้รับระบบที่มีความสะดวกในการตรวจสอบข้อมูลของคนไข้ สามารถดูข้อมูลของคนไข้ได้แบบ real-time จากที่ไหนก็ได้
- 1.4.1.2. ได้รับข้อมูลสุขภาพประกอบการวินิจฉัยโรค เนื่องจากการเก็บข้อมูลสุขภาพของผู้ป่วยย้อนหลัง

1.4.2. ผู้ป่วย

- 1.4.2.1. ได้รับความสะดวกในการไปพบแพทย์ เนื่องจากการเก็บข้อมูลสุขภาพย้อนหลัง
- 1.4.2.2. ได้รับความปลอดภัยในชีวิต เนื่องจากการมีอุปกรณ์วัดข้อมูลสุขภาพตลอดเวลา เมื่อมีเหตุฉุกเฉินจะมีระบบแจ้งเตือนทำให้แพทย์สามารถรักษาได้อย่างทันทั่วถึง

1.4.3. นักวิจัย

- 1.4.3.1. ได้รับประโยชน์จากการเก็บข้อมูลใน Data Lake สามารถนำเอาข้อมูลไปใช้ในการศึกษาวิจัยต่อได้ในอนาคต
- 1.4.3.2. ได้รับประโยชน์จากการเก็บรวบรวมข้อมูลที่เป็นระเบียบมากขึ้น ทำให้สะดวกในการนำเอาข้อมูลไปใช้
- 1.4.3.3. ได้รับประโยชน์ในการทำ Data Dashboard ซึ่งสามารถช่วยให้เข้าใจข้อมูลได้ดียิ่งขึ้น

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1. ทฤษฎีที่เกี่ยวข้อง

2.1.1. อุปกรณ์ทางการแพทย์

อุปกรณ์ที่ใช้ในโรงพยาบาลที่สามารถส่งต่อข้อมูลได้ [4]

2.1.2. Structured Data

Structured data ข้อมูลที่มีโครงสร้างชัดเจน หรือมีจำนวนและชื่อ attribute ที่แน่นอนซึ่งเป็นข้อมูลที่นักวิทยาศาสตร์สามารถข้อมูลสามารถนำไปวิเคราะห์ใช้งานต่อได้ง่าย ตัวอย่างเช่น ไฟล์ csv หรือไฟล์ excel [5]

2.1.3. Unstructured Data

Unstructured data ข้อมูลที่ไม่มีโครงสร้างหรือไม่สามารถระบุโครงสร้างที่ชัดเจนได้ เช่น ไฟล์เสียง รูปภาพ วิดีโอ [6]

2.1.4. Messaging Systems

เป็นระบบที่มีหน้าที่ส่งข้อมูลจากแอปพลิเคชันหนึ่งไปยังแอปพลิเคชันหนึ่งโดย แอปพลิเคชันที่ต้องการส่งข้อมูลไม่จำเป็นต้องรอการตอบกลับ ทำให้แอปพลิเคชันสามารถส่งข้อมูลได้ต่อเนื่องโดยไม่ต้องหยุดการทำงาน [7]

2.1.5. Kafka

Kafka คือ Messaging systems แบบหนึ่งในลักษณะ Publish/Subscribe ที่รองรับการ scale ในลักษณะ Horizontal Scale และด้วยคุณสมบัติของ Partition ของ Kafka ที่รองรับ MultiWrite ทำให้รองรับ events จำนวนมากได้ จึงเหมาะกับงานที่เป็น event streaming [8]

2.1.6. MQTT protocol

โพรโทคอลในการส่งข้อมูลระหว่างอุปกรณ์ที่พัฒนามาเพื่อใช้ในระบบ IoT มีทำงานแบบ Broker and Clients Network ซึ่งถูกออกแบบมาเพื่อใช้สื่อสารในระบบเครือข่ายที่มีทรัพยากรค่อนข้างจำกัด ใช้งาน Bandwidth ต่ำและสามารถ publish-subscribe ข้อมูลระหว่าง Device เพื่อสื่อสารกันระหว่างอุปกรณ์ [9]

2.1.7. Kafka Connector

ทำหน้าที่แปลงข้อมูลจาก source systems ไปยัง target systems โดยที่ไม่ต้องสร้าง consumer เพื่อมา subscribe MQTT broker เอง [10]

2.1.8. Web Service

ส่วนหนึ่งของซอฟต์แวร์ที่ให้บริการผ่านอินเทอร์เน็ตโดยมีการแลกเปลี่ยนข้อมูลด้วยวิธีที่เป็นมาตรฐานในรูปแบบที่เป็นมาตรฐาน [11]

2.1.9. REST API

REST ย่อมาจาก Representational State Transfer เป็นข้อกำหนดการส่งข้อมูลระหว่าง Server-Client รูปแบบหนึ่งซึ่งอยู่บนพื้นฐานของ HTTP Protocol เป็นการสร้าง Web Service เพื่อแลกเปลี่ยนข้อมูลกันผ่านแอปพลิเคชัน วิธีหนึ่ง ซึ่งส่งข้อมูลได้หลายชนิด ไม่ว่าจะเป็น Text, XML, JSON REST API คือ interface สำหรับติดต่อสื่อสารของแอปพลิเคชันโดยใช้ REST [12] [13]

2.1.10. Data Lake

Data Lake คือที่เก็บข้อมูลส่วนกลางซึ่งสามารถเก็บข้อมูลทุกรูปแบบ คือ structured data semi-structured data และ unstructured data ซึ่งนิยมใช้เก็บข้อมูลดิบเนื่องจากมีความยืดหยุ่นในการเก็บข้อมูล [14]

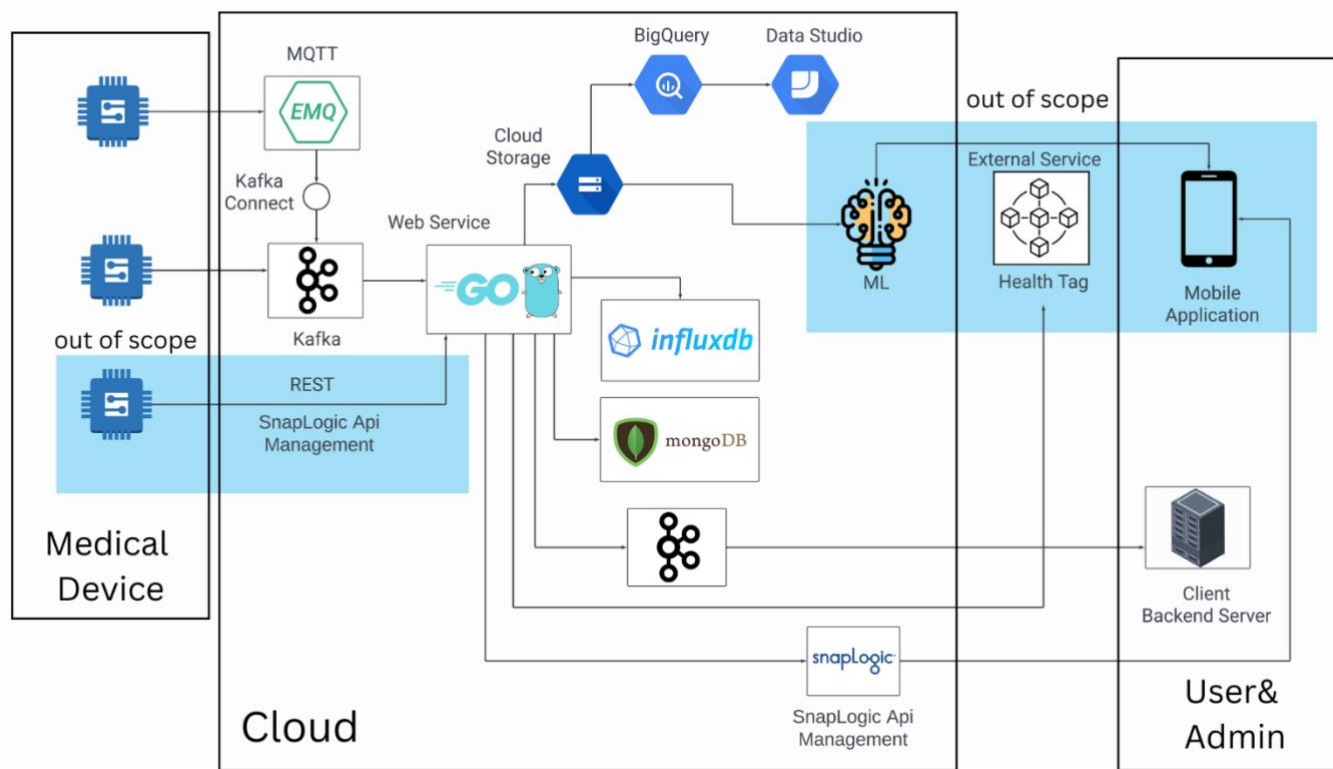
2.1.11. Data Warehouse

Data Warehouse คือ รูปแบบของระบบจัดการข้อมูลที่สามารถเก็บข้อมูลแบบ structured data ได้ดี และถูกออกแบบมาเพื่อสนับสนุนการสร้าง Business Intelligence (BI) และการวิเคราะห์ข้อมูลโดยเฉพาะ นอกจากนั้นยังมีความสามารถในการสืบค้นข้อมูล (Query) จำนวนมาก [15]

3. วิธีการที่นำเสนอ/แนวทางที่คิดว่าจะทำ

3.1. สถาปัตยกรรมของระบบ

คณะผู้จัดทำได้ทำการศึกษาและออกแบบสถาปัตยกรรมของระบบซึ่งประกอบไปด้วย ส่วนที่เป็นอุปกรณ์ทางการแพทย์ ส่วนระบบหลังบ้านและฐานข้อมูลซึ่งอยู่บน cloud และระบบติดต่อกับผู้ใช้งานและแอดมิน โดยนำเสนอสถาปัตยกรรมได้ดังรูปที่ 2



รูปที่ 2 สถาปัตยกรรมของระบบ

3.2. ระบบภายใน

ระบบภายในแบ่งออกเป็น 4 ส่วน

3.2.1. ระบบการส่งข้อมูลของอุปกรณ์การแพทย์

การส่งข้อมูลของอุปกรณ์ทางการแพทย์จะแบ่งออกเป็น 5 รูปแบบ

- อุปกรณ์ที่ไม่เป็น Time Series ซึ่งจะส่งข้อมูลโดยใช้ Bluetooth ส่งต่อให้สมาร์ทโฟน และ สมาร์ทโฟนจะส่งข้อมูลให้กับเซิร์ฟเวอร์โดยใช้ REST API
- อุปกรณ์ที่เชื่อมต่อผ่าน Mobile Application และใช้ MQTT protocol ในการส่งข้อมูลให้กับเซิร์ฟเวอร์

- อุปกรณ์ที่เชื่อมต่อผ่าน Arduino หรือ Raspberry Pi และใช้ MQTT protocol ในการส่งข้อมูลให้กับเซิร์ฟเวอร์
- อุปกรณ์ที่เชื่อมต่ออินเทอร์เน็ตและส่งข้อมูลโดยใช้ MQTT protocol ให้กับเซิร์ฟเวอร์โดยตรง
- อุปกรณ์ที่เชื่อมต่อกับระบบอื่นอยู่แล้ว จะมีการส่งข้อมูลในรูปแบบของ MQTT protocol, Kafka และ REST API

เพราะฉะนั้นจึงสรุปการส่งข้อมูลได้เป็น 3 ส่วนดังรูปที่ 3 ได้แก่

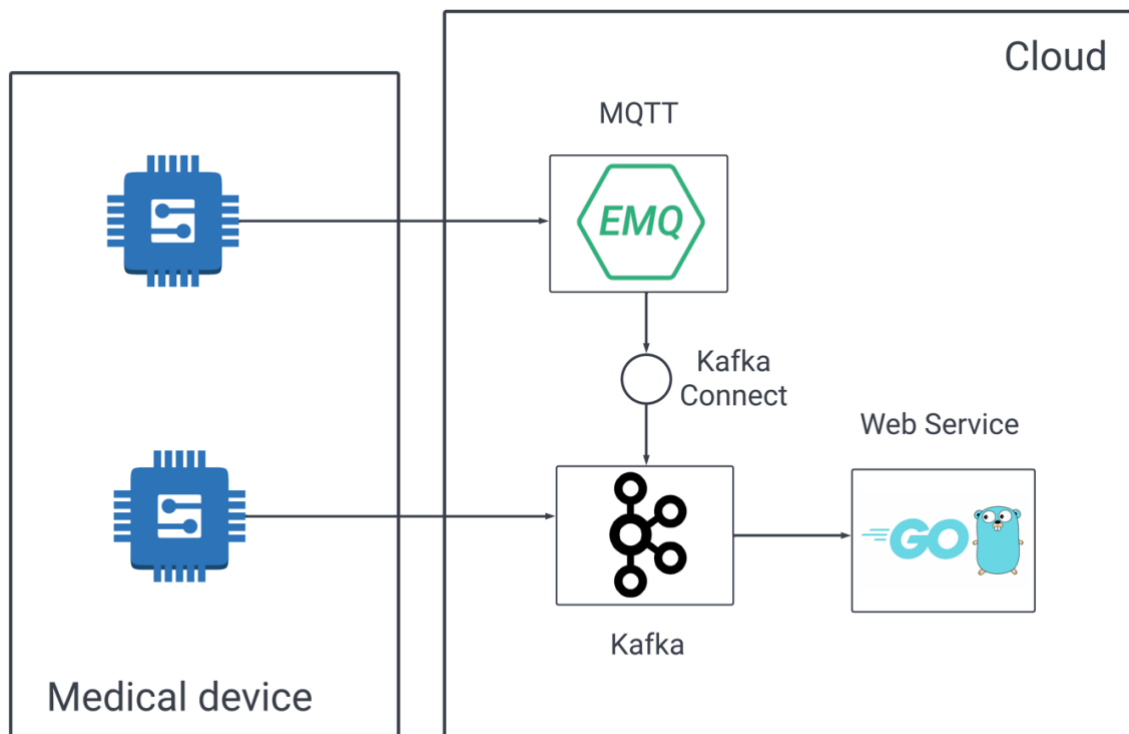
3.2.1.1. การส่งข้อมูลผ่าน REST API (อยู่นอกขอบเขตของโครงการ)

3.2.1.2. การส่งข้อมูลผ่าน Kafka server

จะมีการออกแบบ topic โดย 1 โมเดลอุปกรณ์การแพทย์จะส่งไปยัง topic เดียวกัน กล่าวคืออุปกรณ์ชนิดเดียวกันจะมีการส่งข้อมูลไปยัง topic เดียวกัน และส่งข้อมูลให้กับ web service เพื่อประมวลผลในลำดับถัดไป

3.2.1.3. การส่งข้อมูลผ่าน MQTT protocol

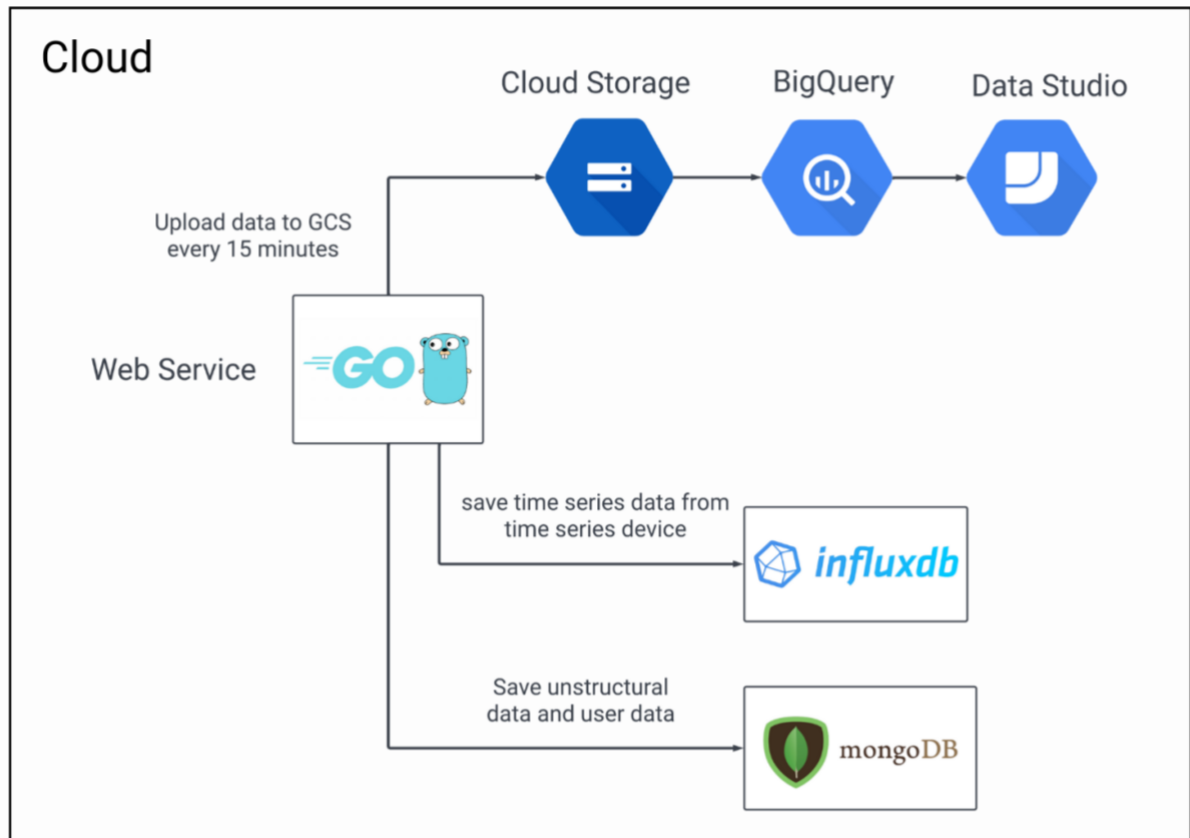
มีการใช้ Kafka connector เพื่อนำข้อมูลจาก EMQX ซึ่งเป็น MQTT Messaging Platform เชื่อมต่อไปยัง Kafka server และส่งประมวลผลที่ web service เดียวกัน



รูปที่ 3 การส่งข้อมูลของอุปกรณ์ IoT

3.2.2. ระบบการจัดการข้อมูล

ระบบนี้จะเป็นการรับข้อมูลจาก Kafka server เพื่อเก็บข้อมูลจากอุปกรณ์การแพทย์ลงสู่ InfluxDB database และมีการส่งข้อมูลขึ้น Google Cloud Storage (Data Lake) ทุกๆ 15 นาที เพื่อจัดเก็บข้อมูลและทำ Data Pipeline ในลำดับถัดไปโดยมีการทำงานดังรูปที่ 4



รูปที่ 4 ระบบการจัดการข้อมูล

3.2.2.1. การออกแบบการเก็บข้อมูลลง InfluxDB database

ระบบของเราใช้ InfluxDB database ในการเก็บข้อมูลจากอุปกรณ์การแพทย์ โดยมีการออกแบบการเก็บข้อมูลดังนี้ 1 โมเดลอุปกรณ์การแพทย์จะจัดเก็บลงใน 1 bucket และมีการแบ่งแยกชนิดอุปกรณ์โดยใช้ InfluxDB structure ที่เรียกว่า `_measurement` ซึ่งจะบันทึก device's id ซึ่งจะเป็นค่าที่ระบุว่าเป็นข้อมูลที่มาจากอุปกรณ์การแพทย์เครื่องไหน และค่าที่วัดได้จากอุปกรณ์ทางการแพทย์ ตัวอย่างเช่น blood pressure, o2 saturation, heart rate จะบันทึกลงใน InfluxDB structure ที่เรียกว่า `_field`

3.2.2.2. การออกแบบการจัดเก็บข้อมูลลง MongoDB database

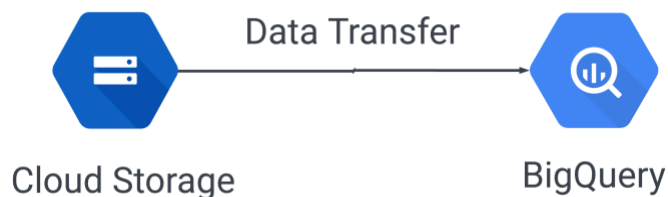
ระบบของเราใช้ MongoDB database ในการเก็บข้อมูลของผู้ใช้งาน, สิทธิการเข้าถึงข้อมูล, ข้อมูลผู้ใช้งานเสนอขอข้อมูลแบบ real-time โดยมีการออกแบบดังนี้คือ 1 collection จะเก็บข้อมูล 1 ชนิดจากที่กล่าวมา

3.2.2.3. การอัปโหลดข้อมูลขึ้น Google Cloud Storage (GCS)

ระบบของเรามีการใช้ Job Scheduler ที่จะคอยทำงานทุกๆ 15 นาที เพื่อดึงข้อมูลจาก InfluxDB และสร้าง csv file format เพื่ออัปโหลดข้อมูลเข้าสู่ GCS ซึ่งเป็น Data Lake ในส่วนของการออกแบบการจัดเก็บข้อมูลใน GCS มีการออกแบบดังนี้คือ model/year/month/day/device_id_timestamp.csv และมีการเก็บผลลัพธ์การทำงานเข้า Elasticsearch เพื่อสามารถติดตามได้ว่าการอัปโหลดได้สำเร็จหรือไม่

3.2.2.4. การทำ Data Warehouse

ระบบจะดึงข้อมูลจาก Google Cloud Storage (GCS) ไปบันทึกลงในตารางของ Google Cloud BigQuery โดยใช้ Data Transfer ซึ่งเป็นบริการภายในของ BigQuery โดยกำหนดให้ทำงานทุก ๆ 15 นาที ตามรูปที่ 5



รูปที่ 5 การส่งข้อมูลจาก GCS ไป BigQuery

3.2.2.5. การทำ Time Series Database Clustering

เนื่องจาก database นั้นจะต้องทำงานได้อยู่ตลอดเวลาและสามารถรับข้อมูลได้เป็นจำนวนมากจึงเพิ่มการใช้งาน Kubernetes เพื่อให้มีกำลังการทำงานสูงขึ้นรวมทั้งมีการจัดการเรื่องการล่มของของฐานข้อมูลหนึ่งๆ ให้สามารถเริ่มต้นทำงานต่อเนื่องจากเดิม

3.2.3.ระบบการติดต่อกับผู้ใช้งาน

ระบบติดต่อกับผู้ใช้งานมีลักษณะดังรูปที่ 6 โดยประกอบด้วย 3 ส่วนที่สำคัญ ดังนี้

3.2.3.1. การลงทะเบียนผู้ใช้งาน

ระบบของเรามี Web Service ที่มี API ไว้ให้ผู้ใช้งานในการลงทะเบียนเพื่อใช้งานระบบ โดยระบบจะมีการบันทึกข้อมูลของผู้ใช้งานลง MongoDB และมีการสร้าง topic ใน Kafka server เพื่อรองรับการส่งข้อมูลแบบ real-time

3.2.3.2. การทำ Dashboard สำหรับผู้ใช้

ระบบมีการทำ Dashboard เพื่อดูสถานะของอุปกรณ์และข้อมูลสุขภาพของผู้ป่วยแต่ละคน โดยใช้ Google Data Studio ซึ่งสามารถดึงข้อมูลจากตารางใน BigQuery มาแสดงได้เลยโดยไม่ต้องใช้เครื่องมืออื่นเพิ่มเติม

3.2.3.3. การส่งข้อมูล IoT ไปให้ผู้ใช้งาน

ระบบของเราจะมีการเช็คสิทธิในการเข้าถึงข้อมูลของผู้ใช้ก่อนจะมีการส่งข้อมูล และการส่งข้อมูลของระบบแบ่งออกเป็น 2 ประเภท คือ

- Historical Data ซึ่งเป็นข้อมูลย้อนหลังของอุปกรณ์ต่างๆ ซึ่งทางระบบจะมี API ในการส่งข้อมูลย้อนหลังให้กับผู้ใช้งาน
- Real-Time Data ระบบจะมี API ในการให้ผู้ใช้งานสามารถขอข้อมูลได้ หลังจากนั้นระบบจะมีการบันทึกข้อมูลการร้องขอลงสู่ MongoDB และระบบจะค่อยๆ ดึงข้อมูลร้องขอจาก MongoDB และสร้าง Kafka consumer เพื่ออ่านข้อมูลตามเงื่อนไขของผู้ใช้งาน และส่งข้อมูล ผ่าน Kafka server โดยมีการออกแบบ topic เป็นดังนี้คือ 1 topic ต่อ 1 ผู้ใช้งาน

3.2.4. ระบบแอดมิน

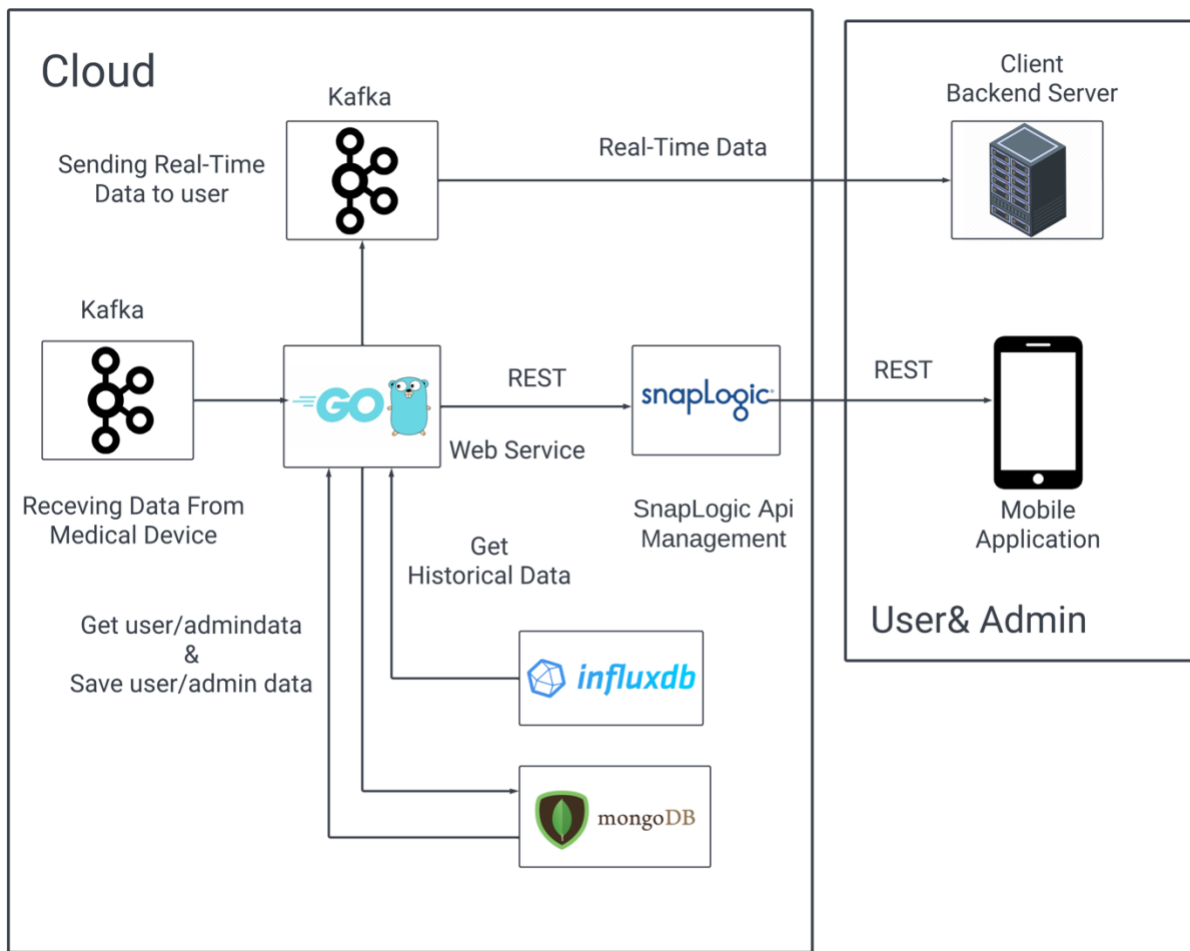
ระบบแอดมินมีลักษณะดังรูปที่ 6 โดยประกอบด้วย 2 ส่วนที่สำคัญ ดังนี้

3.2.4.1. การลงทะเบียนอุปกรณ์ IoT

ระบบของเรามี Web Service ที่ทำ API ในการให้แอดมินลงทะเบียนอุปกรณ์ IoT ซึ่งจะเป็นข้อมูลที่รายละเอียดของอุปกรณ์ว่าสามารถวัดค่าอะไรได้บ้าง และประเภทข้อมูลของค่านั้นๆ ซึ่งระบบจะมีการบันทึกค่าเหล่านั้นใน Configuration File เพื่อใช้อ้างอิงในการทำงานของระบบ

3.2.4.2. การจัดการสิทธิของผู้ใช้งานในการเข้าถึงข้อมูล

ระบบของเรามี Web Service ที่ทำ API ในการให้แอดมินสร้าง แก้ไข และ ลบ สิทธิต่างๆได้ นอกจากนี้สามารถเพิ่มหรือลดสิทธิการเข้าถึงของผู้ใช้งานในระบบได้



รูปที่ 6 ระบบการติดต่อกับผู้ใช้งานและระบบแอดมิน

4. ผลการวิเคราะห์

4.1. ภาษาที่เลือกใช้

ระบบ Web Service คณะผู้จัดทำเลือกใช้ภาษา GO ในการพัฒนาเนื่องจากเป็นภาษาที่ทำงานได้เร็ว อีกทั้งยังรองรับการทำ Concurrent Programming และ Multithreading เพราะฉะนั้นจึงเหมาะกับการรองรับ request เป็นจำนวนมาก นอกจากนี้ยังมี standard library รองรับการทำงานที่หลากหลาย [16]

4.2. ระบบฐานข้อมูลที่ใช้

เนื่องจากระบบมีการรองรับการบันทึกข้อมูลจากอุปกรณ์การแพทย์ซึ่งมีลักษณะข้อมูลเป็น Time Series Data เพราะฉะนั้นจึงเลือกใช้ InfluxDB database ซึ่งมีคุณสมบัติการเขียนข้อมูลได้เร็ว อีกทั้งยังมีการออกแบบ query ที่เหมาะกับ Time Series Data นอกจากนี้ยังมี API ให้ Web Service สามารถติดต่อกับฐานข้อมูลได้สะดวก [17]

นอกจากมีข้อมูลประเภท Time Series Database ระบบยังมีการรองรับข้อมูลของผู้ใช้ ข้อมูลสิทธิการเข้าถึงข้อมูล คณะผู้จัดทำจึงเลือกใช้ MongoDB เนื่องจาก MongoDB เป็น NoSQL ทำให้ง่ายต่อการเปลี่ยนแปลงโครงสร้างของข้อมูล อีกทั้งยังมี driver ที่ช่วยในการติดต่อกับฐานข้อมูลที่รองรับภาษา Go ที่ผู้จัดทำเลือกใช้ และยังสามารถ scale ฐานข้อมูลได้ง่าย [18]

4.3. Library ในการให้ Web Service ติดต่อกับ Kafka server

คณะผู้จัดทำเลือกใช้ SARAMA library เนื่องจากมี MIT license อีกทั้งยังมีบริการทั้ง low-level API และ high-level API ในการติดต่อและจัดการ Kafka server [19]

4.4. เครื่องมือในการจัดการ Logs ของ web service

คณะผู้จัดทำเลือกใช้ Elasticsearch ร่วมกับ Kibana เนื่องจากเครื่องมือทั้งสองมีการพัฒนาให้สามารถทำงานร่วมกันได้ดี อีกทั้ง Elasticsearch สามารถทำงานได้รวดเร็วเนื่องจากมีการออกแบบการจัดเก็บข้อมูลโดย BKD tree นอกจากนี้ยังมี API ให้ Web Service สามารถติดต่อได้ง่าย [20] [21]

4.5. การทำ Data Lake

ระบบนี้จะทำการเก็บข้อมูลจากอุปกรณ์ IoT ทางด้านการแพทย์ซึ่งข้อมูลเหล่านั้นมีลักษณะแบบ semi-structure เพราะอุปกรณ์ IoT แต่ละประเภทอาจเก็บข้อมูลลักษณะแตกต่างกัน เช่น อุปกรณ์ประเภทที่ 1 สามารถเก็บค่าอุณหภูมิร่างกายและอัตราการเต้นของหัวใจได้ ส่วนอุปกรณ์ประเภทที่ 2 เก็บค่าอัตราการเต้นของหัวใจได้อย่างเดียว เป็นต้น

ใช้ Google Cloud Storage (GCS) ในการทำ Data Lake เนื่องจากมี Data Retention ทำให้สามารถเก็บข้อมูลที่มีอยู่ไว้ได้หากมีการเปลี่ยนแปลงกฎการรักษาข้อมูล (Retention rule) ซึ่งข้อดีนี้เป็นคุณสมบัติที่ดีในการเก็บข้อมูลด้านสุขภาพ อีกทั้งยังมี interface ที่ทำให้ผู้ใช้งานใช้งานได้ง่าย [22]

4.6. การทำ Data Warehouse

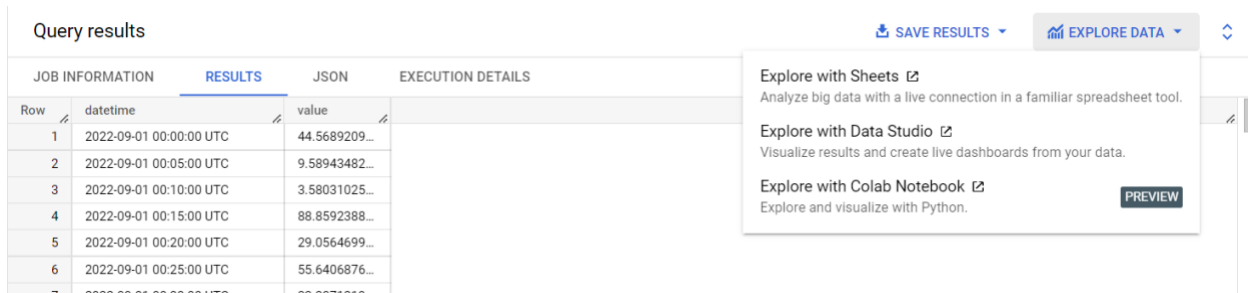
ในการทำ Data Warehouse เลือกใช้ Google Cloud BigQuery เนื่องจากเป็นบริการของ Google เช่นเดียวกันกับ GCS ทำให้สามารถเชื่อมต่อกันได้ง่าย มี API หรือ บริการภายในในการเชื่อมต่อกับ GCS

ระบบจะต้องนำข้อมูลที่อยู่ใน GCS ไปใส่ในตารางใน BigQuery ในส่วนนี้เลือกใช้ Data Transfer ซึ่งเป็นบริการใน BigQuery ที่จะเช็คข้อมูลใน GCS ทุก ๆ 15 นาทีว่ามีข้อมูลเพิ่มหรือไม่ ถ้ามีจะนำข้อมูลที่เพิ่มมานั้นไปต่อท้ายในตารางของ BigQuery เหตุผลที่ใช้ 15 นาที เพราะว่าเป็นคาบที่น้อยที่สุดที่สามารถตั้งค่าได้ใน Data Transfer [23]

4.7. การทำ Dashboard

Dashboard มีความสำคัญมากต่อการ monitor ข้อมูลของผู้ใช้หรือบุคลากรทางการแพทย์ที่ต้องการตรวจสอบสถานะโดยภาพรวมของอุปกรณ์ IoT หรือดูค่าของอุปกรณ์ IoT ชนิดหนึ่ง ๆ ทั้งในแบบ Real-Time และแบบข้อมูลย้อนหลัง เพื่อติดตามดูแลสุขภาพของคนไข้ เมื่อเกิดเหตุฉุกเฉิน แพทย์จะสามารถรักษาอาการอย่างทัน่วงที

การทำ Dashboard สำหรับผู้ใช้ ใช้ Google Data Studio ซึ่งเป็นบริการของ Google เช่นเดียวกันกับ GCS และ BigQuery เนื่องจากใน BigQuery สามารถ Explore Data โดยใช้ Data Studio ได้ ทำให้ไม่ต้องใช้ API หรือ เครื่องมืออื่น ๆ เพิ่มเติม นอกจากนี้เมื่อมีการเปลี่ยนแปลงของตารางใน BigQuery จะทำการเปลี่ยนแปลงข้อมูลใน Data Studio โดยอัตโนมัติดังรูปที่ 7



JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	datetime	value		
1	2022-09-01 00:00:00 UTC	44.5689209...		
2	2022-09-01 00:05:00 UTC	9.58943482...		
3	2022-09-01 00:10:00 UTC	3.58031025...		
4	2022-09-01 00:15:00 UTC	88.8592388...		
5	2022-09-01 00:20:00 UTC	29.0564699...		
6	2022-09-01 00:25:00 UTC	55.6406876...		
7	2022-09-01 00:30:00 UTC	22.2071219...		

รูปที่ 7 GUI ของ BigQuery

4.8. การเชื่อมต่อ MQTT กับ Kafka

การเชื่อมต่อของ MQTT กับ Kafka สำคัญเป็นอย่างมากเนื่องจาก เพราะการรับข้อมูลโดยใช้ MQTT เหมาะกับการต่อกับอุปกรณ์จำพวก sensor แต่ไม่เหมาะกับการเก็บข้อมูลจำนวนมากจึงต้องมีการเชื่อมต่อกับ Kafka เพื่อเก็บข้อมูลให้มีประสิทธิภาพและรวบรวมของอุปกรณ์ทั้งหมดไว้ในที่เดียวเพื่อให้ง่ายต่อการส่งต่อ

การเชื่อมต่อ MQTT กับ Kafka ใช้ Confluent Platform ในการเชื่อมต่อโดยใช้ Connector Service ซึ่งสามารถทำงานร่วมกันได้เป็นอย่างดีและส่งข้อมูลได้มีประสิทธิภาพ [24]

5. ผลกระทบทางสังคมของโครงการ

ทางผู้จัดทำได้ทำการพิจารณาผลกระทบทางสังคมที่จะเกิดจากโครงการที่กำลังพัฒนาขึ้นนี้ โดยจากการวิเคราะห์ได้ผลกระทบทางบวกออกมาดังนี้

- 5.1. เกิดแหล่งข้อมูลความรู้ทางด้านสุขภาพของประเทศไทยในอนาคต สามารถนำข้อมูลมาวิเคราะห์แนวโน้มสุขภาพของคนไทย
- 5.2. ช่วยให้การรักษาทางไกลสามารถทำได้อย่างมีประสิทธิภาพมากขึ้น ทำให้ช่วยเพิ่มพื้นที่สำหรับผู้ป่วยฉุกเฉินในโรงพยาบาล ลดความแออัดของผู้ป่วย
- 5.3. ลดงบประมาณในการดูแลระบบฐานของข้อมูลของแต่ละโรงพยาบาลรัฐบาล เนื่องจากแพลตฟอร์มจะเป็นศูนย์กลางในการรวบรวมข้อมูล

6. บทสรุปและงานที่จะทำต่อไป

คณะผู้จัดทำได้มีการเรียนรู้เครื่องมือในการพัฒนาระบบ นอกจากนี้ได้ดำเนินการสร้างระบบที่รองรับการนำข้อมูลเข้าโดยผ่าน MQTT protocol Kafka server เพื่อบันทึกข้อมูลที่ได้รับไปยัง database ที่ออกแบบไว้และมีบริการส่งข้อมูลย้อนหลังและข้อมูลแบบ real-time ให้แก่ผู้ใช้งาน สุดท้ายได้มีการสร้างบริการให้กับแอดมินเพื่อจัดการกับอุปกรณ์การแพทย์และสิทธิของผู้ใช้งานในการเข้าถึงข้อมูลของอุปกรณ์ ดังรูปที่ 8

Task	Responsibility	Duration(Weeks)	2022											
			AUG			SEP			OCT			NOV		
1 เลือกหัวข้อโปรเจกต์		4												
2 ศึกษาโครงสร้างของระบบ		4												
2.1 AWS IoT Core	ทุกคน	1												
2.2 MQTT	ทุกคน	3												
2.3 Kafka	ทุกคน	3												
2.4 InfluxDB	ทุกคน	1												
2.5 Flutter	ทุกคน	2												
2.6 HealthTag	ทุกคน	1												
3 ระบบการรับข้อมูลจากอุปกรณ์การแพทย์		7												
3.1 รองรับข้อมูลที่ส่งผ่าน MQTT protocol	มารินญา	4												
3.2 รองรับการรับข้อมูลจาก Kafka	ณิชกานต์	5												
4 ระบบการจัดการข้อมูล		8												
4.1 การบันทึกข้อมูลลง influxDB และ mongoDB	ณิชกานต์	4												
4.3 การบันทึกข้อมูลลง data lake, data warehouse	ณิชกานต์	5												
4.5 การทำ time series database clustering	ณัฐภัทร	4												
5 ระบบการติดต่อผู้ใช้งาน		5												
5.1 การลงทะเบียนผู้ใช้งาน	ณิชกานต์	5												
5.2 การทำ dashboard สำหรับผู้ใช้	ศิวกาญจน์	3												
5.3 การส่งข้อมูล IoT ไปให้ผู้ใช้งาน	ณิชกานต์	2												
6 ระบบแอดมิน		5												
6.1 การลงทะเบียนอุปกรณ์ IoT	ณิชกานต์	2												
6.2 การจัดการสิทธิการเข้าถึงของผู้ใช้งานในการเข้าถึงข้อมูล	ณิชกานต์	3												

รูปที่ 8 งานที่ทำแล้วเสร็จในปัจจุบัน

ในอนาคตคณะผู้จัดทำได้มีการวางแผนในการทดสอบระบบ เพิ่มประสิทธิภาพในส่วน of reliability และ availability และสุดท้ายจะมีการวัดประสิทธิภาพของระบบเปรียบเทียบระหว่างก่อนและหลังจากการเพิ่มประสิทธิภาพ

	Task	Responsibility	Duration(Weeks)	2022											
				DEC	JAN	FEB	MAR	APR							
7	ทดสอบระบบการทำงานระบบ		5												
7.1	ทดสอบระบบการรับข้อมูล	มารินญา	2												
7.2	ทดสอบระบบการจัดการข้อมูล	ณัฏกานต์	2												
7.3	ทดสอบระบบการติดต่อผู้ใช้งาน	ศิวภาณุจน์	2												
7.4	ทดสอบระบบแอดมิน	ณัฐภัทร	2												
8	เพิ่มประสิทธิภาพของระบบ		7												
8.1	เรียนรู้การเพิ่มประสิทธิภาพของระบบ	ทุกคน	3												
8.2	เพิ่มประสิทธิภาพระบบการรับข้อมูล	มารินญา	3												
8.3	เพิ่มประสิทธิภาพระบบการจัดการข้อมูล	ณัฏกานต์	2												
8.4	เพิ่มประสิทธิภาพระบบการติดต่อผู้ใช้งาน	ศิวภาณุจน์	2												
8.5	เพิ่มประสิทธิภาพระบบแอดมิน	ณัฐภัทร	2												
9	วัดผลการเพิ่มประสิทธิภาพของระบบ		4												
9.1	วัดผลเพิ่มประสิทธิภาพระบบการรับข้อมูล	มารินญา	1												
9.2	วัดผลเพิ่มประสิทธิภาพระบบการจัดการข้อมูล	ณัฏกานต์	1												
9.3	วัดผลเพิ่มประสิทธิภาพระบบการติดต่อผู้ใช้งาน	ศิวภาณุจน์	1												
9.4	วัดผลเพิ่มประสิทธิภาพระบบแอดมิน	ณัฐภัทร	1												
10	รายงานและเอกสารต่างๆ	ทุกคน	3												

รูปที่ 9 งานที่จะทำต่อในอนาคต

7. เอกสารอ้างอิง

- [1] Google Cloud. (ม.ป.ป.). Google Cloud Pricing Calculator. สืบค้นเมื่อ 3 พฤศจิกายน 2565. จาก. <https://cloud.google.com/products/calculator/#id=05c923fb-db28-416f-afdd-527421f6cc5f>
- [2] Hansa Manakitsomboon. (2565). Number of hospitals in Thailand from 2011 to 2020. สืบค้นเมื่อ 3 พฤศจิกายน 2565. จาก. <https://www.statista.com/statistics/1114483/thailand-number-of-hospitals/>
- [3] HealthTAG. (2565). HealthTAG. สืบค้นเมื่อ 30 ตุลาคม 2565. จาก. <https://healthtag.io/th>
- [4] อภิรักษ์ คงคาเพชร. (2560). ความหมาย ความเป็นมา แนวคิด ทฤษฎี ในการคุ้มครองผู้บริโภคที่เกี่ยวข้องกับเครื่องมือแพทย์ชนิดเต้านมเทียมซิลิโคนใช้ฝังในร่างกาย. สืบค้นเมื่อ 3 พฤศจิกายน 2565. จาก. <http://dspace.spu.ac.th/bitstream/123456789/5397/10/10.%E0%B8%9A%E0%B8%97%E0%B8%97%E0%B8%B5%E0%B9%88%202.pdf>
- [5] TIBCO. (ม.ป.ป.). What is Structured Data? สืบค้นเมื่อ 29 ตุลาคม 2565. จาก. <https://www.tibco.com/reference-center/what-is-structured-data>
- [6] Mongo. (ม.ป.ป.). Unstructured Data. สืบค้นเมื่อ 29 ตุลาคม 2565. จาก. <https://www.mongodb.com/unstructured-data>
- [7] Pethuru Raj. (2561). A Deep Dive into NoSQL Databases: The Use Cases and Applications. สืบค้นเมื่อ 3 พฤศจิกายน 2565. จาก. <https://www.sciencedirect.com/topics/computer-science/messaging-system>
- [8] Kafka. (ม.ป.ป.). Introduction. สืบค้นเมื่อ 28 ตุลาคม 2565. จาก. <https://kafka.apache.org/intro>

- [9] Sonic Automation. (2563). MQTT กับงาน Industrial Internet of Things (IoT). สืบค้นเมื่อ 28 ตุลาคม 2565. จาก. [https://sonicautomation.co.th/mqtt-for-iiot-application/#:~:text=MQTT%20\(Message%20Queueing%20Telemetry%20Transport,MQTT%20%E0%B8%96%E0%B8%B9%E0%B8%81%E0%B8%9E%E0%B8%B1%E0%B8%92%E0%B8%99%E0%B8%B2%E0%B8%82%E0%B8%B6%E0%B9%89%E0%B8%99%E0%B8%A1%E0%B8%B2](https://sonicautomation.co.th/mqtt-for-iiot-application/#:~:text=MQTT%20(Message%20Queueing%20Telemetry%20Transport,MQTT%20%E0%B8%96%E0%B8%B9%E0%B8%81%E0%B8%9E%E0%B8%B1%E0%B8%92%E0%B8%99%E0%B8%B2%E0%B8%82%E0%B8%B6%E0%B9%89%E0%B8%99%E0%B8%A1%E0%B8%B2)
- [10] Confluent. (ม.ป.ป.). Kafka Connect. สืบค้นเมื่อ 28 ตุลาคม 2565. จาก. <https://docs.confluent.io/platform/current/connect/index.html#kafka-connect>
- [11] Team Cleo. (ม.ป.ป.). What Are Web Services? Easy-to-Learn Concepts with Examples. สืบค้นเมื่อ 3 พฤศจิกายน 2565. จาก. <https://www.cleo.com/blog/knowledge-base-web-services>
- [12] <https://medium.com/@Biewz/ความหมายแตกต่างระหว่าง-web-application-และ-web-service-aa12b992d8ae>
- [13] RedHat. (2563). What is a REST API? สืบค้นเมื่อ 2 พฤศจิกายน 2565. จาก. <https://www.redhat.com/en/topics/api/what-is-a-rest-api>
- [14] Google Cloud. (ม.ป.ป.). What is a Data Lake? สืบค้นเมื่อ 2 พฤศจิกายน 2565. จาก. <https://cloud.google.com/learn/what-is-a-data-lake>
- [15] Oracle. (ม.ป.ป.). What Is a Data Warehouse? สืบค้นเมื่อ 2 พฤศจิกายน 2565. จาก. <https://www.oracle.com/database/what-is-a-data-warehouse/>
- [16] GO. (ม.ป.ป.). Case studies. สืบค้นเมื่อ 3 พฤศจิกายน 2565. จาก. <https://go.dev/solutions/#>
- [17] InfluxData. (ม.ป.ป.). InfluxData. สืบค้นเมื่อ 30 ตุลาคม 2565. จาก. <https://www.influxdata.com/>
- [18] MongoDB. (ม.ป.ป.). Why Use MongoDB and When to Use It? สืบค้นเมื่อ 30 ตุลาคม 2565. จาก. <https://www.mongodb.com/why-use-mongodb>
- [19] Shopify. (2562). Sarama. สืบค้นเมื่อ 25 ตุลาคม 2565. จาก. <https://pkg.go.dev/github.com/shopify/sarama#section-readme>
- [20] Elastic. (ม.ป.ป.). What is Kibana? สืบค้นเมื่อ 25 ตุลาคม 2565. จาก. <https://www.elastic.co/what-is/kibana>
- [21] Elastic. (ม.ป.ป.). Elasticsearch. สืบค้นเมื่อ 25 ตุลาคม 2565. จาก. <https://www.elastic.co/elasticsearch/>
- [22] Olga Weis. (2564). Differences between Google Cloud Storage and Amazon S3. สืบค้นเมื่อ 1 พฤศจิกายน 2565. จาก. <https://cloudmounter.net/amazon-s3-vs-google-cloud-storage/>

[23] Google Cloud. (ม.ป.ป.). BigQuery Data Transfer Service API Client Libraries. สืบค้นเมื่อ 26 ตุลาคม 2565. จาก.

<https://cloud.google.com/bigquery/docs/reference/datatransfer/libraries#client-libraries-install-go>

[24] Confluent. (ม.ป.ป.). MQTT Source Connector for Confluent Cloud. สืบค้นเมื่อ 2 พฤศจิกายน 2565. จาก. <https://docs.confluent.io/cloud/current/connectors/cc-mqtt-source.html#mqtt-source-connector-for-ccloud>