โครงการ

การเรียนการสอนเพื่อเสริมประสบการณ์

ชื่อโครงงาน แอปพลิเคชันช่วยในการตรวจสอบคุณภาพสำหรับกระบวนการก่อสร้าง (ระบบส่วนหลังบ้าน)
Construction process quality control assisted application (Back-end)

ชื่อนิสิต นายคุปติพงศ์ สุวรรณไตรย์
นายวัชรพงศ์ พงศ์สุทธิศรัทธา

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์
สาขาวิชาวิทยาการคอมพิวเตอร์

ปีการศึกษา 2562

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

แอปพลิเคชันช่วยในการตรวจสอบคุณภาพสำหรับกระบวนการก่อสร้าง (ระบบส่วนหลังบ้าน)

คุปติพงศ์ สุวรรณไตรย์

วัชรพงศ์ พงศ์สุทธิศรัทธา

โครงการนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตร์บัณฑิต

สาขาวิชาคณิตศาสตร์/วิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2562

Construction process quality control assisted application (Back-end)

Kuptipong Suwannatrai

Watcharapong Pongsuttisatta

A Project Submitted in Partial Fulfillment of the Requirements

for the Degree of Bachelor of Science Program in Computer Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2019

นายคุปติพงศ์ สุวรรณไตรย์, นายวัชรพงศ์ พงศ์สุทธิศรัทธา: แอปพลิเคชันช่วยในการตรวจสอบคุณภาพสำหรับกระบวนการก่อสร้าง (ระบบส่วนหลังบ้าน). (Construction process quality control assisted application (Back-end)) อ.ที่ปรึกษาโครงงานหลัก : รองศาสตราจารย์ ดร.นกุล คูหะโรจนานนท์, 144 หน้า.

คณะผู้จัดทำได้พัฒนาแอปพลิเคชัน ที่พัฒนาโดย React Native มาช่วยเหลือในการตรวจสอบคุณภาพสำหรับกระบวนการก่อสร้าง เพื่อลดความผิดพลาดในการสื่อสาร

ผู้จัดทำแอปพลิเคชันได้ออกแบบระบบ Backend ที่เป็น Cloud Services โดยเลือกใช้เป็น AWS (Amazon Web Services) ซึ่งไม่จำเป็นจะต้องใช้เครื่องเซิฟเวอร์ขนาดใหญ่ในการพัฒนา ในการพัฒนาแอปพลิเคชัน React Native นั้น ผู้จัดทำแอปพลิเคชันในส่วนหลังบ้านเลือกหยิบ Services ที่สำคัญต่อการสร้างแอปพลิเคชันหลัก ๆ มาดังนี้ ได้แก่ AWS Lambda, Amazon API Gateway, Amazon S3, Amazon CloudWatch, AWS Amplify, Amazon Cognito, AWS IAM, Amazon และ Amazon DynamoDB อีกทั้งผู้จัดทำแอปพลิเคชันในส่วนหลังบ้านยังใช้ Services อื่น ๆ นอกเหนือจากของ Amazon ได้แก่ Expo Server, Firebase Cloud Messaging

ผู้จัดทำได้นำแอปพลิเคชันไปทดสอบกับผู้ใช้งานจริง โดยทดสอบในเชิงฟังก์ชันและเชิงประสิทธิภาพ ได้ผลตอบรับที่น่าพึงพอใจ อีกทั้งยังทดสอบกับนักพัฒนาในส่วนหน้าบ้านที่เป็นผู้ใช้บริการหลักของระบบหลังบ้าน ได้รับผลตอบรับที่ดีเช่นกัน

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ ลายมือชื่อนิสิต .....

ลายมือชื่อนิสิต .....

สาขาวิชา วิทยาการคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาโครงงานหลัก .....

ปีการศึกษา 2562

5933608323, 5933651223: MAJOR COMPUTER SCIENCE

KEYWORDS : Construction / Application / Back-end / Amazon Web Service

Kuptipong Suwannatrai, Watcharapong Pongsuttisatta: Construction process quality control assisted application (Back-end). ADVISOR : Nagul Cooharojananone, Ph.D., 144 pp.

We intend to develop the application in React Native framework that helps with the validation process during construction in order to enhance the communication efficiency within the organization.

Backend development team chooses AWS (Amazon Web Services) to deploy our backend system. We pick AWS Lambda, Amazon API Gateway, Amazon S3, Amazon CloudWatch, AWS Amplify, Amazon SNS, Amazon Cognito, AWS IAM and the last one is Amazon DynamoDB. Not only Amazon services, we also include Expo Server and Firebase Cloud Messaging to take care of push notifications.

Our application has been tested by real users of reliable construction companies in field of performance and functional. The result is satisfactory. In backend, target users are frontend developers so we invite them to test our backend system, favorable outcome returned.

Department :Mathematics and Computer Science Student's Signature

Student's Signature

Field of Study : Computer Science Advisor's Signature

Academic Year : 2019

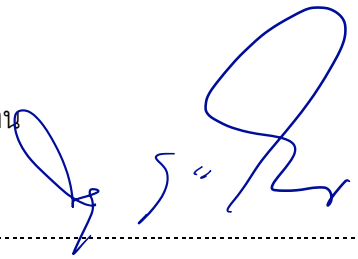
หัวข้อโครงงาน	แอปพลิเคชันช่วยในการตรวจสอบคุณภาพสำหรับกระบวนการก่อสร้าง (ระบบส่วนหลังบ้าน)
โดย	นายคุปติพงศ์ สุวรรณไตรย์ นายวัชรพงศ์ พงศ์สุทธิศรัทธา
สาขาวิชา	วิทยาการคอมพิวเตอร์
อาจารย์ที่ปรึกษาโครงงานหลัก	รองศาสตราจารย์ ดร.นกุล คูหะโรจนานนท์

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับโครงงานฉบับนี้เป็นส่วนหนึ่ง ของการศึกษาตามหลักสูตรปริญญาบัณฑิต ในรายวิชา 2301499 โครงงานวิทยาศาสตร์ (Senior Project)

..... หัวหน้าภาควิชาคณิตศาสตร์

(ศาสตราจารย์ ดร.กฤษณะ เนียมมณี) และวิทยาการคอมพิวเตอร์

คณะกรรมการสอบโครงงาน



..... อาจารย์ที่ปรึกษาโครงงานหลัก

(รองศาสตราจารย์ ดร.นกุล คูหะโรจนานนท์)

..... กรรมการ

(รองศาสตราจารย์ ดร.พีระพนธ์ โสพัศสถิตย์)

..... กรรมการ

(รองศาสตราจารย์ ดร.ภัทรสินี ภัทรโกศล)

กิตติกรรมประกาศ

การจัดทำแอปพลิเคชันช่วยในการตรวจสอบคุณภาพสำหรับกระบวนการก่อสร้าง สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความอนุเคราะห์และความช่วยเหลือจากคณาจารย์และบุคลากรต่าง ๆ ในตลอดระยะเวลาการดำเนินโครงงาน ผู้พัฒนาจึงขอขอบคุณในความช่วยเหลือต่าง ๆ ที่มอบให้ ดังนี้

ขอขอบพระคุณ รองศาสตราจารย์ ดร.นกุล คูหะโรจนานนท์ อาจารย์ที่ปรึกษาโครงงาน ที่คอยให้คำแนะนำปรึกษาชี้แนะแนวความคิดในการทำโครงงาน ตลอดจนช่วยปรับปรุงแก้ไขข้อบกพร่องต่าง ๆ ด้วยความเอาใจใส่ จึงทำให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบพระคุณ รองศาสตราจารย์ ดร. พีระพนธ์ โสพัศสถิตย์ กรรมการสอบโครงงาน ที่ช่วยแนะแนวมุมมองในการออกแบบวิศวกรรมซอฟต์แวร์ ตักเตือนถึงเรื่องจรรยาบรรณของวิชาชีพ รวมถึงการใช้แผนภาพและเรื่องการทำงานของซอฟต์แวร์ต่าง ๆ ในวิชาเรียน จึงทำให้โครงงานมีความสมบูรณ์ และมีประสิทธิภาพมากยิ่งขึ้น

ขอขอบพระคุณ รองศาสตราจารย์ ดร. ภัทรสินี ภัทรโกศล กรรมการสอบโครงงาน ที่ช่วยให้คำแนะนำในเรื่องของการออกแบบระบบเป็นอย่างดี ทำให้สามารถออกแบบระบบออกมาได้อย่างสมบูรณ์ สามารถทดสอบได้อย่างมีประสิทธิภาพ และยังช่วยชี้แนะข้อบกพร่องของโครงงานที่อาจจะทำให้เกิดข้อผิดพลาดได้อีก จึงทำให้โครงงานมีความสมบูรณ์แบบมากยิ่งขึ้น

ขอขอบพี่ ๆ และผู้มีอุปการะคุณจากบริษัท Team built CO.,LTD., บริษัท ปริญสิริ จำกัด (มหาชน), สำนักวิจัยและพัฒนาทาง กรมทางหลวง ที่ให้ความร่วมมือในการให้ข้อมูลความต้องการในการนำมาพัฒนาโครงงานได้อย่างครบถ้วนซึ่งทำให้โครงงานสำเร็จไปได้ด้วยดี

ขอขอบคุณเพื่อน ๆ ทุกคนที่คอยให้ความช่วยเหลือ และให้คำปรึกษาเสมอมา จึงทำให้โครงงานนี้ประสบความสำเร็จได้

สุดท้ายนี้ ขอขอบคุณความกรุณาอันดีจากทุกท่านที่ได้กล่าวนามมาไว้ข้างต้น รวมถึงบุคคลท่านอื่นที่ไม่ได้กล่าวนามไว้ ณ ที่นี้ด้วย สำหรับความช่วยเหลือและคำแนะนำต่าง ๆ ที่คอยผลักดันให้โครงงานนี้ประสบผลสำเร็จไปได้ด้วยดี

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

ในปัจจุบันมีผู้ใช้หลากหลายประเภทใช้แอปพลิเคชันสื่อสารที่มีชื่อเสียงอย่างเช่น LINE แต่แอปพลิเคชันดังกล่าวกลับไม่สามารถตอบโจทย์ผู้ใช้ได้ครบ ทำให้เกิดปัญหาในการสื่อสารในองค์กร โดยส่งผลกระทบต่อเวลา, ค่าใช้จ่าย และบุคลากร หากเราจัดทำแอปพลิเคชันที่จะมาช่วยเหลือในการตรวจสอบคุณภาพสำหรับกระบวนการก่อสร้าง จะทำให้การสื่อสารระหว่างทีมก่อสร้างเป็นไปในแบบแผนเดียวกันและมีมาตรฐานเดียวกัน แต่ระบบฐานข้อมูลและการเก็บข้อมูลที่ใช้ในปัจจุบันนั้นจำเป็นจะต้องวางเครื่องเซิฟเวอร์ที่มีค่าใช้จ่ายเกินขนาดของแอปพลิเคชันเกินไป และเพื่อศึกษาระบบฐานข้อมูลใหม่ ๆ , บริการใหม่ ๆ ที่เพิ่งกำเนิดขึ้นมาในยุคปัจจุบัน ซึ่งให้บริการในรูปแบบ Cloud Services ยกอย่างเช่น Amazon Web Services (AWS), Google Cloud ซึ่งไม่จำเป็นจะต้องใช้เครื่องเซิฟเวอร์ขนาดใหญ่ และจ่ายเงินตามที่ใช้งานจริง

ผู้พัฒนาเห็นว่าหากมีช่องทางที่จะนำบริการเหล่านี้มาใช้กับแอปพลิเคชันที่ผู้พัฒนาจะพัฒนาซึ่งกล่าวมาในข้างต้น จะทำให้ระบบส่วนหน้าบ้านสามารถใช้บริการที่รวดเร็วกว่าเดิม, มีค่าใช้จ่ายในการพัฒนาน้อยลง และสามารถนำไปผสานรวบรวมกับบริการอื่น ๆ ใน Amazon Web Services ได้อย่างครบครัน

1.2 วัตถุประสงค์

1. เพื่อสร้างนวัตกรรมที่สามารถตอบโจทย์โครงการก่อสร้างต่างๆให้มีการติดต่อสื่อสารกันที่ง่ายรวดเร็วและมีประสิทธิภาพ และเป็นแบบแผนมากยิ่งขึ้น
2. สามารถตรวจสอบการดำเนินงานในหน้างานได้โดยที่วิศวกรระดับสูงไม่จำเป็นต้องลงพื้นที่หน้างาน
3. สามารถดูเข้าถึงข้อมูลของโครงการผ่านทางระบบออนไลน์แบบ Realtime

1.3 ขอบเขตของโครงงาน

1. บริษัทก่อสร้างที่ให้เก็บความต้องการของผู้ใช้งาน
    1.1. บริษัท Team built CO.,LTD.
    1.2. สำนักวิจัยและพัฒนาทาง กรมทางหลวง
    1.3. บริษัท ปริญสิริ จำกัด (มหาชน)
2. แอปพลิเคชันสามารถใช้ได้กับอุปกรณ์ Apple iPhone 6

1.4 ขั้นตอนการวิจัย

1. ศึกษาการใช้งาน Amazon Web Services (AWS)
    1.1. AWS Amplify
    1.2. Amazon DynamoDB
    1.3. AWS Lambda

1.4. Amazon S3 (Storage)

2. เก็บรวมรวบความต้องการของผู้ใช้งาน
3. ปรึกษากับระบบส่วนหน้าบ้านเพื่อออกแบบฐานข้อมูลเบื้องต้น
4. จัดการสร้าง Services ต่าง ๆ สำหรับเรียกใช้งาน
5. นำระบบหลังบ้านมาเชื่อมต่อกับระบบส่วนหน้าบ้าน
6. ทดสอบประสิทธิภาพของระบบ
7. สรุปผลการดำเนินงานและจัดทำเอกสารประกอบโครงการ

ตารางที่ 1.1 ขั้นตอนการดำเนินงาน

ขั้นตอนการดำเนินงาน	ปี พ.ศ. 2562				ปี พ.ศ. 2563		
	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
1. ศึกษาการใช้งาน Amazon Web Services (AWS)							
2. เก็บรวมรวบความต้องการของผู้ใช้งาน							
3. ปรึกษากับระบบส่วนหน้าบ้านเพื่อออกแบบฐานข้อมูลเบื้องต้น							
4. จัดการสร้าง Services ต่าง ๆ สำหรับเรียกใช้งาน							
5. นำระบบหลังบ้านมาเชื่อมต่อกับระบบส่วนหน้าบ้าน							
6. ทดสอบประสิทธิภาพของระบบ							
7. สรุปผลการดำเนินงานและจัดทำเอกสารประกอบโครงการ							

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ต่อผู้พัฒนาโครงงาน

1. ได้รับทักษะการออกแบบฐานข้อมูล โดยสามารถนำไปใช้ได้จริงในภาคอุตสากรรม
2. ได้พัฒนาความรู้ในด้าน API, Cloud Database and Storage และ Cloud Computing
3. ได้พัฒนาความรู้เกี่ยวกับ Tools ใหม่ ๆ ที่ใช้กันอย่างแพร่หลายในปัจจุบัน
4. ได้พัฒนาทักษะการคิด วิเคราะห์ และการทำงานเป็นทีม

ประโยชน์ที่ได้จากโครงงานที่พัฒนาขึ้น

1. เพิ่มประสิทธิภาพด้านการสื่อสารของโครงการก่อสร้าง
2. สามารถดูข้อมูลของโครงการผ่านทางระบบออนไลน์แบบประมวลแบบทันท่วงที
3. ลดปริมาณการใช้วิศวกรระดับสูงลงไปที่หน้างาน

1.6 โครงสร้างของรายงาน

บทที่ 2 จะกล่าวถึงหลักการและทฤษฎีที่เกี่ยวข้อง มีการอธิบาย Framework Library และความรู้ในด้านต่าง ๆ เพิ่มเติมที่ทางผู้พัฒนาเว็บแอปพลิเคชันใช้ในการพัฒนา เช่น JavaScript, Node.js, NoSQL Database, Serverless, AWS เป็นต้น

บทที่ 3 จะกล่าวถึงการวิเคราะห์และออกแบบระบบเริ่มตั้งแต่การเก็บรวบรวมข้อมูล การวิเคราะห์ระบบ ความต้องการเชิงหน้าที่ของระบบ การออกแบบระบบ การออกแบบฐานข้อมูล แผนภาพกิจกรรม และการออกแบบส่วนต่อประสานผู้ใช้

บทที่ 4 จะกล่าวถึงการพัฒนาระบบและการทดสอบระบบ

บทที่ 5 จะกล่าวถึงข้อสรุปปัญหาและแนวทางแก้ไข และข้อเสนอแนะ

ในส่วนของภาคผนวกจะประกอบด้วยภาคผนวก ก. แบบเสนอหัวข้อโครงงาน ภาคผนวก ข. คู่มือการใช้งาน AWS และประวัติของผู้เขียน

บทที่ 2
ความรู้พื้นฐานและทฤษฎีที่เกี่ยวข้อง

บทนี้จะกล่าวถึงหลักการและทฤษฎีที่เกี่ยวข้องในการพัฒนาแอปพลิเคชันช่วยในการตรวจสอบคุณภาพสำหรับกระบวนการก่อสร้าง (ระบบส่วนหลังบ้าน)

2.1 ซอฟต์แวร์ที่เกี่ยวข้อง

ผู้พัฒนาใช้ Editor ในการพัฒนา 2 โปรแกรม คือ

2.1.1 Visual Studio Code

Visual Studio Code หรือ VS Code เป็นโปรแกรม Code Editor ที่ใช้ในการแก้ไขและปรับแต่งโค้ด โดยบริษัท Microsoft มีการพัฒนาเป็นรูปแบบ Open Source ซึ่งสามารถนำมาใช้งานได้ฟรีเหมาะสำหรับ นักพัฒนาโปรแกรมที่ต้องการใช้งานข้ามแพลตฟอร์ม รองรับการใช้งานทั้งบน Windows, macOS และ Linux สามารถเชื่อมต่อกับ Git ได้สามารถดูการเปลี่ยนแปลงของไฟล์ได้ [8]

2.1.2 AWS Console Editor

คือ Tools ของ AWS สำหรับการแก้ไขปรับแต่งโค้ดซึ่งจะอยู่ในทุก Service ของ AWS ทำให้การใช้งานแต่ละ Service นั้นสามารถแก้ไขได้ทันที ด้วยความที่ AWS Console Editor นั้นอยู่บน Website ของ AWS ทำให้สามารถเปิดและแก้ไข้ได้จากทุก Browser และทุกแพลตฟอร์ม

2.2 JavaScript

คือภาษาคอมพิวเตอร์สำหรับการเขียนโปรแกรมอินเทอร์เน็ตที่กำลังได้รับความนิยมอย่างสูง ซึ่งเป็นภาษาเชิงวัตถุ สามารถตอบสนองผู้ใช้งานได้มากขึ้น ซึ่งมีวิธีการทำงานคือ แปลความและดำเนินงานไปทีละคำสั่ง หรือ Object Oriented Programming สามารถทำงานข้ามแพลตฟอร์มได้ โดยทำงานร่วมกับภาษา HTML และ ภาษา Java [9]

2.2.1 การใช้งาน JavaScript

ตัวแปร (Variable)

JavaScript มีระบบตัวแปรแบบ Dynamic Typing หรือชนิดตัวแปรจะเป็นชนิดใดก็ได้ สามารถเปลี่ยนได้ตามค่าที่ตัวมันเก็บเหมือนกับภาษา Python หรือ Ruby คือการไม่จำเป็นต้องประการ Type ของตัวแปร เช่น

```
var a = 5
var str = 'abcd'
var f = 1.23
var b = true
```

Var เป็นตัวแปรแบบ Global scope ซึ่งเป็นตัวแปรที่สามารถเรียกใช้ได้ทุกที่

Let เป็นตัวแปรแบบ Block scope ซึ่งเป็นตัวแปรที่สามารถเรียกใช้ได้เฉพาะฟังก์ชัน

Const เป็นตัวแปรแบบ Block scope ซึ่งเป็นตัวแปรที่ไม่สามารถเปลี่ยนค่าได้ยกเว้น เปลี่ยนค่าภายใน Object

ชุดของข้อมูล (Array)

เป็นตัวแปรพิเศษซึ่งสามารถเก็บมากกว่าหนึ่งค่าในแต่ละครั้ง เช่น

```
var arr = ["Apple","Bird","Orange"]
```

ฟังก์ชัน (Function)

ฟังก์ชัน หรือชุดคำสั่งที่รวม Statement การทำงานเอาไว้ สามารถเรียกชื่อมาใช้งานตามที่เราต้องการ

```
function functionName (parameter) {
  //statement...
  return something
}
```

ตัวแปร this

เมื่อสร้าง function ขึ้นมาแล้วทาง Js (Javavscript) จะเติมพารามิเตอร์ในตำแหน่งแรกให้เราโดยอัตโนมัติ คือ ตัวแปร this แต่ไม่จำเป็นต้องพิมพ์ลงไปในฟังก์ชันเพราะทาง Js ได้สร้างไว้ให้อยู่แล้ว และในภาษา Js คำว่า this จะอ้างถึง Context หรือ บริบทในขณะนั้น

```
function add (this,a,b) {
  return a+b
}
```

For of

เป็นการวนลูปเหมือนกับ Foreach เช่น

```
var data = [10, 20, 30, 40]
for ( item of data){
  console.log(item)
}
//output : 10 20 30 40
```

Module

โมดูลคือการยกโค้ดออกเป็นไฟล์ ๆ แล้วโหลดโค้ดเข้ามาประกอบ คล้าย ๆ กับ import ใน JAVA ซึ่งหากเรา ต้องการโหลดเพียงอย่างเดียวเข้ามาใช้ สามารถประกาศได้ด้วยการกำหนด default

```
//User.js
export default class User{
  //code..
}
//main app
import User from './User.js'
```

Promise

เป็นการเขียนโปรแกรมแบบ Asynchronous โดยจะรับ parameter เป็นฟังก์ชัน ที่ปกติจะใช้รูปแบบ (resolve,reject) => {...} ซึ่งจะสั่งรัน resolve เมื่อโค้ดทำงานเสร็จ หรือ reject เมื่อเกิด Error สำหรับ resolve จะใช้ .then แต่ถ้าเกิด reject จะใช้ .catch เช่น

```
let p = new Promise( (resolve,reject)=>{
  if (success){
    resolve(1)
  }
  else{
    reject(2)
  }
})
p.then((val)=>{
  console.log()
}).catch((val)=>{
  console.log()
})
```

2.3 Node.js

คือ Cross Platform Runtime Environment ซึ่งจะใช้ภาษา JavaScript ในการเขียนและมีการการันตี ด้วย Chrome's V8 JavaScript โดยตอนเริ่มต้นก็คือ V8 ของ Google ที่พัฒนาสามารถใช้กันได้อย่าเสรี สามารถ นำไปใช้กับงานส่วนใดก็ได้ที่สามารถรัน JavaScript ได้ และมีการนำมาทำเป็น server interpreter และได้มีภาษาที่เขียนและใช้งานบน server แบบรวดเร็วจึงเป็นที่มาของ Node.js ซึ่งการทำงานจะมีการใช้ call back ในการส่งข้อมูลไปและให้ส่งค่ากลับมา โดยที่ไม่ต้องให้โปรแกรมที่เหลือรอการทำงานจึงมีความรวดเร็วสูง [10]

2.3.1 Call back

การทำงานแบบ call back เป็น Asynchronous function ซึ่งมีความรวดเร็วกว่าแบบ Synchronous function ตัวอย่างต่อไปนี้จะเปรียบเทียบลำดับการทำงานของ Function ทั้งสองแบบ เพื่อแสดงให้เห็นกระบวนการทำงานที่ไม่ต้องรอข้อมูลเพื่อทำงานในบรรทัดถัดไปของ Asynchronous function และกระบวนการทำงานที่ต้องรอข้อมูลเพื่อทำงานในบรรทัดถัดไปของ Synchronous function

```
1| var fs = require('fs')
2| fs.readFile('input.txt', function (err,
3| if (err){
4|    return console.err(err)
5|  }
6|  console.log(data.toString())
7| })
8| console.log("Program Ended")
```

จากตัวอย่าง

บรรทัดที่ 1 เป็นการเรียก Filesystem เพื่อมาใช้งานฟังก์ชันผ่านตัวแปร fs

บรรทัดที่ 2 เป็นการอ่านค่าไฟล์ชื่อว่า input.txt โดยให้ทำการเรียก callback โดยมี parameter คือ err และ data

บรรทัดที่ 3 ตรวจสอบเงื่อนไขการใช้ callback ในกรณีที่อ่านค่าไฟล์ไม่สำเร็จให้แสดงข้อผิดพลาด

บรรทัดที่ 6 สั่งแสดงค่าที่ได้จากการอ่านไฟล์ ในกรณีที่อ่านไฟล์ได้สำเร็จ

บรรทัดที่ 8 สั่งแสดงข้อความ Program Ended

โดยลำดับการทำงานเป็นดังนี้ บรรทัดที่ 1, 2, 8, 3, 6 โปรแกรมจะทำงานต่อเนื่องโดยไม่ต้องรอให้มีการอ่านค่าไฟล์เสร็จก่อน

```
1| var fs = require('fs')
2| var content = fs.readFileSync('input.txt','utf8')
3| console.log(content)
```

บรรทัดที่ 1 เป็นการเรียก Filesystem เพื่อมาใช้งานฟังก์ชันผ่านตัวแปร fs บรรทัดที่ 2 เป็นการอ่านค่าไฟล์ชื่อว่า input.txt โดยเก็บไว้ที่ตัวแปร contents

บรรทัดที่ 3 สั่งแสดงข้อมูลจากไฟล์ input.txt โดยรอข้อมูลจาก contents โดยลำดับการทำงานเป็นดังนี้

บรรทัดที่ 1, 2, 3 โดยบรรทัดที่ 3 จะต้องรอให้มีการอ่านค่าไฟล์ input.txt จากบรรทัดที่ 2 เสร็จสิ้นก่อน

2.4 Express.js

คือ Web Application Framework บน Node.js ที่ได้รับความนิยมตัวหนึ่ง ซึ่ง Express.js จะมี module ต่าง ๆ ที่ช่วยจัดการแอปพลิเคชันให้สะดวกขึ้น เช่น การทำ Routing, Middleware การจัดการ Request และ Response เป็นต้น Express.js คือตัวกลางในการประสานและติดต่อระหว่างฝั่งหน้าบ้านและหลังบ้าน ซึ่ง Express.js จะมี method ในการใช้งานดังนี้

GET

เป็น Method สำหรับการรับค่าจากฝั่งหลังบ้านไปยังหน้าบ้าน

```
API.get(apiName: any, path: any, init: any)
```

POST

เป็น Method สำหรับการเพิ่มค่าใหม่จากฝั่งหน้าบ้านส่งไปยังฝั่งหลังบ้าน

```
API.post(apiName: any, path: any, init: any)
```

PUT

เป็น Method สำหรับการแก้ไขค่าใหม่จากฝั่งหน้าบ้านส่งไปยังฝั่งหลังบ้าน

```
API.put(apiName: any, path: any, init: any)
```

DELETE

เป็น Method สำหรับการลบค่าจากฝั่งหน้าบ้านส่งไปยังฝั่งหลังบ้าน

```
API.delete(apiName: any, path: any, init: any)
```

2.4.1 RESTful API

API (Application Programming Interface) เป็นส่วน Interface ที่เชื่อมแอปพลิเคชันฝั่งหน้าบ้านกับภายนอก เพื่อใช้ข้อมูลและเข้าถึงข้อมูล การออกแบบ API ที่ดีจะทำให้ใช้งานได้ง่ายจะทำให้ผู้พัฒนาทั้งสองฝั่งทำงานได้ราบรื่นขึ้น [11]

API Endpoint

ตัวอย่างการเขียน API สำหรับ Companies ที่จะมี API ในการจัดการ Employees

```
/getAllEmployees //เป็น API ที่จะตอบสนองเป็นรายชื่อของพนักงานทั้งหมด และ API อื่น ๆ ของ companies ก็มีดังนี้
/addNewEmployee //เพิ่มพนักงานใหม่
/updateEmployee //อัพเดตข้อมูลพนักงาน
/deleteEmployee //ลบพนักงาน 1 คน
/deleteAllEmployees //ลบพนักงานทั้งหมด
/promoteEmployee //เลื่อนขั้นพนักงาน 1 คน
/promoteAllEmployees //เลื่อนขั้นพนักงานทั้งหมด
/getAllEmployees //เป็น API ที่จะตอบสนองเป็นรายชื่อของพนักงานทั้งหมด และ API อื่น ๆ ของ companies ก็มีดังนี้
/addNewEmployee //เพิ่มพนักงานใหม่
/updateEmployee //อัพเดตข้อมูลพนักงาน
/deleteEmployee //ลบพนักงาน 1 คน
/deleteAllEmployees //ลบพนักงานทั้งหมด
/promoteEmployee //เลื่อนขั้นพนักงาน 1 คน
/promoteAllEmployees //เลื่อนขั้นพนักงานทั้งหมด
```

จากตัวอย่างข้างบนเป็นการเขียน API ที่ไม่ดี ซึ่ง API ที่ดี URL ควรจะมีแค่ resource (คำนาม) และไม่ควรจะมีคำของ action หรือ verbs (คำกิริยา) แต่ API path ที่สร้างมานี้ /addNewEmployee มีคำของ action คือ addNew ตามด้วย Resource คือ Employee

/companies เป็น API Endpoint ที่ดี และหากต้องการมี Action ใด ๆ ที่ต้องทำกับ resource ของ companies เราสามารถ ใช้ 4 methods จากข้อ 2.4 เพื่อช่วยให้ผู้พัฒนาเข้าใจตรงกันได้ เช่น

```
method GET path /companies/34 //เป็น API ที่จะตอบสนองเป็นข้อมูลของ companies ทั้งหมด
method GET path /companies/34 //เป็น API ที่จะตอบสนองเป็นข้อมูลของ companies 34
method DELETE path /companies/34 //เป็น API ที่จะลบข้อมูล companies 34
method GET path /companies/34 //เป็น API ที่จะตอบสนองเป็นข้อมูลของ companies ทั้งหมด
method GET path /companies/34 //เป็น API ที่จะตอบสนองเป็นข้อมูลของ companies 34
method DELETE path /companies/34 //เป็น API ที่จะลบข้อมูล companies 34
```

2.4.2 HTTP response status code (รหัสสถานะ)

เมื่อ client สร้าง request ไปยัง server ผ่าน API, ควรจะรู้ผลสะท้อนกลับ ไม่ว่าการ request นั้น จะ Pass, Fail หรือการ request นั้นผิด โดยที่ HTTP status จะเป็นรหัสมาตรฐานที่บอกสถานะของการ request ซึ่งจะมีหลากหลายในแต่ละสถานการณ์ ซึ่ง Server เองควรจะ return status code ให้ถูกต้อง

ต่อไปนี้จะเป็น HTTP status code ที่สำคัญโดยแบ่งออกเป็นหมวดดังนี้

2xx (หมวด Success)

เป็น status code ที่บอกว่าการ request นั้นได้รับแล้วและกระทำตาม method สำเร็จโดย Server

- **200 Ok** เป็นมาตรฐานของ HTTP response นั้น Success สำหรับ GET, PUT หรือ POST
- **201 Create** เป็น response สำหรับข้อมูลใหม่ได้ถูกสร้างขึ้น ใช้สำหรับ POST
- **204 No Content** เป็น response สำหรับ request ที่ดำเนินการ Success แต่ไม่ได้ return ข้อมูลกลับ

3xx (หมวด Redirection)

- **304 Not Modified** เป็น status code ที่บอกว่า client ได้รับการ response แล้วอยู่ใน cache และไม่จำเป็นจะต้องส่งผ่านข้อมูลเดิมอีกครั้ง

4xx (หมวด Client error)

โดยที่ status code เหล่านี้จะบอก client ว่าสิ่งที่ request มานั้น Failed

- **400 Bad Request** บอกว่า request ที่ส่งมาโดย client นั้นไม่ถูกดำเนินการ และ Server ไม่เข้าใจว่า request เกี่ยวกับอะไร
- **401 Unauthorized** บอกว่า client ไม่ได้รับอนุญาตในการเข้าถึง Resource และควรจะ request ใหม่ด้วย credential
- **403 Forbidden** บ่งบอกว่า request นั้นถูกต้องและ client ได้รับการอนุญาต แต่ Client ไม่ได้รับการอนุญาตให้เข้าถึง Resource หรือหน้าเพจด้วยเหตุผลบางประการ เช่น บางครั้ง Client ที่ได้รับอนุญาต ไม่อนุญาตให้เข้าถึงระบบไฟล์
- **404 Not Found** บ่งบอกว่า resource ที่ request มานั้น ไม่ว่างใช้งานตอนนี้

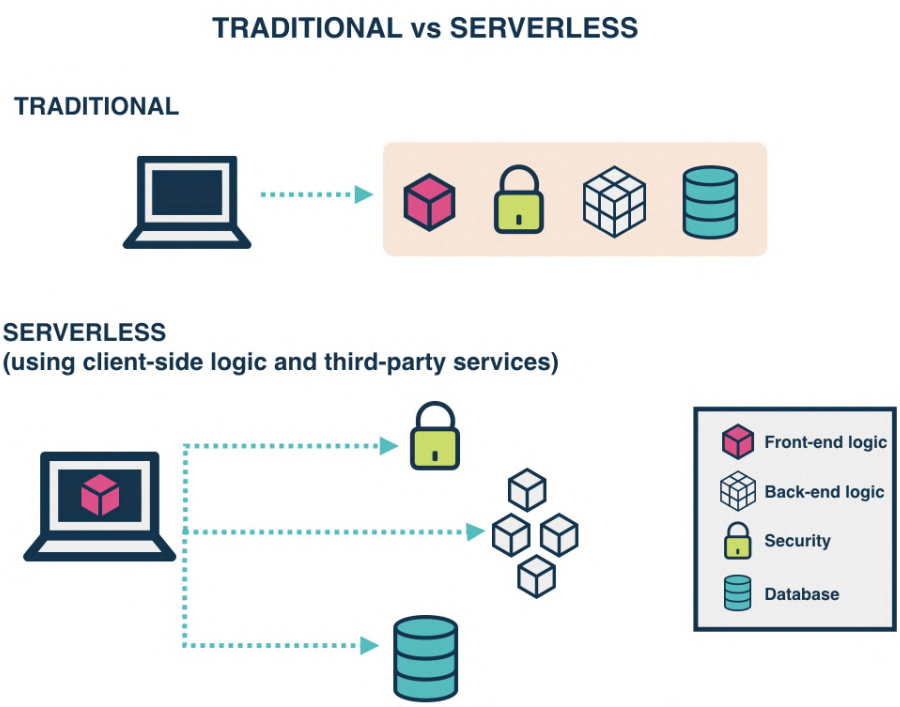
- **405 Gone** บ่งบอกว่า resource ที่ต้องการนั้นไม่มีอยู่แล้ว หรืออาจจะย้ายไปที่อื่น

5xx (หมวด Server error)

- 500 Internal Server Error บ่งบอกว่าการ request นั้นถูกต้อง แต่ server มีความสับสนและจะบริการด้วยเงื่อนไขที่คาดการไม่ได้
- 503 Service Unavailable บ่งบอกว่า server ใช้การไม่ได้ หรือไม่ว่างที่จะรับและดำเนินการ request โดยส่วนใหญ่แล้ว server อยู่ในช่วงบำรุงรักษา

2.5 Serverless

คือ การทำงาน Service ต่าง ๆ ผ่านระบบ Cloud Computing ซึ่งผู้พัฒนาไม่จำเป็นต้องซื้อ หรือเช่า Server เป็นของตัวเอง โดยทั่วไปเมื่อเราต้องการรัน Server ให้คนอื่นใช้งาน เราจะต้องเปิดเครื่องทิ้งไว้ตลอดเวลา เพื่อรอรับคนเข้ามาใช้งาน แล้วก็อาศัยประสบการณ์ สถิติ หรือจาก Monitoring เพื่อจะเปิดเครื่องที่เหมาะสมทิ้งไว้ แต่ระบบส่วนใหญ่อาจจะมี Peak hour เพียง 2–3 ชั่วโมงต่อวัน ทำให้ในช่วงเวลาปกติ เราจะพยายามไม่ให้ระบบกิน Resource เยอะนัก เพื่อรอรับ Peak hour ทำให้มี Resource Idle เพียงพอ และหากวันใดมี Load เข้ามาเยอะ และคาดเดาไม่ได้ เช่น ในระดับ 5 ถึง 10 เท่าของปกติ เราก็อาจจะต้อง Setup load balancer และสร้าง Auto scale policy เพื่อรองรับโหลดขนาดมหึมาและทำให้ Resource idle ไม่มากเกินไป ซึ่งทำให้เกิดปัญหาต่างตามมาได้ เช่น ต้องเซ็ตอัพค่าใหม่ ค่าใช้จ่ายที่เปลี่ยนแปลง รวมถึงการบำรุงรักษาเพื่อให้ระบบ Available ตลอดเวลา Serverless คือการที่เราผลักภาระการจัดการเรื่องนี้ให้บริการภายนอกดูแล และเสียค่าใช้จ่ายตามคนเข้ามาใช้จริง เพื่อที่จะขจัดการที่เราต้องเปิด Server ทิ้งไว้ตลอดเวลาได้ [12]



ภาพที่ 2.1 เปรียบเทียบการใช้ระบบ Server และ ระบบ Serverless

ที่มา: http://teerapuch.com/technology/what-serverless/

2.6 NoSQL Database

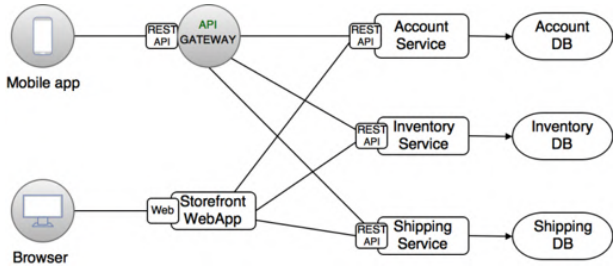
NoSQL ย่อมาจาก Not Only SQL คือเทคโนโลยีฐานข้อมูลที่ถูกออกแบบมาสำหรับการอ่าน-เขียนข้อมูลในปริมาณมากที่ SQL ไม่สามารถตอบโจทย์ได้ดีพอ ซึ่ง NoSQL มีการทำ Cache layer โดยอ่านข้อมูลผ่าน Cache แทนการอ่านจากฐานข้อมูลทำให้รับปริมาณการใช้งานได้มากขึ้น ซึ่ง NoSQL Database ได้ถูกคิดค้นขึ้นมาเพื่อแก้ปัญหาหลัก ๆ 2 อย่างที่มีใน SQL Database คือ

1) เพิ่มความสามารถในการจัดเก็บ unstructured data (หรือข้อมูลที่มีรูปแบบไม่แน่นอน)

2) เพิ่มความสามารถในการขยายระบบในรูปแบบแนวนอน (Horizontal Scalability) เพื่อรองรับปริมาณข้อมูลที่มากขึ้น [13]

2.7 Microservice Architecture

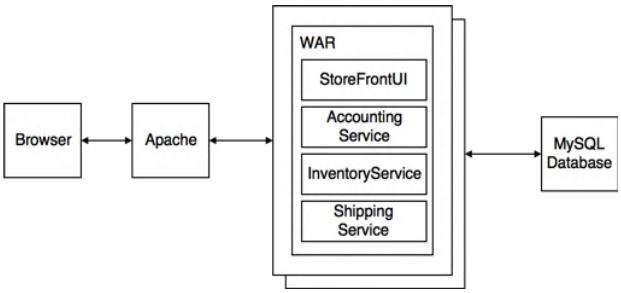
คือสถาปัตยกรรมการออกแบบ และพัฒนาซอฟต์แวร์ ระบบหรือบริการ (Service) สถาปัตยกรรมที่มีการจัดกลุ่มของแอปพลิเคชันเป็น Collection ในลักษณะ Loosely Coupled หรือเชื่อมต่อกันแบบไม่ผูกติดซึ่งกันและกันมากนัก



ภาพที่ 2.2 Microservice Architecture

ที่มา: http://www.softnix.co.th/2018/08/09/microservices-in-10-minutes/

Monolithic Architecture คือสถาปัตยกรรมการออกแบบ และพัฒนาซอฟต์แวร์ ระบบหรือบริการ (Service) ที่รวมทุกส่วนเอาไว้ในสภาพแวดล้อม (Environment) เดียวกัน ใช้ฐานข้อมูลเดียวกัน (Database)



ภาพที่ 2.3 Monolithic Architecture

ที่มา: http://www.softnix.co.th/2018/08/09/microservices-in-10-minutes/

จะเห็นได้ว่าความแตกต่างของ Microservices Architecture และ Monolithic Architecture นั้นค่อนข้างชัดเจน ซึ่งเหตุผลที่ผู้พัฒนาเลือกใช้ Microservice Architecture แทน Monolithic Architecture คือ

1) Microservices Architecture จัดการความซับซ้อนของ Application ได้ง่าย

2) Microservices Architecture สามารถเพิ่ม Services ที่จะเกิดในอนาคตได้ง่าย

3) Microservices Architecture ทำให้สามารถ Deploy ได้รวดเร็วเนื่องจากไม่ต้องรอให้ Services ต่าง ๆ เสร็จหมดก็สามารถ Deploy ได้

4) Microservices Architecture แต่ละ Services เป็นอิสระต่อกันทำให้ สามารถออกแบบให้รองรับการ Scale ได้เลย [14]

2.8 Amazon web service

2.8.1 AWS Amplify

คือแพลตฟอร์ม AWS ที่ติดตั้งที่ฝั่งหน้าบ้านที่จะคอยประสานกับฝั่งหลังบ้านที่เป็นระบบ Serverless ซึ่ง AWS Amplify สามารถเรียก Service ต่าง ๆ ที่เราสร้างในฝั่งหลังบ้าน เช่น AWS Lambda ผ่าน Amazon API Gateway, AWS Cognito, Amazon S3 (Storage)

2.8.2 Amazon DynamoDB

คือ Database แบบ NoSQL Database มีความสามารถในเรื่อง Scalability และยังสามารถ Hook event โดยใช้ AWS Lambda ซึ่ง DynamoDB สามารถจัดการเพิ่มลบแก้ไข และจัดการ Database ได้จาก Console ของทาง AWS อีกด้วย พร้อมทั้งมี DynamoDB stream ที่จะบันทึกการปรับเปลี่ยนในระดับรายการในตาราง DynamoDB อีกด้วยโครงสร้าง Amazon DynamoDB ประกอบไปด้วย

Tables

Tables ของ DynamoDB จะเหมือนกับ Database อื่น ๆ ทั่วไป

Items

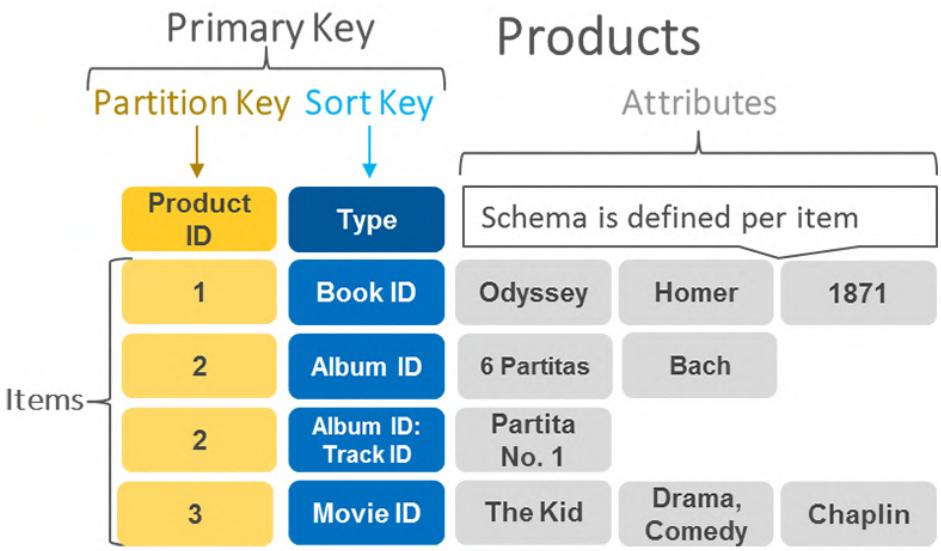
แต่ละ Table จะมี Items อยู่ข้างใน ซึ่งมีได้ตั้งแต่ 0 แถวและมากกว่า

Primary Key

แต่ละ Item จะมี Primary Key สำหรับ Query ซึ่ง Primary Key ของ DynamoDB สามารถมีได้ทั้ง 1 หรือ 2 keys โดยจะเรียกว่า Partition Key และ Sort Key

Attributes

แต่ละ Item จะมี Attributes ซึ่งอาจจะเหมือนหรือไม่เหมือนกันก็ได้ใน Table เดียวกัน [15]



ภาพที่ 2.8.2.1 รูปแบบตารางของ DynamoDB

ที่มา: http://aws.amazon.com/blogs/database/choosing-the-right-dynamodb-partition-key/

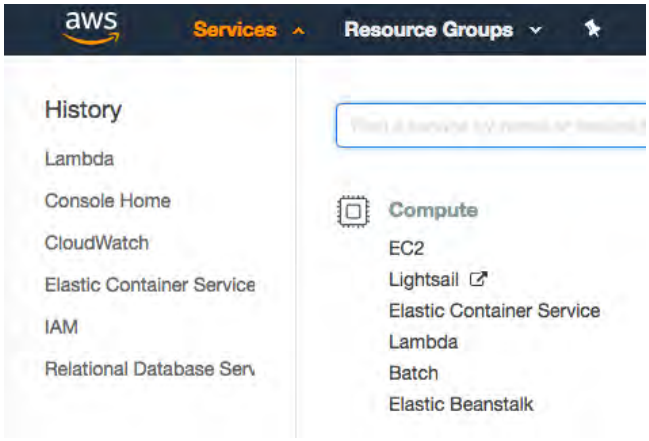
2.8.3 AWS Lambda

คือบริการด้านการทำ Serverless หรือ FaaS (Function as a Service) เราสามารถสร้าง function หรือ API ที่ต้องการใช้งานโดยไม่ต้อง Setup Server ซึ่งสามารถเขียน ได้จากทั้ง Editor ของ AWS Lambda tools (ภาพที่ 2.8.3.1.4) และ Editor ภายนอกแล้ว Upload ไปที่ AWS Lambda ซึ่ง AWS Lambda สามารถ config ค่าต่าง ๆ ได้มากมาย เช่น ระบบ Authorization หรือการตั้งค่า API Key

AWS Lambda นั้นจุดเด่นอยู่ที่ความสามารถในการเรียก Services ต่าง ๆ ของ AWS ที่มีอยู่มากมายมาใช้ได้ทั้งหมดทำให้ง่าย หากผู้พัฒนาเลือกใช้ Services ของ AWS อยู่แล้ว และค่าใช้จ่ายจะคิดตามการใช้งานจริงของผู้พัฒนา [16]

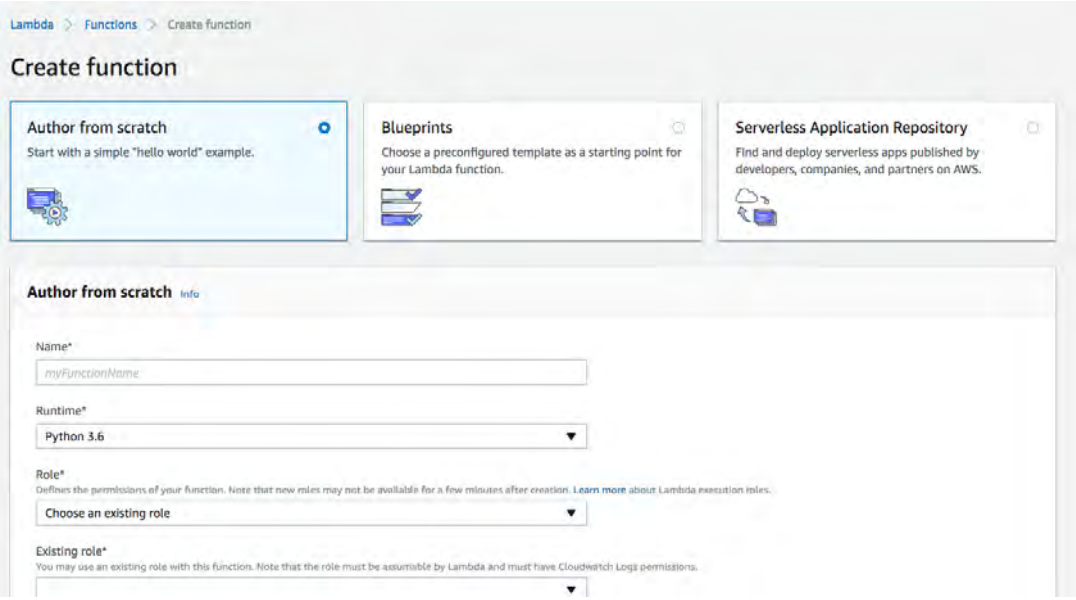
2.8.3.1 วิธีการ Create Function

เลือกที่ Menu Services > Clisk Lambda

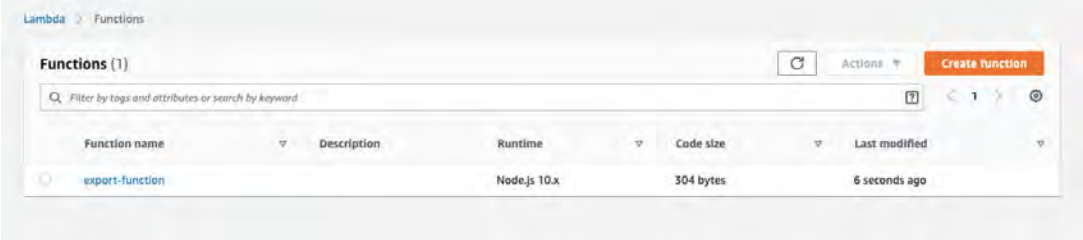


ภาพที่ 2.8.3.1.1 เข้าสู่ Lambda Service

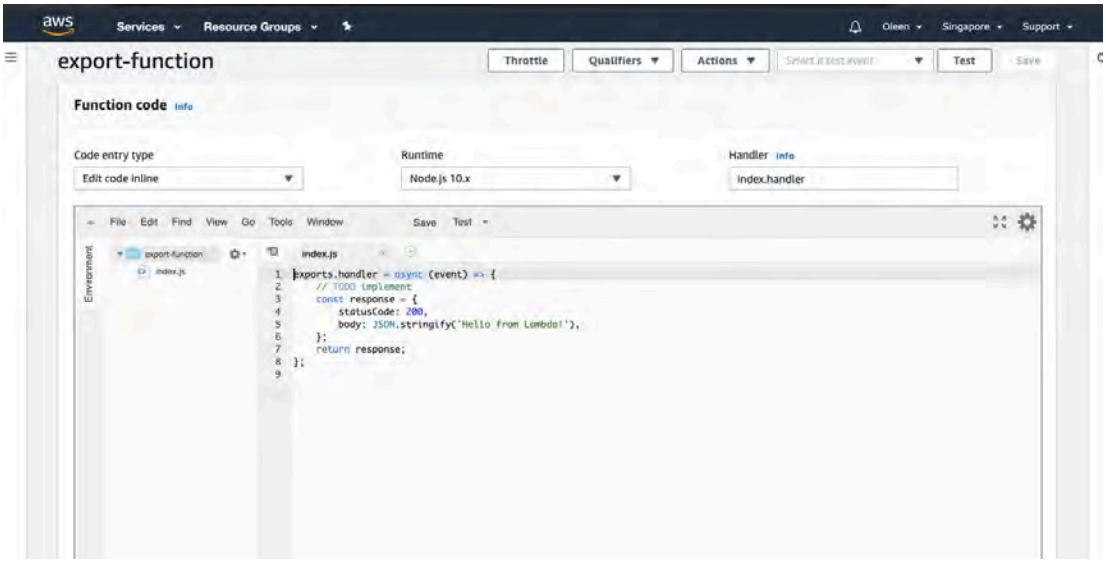
กด Create function แล้ว ตั้งชื่อ Function ของเรา จากนั้นเลือก Runtime, Role, Existing role ตามภาพที่ 2.8.3.1.2 และจะได้ Lambda function ตามภาพที่ 2.8.3.1.3



ภาพที่ 2.8.3.1.2 สร้าง Lambda function



ภาพที่ 2.8.3.1.3 Lambda function หลังจากการ create



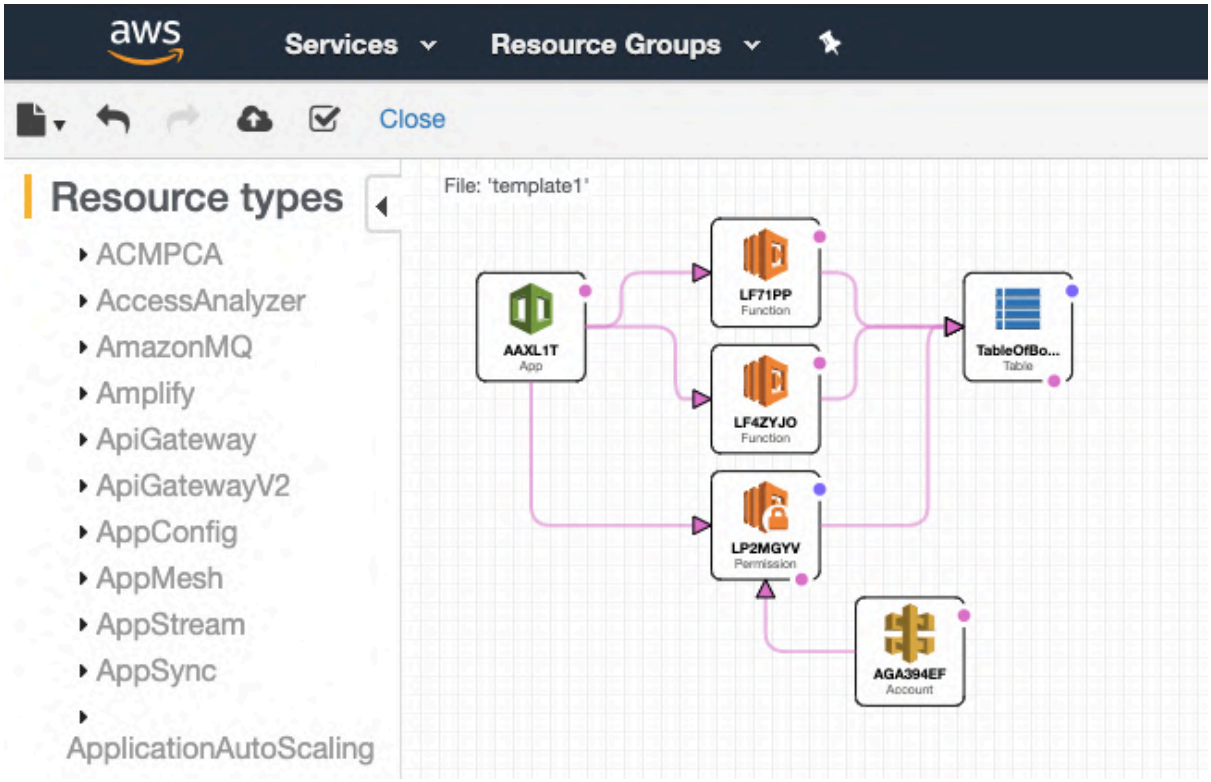
ภาพที่ 2.8.3.1.4 Lambda function editor

2.8.4 Amazon S3 (Storage)

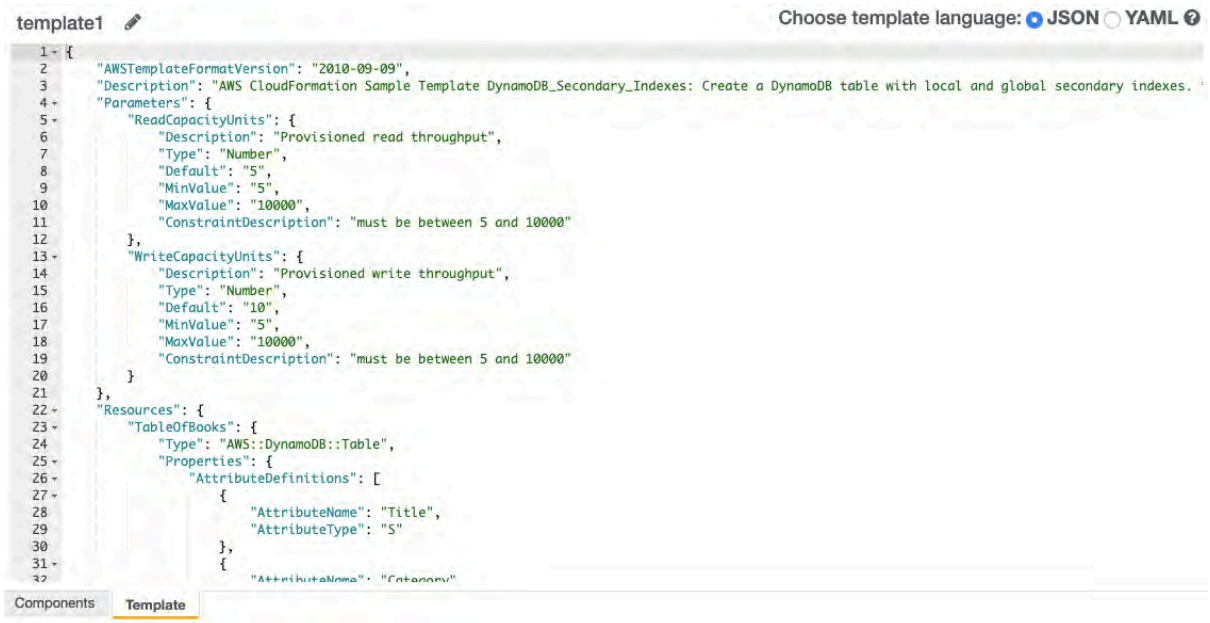
คือบริการเก็บข้อมูลแบบ Object มีชื่อเรียกอย่างเป็นทางการว่า Object storage ในการใช้งาน S3 ผู้ใช้เพียงแค่สร้าง Bucket สำหรับเก็บข้อมูลขึ้นมาการสร้าง Bucket ขึ้นมานั้นชื่อแต่ละ Bucket จะต้องห้ามซ้ำกัน เมื่อสร้าง Bucket แล้ว ผู้ใช้ถึงสามารถเก็บ Object ลงในถังตามที่ต้องการใช้งาน หลังจากการสร้าง Bucket แล้วผู้พัฒนาต้องกำหนด Permission ให้กับ Bucket ด้วยว่า ใครมีสิทธิ์เข้าถึง และสามารถทำ Action อะไรได้บ้าง

2.8.5 CloudFormation

คือ Tools สำหรับช่วยให้เราสามารถสร้าง Services ต่าง ๆ ผ่าน Design Template โดยที่เราสามารถออกแบบ Template ขึ้นมาเอง โดยเราสามารถเลือก Service ต่าง ๆ ของ AWS มาสร้างได้ดังที่แสดงดังภาพที่ 2.8.5.1 หรือสามารถเลือกใช้จาก Template ที่ AWS ได้เตรียมไว้ให้ก็ได้ ซึ่ง Template ที่ได้จะออกมาเป็น JSON ไฟล์ ดังที่แสดงในภาพที่ 2.8.5.2 ซึ่งทำให้ผู้พัฒนาออกแบบและเข้าใจระบบได้ง่ายขึ้น [17]



ภาพที่ 2.8.5.1 การออกแบบ Template เองของ AWS CloudFormation



ภาพที่ 2.8.5.2 ผลลัพธ์ JSON ไฟล์ จากการออกแบบ Template

บทที่ 3

การวิเคราะห์ระบบและออกแบบระบบ

ในบทนี้จะกล่าวถึงการเก็บรวบรวมข้อมูล, ความต้องการของผู้ใช้, การวิเคราะห์ระบบ, การออกแบบระบบโดยจะใช้ Architecture Diagram, Entity Relationship Diagram และ Sequence Diagram ในการอธิบาย และจะกล่าวถึงการออกแบบส่วนต่อประสานผู้ใช้ด้วย

3.1 การเก็บรวบรวมข้อมูล

เพื่อการศึกษาและรวบรวมความต้องการของผู้ใช้งาน
ผู้พัฒนาจึงมีการจัดและรวบรวมความต้องการของผู้ใช้งานผ่านทางการสัมภาษณ์
โดยการเดินทางไปสัมภาษณ์จากผู้ใช้งานจริง จำนวน 3 หน่วยงานตามระยะเวลาดังนี้

- 30 พฤษภาคม พ.ศ. 2562 สัมภาษณ์หน่วยงาน สำนักวิจัยและพัฒนาทาง กรมทางหลวง
- 17 ตุลาคม พ.ศ. 2562 สัมภาษณ์หน่วยงาน บริษัท Team built CO., LTD.
- 17 พฤศจิกายน พ.ศ. 2562 สัมภาษณ์หน่วยงาน บริษัท ปริญสิริ จำกัด (มหาชน)
- 19 ธันวาคม พ.ศ. 2562 สัมภาษณ์หน่วยงาน บริษัท Team built CO., LTD.
- 22 ธันวาคม พ.ศ. 2562 สัมภาษณ์หน่วยงาน บริษัท ปริญสิริ จำกัด (มหาชน)

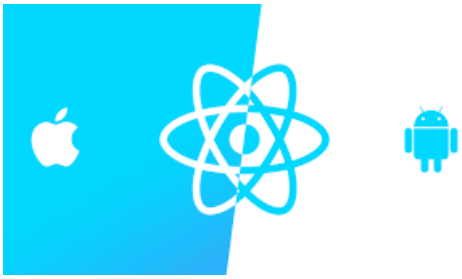
ภาพที่ 3.1.1 เก็บความต้องการที่สำนักวิจัยและพัฒนาทาง กรมทางหลวง

ภาพที่ 3.1.2 เก็บความต้องการที่ บริษัท Team built CO., LTD.

ภาพที่ 3.1.3 เก็บความต้องการที่ บริษัท ปริญสิริ จำกัด (มหาชน)

ก่อนที่จะทำสามารถวิเคราะห์และออกแบบระบบหลังบ้านได้ ผู้พัฒนาจำเป็นจะต้องสอบถามถึงการใช้งานจากผู้ใช้และทรัพยากรต่าง ๆ ได้แก่ งบประมาณทางการเงิน, ระบบปฏิบัติการอุปกรณ์สมาร์ทโฟนที่ลูกค้าใช้, เฟรมเวิร์คที่นักพัฒนาระบบหน้าบ้านเลือกใช้, ระยะเวลาการใช้งานระบบและความต้องการพิเศษ เพื่อนำข้อมูลมาประกอบการตัดสินใจในการออกแบบระบบ

ผู้ใช้หลักของซอฟแวร์นี้ได้แก่ Project Manager, Foreman และ Admin ผู้พัฒนาสามารถสรุปได้เบื้องต้นว่า ผู้ใช้งานต้องการแอปพลิเคชันสำหรับช่วยในการสื่อสารภายในองค์กรการก่อสร้าง โดยเป็นเครื่องมือช่วยตรวจสอบคุณภาพในกระบวนการก่อสร้าง โดยจะเป็นระบบที่สามารถเพิ่มหัวข้อ (Title) รายละเอียด (Description) มัลติมีเดีย (Picture) และในแต่ละงานยังสามารถเพิ่มความเห็นของปัญหาได้ (Comment) ซึ่งจะเป็นการสื่อสารเฉพาะภายในองค์กรก่อสร้าง



ภาพที่ 3.1.4 React Native Framework

ที่มา: medium.com/@sa.ruangwit/lets-code-101-เมื่อ-native-android-ต้องย้ายไปเขียน-react-native-3b7993bc51b7

โดยการพัฒนาในส่วนหน้าบ้านนั้นเลือกใช้ที่จะพัฒนาเป็นรูปแบบ “แอปพลิเคชันบนสมาร์ทโฟน” ผู้พัฒนาเลือกใช้ Framework เป็น React Native เพื่อที่จะพัฒนาได้ทั้งระบบปฏิบัติการ iOS และ Android อีกทั้งยังพัฒนาโดยภาษา Javascript และ JSX ทำให้ง่ายต่อการยืม Library หรือ Package ที่สำคัญต่อการพัฒนาได้จาก npm (Node Package Manager)

ทางทีมพัฒนาหลังบ้านได้เลือกใช้ AWS Serverless Application โดยจะวางโครงสร้างระบบหลังบ้านไว้ใน Cloud ทั้งหมด โดยจะมี ข้อดี/ข้อเสีย ดังที่กล่าวไว้ข้างต้นในบทที่ 2

เพื่อที่จะเรียนรู้การใช้งาน AWS และนำไปใช้ในอนาคตนั้น ผู้พัฒนาเห็นตรงกันว่าการนำ AWS เข้ามาใช้ในการวางระบบหลังบ้านนั้นเป็นผลดียิ่ง เพราะลูกค้ามีเจตนาที่จะใช้ระบบ Cloud อยู่แล้ว อีกทั้งค่าใช้จ่ายและค่าบำรุงรักษาจะเป็นการจ่ายตามการใช้งานจึงเหมาะอย่างยิ่งกับการนำมาทดลองใช้กับแอปพลิเคชันขนาดเล็กที่ผู้พัฒนากำลังวิจัยอยู่

3.2 ความต้องการเชิงหน้าที่ (Functional Requirement Specifications)

จากการรวบรวมความต้องการด้วยการสัมภาษณ์บุคคล, การวิเคราะห์เอกสารและการมีส่วนร่วมในการสร้างความต้องการเพื่อใช้ภายในแอปพลิเคชัน ซึ่งผู้พัฒนาสามารถเขียนเป็นความต้องการเชิงหน้าที่ (Functional Requirements: FR) โดยแสดงเป็นแผนภาพยูเคส ดังภาพที่ 3.2.1

ภาพที่ 3.2.1 แผนภาพยูสเคสแสดงหน้าที่การทำงานของระบบช่วยในการตรวจสอบคุณภาพในกระบวนการก่อสร้าง

จากแผนภาพยูสเคสแสดงหน้าที่การทำงานของระบบการสื่อสารที่ใช้ในการตรวจสอบคุณภาพของกระบวณการก่อสร้าง แบ่งผู้ใช้งานออกเป็น 3 กลุ่มดังนี้

1. ผู้ดูแลระบบซอฟแวร์ หรือ Admin Software เป็นผู้ที่กำหนดสิทธิ์และหน้าที่ให้กับผู้งานคนอื่น
2. ผู้จัดการโปรเจค หรือ Project Manager เป็นผู้จัดการโครงการก่อสร้างสังกัดบริษัทตามขอบเขตของโครงงาน คือ 2 บริษัท ได้แก่ บริษัท ปริญสิริ จำกัด (มหาชน), Team built CO., LTD และอีก 1 สำนักงานได้แก่ สำนักวิจัยและพัฒนาทาง กรมทางหลวง

3. หัวหน้าคนงาน หรือ Foreman ที่อยู่ภายในสังกัดตามบริษัท ปริญสิริ จำกัด (มหาชน), Team built CO., LTD และอนุวิศวกรที่อยู่ภายในสังกัด สำนักวิจัยและพัฒนาทาง กรมทางหลวง

จากแผนภาพดังกล่าวผู้พัฒนาสามารถรวบรวมและสรุปออกมาเป็นข้อกำหนดความต้องการด้านซอฟแวร์ในระบบหลังบ้านเบื้องต้น ดังนี้

ตารางที่ 3.1 Functional Requirement

Functional Requirement No.	Title and Description	
FR1	Title	Registration
	Description	ผู้ใช้สามารถสมัครสมาชิกเพื่อใช้งานระบบได้
FR2	Title	Login
	Description	ผู้ใช้สามารถลงชื่อเข้าใช้งานระบบได้
FR3	Title	Profile Picture
	Description	ผู้ใช้สามารถ เพิ่ม/แก้ไข รูปภาพส่วนตัวได้
FR4	Title	Verification
	Description	ผู้ใช้สามารถยืนยันตัวตนได้
FR5	Title	Push Notification
	Description	ผู้ใช้สามารถรับสารแจ้งเตือนขณะใช้แอปพลิเคชันได้
FR6	Title	Role based
	Description	การเข้าถึงข้อมูลจะขึ้นอยู่กับตำแหน่งของผู้ใช้ (Role)
FR7	Title	Get Data
	Description	ผู้ใช้สามารถเรียกข้อมูลจากเซิฟเวอร์ได้
FR8	Title	Edit Data
	Description	ผู้ใช้สามารถแก้ไขข้อมูลในเซิฟเวอร์ได้
FR9	Title	Add Data
	Description	ผู้ใช้สามารถเพิ่มข้อมูลเข้าสู่เซิฟเวอร์ได้
FR10	Title	Delete Data
	Description	ผู้ใช้สามารถลบข้อมูลออกจากเซิฟเวอร์ได้

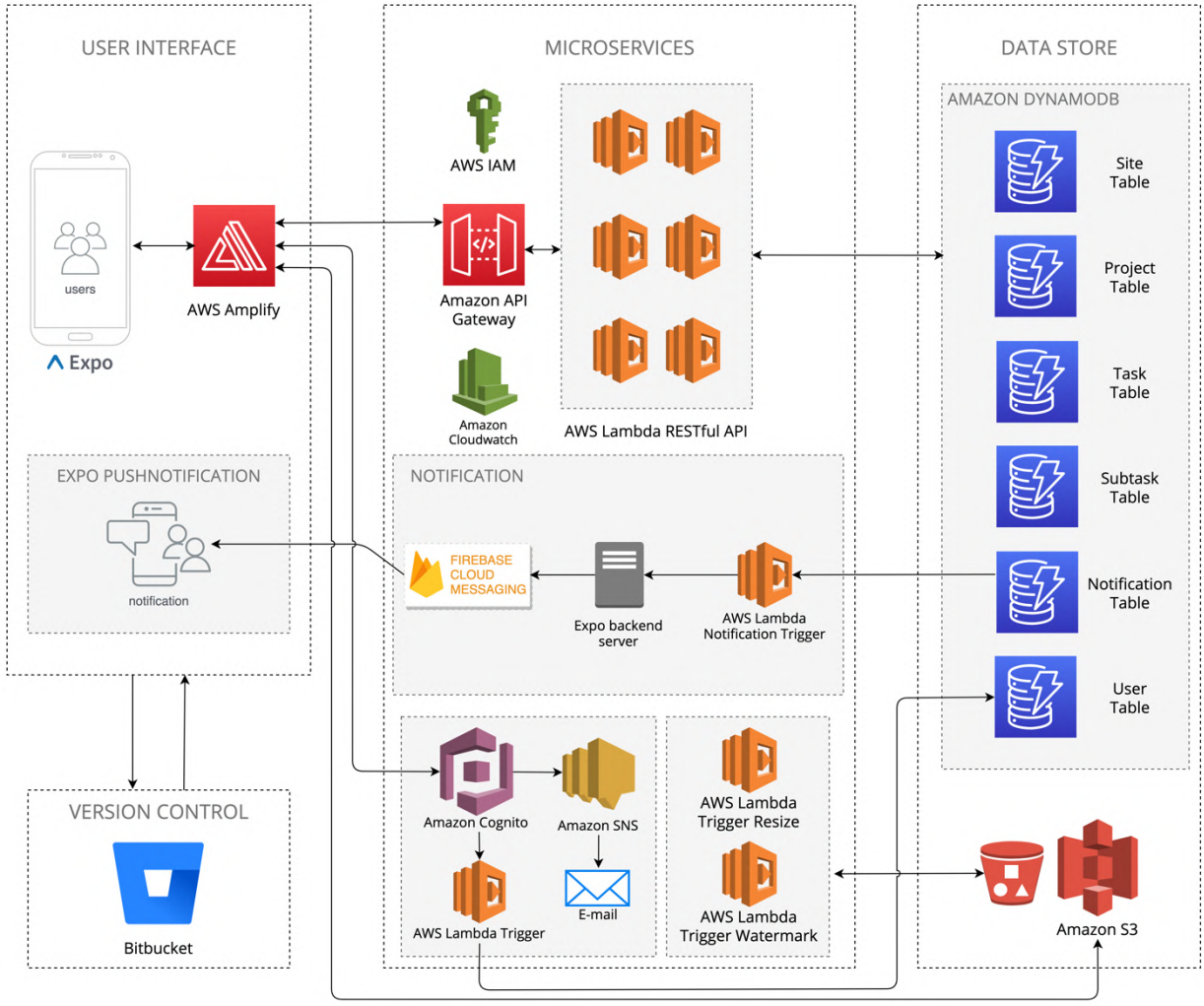
3.3 ความต้องการเชิงคุณภาพ (Functional Requirement Specifications)

ตารางที่ 3.2 Non-Functional Requirement

Non-Functional Requirement No.	Title and Description	
NFR1	Title	Timestamp Watermark
	Description	หลังผู้ใช้อัพโหลดภาพแล้ว ภาพจะมีการเขียนวันที่ทับลงไปโดยอัตโนมัติ
NFR2	Title	Resize Image
	Description	ระบบบีบภาพอัตโนมัติ เพื่อไม่ให้ภาพขนาดใหญ่จนเกินไป และช่วยให้การโหลดรูปเป็นไปได้รวดเร็ว
NFR3	Title	Monitoring Backend
	Description	สามารถตรวจสอบ Log / การทำงานทั้งหมดของส่วนหลังบ้านได้ตลอดเวลา
NFR4	Title	Storage Security
	Description	ข้อมูลและภาพจะต้องเป็นความลับและสามารถถูกเรียกได้โดยเจ้าของ และผู้เกี่ยวข้องเท่านั้น
NFR5	Title	Scalabiliy
	Description	สามารถรองรับการประมวลผลที่มากขึ้นได้ในอนาคต เมื่อมีผู้ใช้และจำนวนงานมากขึ้น
NFR6	Title	Fast Loading
	Description	ระบบสามารถประมวลผลได้รวดเร็วและตอบสนองได้ทันท่วงทีต่อการใช้งาน

3.4 การออกแบบระบบ (System Design)

ในการอธิบายโครงสร้างระบบ ผู้พัฒนาจะอธิบายโดยใช้ Architecture Diagram และตารางที่แสดงถึง Resources ทั้งหมดในเบื้องต้น



ภาพที่ 3.4.1 Overall Architecture Diagram

จากภาพที่ 3.4.1 สามารถอธิบายระบบแบ่งเป็นสามส่วนได้ดังนี้

<u>User Interface</u>

แอปพลิเคชันในส่วนหน้าบ้านพัฒนาโดยใช้ React Native Framework และใช้ Node Package ที่สำคัญในการเชื่อมต่อและใช้งานร่วมกับกับส่วนหลังบ้านดังนี้

ตารางที่ 3.3 Node Package ที่เกี่ยวข้องกับการพัฒนาระบบหลังบ้าน

ลำดับ	Node Package	เวอร์ชัน	คำอธิบาย	ฟังก์ชัน, คลาส
1	expo	36.0.0	Core package สำหรับ expo framework ทั้งหมด ถูกใช้เพื่อรันแอปพลิเคชัน	{Notification}
2	expo-cli	3.17.18	Command Line Interface สำหรับสร้าง, แก้ไขและเรียกใช้งาน expo platform	
3	expo-image-manipulator	8.0.0	เพื่อ Resize ภาพ และ Compress ภาพ เบื้องต้นก่อนจะนำส่งมาใน Amazon S3	ManipulateAsync()
4	expo-permissions	8.0.0	เพื่อขอ Permission ในการใช้งานฟังก์ชั่นของมือถือ ได้แก่ ▫ Notifications ▫ Camera Roll ▫ Calendar	getAsync(), askAsync()
5	@aws-amplify/cli	4.18.0	Command Line Interface รวมรวม toolchain สำหรับเรียกใช้งานเพื่อควบคุม environment หลังบ้านทั้งหมด โดยสามารถ pull, add, delete ทรัพยากร AWS และทดสอบบน local ก่อนจะ test และ deploy ขึ้นระบบ Cloud ได้	
6	aws-amplify	2.2.6	AWS Amplify core library	{Auth, API, Storage}
7	aws-amplify-react-native	3.2.0	AWS Amplify สำหรับแอปพลิเคชัน React Native โดยเฉพาะ	

โดยจะเชื่อมแอปพลิเคชันกับระบบหลังบ้านโดยใช้ aws-amplify เป็นหลัก โดยฟังก์ชันที่เรียกใช้ได้แก่ Auth, API, Storage

- **Auth** เพื่อลงชื่อเข้าใช้, ลงทะเบียน, ยืนยันตัวตนและเรียก Credentials, Tokens
- **API** เรียกใช้เพื่อ Request ไปที่ RESTful API Lambda Function
- **Storage** เรียกใช้เพื่ออัพโหลดรูปไปที่ S3 Bucketและสำหรับขอ URL สำหรับโหลดรูป

ในส่วนของ Push Notification ได้เชื่อมต่อกับระบบหลังบ้านด้วย expo ฟังก์ชันที่เรียกใช้ได้แก่ Notifications โดยจะวางเป็น Event Listener และผูกกับระบบปฏิบัติการ Android ด้วย FCM (Firebase Cloud Messaging)

Microservices

Services ที่ใช้ในส่วนหลังบ้านได้แก่

ตารางที่ 3.4 บริการที่เลือกมาใช้พัฒนา

ลำดับ	สัญรูป	ชื่อบริการ	คำอธิบาย
1		AWS Lambda	เรียกใช้โค้ดได้โดยไม่ต้องมีการจัดเตรียมหรือจัดการเซิร์ฟเวอร์ จ่ายเฉพาะสำหรับเวลาการประมวลผลที่ใช้
2		Amazon API Gateway	บริการที่มีการจัดการเต็มรูปแบบ ซึ่งทำให้นักพัฒนาสามารถสร้าง เผยแพร่ บำรุงรักษา เฝ้าติดตาม และรักษาความปลอดภัยของ API
3		Amazon Cognito	สามารถเพิ่มการสมัครใช้งาน การลงชื่อเข้าใช้ และการควบคุมการเข้าถึงในแอปบนเว็บและ อุปกรณ์เคลื่อนที่ได้อย่างรวดเร็วและง่ายดาย
4		Amazon Cloudwatch	บริการข้อมูลและข้อมูลเชิงลึกที่ปฏิบัติตามได้เพื่อเฝ้าสังเกตแอปพลิเคชัน ตอบสนองต่อการเปลี่ยนแปลงของประสิทธิภาพ ตลอดทั้งระบบ ปรับการใช้ทรัพยากรให้เหมาะสม และดูภาพรวมการทำงานของแอปพลิเคชัน
5		Amazon SNS	บริการส่งข้อความแบบ Pub/Sub ที่มีการจัดการอย่างเต็มที่ ซึ่งทนทาน ปลอดภัย และมีความพร้อมใช้งานสูง

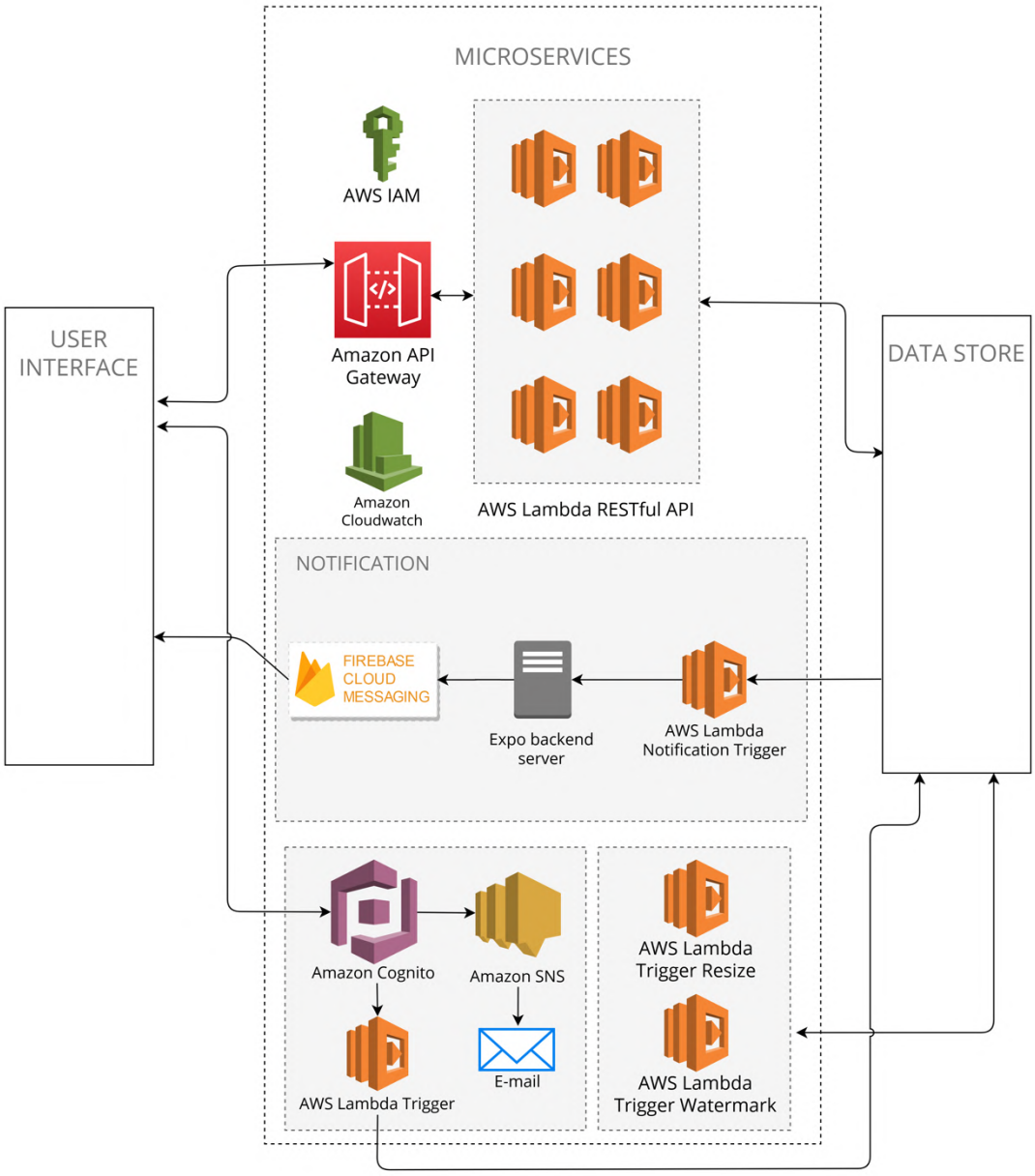
			โดยช่วยให้คุณแยกแยะไมโครเซอร์วิส ระบบแบบกระจาย และแอปพลิเคชันแบบไร้เซิร์ฟเวอร์ได้
6		AWS IAM	จัดการสิทธิ์การเข้าถึงทรัพยากรและบริการของ AWS อย่างปลอดภัย คุณสามารถสร้างและจัดการผู้ใช้และกลุ่ม AWS ได้โดยใช้ IAM และใช้สิทธิ์เพื่ออนุมัติหรือปฏิเสธไม่ให้ผู้ใช้เข้าถึงทรัพยากร AWS ได้
7		Expo Notification API	จัดการส่ง Notification ไปยัง Mobile Application ทั้งสองแพลตฟอร์ม
8		Firebase Cloud Messaging	บริการส่งข้อความแจ้งเตือนข้ามแพลตฟอร์ม

ขั้นตอนการออกแบบ Microservices จะถูกอธิบายในหัวข้อ 3.5

Data store

การเก็บข้อมูลใน Cloud ผู้พัฒนาเลือกใช้ Amazon DynamoDB และ Amazon S3 เพราะสามารถเรียกใช้ได้ง่าย สะดวก อีกทั้งยังรองรับการพัฒนาบน AWS Amplify อย่างเต็มรูปแบบ (AWS Amplify Documentation: https://aws-amplify.github.io/docs/)

3.5 การออกแบบ Microservices



ภาพที่ 3.5.1 Microservices Architecture

ในส่วนของ Microservices ผู้พัฒนาขอแบ่งเป็น 4 ส่วน ได้แก่

- RESTful API Service
- Notification Service
- Authentication Service
- Trigger Service

ผู้พัฒนาใช้ Lambda Function ในการประมวลผลทั้งหมด ซึ่งพัฒนาโดยใช้ Node.js เป็นหลัก โดยจะใช้กับทุก ๆ Service สามารถดูรายละเอียดได้ที่ตาราง 3.6

ในส่วน RESTful API จะเชื่อมต่อกับ Amazon API Gateway โดยสร้าง AWS Lambda Function เป็น Node.js เวอร์ชัน 10.x และติดตั้ง Express Framework ในทุก ๆ ฟังก์ชันให้เสร็จสิ้นโดยจะสร้างทั้งหมด 6 API และ 6 Function ดังตารางที่ 3.5 และ 3.6 และจะระบุรายละเอียด Express API อย่างระเอียดอีกครั้งในหัวข้อที่ 3.5

ตารางที่ 3.5 API Resource

ลำดับ	API Resource	Path
1	siteAPI	/sites
2	projectAPI	/projects
3	taskAPI	/tasks
4	subtaskAPI	/subtasks
5	userAPI	/users
6	notificationAPI	/notifications

ตารางที่ 3.6 Lambda Function Resource

ลำดับ	Function Resource	คำอธิบาย
1	siteFunction	Express RESTful API ที่เกี่ยวข้องกับ Site
2	projectFunction	Express RESTful API ที่เกี่ยวข้องกับ Project
3	taskFunction	Express RESTful API ที่เกี่ยวข้องกับ Task
4	subtaskFunction	Express RESTful API ที่เกี่ยวข้องกับ Subtask
5	userFunction	Express RESTful API ที่เกี่ยวข้องกับ User
6	notificationFunction	Express RESTful API ที่เกี่ยวข้องกับ Notification
7	registerTrigger	Trigger Function เมื่อผู้ใช้สมัครและยืนยันตัวตนสำเร็จ
8	watermarkTrigger	Trigger Function เมื่อมีภาพอัพโหลดเข้าไปใหม่ใน Amazon S3
9	notificationTrigger	Trigger Function เมื่อมี Notification ใหม่เข้ามาในตาราง
10	taskTrigger	Trigger Function เมื่อมีการเพิ่ม Task เข้าไปในตาราง taskTable
11	subtaskTrigger	Trigger Function เมื่อมีการเพิ่ม Subtask เข้าไปในตาราง subtaskTable

ในส่วน Notification ผู้พัฒนาเลือกใช้เป็น Expo backend server เพราะสามารถส่งข้อความแจ้งเตือนไปยังได้หลาย ๆ แพลตฟอร์มโดยแยก Firebase Cloud Messaging สำหรับ Android และ APNS สำหรับ iOS (ในการพัฒนาเบื้องต้น ผู้พัฒนาจะเน้นไปที่ Firebase Cloud Messaging)

ระบบ Authentication ทาง AWS ได้แนะนำให้ใช้ Amazon Cognito เพราะมีฐานระบบ Credentials, Tokens, Groups ให้เรียกใช้ได้อย่างสะดวก ผู้พัฒนายังเลือกใช้ Lambda Trigger เป็น Post Confirmation เพื่อสร้างฐานข้อมูลผู้ใช้ใน DynamoDB อีกด้วย ดูรายละเอียด Resource ได้ที่ตารางที่ 3.7 ดูรายละเอียดแผนภาพขั้นตอนได้ที่หัวข้อ 3.6.1 และ 3.6.2

ตารางที่ 3.7 Authentication Resource

Authentication Resource	คำอธิบาย
ultcollabAuth	Cognito Identity Pool สำหรับเก็บ Credentials ของผู้ใช้ รวมทั้ง Guest
userPoolGroups	User Pool สำหรับเก็บข้อมูลผู้ใช้ทั้งหมด รวมทั้ง Groups ได้แก่ foreman, projectmanager และ admin

การ Authentification โดยจะแบ่งผู้ใช้ออกเป็น 3 Groups ตามสิทธิ์ Credentials ของผู้ใช้นั้น ๆ ดูรายละเอียดได้ที่ตาราง 3.8

ตารางที่ 3.8 ตารางเปรียบเทียบสิทธิ์ของผู้ใช้ตาม Group

ฟังก์ชัน	ผู้ใช้งาน		
	ผู้ดูแลระบบซอฟต์แวร์	ผู้จัดการโปรเจค	หัวหน้าคนงาน
1. กำหนดหน้าที่ให้กับผู้ใช้งานอื่น ๆ	O	X	X
2. ระบบการแจ้งเตือน	O	O	O
3.สามารถเพิ่ม ลบ แก้ไข ข้อมูลภายในโครงการ	O	O	X
4.สามารถเพิ่ม ลบ แก้ไข้ ข้อมูลภายในโปรเจค	O	O	X
5.สามารถเพิ่ม ลบ แก้ไข้ ข้อมูลภายในงาน	O	O	X
6.สามารถเพิ่ม ข้อมูลภายในงานย่อย	O	O	O
7.สามารถแสดงความคิดเห็นเพิ่มเติมในงานย่อย	O	O	O

ในการ Trigger ผู้พัฒนาสร้าง Lambda Function ที่เป็น Node.js เพื่อรับ event จาก Resource ต่าง ๆ โดยจะนำข้อมูลไปประมวลผล, ส่งต่อและบันทึกค่า โดยจะมีทั้งหมด 5 Functions ด้วยกันดังที่กล่าวเบื้องต้นไปในตารางที่ 3.6 แล้ว ได้แก่

- registerTrigger
- watermarkTrigger
- notificationTrigger
- taskTrigger
- subtaskTrigger

โดย registerTrigger ถูกเรียกใช้เพื่อเพิ่มผู้ใช้เข้าไปใน Groups ของ Authentication แยกตาม projectmanager, foreman และ admin อีกทั้งยังใช้ในการสร้าง Item ใหม่ใน userTable อีกด้วย

watermarkTrigger ผู้พัฒนาออกแบบโดยใช้ Jimp - Image Processing เข้ามาช่วยในการเขียนลายน้ำวันที่ลงบนภาพก่อสร้าง โดยในส่วนนี้เป็น Requirement จากทางลูกค้า

notificationTrigger ผู้พัฒนาออกแบบโดยใช้ในเก็บ event จาก notificationTable เพื่อนำการแจ้งเตือนส่งไปถึงมือผู้ใช้

taskTrigger ผู้พัฒนาสร้างฟังก์ชันนี้มาเพื่อดัก Event การเพิ่ม Item ใหม่เข้ามาใน taskTable (มีการเพิ่มงานใหม่จากผู้ใช้) เพื่อนำไปสร้าง Notification Item ลงบน notificationTable อีกทีหนึ่ง เพื่อสร้างความสามารถ Realtime

subtaskTrigger เบื้องต้นผู้พัฒนาออกแบบให้เหมือนกับ taskTrigger แต่จะเป็นการดัก Event จาก subtaskTable แทน

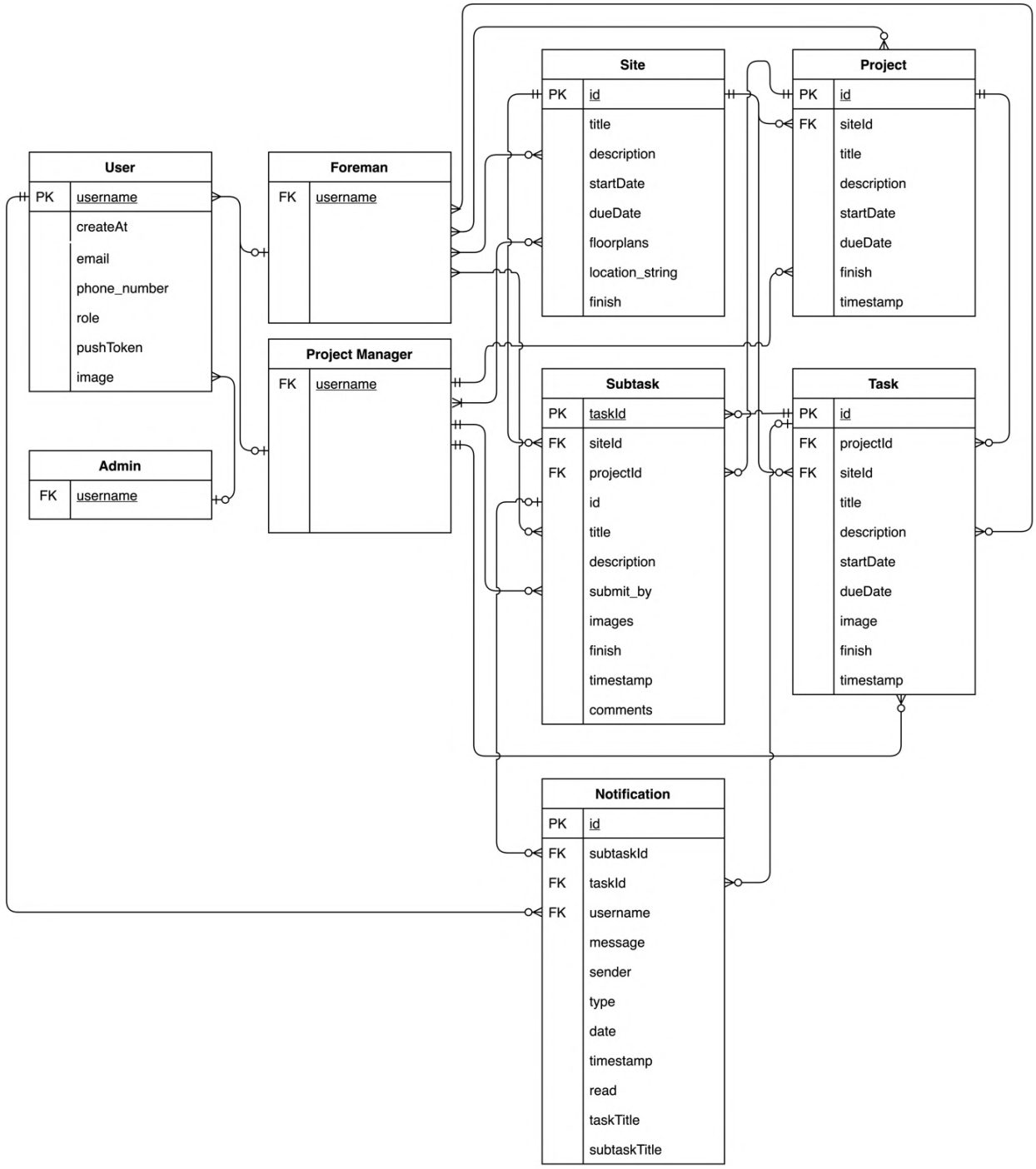
3.5 การออกแบบฐานข้อมูล, คลังข้อมูล และ API (Database, Storage and API Design)

ส่วนนี้จะอธิบายถึงการออกแบบโครงสร้างฐานข้อมูล (Schema) โดยจะใช้ Entity Relationship Diagram ในการช่วยอธิบาย และการออกแบบคลังข้อมูลกับการออกแบบ RESTful API โดยผู้พัฒนาขอเลือกใช้ตารางและ Architecture Diagram ในการอธิบาย และจะแสดงลำดับการใช้งานในหัวข้อที่ 3.6.7 ต่อไป

ก่อนจะอธิบายแบบระเอียดผู้พัฒนาขอแสดง Resource ทั้งหมดในหัวข้อนี้ ดังตารางที่ 3.5.1

ตารางที่ 3.5.1 Storage Resource

Storage Resource	คำอธิบาย
bucket	Amazon S3 Bucket สำหรับเก็บรูปภาพหรือไฟล์อื่นๆ
siteTable	DynamoDB Table สำหรับเก็บ Sites
projectTable	DynamoDB Table สำหรับเก็บ Projects
taskTable	DynamoDB Table สำหรับเก็บ Tasks
subtaskTable	DynamoDB Table สำหรับเก็บ Subtasks
userTable	DynamoDB Table สำหรับเก็บ Users
notificationTable	DynamoDB Table สำหรับเก็บ Notifications



ภาพที่ 3.5.1 แผนภาพ Entity Relationship Diagram แสดงความเชื่อมโยงของฐานข้อมูล

จากแผนภาพที่ 3.5.1 เนื่องจากฐานข้อมูลเป็นแบบ NoSQL ผู้พัฒนาจึงสามารถเขียน Schema เป็นตัวอย่าง JSON ง่าย ๆ ออกมาได้ดังตารางที่ 3.5.2

ตารางที่ 3.5.2 JSON Schema ของฐานข้อมูล

DynamoDB Table	JSON Schema Example

siteTable	{
	"description": "พรีเมี่ยมทาวน์โฮมฟังก์ชั่นบ้านเดี่ยว",
	"dueDate": "2020-12-12",
	"finish": false,
	"floorplans": [
	"sites/ดินแด17ea259d-7a67-4fbd-9aa7-25310f23953c/fp1.jpg",
	"sites/ดินแด17ea259d-7a67-4fbd-9aa7-25310f23953c/fp2.jpg",
	"sites/ดินแด17ea259d-7a67-4fbd-9aa7-25310f23953c/fp3.jpg"
	],
	"foremans": ["kuptipong", "tester2", "tester3", "tester4"],
	"id": "sitealpha2",
	"location_string": "นครปฐม",
	"projectmanagers": ["siradathb", "watcharapong"],
	"startDate": "2020-04-12",
	"title": "Foret Salaya By ปริญสิริ"
	}

projectTable	{
	"description": "Phase-01",
	"dueDate": "2020-05-8",
	"finish": false,
	"id": "9e70f01a-316c-43b1-9f53-d85ab85c9738",
	"images": [],
	"projectmanager": "siradathb",
	"siteId": "sitealpha1",
	"startDate": "2020-04-8",
	"timestamp": "1586289083824",
	"title": "Phase-01"
	}

taskTable	{ "blueprints": [],

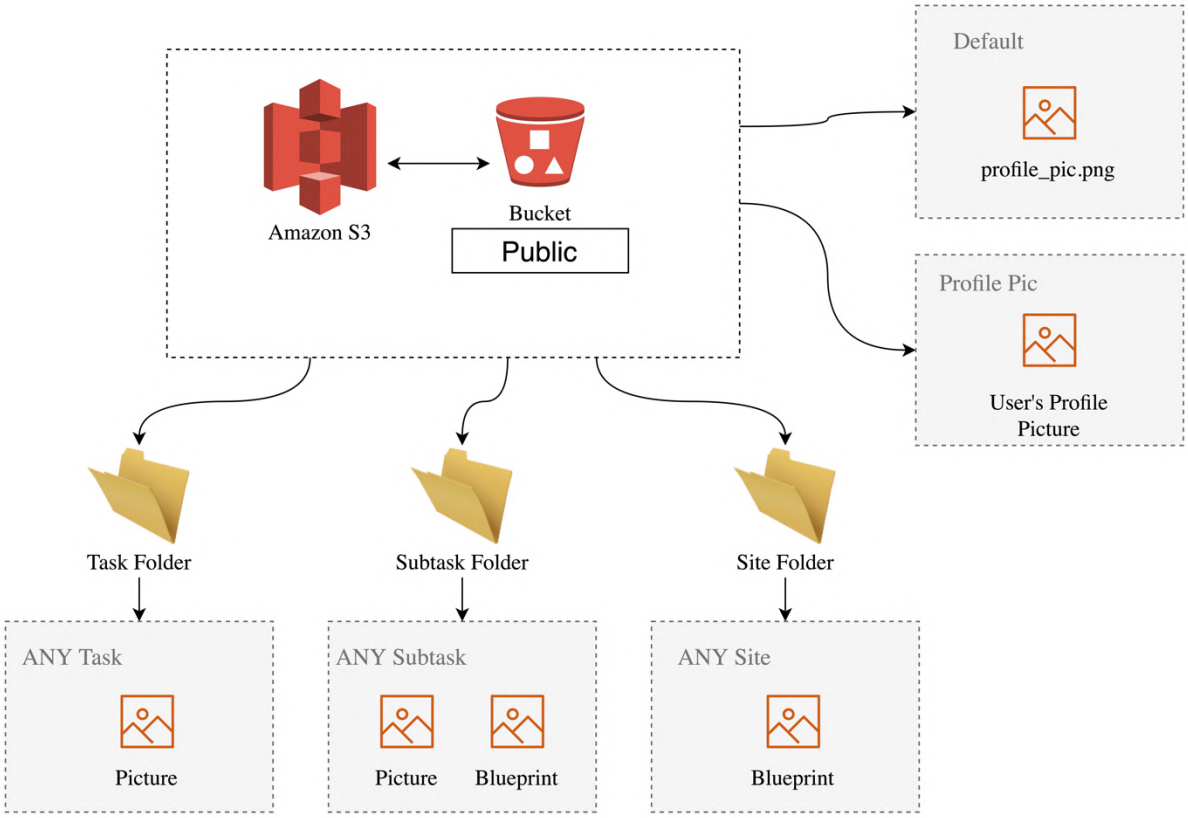
	"description": “ใช้ปูชนิด 2”, "dueDate": "2020-05-16", "finish": false, "foremans": ["kuptipong"], "id": "35684793-4b3c-4e9c-b3dd-bc4dfe064d4b", "image": "tasks/รื้อถ630adc22-1aee-402d-96d7-fb431056ce24/5b33ffe5-4636-46e6-a149-67bf127a21b7.jpeg", "projectId": "7db1f9be-0881-45b0-82fb-c45ca3173221", "projectmanager": "watcharapong", "siteId": "sitealpha2", "startDate": "2020-04-16", "timestamp": "1586990932632", "title": "รื้อถนน" }

subtaskTable	{
	"comments": [
	{
	"comment": "การดำเนินการเสร็จสิ้น 50%",
	"id": "4c3b4138-14bc-4ff7-bc27-f284233bd6eb",
	"images": [],
	"submit_by": "kuptipong",
	"timestamp": "1586010107957"
	}
	],
	"description": "จัดการพื้นที่ วางโครงสร้างชั้นรากฐาน",
	"finish": false,
	"id": "5ee4a6d8-601c-4aff-ab55-fe04ce0364d6",
	"images": [
	"subtasks/วางโค7dff5cbd-a9d6-4f98-9421-f50a769318e7/9905bd6e-ba1b-4acc-8014-332a3d581f79.jpeg",
	"subtasks/วางโค7dff5cbd-a9d6-4f98-9421-f50a769318e7/f128f641-12c0-400a-8ce3-cd5063ad89e4.jpeg"
	],

	"projectId": "7db1f9be-0881-45b0-82fb-c45ca3173221", "siteId": "sitealpha2", "submit_by": "watcharpong", "taskId": "6243ca37-dddc-4e61-b259-99bd16effa98", "timestamp": "1586009486394", "title": "วางโครง" }

userTable	{
	"createdAt": "2020-02-14",
	"email": "watcharapong.pstst@gmail.com",
	"image": "ProfilePic/watcharapong.jpeg",
	"phone_number": "+66860171331",
	"pushtoken": "ExponentPushToken[vet5WKDqit0MNTmaT79xZK]",
	"role": "projectmanager",
	"username": "watcharapong"
	}

notificationTable	{
	"date": "2020-04-4",
	"id": "b391c71c-1697-4b7a-8ded",
	"message": "watcharapong ได้เพิ่มคุณเข้าสู่ รากฐาน",
	"read": false,
	"sender": "watcharapong",
	"timestamp": "1586009421394",
	"type": "assign",
	"username": "kuptipong"
	"subtaskId": null,
	"subtaskTitle": null,
	"taskId": "kbas562x-ax32",
	"taskTitle": "รากฐาน",
	}



ภาพที่ 3.5.2 แผนภาพ Architecture Diagram ของคลังข้อมูล

จากภาพที่ 3.5.2 ผู้พัฒนาแบ่ง Amazon S3 Storage Bucket ออกเป็น 5 ส่วน ได้แก่ Site, Task, Subtask, Default, Profile Pic

- Site ใช้สำหรับเก็บ Blueprint ของ Site นั้น ๆ
- Task ใช้สำหรับเก็บภาพของ Task นั้น ๆ
- Subtask ใช้สำหรับเก็บภาพของ Subtask นั้น ๆ
- Default ใช้สำหรับเก็บข้อมูลพื้นฐาน เช่น ภาพ Default ของผู้ใช้
- Profile Pic ใช้สำหรับเก็บภาพ Profile ของผู้ใช้แต่ละคน

โดย Bucket ที่เลือกใช้นั้น ผู้พัฒนาจะใช้เป็น Public ทั้งแอปพลิเคชันเพื่อให้แต่ละผู้ใช้สามารถเห็นรูปภาพที่อัพโหลดโดยคนอื่นได้

ภาพที่ 3.5.3 แผนภาพ Architecture Diagram ของ RESTful API

API ที่สร้างขึ้นสำหรับให้นักพัฒนาหน้าบ้านนั้นเรียกใช้ดังที่ได้ถูกออกแบบมาเบื้องต้นในหัวข้อที่ 3.5 นั้น ได้แก่

- siteAPI
- projectAPI
- taskAPI
- subtaskAPI
- userAPI
- notificationAPI

ผู้พัฒนาจะขออธิบายในแต่ละ API ที่ถูกออกแบบอย่างละเอียด โดยคำนึงถึงความครอบคลุมในการเรียกใช้งานของนักพัฒนาหน้าบ้าน สามารถอธิบายได้ดังต่อไปนี้ (การเรียกใช้งานเบื้องต้นผู้พัฒนาระบุไว้แล้วในเอกสารนี้ หัวข้อที่ 3.7.4)

ในส่วนของ siteAPI นั้น ผู้พัฒนาเลือกออกแบบสำเร็จออกมาได้ดังตารางที่ 3.5.1

ตารางที่ 3.5.1 siteAPI

Path	Method	Request Body	Parameter	Response Data
/sites	GET	ไม่มี	username	ข้อมูล JSON ของทุก Site ที่ username กำลังร่วมอยู่
/sites/byid/:site_id	GET	ไม่มี	site_id	ข้อมูล JSON ของ Site
/sites/foremans/:site_id	GET	ไม่มี	site_id	ข้อมูลรายชื่อ foremans ทั้งหมดของ Site
/sites/projectmanagers/:site_id	GET	ไม่มี	site_id	ข้อมูล projectmanagers ทั้งหมดของ Site

ตารางที่ 3.5.2 projectAPI

Path	Method	Request Body	Parameter	Response Data
/projects/bysiteid/:site_id	GET	ไม่มี	site_id	ข้อมูล JSON ของทุก Project ที่อยู่ใน site_id
/projects/byid/:project_id	GET	ไม่มี	project_id	ข้อมูล JSON ของ Project ที่ระบุไอดี project_id
/projects	POST	String title String startDate String dueDate String projectmanager Array foremans String description String siteId	ไม่มี	สร้าง Project ใหม่ด้วย Body ที่กำหนด
/projects/:project_id	PUT	String title String startDate String dueDate String description	project_id	ข้อมูล JSON ที่อัพเดทแล้ว

/projects/:project_id	DELETE	ไม่มี	project_id	Successful Message

ตารางที่ 3.5.3 taskAPI

Path	Method	Request Body	Parameter	Response Data
/tasks	GET	ไม่มี	username, project_id	ข้อมูล JSON ของทุก Tasks ของ username ที่อยู่ใน project_id
/tasks/byid	GET	ไม่มี	task_id	ข้อมูล JSON ของ tasks ที่ระบุไอดี tasks_id
/tasks/foremans/:task_id	GET	ไม่มี	task_id	ข้อมูล JSON ของ foremans ที่ระบุไอดี tasks_id
/tasks	POST	Array images String title String startDate String dueDate String projectmanager Array foremans String description Array blueprints Array images String siteId	ไม่มี	Successful Message

		String projectId		
/tasks/:task_id	PUT	String title String startDate String dueDate String description	ไม่มี	ข้อมูล JSON ที่อัพเดทแล้ว
/tasks/finish/:task_id	PUT	ไม่มี	task_id	ข้อมูล JSON ที่อัพเดทแล้ว
/tasks/removeforeman/:task_id	PUT	String foreman	task_id	ข้อมูล JSON ที่อัพเดทแล้ว
/tasks/addforemans/:task_id	PUT	Array foremans	task_id	ข้อมูล JSON ที่อัพเดทแล้ว
/tasks/:task_id	DELETE	ไม่มี	task_id	Successful Message

ตารางที่ 3.5.4 subtaskAPI

Path	Method	Request Body	Parameter	Response Data
/subtasks/:task_id	GET	ไม่มี	:task_id	ข้อมูล JSON ของ subtasks ทั้งหมดที่ระบุด้วย task_id
/subtasks	POST	String title String description String submit_by String siteId String taskId String projectId Array images	ไม่มี	Successful Message
/subtasks/comment	POST	JSON subtask (JSON data of subtask)	ไม่มี	Successful Message

/subtasks	PUT	String title String description JSON subtask (JSON data of subtask)	ไม่มี	ข้อมูล JSON ที่อัพเดทแล้ว
/subtasks/comment/	PUT	String comment String submit_by Array images JSON subtask (JSON data of subtask)	ไม่มี	ข้อมูล JSON ที่อัพเดทแล้ว
/subtasks	DELETE	JSON subtask (JSON data of subtask)	ไม่มี	Successful Message

ตารางที่ 3.5.5 userAPI

Path	Method	Request Body	Parameter	Response Data
/users/:username	GET	ไม่มี	username	ข้อมูล JSON ของ username ที่ระบุ
/users	GET	ไม่มี	ไม่มี	ข้อมูล JSON ของ users ทั้งหมดในแอพ
/user/image/:username	GET	ไม่มี	username	ข้อมูล JSON image ของ username ที่ระบุ
/users/pushtoken/:username	GET	ไม่มี	username	ข้อมูล pushtoken ของ username ที่ระบุ
/users/image	PUT	String username String image	ไม่มี	ข้อมูล JSON ของผู้ใช้ที่อัพเดทแล้ว
/users/pushtoken	PUT	String username String token	ไม่มี	ข้อมูล JSON ของผู้ใช้ที่อัพเดทแล้ว

ตารางที่ 3.5.5 userAPI

Path	Method	Request Body	Parameter	Response Data
/users/:username	GET	ไม่มี	username	ข้อมูล JSON ของ username ที่ระบุ
/users	GET	ไม่มี	ไม่มี	ข้อมูล JSON ของ users ทั้งหมดในแอพ
/user/image/:username	GET	ไม่มี	username	ข้อมูล JSON image ของ username ที่ระบุ
/users/pushtoken/:username	GET	ไม่มี	username	ข้อมูล pushtoken ของ username ที่ระบุ
/users/image	PUT	String username String image	ไม่มี	ข้อมูล JSON ของผู้ใช้ที่อัพเดทแล้ว
/users/pushtoken	PUT	String username String token	ไม่มี	ข้อมูล JSON ของผู้ใช้ที่อัพเดทแล้ว

ตารางที่ 3.5.6 notificationAPI

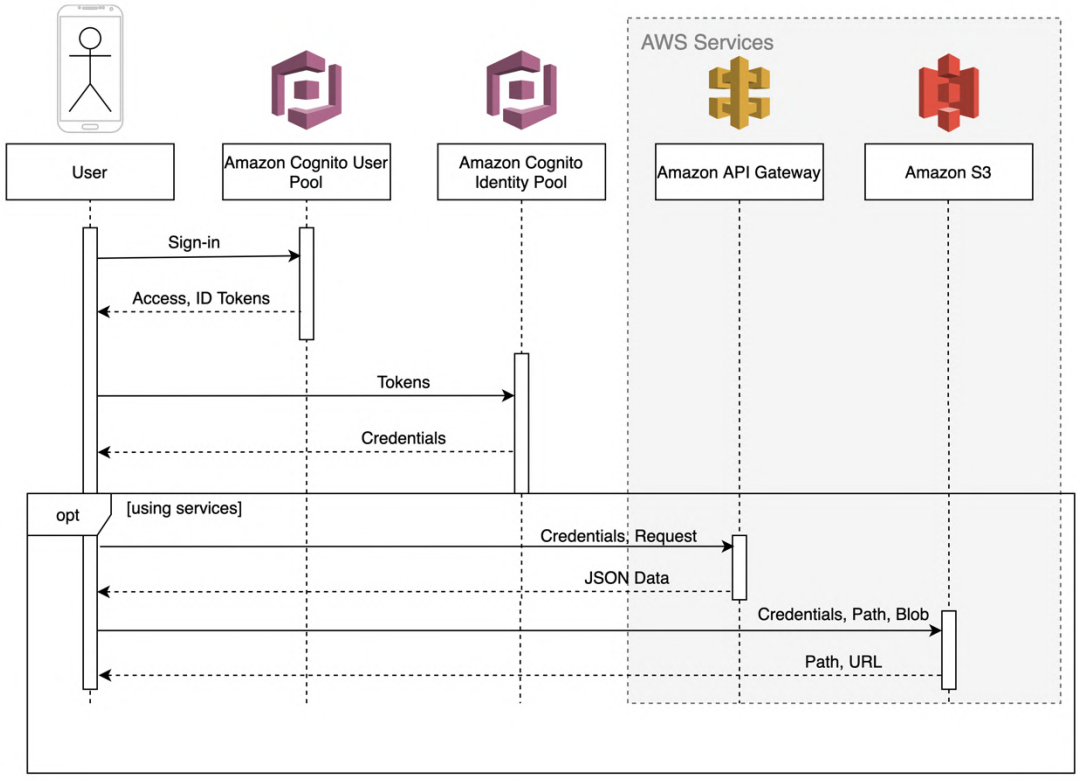
Path	Method	Request Body	Parameter	Response Data
/notifications	GET	ไม่มี	username	ข้อมูล JSON ของ Notification ที่ระบุด้วย username
/notifications/new	GET	ไม่มี	username	ข้อมูล JSON ของ Notification ที่ระบุด้วย username และยังไม่ได้ถูกอ่าน
/notifications/byid	GET	ไม่มี	notification_id	ข้อมูล JSON ของ Notification

				ที่ระบุด้วย notification_id
/notifications/setread	PUT	ไม่มี	notification_id	ข้อมูล JSON ของผู้ใช้ที่อัพเดทแล้ว

3.6 แผนภาพแสดงลำดับการทำงาน (Sequence Diagram)

ส่วนนี้จะอธิบายถึงการทำงานของ Microservices กับผู้ใช้โดยสังเขป โดยจะใช้ Sequence Diagram ในการช่วยอธิบาย

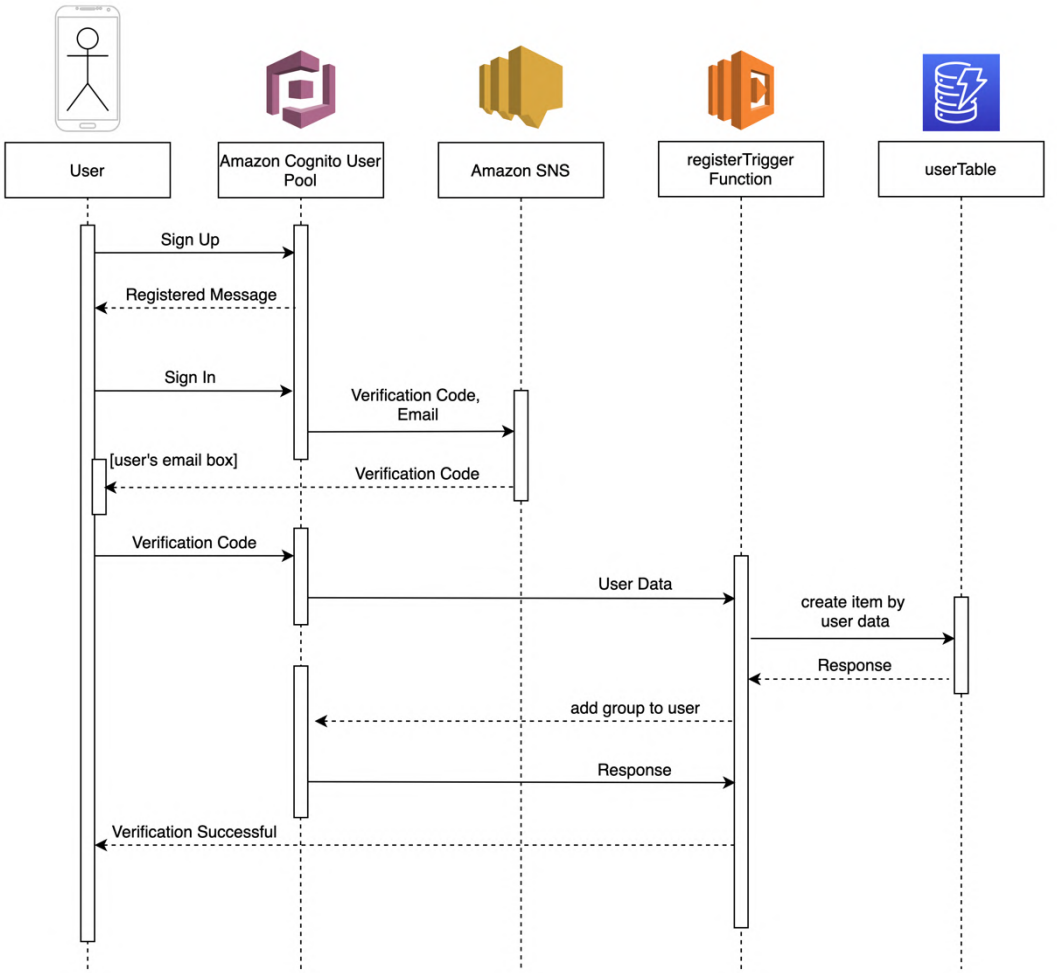
3.6.1 ระบบลงชื่อเข้าใช้



ภาพที่ 3.6.1.1 แผนภาพขั้นตอนการลงชื่อเข้าใช้

จากภาพที่ 3.6.1.1 ผู้ใช้ลงชื่อเข้าใช้โดยการกรอกชื่อผู้ใช้ (Username) และรหัสผ่าน (Password) เพื่อรับ Access และ ID Tokens มาจาก Amazon Cognito User Pool เมื่อได้รับ Tokens มาแล้วผู้ใช้สามารถนำ Tokens เพื่อไปเช็ค Credentials (ว่าเราอยู่ Group อะไร มีสิทธิ์อะไรบ้าง) จากนั้นระบบหลังบ้านจะจดจำ Tokens ของอุปกรณ์มือถือของผู้ใช้เป็นเวลา 30 วัน ในระหว่างนี้ผู้ใช้สามารถนำ Credentials ของตนเองไปเรียกใช้ Services ต่าง ๆ ของ AWS ได้อย่างเต็มที่

3.6.2 ระบบสมัครสมาชิกและยืนยันตัวตน



ภาพที่ 3.6.2.1 แผนภาพขั้นตอนการสมัครสมาชิกและยืนยันตัวตน

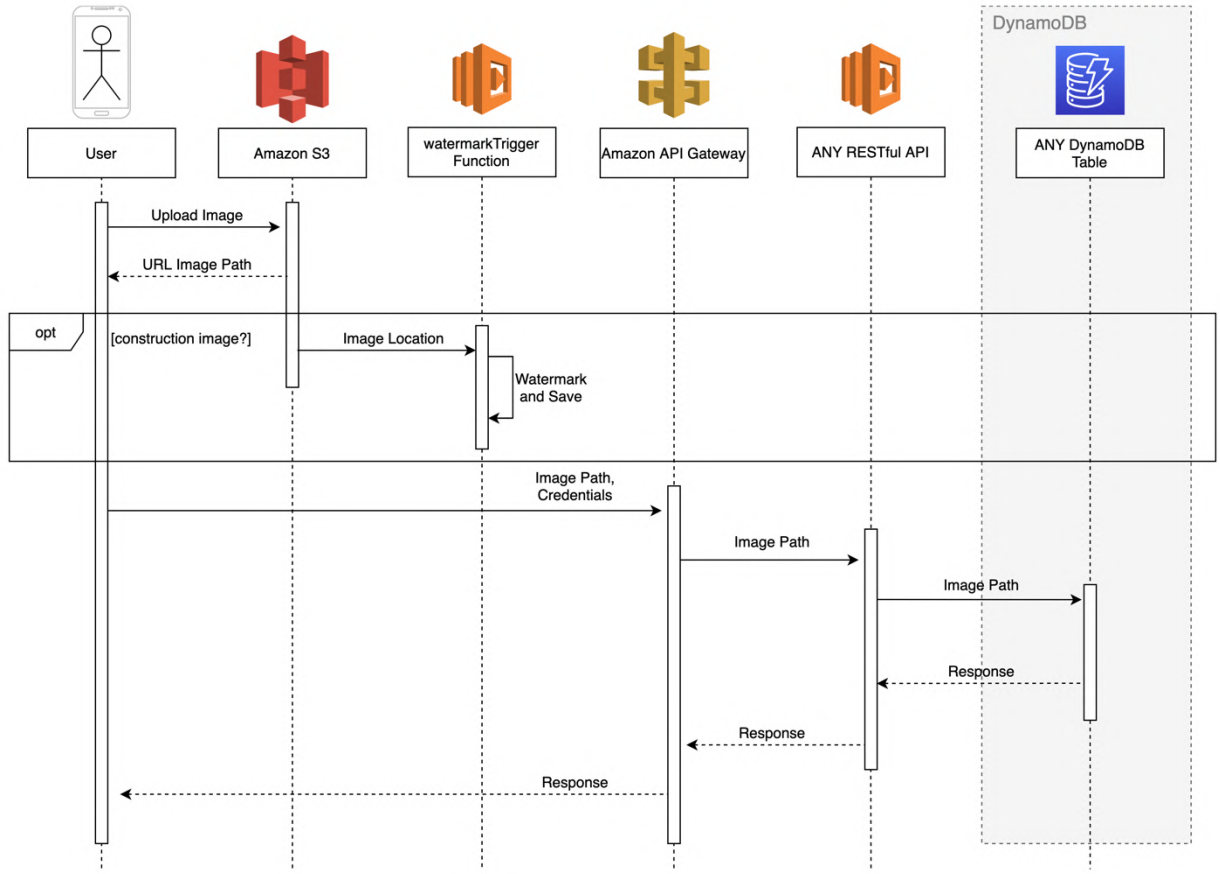
จากภาพที่ 3.6.2.1 ผู้ใช้สมัครสมาชิกโดยกรอกข้อมูลพื้นฐาน ได้แก่ ชื่อผู้ใช้ (Username), รหัสผ่าน (Password), อีเมล (Email), เบอร์โทรศัพท์ (Phone Number), ตำแหน่ง (Role) ส่งไปยัง Amazon Cognito User Pool เพื่อลงทะเบียนและส่งข้อความย้อนกลับมาหากลงทะเบียนสำเร็จ

ผู้ใช้นำชื่อผู้ใช้และรหัสผ่านลงชื่อเข้าใช้ไปยัง User Pool เช่นดังภาพที่ 3.6.1.1 แต่หากผู้ใช้เพิ่งสมัครสมาชิก จำเป็นจะต้องยืนยันตัวตนโดยการนำ Verification Code ที่ระบบจะส่งจาก Amazon SNS ไปยัง Email Box มากรอกที่หน้าแอปพลิเคชัน

หลังจากที่ผู้ใช้ยืนยันตัวตนสำเร็จแล้ว (Post Confirmation) ระบบจะ Trigger ด้วย registerTrigger เพื่อสร้าง Item ลงบน userTable และเพิ่มผู้ใช้ลงบน Group ที่สมัครมาตั้งแต่เริ่มลงบน User Pool อีกด้วย

หากการกระทำทั้งหมดเสร็จสมบูรณ์ไปได้ด้วยดี ระบบจะส่งข้อความ Verification Successful Message กลับไปยังหน้าจอแอปพลิเคชันของผู้ใช้ เป็นอันจบการสมัครสมาชิก

3.6.3 ระบบอัพโหลดรูปภาพ

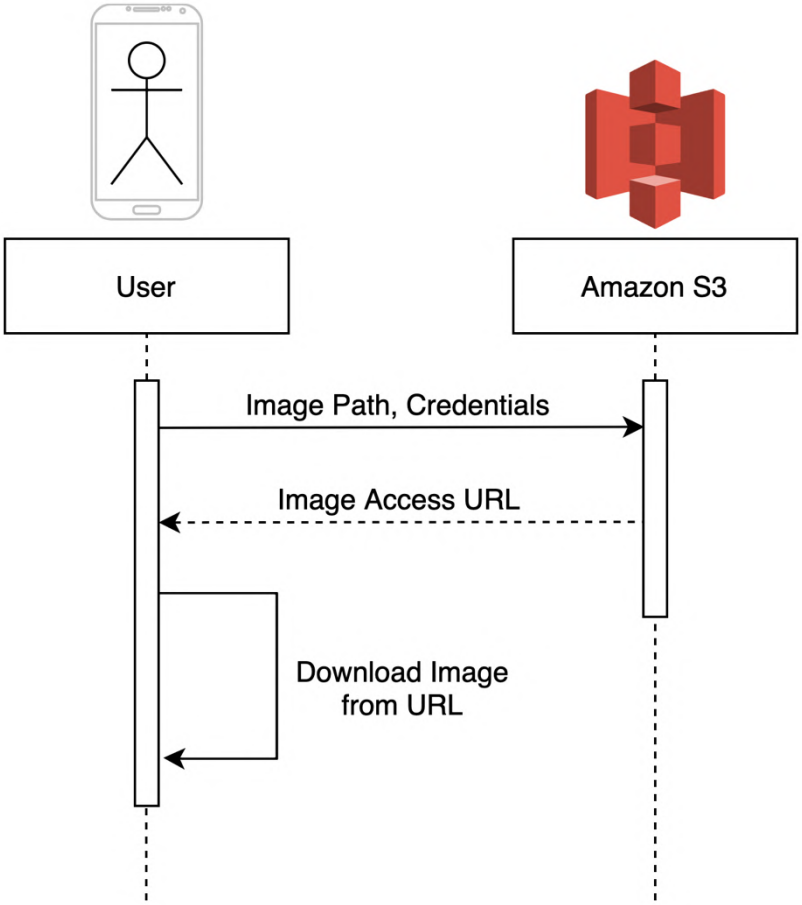


ภาพที่ 3.6.3.1 แผนภาพขั้นตอนการอัพโหลดรูปภาพ

จากภาพที่ 3.6.3.1 ผู้ใช้ส่งอัพโหลดภาพไปยัง Amazon S3 และได้รับ Image Path กลับมา หากผู้ใช้อัพโหลดรูปภาพที่เป็นภาพชนิดก่อสร้าง Amazon S3 จะ Trigger ไปยัง watermarkTrigger เพื่อ Image Processing เติมลายน้ำเป็นวันที่ที่อัพโหลดรูปนั้นลงไป จากนั้นจะบันทึกภาพทับกลับเข้าที่เดิม

จากนั้นผู้ใช้นำ Image Path และ Credentials ส่ง Method PUT, POST ไปยัง API Gateway และส่งต่อไปยัง RESTful API เพื่ออัพเดท, เขียนใหม่ในตารางใดๆของ DynamoDB สามารถดูข้อมูลเชิงลึกได้ที่หัวข้อ 3.5

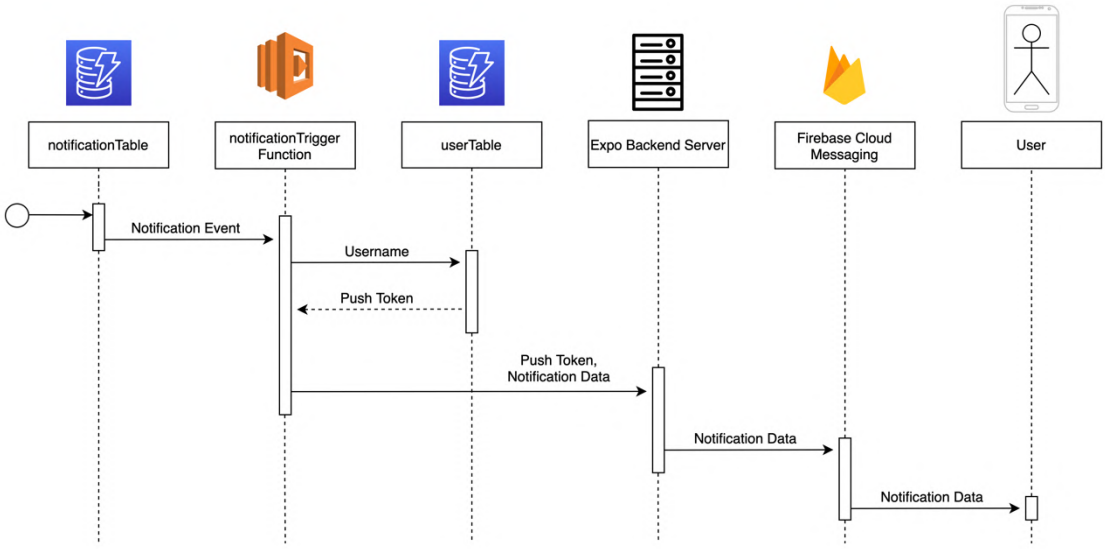
3.6.4 ระบบดาวน์โหลดรูปภาพ



ภาพที่ 3.6.4.1 แผนภาพขั้นตอนการดาวน์โหลดรูปภาพ

จากภาพที่ 3.6.4.1 ผู้ใช้ส่ง Image Path และ Credentials ไปยัง Amazon S3 เพื่อรับ Access URL (มีเวลาหมดอายุ) จากนั้นเมื่อจะแสดงผลภาพนั้นๆก็นำ URL ไปดาวน์โหลดและแสดงผลได้ทันที ยกตัวอย่างเช่น JSX <Image source={{ uri: URL}}/>

3.6.5 ระบบแจ้งเตือน



ภาพที่ 3.6.5.1 แผนภาพขั้นตอนการไหลของข้อมูลแจ้งเตือน

เมื่อมีข้อมูล Notification ใหม่เข้ามา จะถูกบันทึกลง notificationTable แล้ว notificationTrigger จะทำงานทันทีโดยข้อมูลที่นำเข้าจะเป็น Event ของ Notification เพื่อนำไปเขียน JSON Notification Data สำหรับส่งไปยังผู้ใช้ที่ระบุอีกที โดยจะ Query Push Token ของผู้ใช้นั้น ๆ ออกมาจาก userTable และ HTTPS Notification ไปยัง Expo Backend Server โดย Body เป็น Push Token และ Notification Data ที่สร้างขึ้นมา โดย Notification Data ออกแบบด้วย JSON ดังภาพที่ 3.6.5.2 และ Expo Backend Server จะนำ Notification Data ส่งไปยัง FCM เพื่อส่งต่อไปยังแอปพลิเคชันของผู้ใช้ที่ตรงกับ Push Token ตามลำดับ

ผู้พัฒนาออกแบบ type ของ Notification ในเบื้องต้นไว้ทั้งหมด 2 ชนิด ได้แก่

- assign คือ การแจ้งเตือนเมื่อผู้ใช้ได้ถูกรับมอบหมายงาน
- comment คือ การแจ้งเตือนเมื่อผู้ใช้ได้รับ comment ใน subtask ของตัวเอง

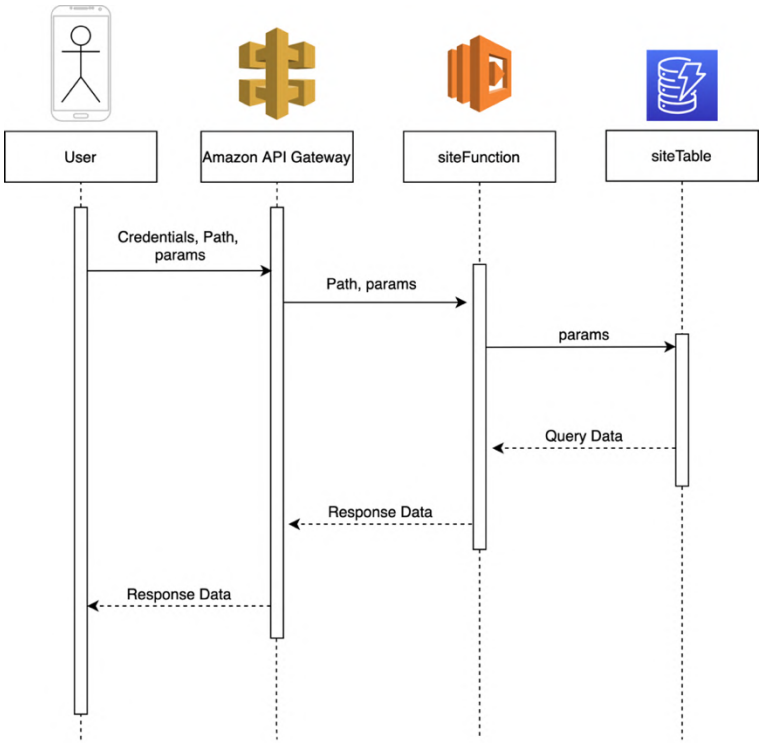
โดยในส่วนของการสมัคร Push Token ผู้พัฒนาจะขอกล่าวถึงในหัวข้อถัดไปที่ 3.6.7.5

```
let noti_body = {
    to: 'pushToken',
    body: 'ข้อความ',
    title: 'หัวข้อ',
    data: {
      type: 'ชนิดของ Notification',
      sender:  'Username ของผู้ส่ง',
      taskId: 'taskId',
      subtaskId: 'subtaskId',
    },
  }
```

ภาพที่ 3.6.5.2 ตัวอย่าง JSON Body ของ Notification Data

3.6.7 ระบบ RESTful API

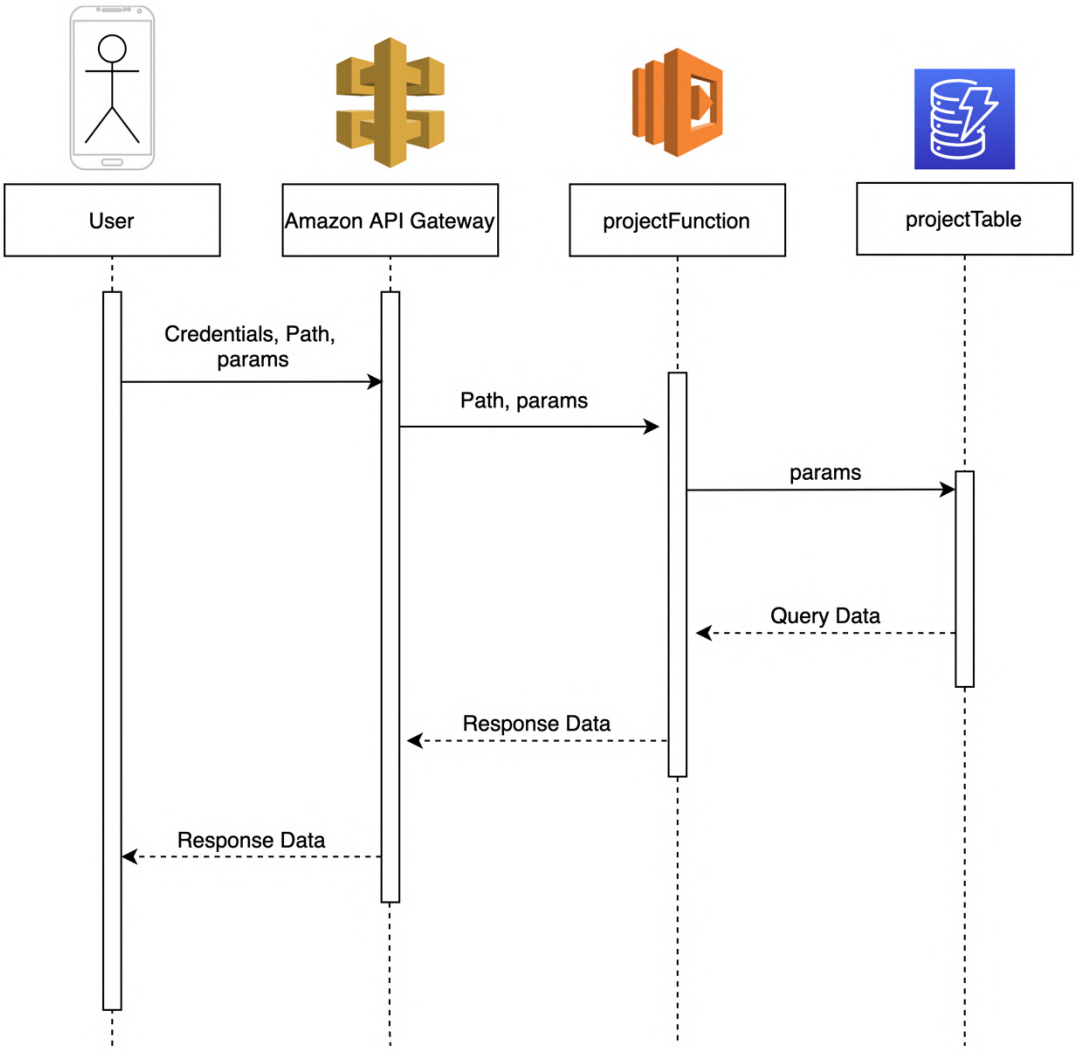
3.6.7.1 siteAPI



ภาพที่ 3.6.7.1.1 แผนภาพขั้นตอนการเรียก siteAPI ด้วย Method GET

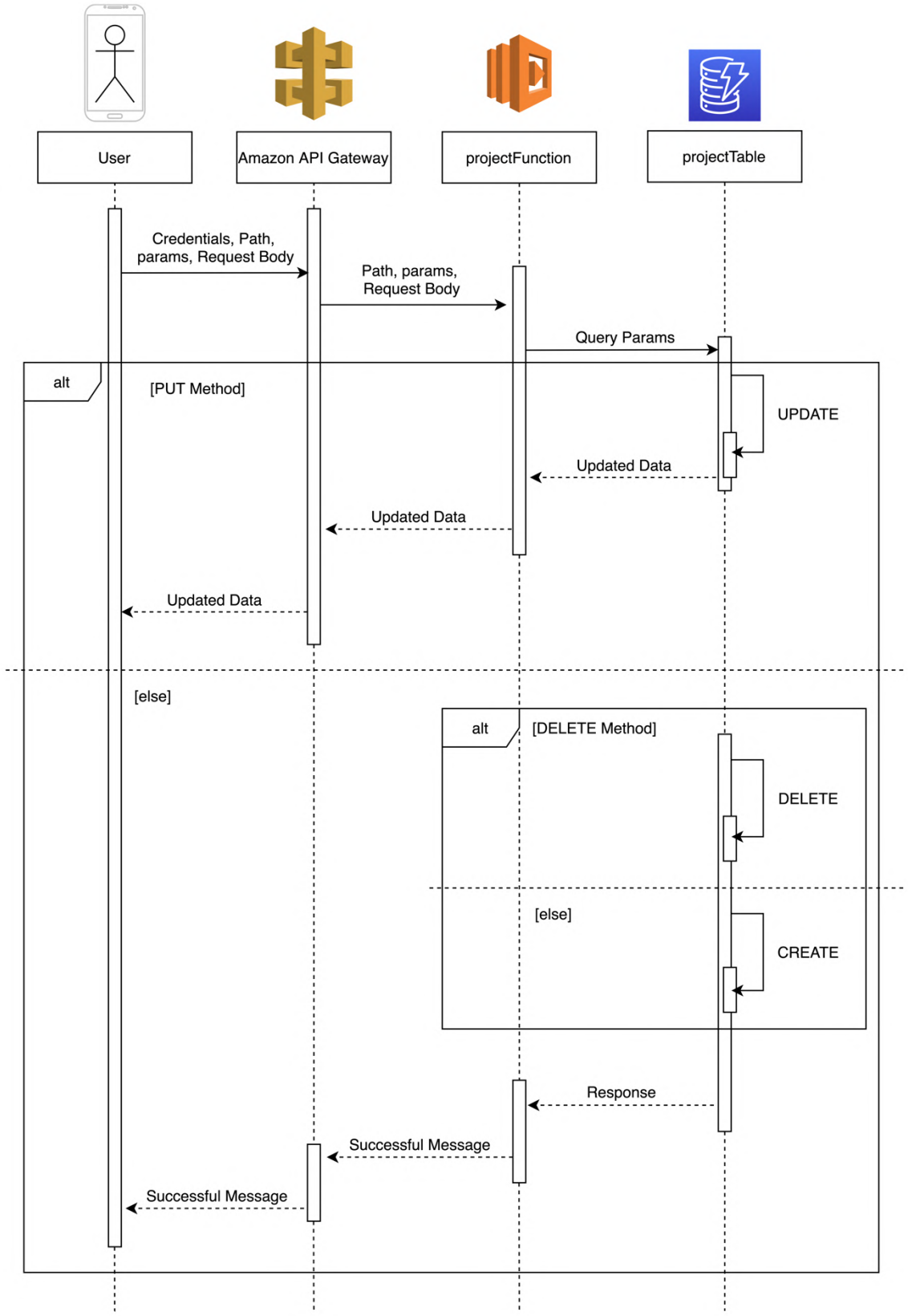
เมื่อส่ง Path, params และ Credentials มาที่ API Gateway แล้ว ระบบจะไปเรียกใช้ siteFunction โดย Route ไปที่ Path ด้วย Method GET เพื่อรันโค้ด Query DynamoDB จาก siteTable จากนั้นจะส่ง Query Data คืนมายัง siteFunction และ Response ด้วย Data ที่ผู้ใช้ร้องขอ

3.6.7.2 projectAPI



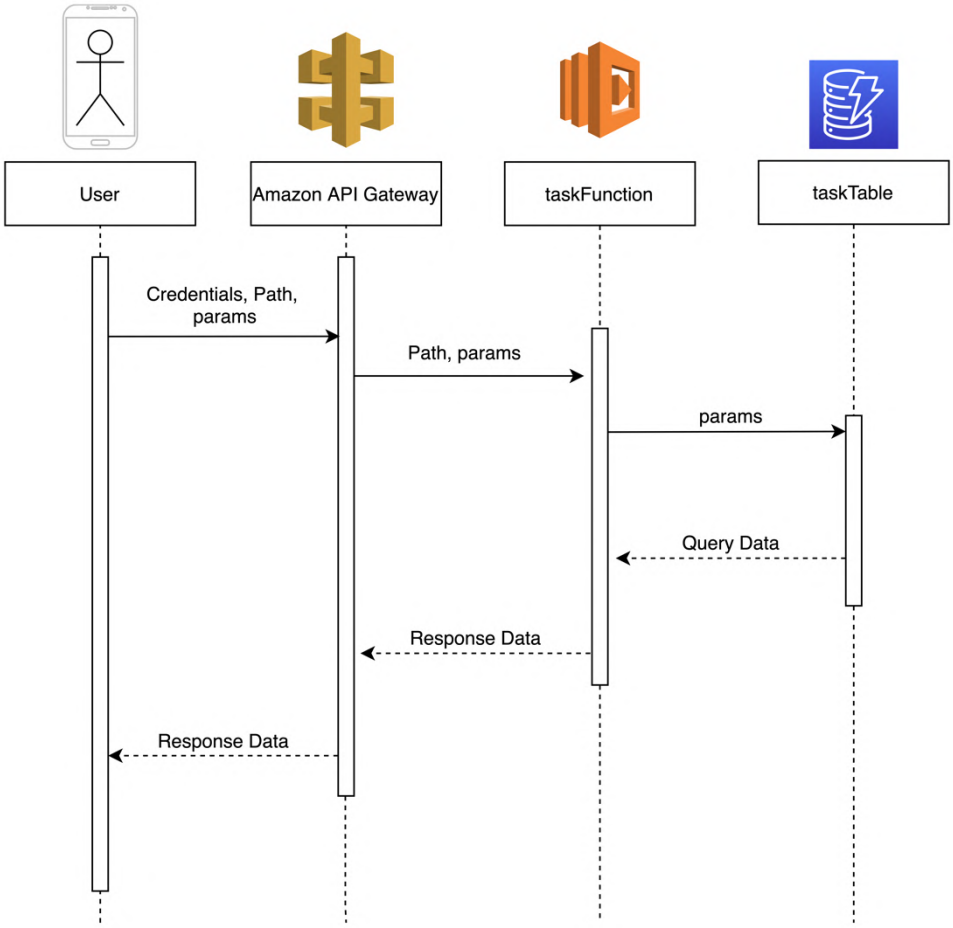
ภาพที่ 3.6.7.2.1 แผนภาพขั้นตอนการเรียก projectAPI ด้วย Method GET

เมื่อส่ง Path, params และ Credentials มาที่ API Gateway แล้ว ระบบจะไปเรียกใช้ projectFunction โดย Route ไปที่ Path ด้วย Method GET เพื่อรันโค้ด Query DynamoDB จาก projectTable จากนั้นจะส่ง Query Data คืนมายัง projectFunction และ Response ด้วย Data ที่ผู้ใช้ร้องขอ



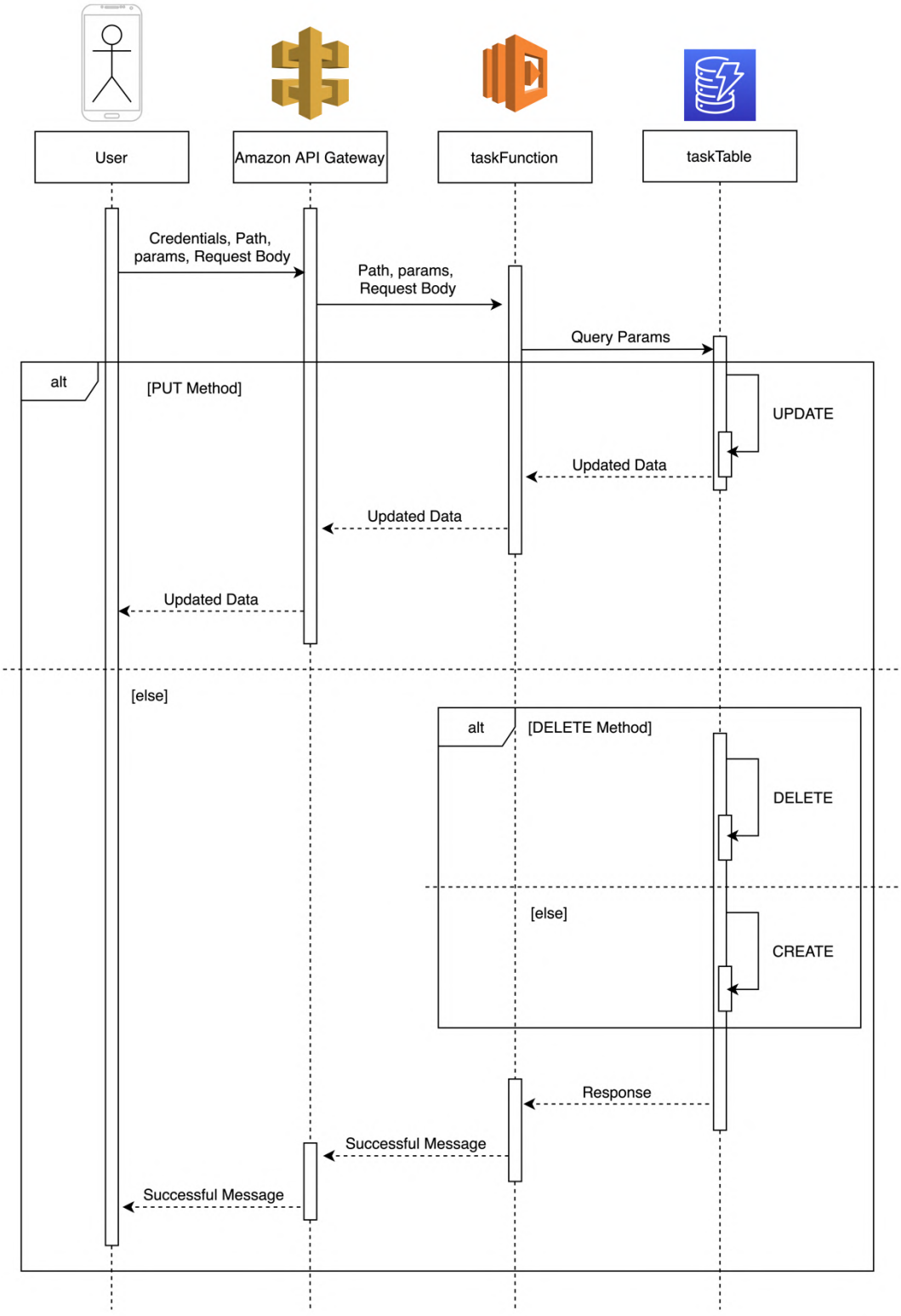
ภาพที่ 3.6.7.3.1 แผนภาพขั้นตอนการเรียก projectAPI ด้วย Method POST, PUT, DELETE

3.6.7.3 taskAPI



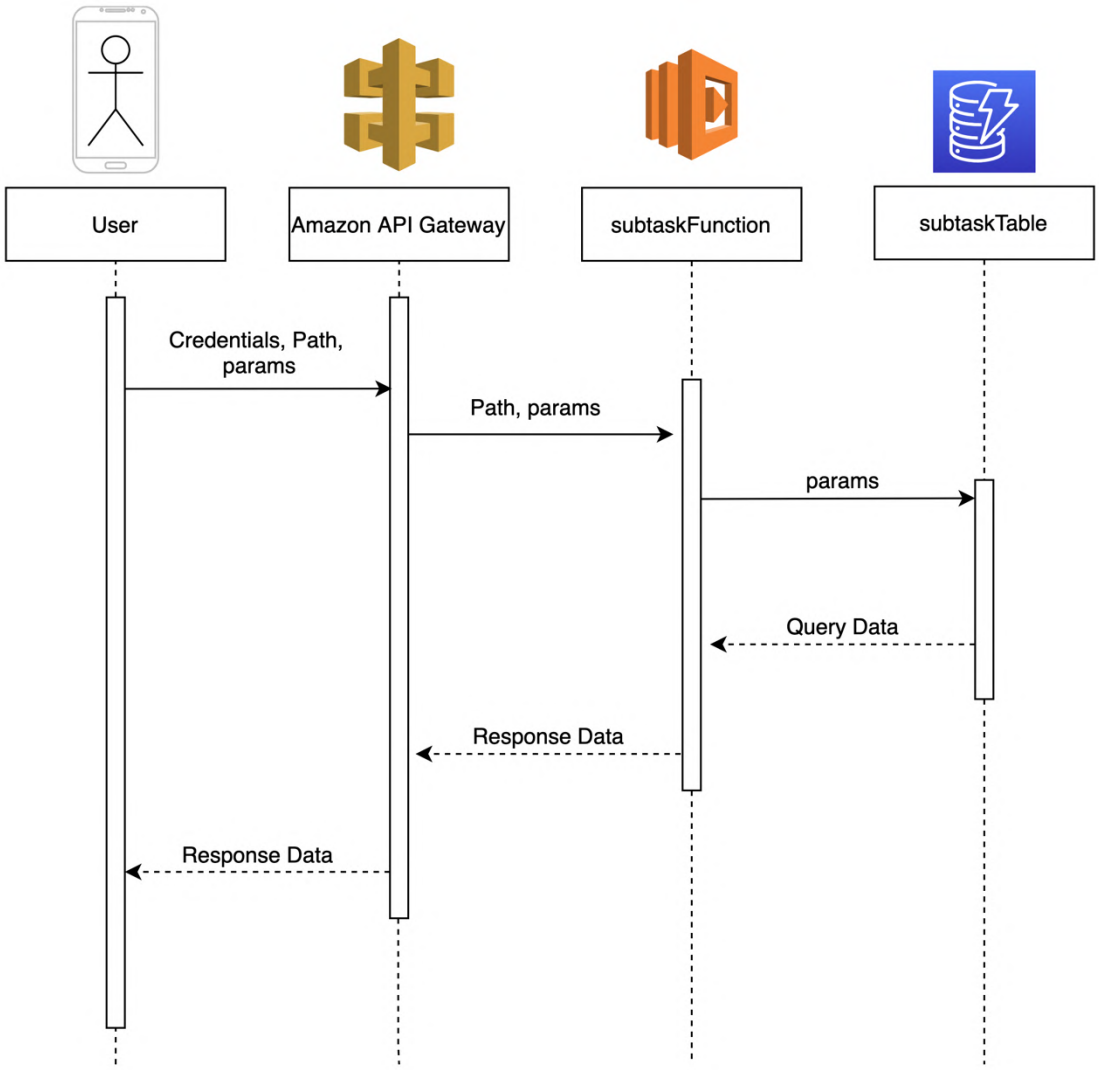
ภาพที่ 3.6.7.3.1 แผนภาพขั้นตอนการเรียก taskAPI ด้วย Method GET

เมื่อส่ง Path, params และ Credentials มาที่ API Gateway แล้ว ระบบจะไปเรียกใช้ taskFunction โดย Route ไปที่ Path ด้วย Method GET เพื่อรันโค้ด Query DynamoDB จาก taskTable จากนั้นจะส่ง Query Data คืนมายัง taskFunction และ Response ด้วย Data ที่ผู้ใช้ร้องขอ



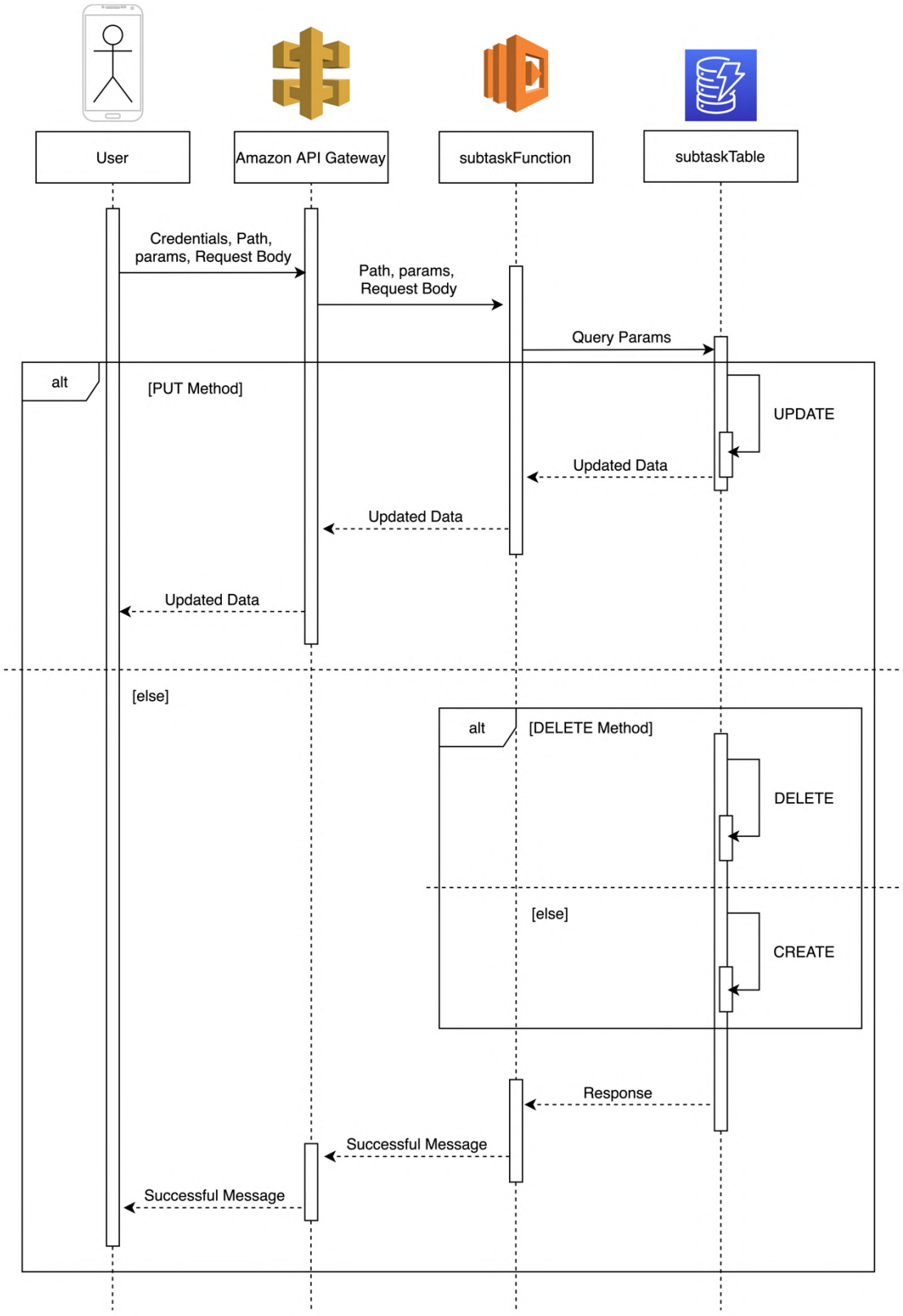
ภาพที่ 3.6.7.3.1 แผนภาพขั้นตอนการเรียก taskAPI ด้วย Method POST, PUT, DELETE

3.6.7.4 subtaskAPI



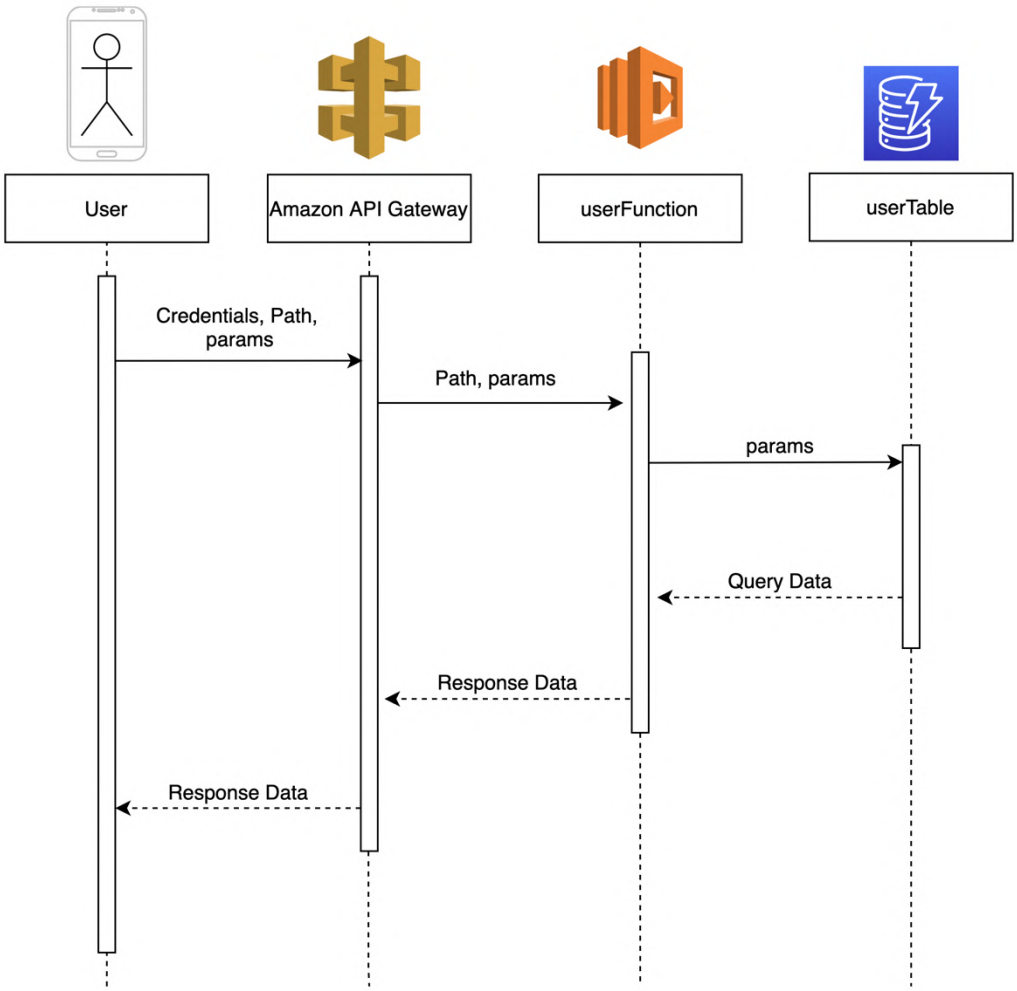
ภาพที่ 3.6.7.4.1 แผนภาพขั้นตอนการเรียก subtaskAPI ด้วย Method GET

เมื่อส่ง Path, params และ Credentials มาที่ API Gateway แล้ว ระบบจะไปเรียกใช้ subtaskFunction โดย Route ไปที่ Path ด้วย Method GET เพื่อรันโค้ด Query DynamoDB จาก subtaskTable จากนั้นจะส่ง Query Data คืนมายัง subtaskFunction และ Response ด้วย Data ที่ผู้ใช้ร้องขอ



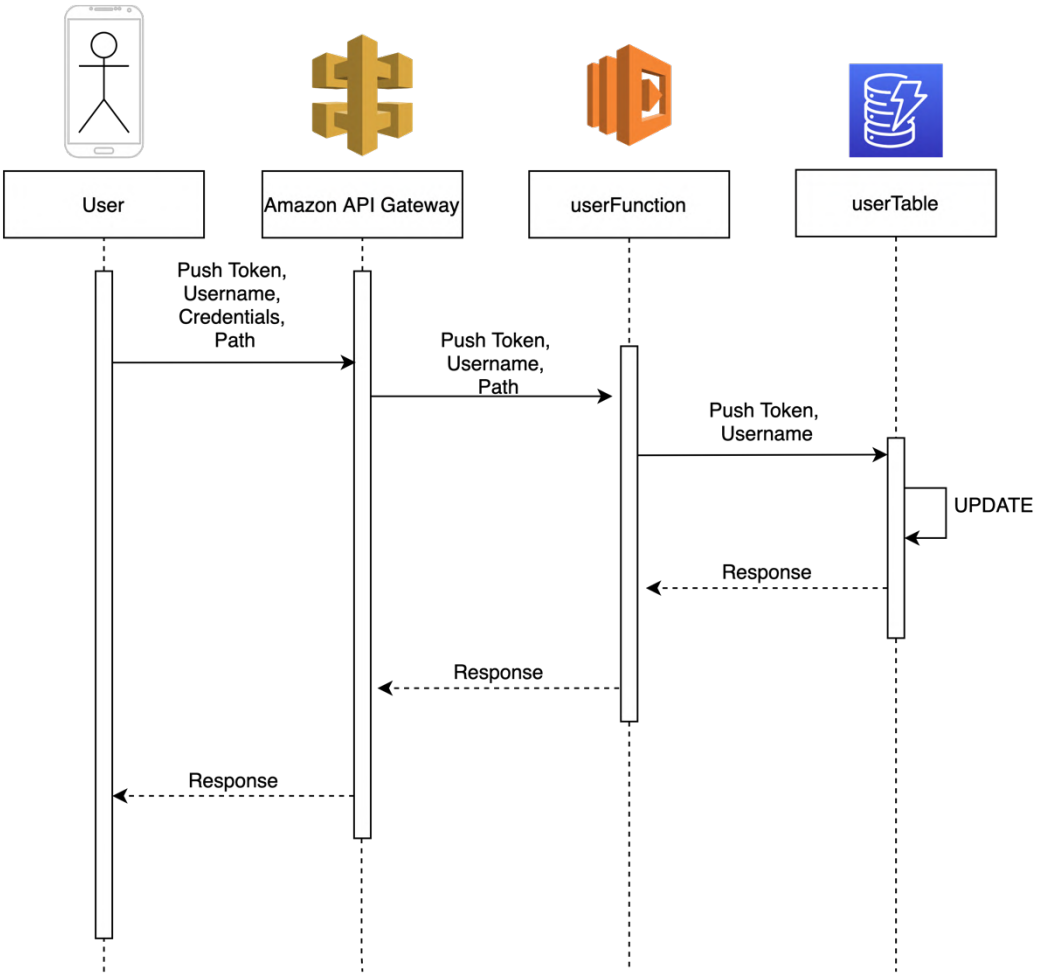
ภาพที่ 3.6.7.4.2 แผนภาพขั้นตอนการเรียก subtaskAPI ด้วย Method POST, PUT, DELETE

3.6.7.5 userAPI



ภาพที่ 3.6.7.5.1 แผนภาพขั้นตอนการเรียก userAPI ด้วย Method GET

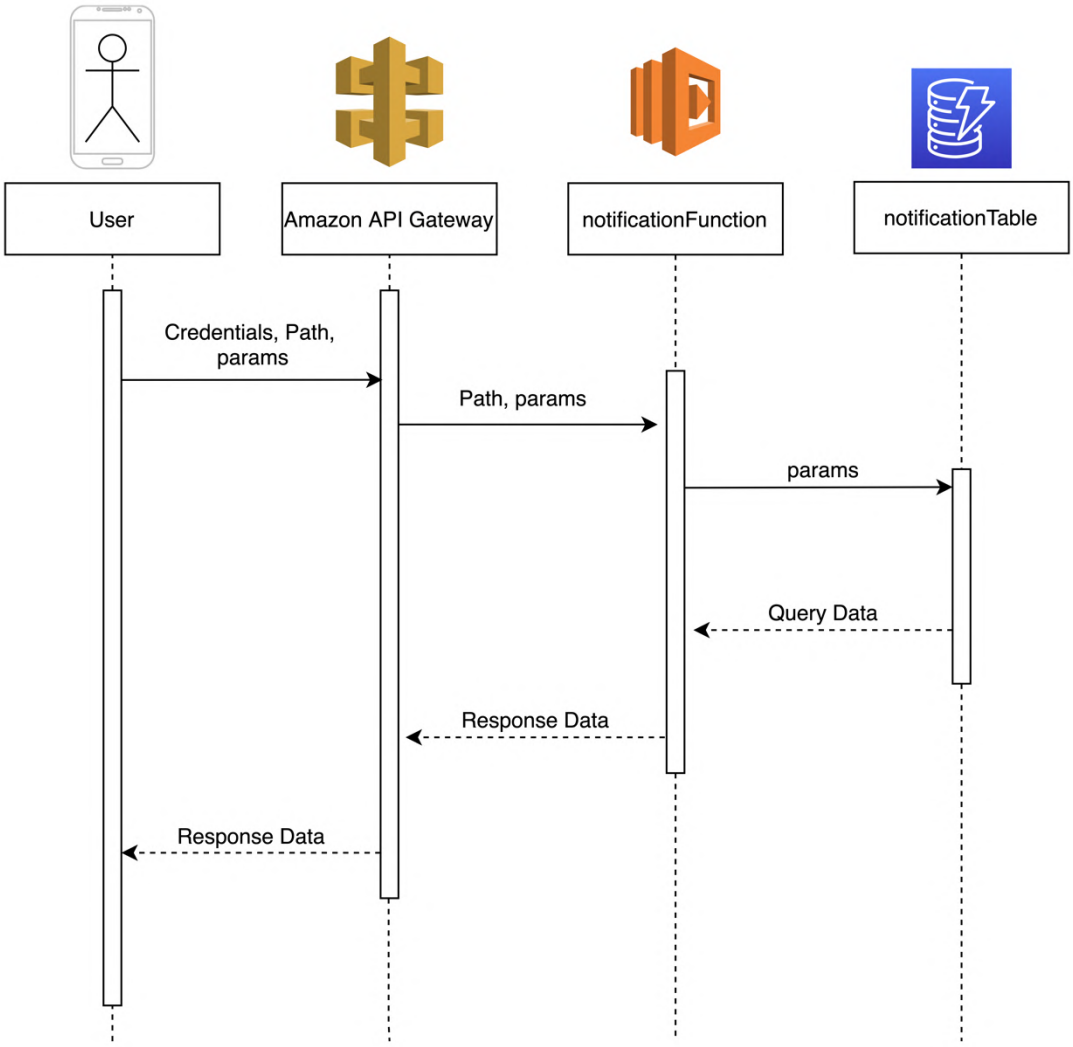
เมื่อส่ง Path, params และ Credentials มาที่ API Gateway แล้ว ระบบจะไปเรียกใช้ userFunction โดย Route ไปที่ Path ด้วย Method GET เพื่อรันโค้ด Query DynamoDB จาก userTable จากนั้นจะส่ง Query Data คืนมายัง userFunction และ Response ด้วย Data ที่ผู้ใช้ร้องขอ



ภาพที่ 3.6.7.5.2 แผนภาพขั้นตอนการสมัคร Push Token ด้วย Method PUT

เมื่อผู้ใช้ Sign In สำเร็จจะเข้าสู่หน้า NotificationScreen (อ้างอิงจากเอกสารผู้พัฒนาส่วนหน้าบ้าน) จะถูกเรียกใช้ API ด้วย HTTP Method PUT ที่เส้นทาง “/users/pushtoken” เพื่อทำการ UPDATE userTable ด้วย Push Token ที่ส่งมาจากผู้ใช้ ณ Attribute pushtoken

3.6.7.6 notificationAPI

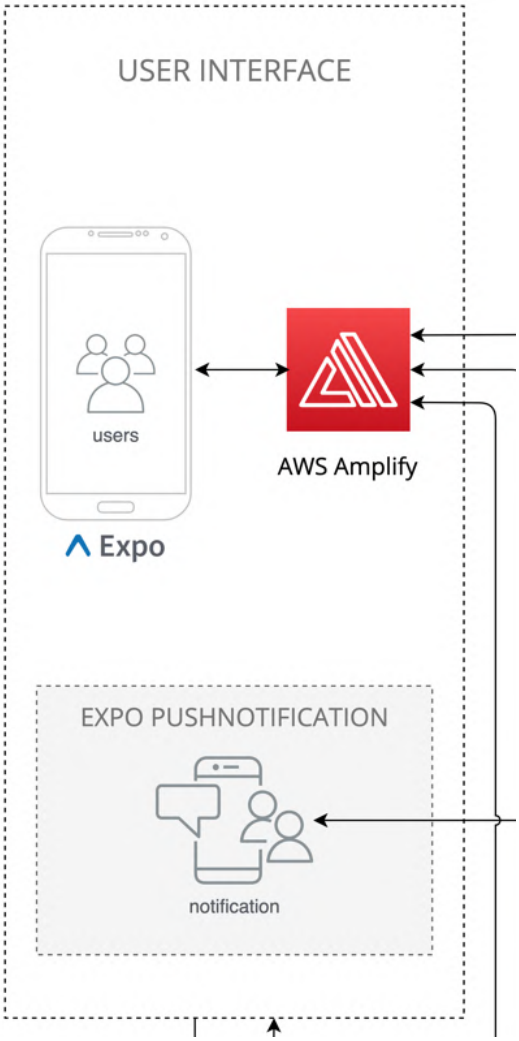


ภาพที่ 3.6.7.6.1 แผนภาพขั้นตอนการเรียก notificationAPI ด้วย Method GET

เมื่อส่ง Path, params และ Credentials มาที่ API Gateway แล้ว ระบบจะไปเรียกใช้ notificationFunction โดย Route ไปที่ Path ด้วย Method GET เพื่อรันโค้ด Query DynamoDB จาก notificationTable จากนั้นจะส่ง Query Data คืนมายัง userFunction และ Response ด้วย Data ที่ผู้ใช้ร้องขอ

3.7 การออกแบบส่วนประสานผู้ใช้ (System Integration with UI)

ในส่วนนี้ผู้พัฒนาจะขออธิบายถึงการออกแบบการเชื่อมต่อกับส่วนหน้าบ้าน (Front-end) จากที่กล่าวมาในหัวข้อที่ 3.4 แล้วว่าผู้พัฒนาหน้าบ้านได้เลือกใช้ expo และ amplify ในการเชื่อมต่อกับระบบหลังบ้าน โดยผู้พัฒนาจะขออธิบายโดยใช้ Architecture Diagram ดังภาพที่ 3.7.1



ภาพที่ 3.7.1 แผนภาพโครงสร้างส่วนหน้าบ้าน

ผู้พัฒนาขออนุญาตนำภาพจากเอกสารของผู้พัฒนาหน้าบ้านมาอ้างอิงเพื่อประกอบความเข้าใจในหัวข้อนี้ โดยจะเรียงจาก

- ลงชื่อเข้าใช้
- สมัครสมาชิก
- การขอ Credentials, เช็คสถานะ
- เรียกใช้ API

- อัพโหลดรูปภาพและร้องขอ URL
- รับการแจ้งเตือน

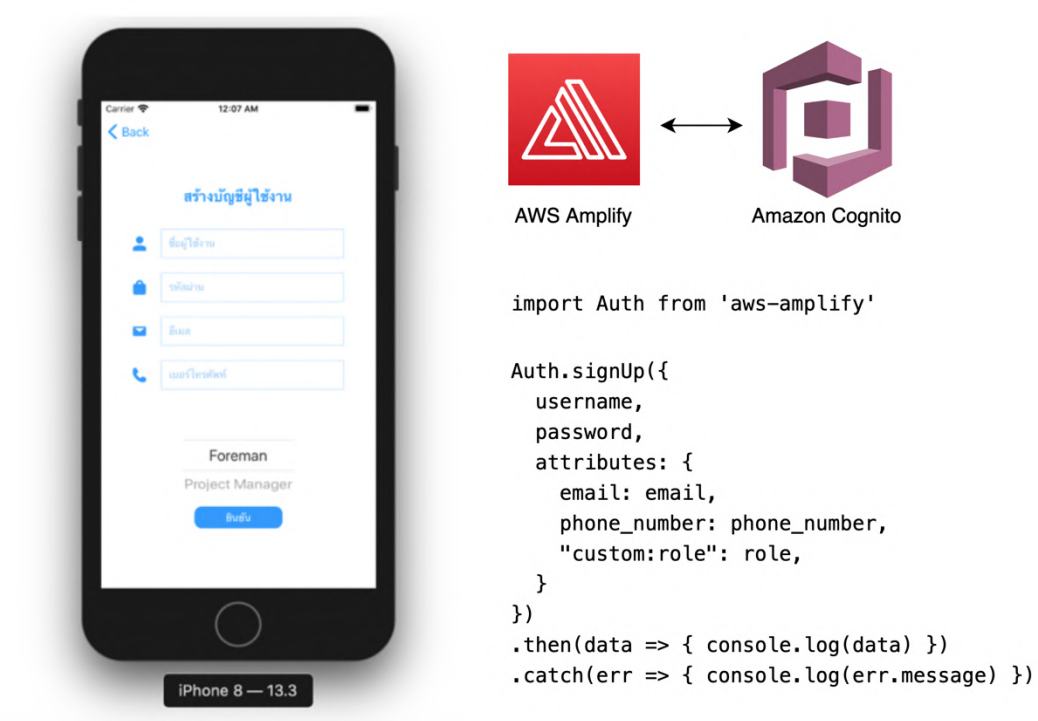
3.7.1 ลงชื่อเข้าใช้

ผู้พัฒนาหน้าบ้านสามารถเรียกใช้ Auth ได้จาก node package ‘aws-amplify’ ได้ โดยใช้ฟังก์ชัน signIn() มี Parameters เป็น (username, password) และจะได้ข้อมูลรายละเอียดของ User กลับมา ดังรูปที่ 3.7.1.1

ภาพที่ 3.7.1.1 ส่วนต่อประสานหน้า Sign In

3.7.2 สมัครสมาชิก

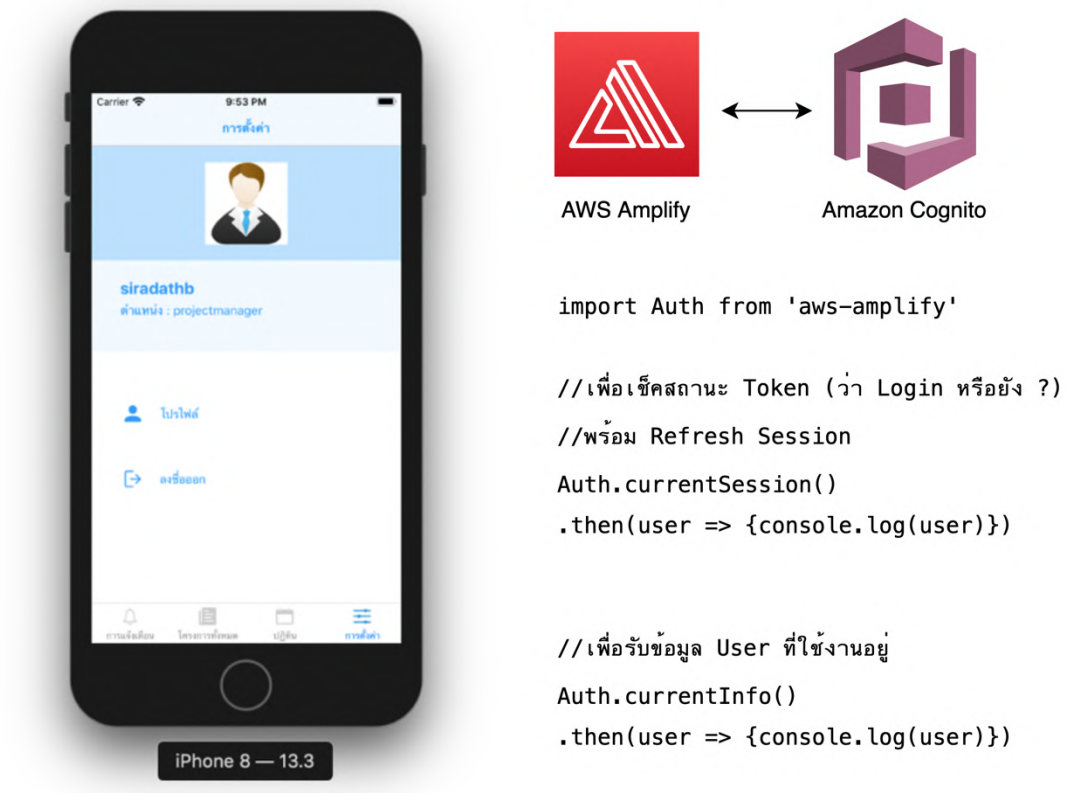
ผู้พัฒนาหน้าบ้านสามารถเรียกใช้ Auth ได้จาก node package ‘aws-amplify’ ได้ โดยใช้ฟังก์ชัน signUp() โดยมี parameter เป็น JSON ดังรูปที่ 3.7.2.1 และจะได้ข้อมูลการสมัคร Return กลับมา



ภาพที่ 3.7.2.1 ส่วนต่อประสานหน้า Sign Up

3.7.3 การขอ Credentials, การเช็คสถานะ

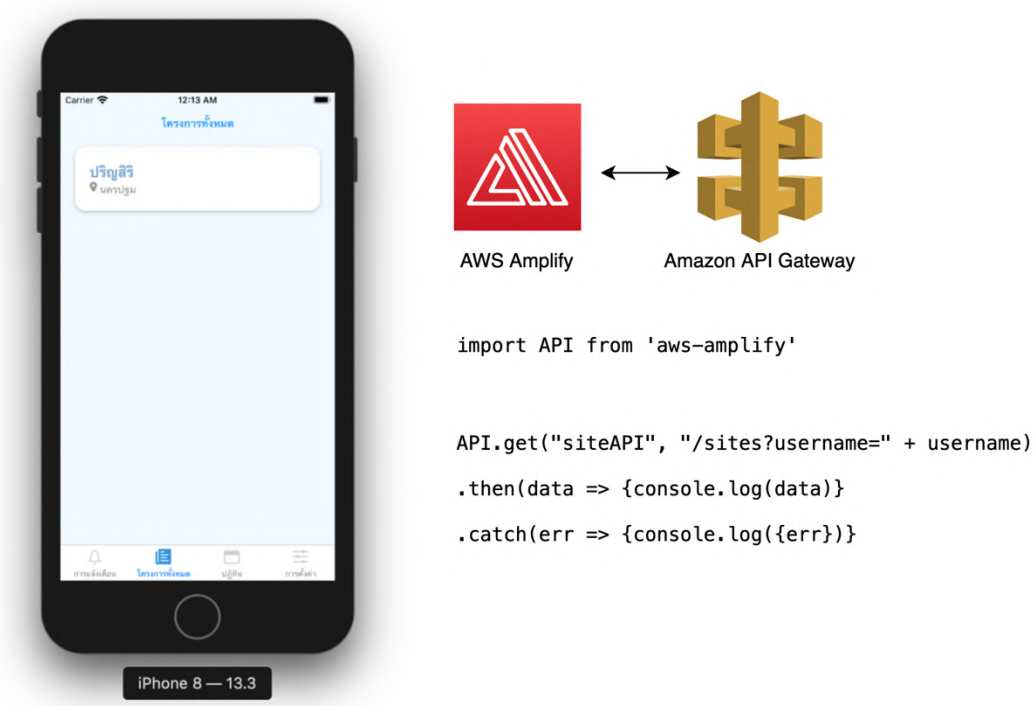
ผู้พัฒนาหน้าบ้านสามารถเรียกใช้ Auth ได้จาก node package ‘aws-amplify’ ได้ โดยใช้ฟังก์ชัน currentSession() เพื่อเช็คสถานะ Token บนเครื่องว่าหมดอายุหรือยังและ ฟังก์ชัน currentInfo() เพื่อรับข้อมูล User ที่ใช้งานอยู่ขณะนั้น ดังรูป 3.7.33



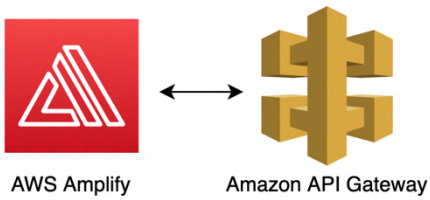
ภาพที่ 3.7.3.1 ส่วนต่อประสานหน้า Setting

3.7.4 เรียกใช้ API

ผู้พัฒนาหน้าบ้านสามารถเรียกใช้ API ได้จาก node package ‘aws-amplify’ ได้ โดยใช้ฟังก์ชัน get(), post(), put(), del() ดังรูปที่ 3.7.4.1 หากมี Request body ด้วยสามารถอ้างอิงได้จากภาพที่ 3.7.4.2 และหลังจากเรียกฟังก์ชันเสร็จสมบูรณ์จะได้ข้อมูลการสมัคร Return กลับมา ฟังก์ชันดังกล่าวมี Parameters เป็น (resource, path, body)



ภาพที่ 3.7.4.1 ส่วนต่อประสานหน้า AllSiteScreen



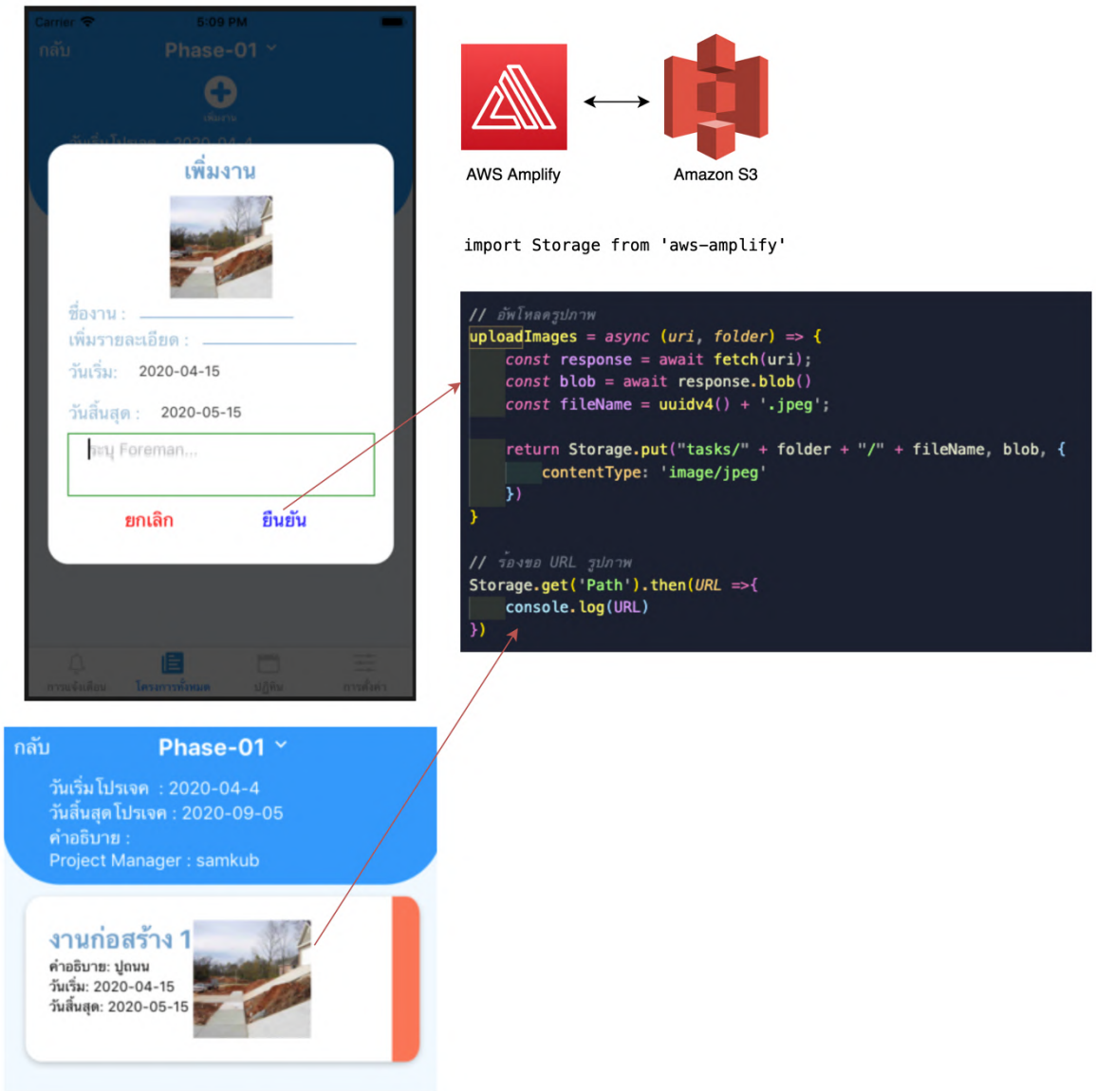

```
import API from 'aws-amplify'


let req = {
  body: {
    projectmanager: 'projectmanager',
    foremans: 'foremans[]',
    startDate: 'dd-mm-yyyy',
    dueDate: 'dd-mm-yyyy',
    description: 'description',
    siteId: 'siteId',
    images: 'images[]',
    title: 'title',
    blueprints: 'blueprints[]',
  }
}
API.post("projectAPI", "/projects", req)
.then(data => {
  console.log(data)
})
.catch(err => {console.log({err})})
```

ภาพที่ 3.7.4.2 ส่วนต่อประสานหน้า SiteScreen – เพิ่มโปรเจค

3.7.5 อัพโหลดรูปภาพและร้องขอ URL

ผู้พัฒนาหน้าบ้านสามารถเรียกใช้ Storage ได้จาก node package ‘aws-amplify’ ได้ โดยใช้ฟังก์ชัน put(), get() โดยมี parameter เป็น JSON ดังรูปที่ 3.7.5.1

- ฟังก์ชัน get() หลังจากเรียกฟังก์ชันเสร็จสมบูรณ์จะได้รับ URL ของรูปภาพนั้น ๆ กลับมา โดยมี Parameter เป็น (path, fIleName, blob, headers)
- ฟังก์ชัน put() หลังจากเรียกฟังก์ชันเสร็จสมบูรณ์จะได้รับ Path ของรูปภาพนั้น ๆ กลับมา โดยมี Parameter เป็น (path)



ภาพที่ 3.7.5.1 ส่วนต่อประสานหน้า Project Screen - อัพโหลดรูปภาพและร้องขอ URL

3.7.6 รับการแจ้งเตือน

ผู้พัฒนาหน้าบ้านสามารถเรียกใช้ Notifications ได้จาก node package ‘expo’ ได้ ดูรูปภาพที่ 3.7.6.1 ประกอบ โดยเรียกใช้ฟังก์ชันได้แก่

- getExpoPushTokenAsync() - เพื่อรับค่า Push Token ของอุปกรณ์เครื่องปัจจุบัน
- addListener() - เพื่อเพิ่มฟังก์ชันที่จะทำงานเมื่อได้รับ Push Notification ใหม่จาก FCM โดยมี Parameter เป็นฟังก์ชันที่ต้องการให้เป็น Listener


```
import { Notifications } from 'expo';
import * as Permissions from 'expo-permissions';


const { status: existingStatus } = await
Permissions.getAsync(Permissions.NOTIFICATIONS);

 // รับ Push Token ของอุปกรณ์เครื่องปัจจุบัน
 Notifications.getExpoPushTokenAsync()


//Subscription Notification ด้วยการเพิ่ม Listener Function
this._notificationSubscription = Notifications.addListener(
this._handleNotification
);
```

ภาพที่ 3.7.6.1 ส่วนต่อประสานหน้า NotificationScreen

บทที่ 4

การพัฒนาและทดสอบระบบ

ในบทนี้จะกล่าวถึงการพัฒนาระบบหลังบ้านของแอปพลิเคชัน โดยจะกล่าวถึงขั้นตอนการพัฒนาระบบและการทดสอบระบบพร้อมระบุภาพประกอบ ซึ่งมีรายละเอียดดังนี้

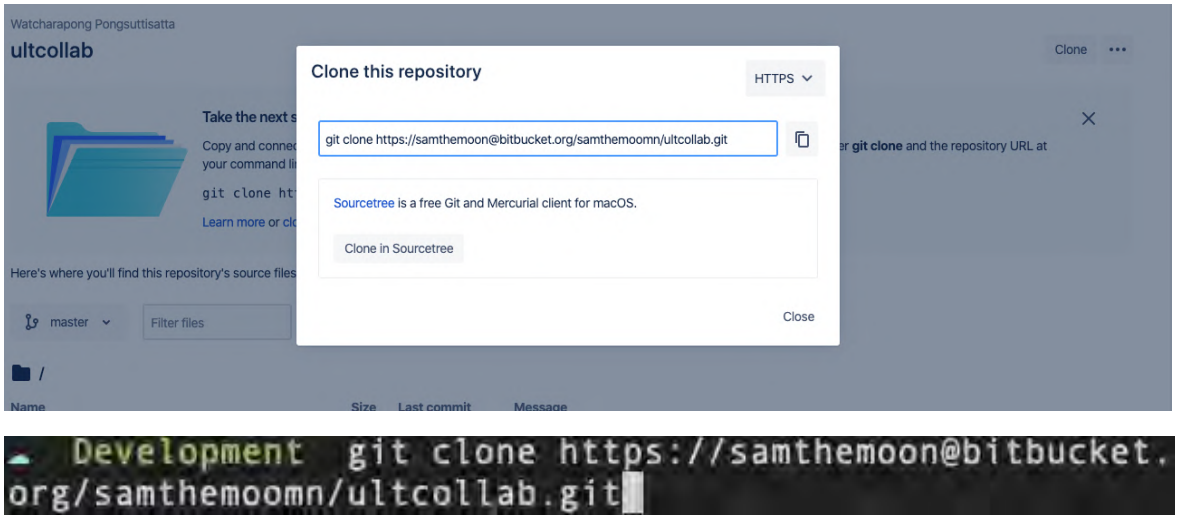
4.1 การพัฒนาระบบ

หลังจากที่เตรียมการทุกอย่างเสร็จสิ้นตามการออกแบบที่กล่าวมาในบทที่ 3 ในหัวข้อนี้ผู้พัฒนาจะอธิบายเริ่มต้นตั้งแต่การสร้างแอปพลิเคชัน (Initialize) ไปจนถึงการสร้างระบบหลังบ้าน (Backend) และจะพูดถึงในส่วนของการรวมระบบหน้าบ้านกับระบบหลังบ้านด้วย

4.1.1 เริ่มต้นสร้างแอปพลิเคชัน

การพัฒนาแอปพลิเคชันนั้น ผู้พัฒนาได้นำโค้ดหลังบ้านไปผูกกับตัวแอปพลิเคชันเลย ทำให้สามารถควบคุมเวอร์ชันของแอปพลิเคชันได้อย่างง่าย โดยใช้ Git Version Control ผู้พัฒนาเลือกใช้บริการของ เว็บไซต์ Bitbucket

เริ่มโดยการสร้าง Repository จากนั้น ให้ทำการ Clone Git มายังคอมพิวเตอร์ของผู้พัฒนา ด้วยการรันคำสั่งดังภาพที่ 4.1.1.1 ใน Terminal



ภาพที่ 4.1.1.1 การ Clone Git Repository

เริ่มสร้างแอปพลิเคชันที่เราจะพัฒนาด้วยคำสั่ง Expo

```
$ expo init ultcollab
```

จากนั้นให้พิมพ์คำสั่ง

```
$ cd ultcollab
```

เพื่อเข้าไปยังโฟลเดอร์ทำงานของเรา และเริ่มแอปพลิเคชันได้โดยคำสั่ง

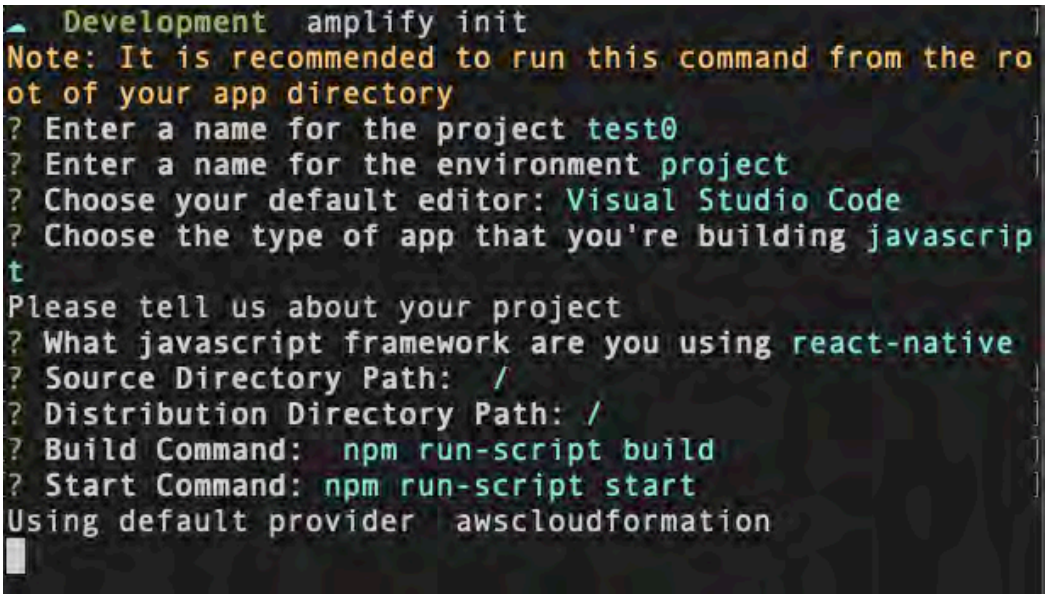
```
$ yarn start
```

4.1.2 เริ่มต้นสร้างระบบหลังบ้าน

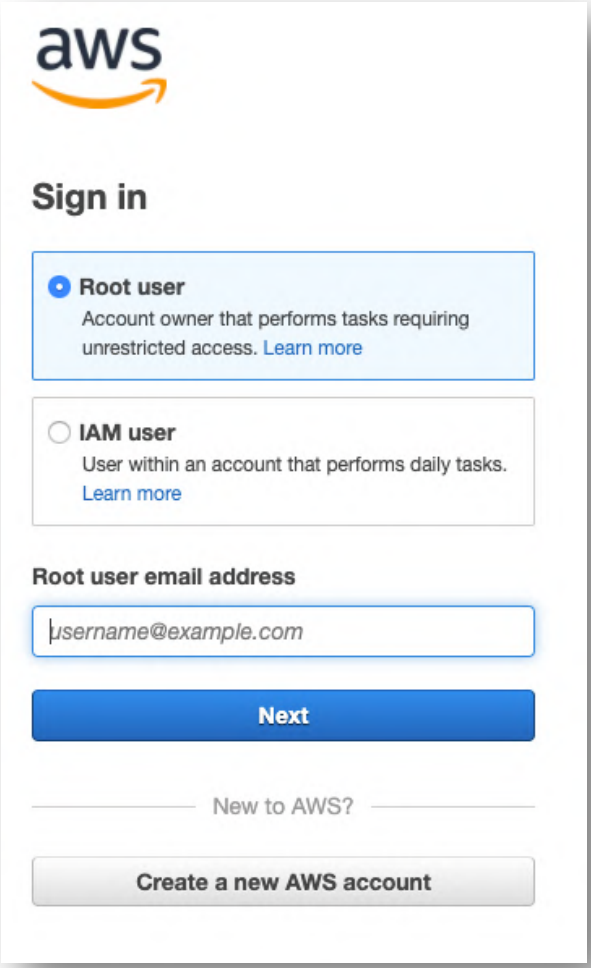
การพัฒนาหลังบ้านเริ่มจากการสร้าง Project และ Environment ที่ต้องการพัฒนาก่อนโดยจะใช้คำสั่งจาก node package Amplify

```
$ amplify init
```

จากนั้นให้กรอกข้อมูลเบื้องต้นให้ครบถ้วน และระบบจะร้องขอการผูกบัญชี AWS เข้ากับตัว Project ให้ทำตามขั้นตอนที่ระบบสอน



ภาพที่ 4.1.2.1 การสร้าง Amplify Projectt



ภาพที่ 4.1.2.2 ลงชื่อเข้าใช้ AWS Management Console

หลังจาก Initialize เสร็จแล้ว เราสามารถเพิ่ม Resource ไปยัง AWS Amplify ได้แล้ว โดยการทำงานจะเหมือนกับ Git มีคำสั่งที่จำเป็น ได้แก่

ตารางที่ 4.1.2.1 คำสั่ง Amplify CLI ที่จำเป็น

คำสั่ง CLI	คำอธิบาย
$ amplify add <category>	เพิ่ม Resource ไปยัง Environment
$ amplify remove <category>	ลบ Resource ออกจาก Environment
$ amplify update	แก้ไข Resource บน Environment
$ amplify push	นำ Resource ที่แก้ไขแล้วทั้งหมด Deploy ไปยัง AWS
$ amplify pull	ดึงข้อมูล Resource ล่าสุดบน AWS มายัง Local
$ amplify status	ตรวจสอบสถานะ Resource ทั้งหมด

ตารางที่ 4.1.2.1 Resource Category ที่เลือกใช้

Resource Category	คำอธิบาย
api	สำหรับสร้าง RESTful API
storage	สำหรับสร้าง S3 Storage, DynamoDB Table
function	สำหรับสร้าง Lambda Function
auth	สำหรับสร้าง User Pool, Groups

เพิ่ม Resource Function ที่ออกแบบมาแล้วในบทที่ 3 ตารางที่ 4.1.2.1 ลงไปยัง Amplify Project ของเรา โดยใช้คำสั่ง

```
$ amplify add function
```

เพิ่ม Resource Storage ที่ออกแบบมาแล้วในบทที่ 3 ตารางที่ 4.1.2.1 ลงไปยัง Amplify Project ของเรา โดยใช้คำสั่ง

```
$ amplify add storage
```

เพิ่ม Resource API ที่ออกแบบมาแล้วในบทที่ 3 ตารางที่ 4.1.2.1 ลงไปยัง Amplify Project ของเรา โดยใช้คำสั่ง

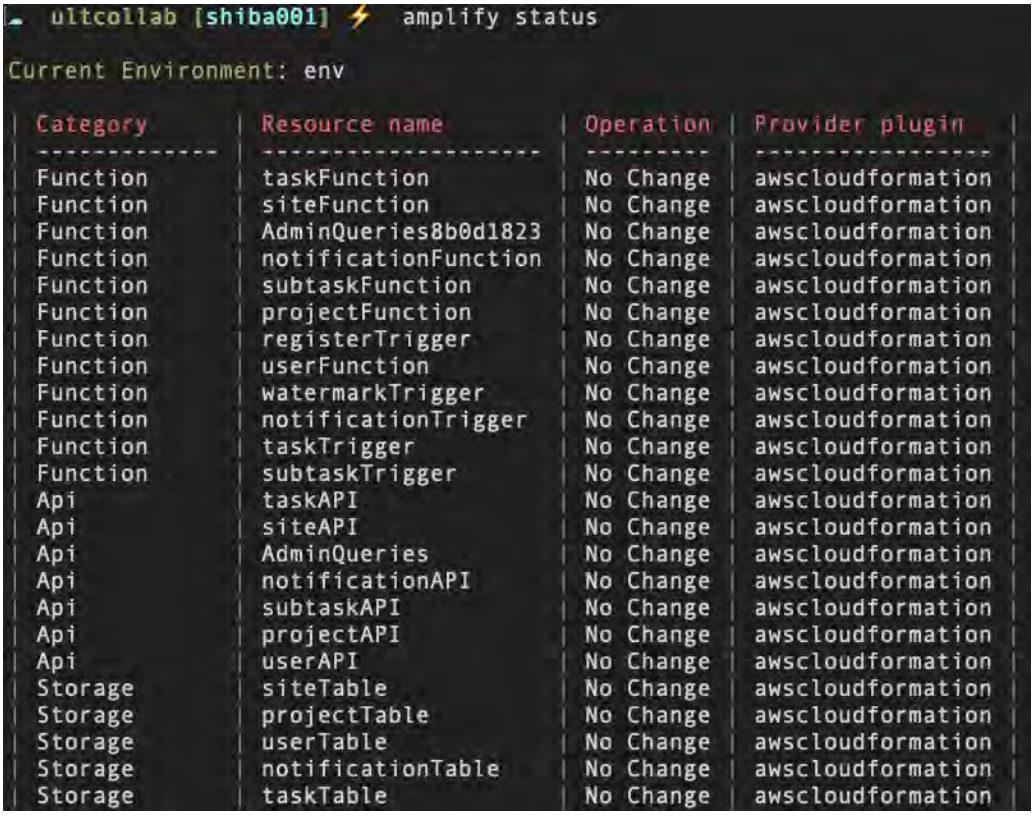
```
$ amplify add api
```

เพิ่ม Resource Auth ที่ออกแบบมาแล้วในบทที่ 3 ตารางที่ 4.1.2.1 ลงไปยัง Amplify Project ของเรา โดยใช้คำสั่ง

```
$ amplify add auth
```

หลังจากเพิ่ม Resource ทั้งหมดแล้ว ให้ทำการเช็คโดยพิมพ์คำสั่ง

```
$ amplify status
```



```
ultcollab [shiba001] ⚡  amplify status

Current Environment: env

| Category | Resource name | Operation | Provider plugin |
| ---------- | ---------- | ---------- | ---------- |
| Function | taskFunction | No Change | awscloudformation |
| Function | siteFunction | No Change | awscloudformation |
| Function | AdminQueries8b0d1823 | No Change | awscloudformation |
| Function | notificationFunction | No Change | awscloudformation |
| Function | subtaskFunction | No Change | awscloudformation |
| Function | projectFunction | No Change | awscloudformation |
| Function | registerTrigger | No Change | awscloudformation |
| Function | userFunction | No Change | awscloudformation |
| Function | watermarkTrigger | No Change | awscloudformation |
| Function | notificationTrigger | No Change | awscloudformation |
| Function | taskTrigger | No Change | awscloudformation |
| Function | subtaskTrigger | No Change | awscloudformation |
| Api | taskAPI | No Change | awscloudformation |
| Api | siteAPI | No Change | awscloudformation |
| Api | AdminQueries | No Change | awscloudformation |
| Api | notificationAPI | No Change | awscloudformation |
| Api | subtaskAPI | No Change | awscloudformation |
| Api | projectAPI | No Change | awscloudformation |
| Api | userAPI | No Change | awscloudformation |
| Storage | siteTable | No Change | awscloudformation |
| Storage | projectTable | No Change | awscloudformation |
| Storage | userTable | No Change | awscloudformation |
| Storage | notificationTable | No Change | awscloudformation |
| Storage | taskTable | No Change | awscloudformation |
```

ภาพที่ 4.1.2.3 พิมพ์คำสั่งเช็คสถานะ Resource

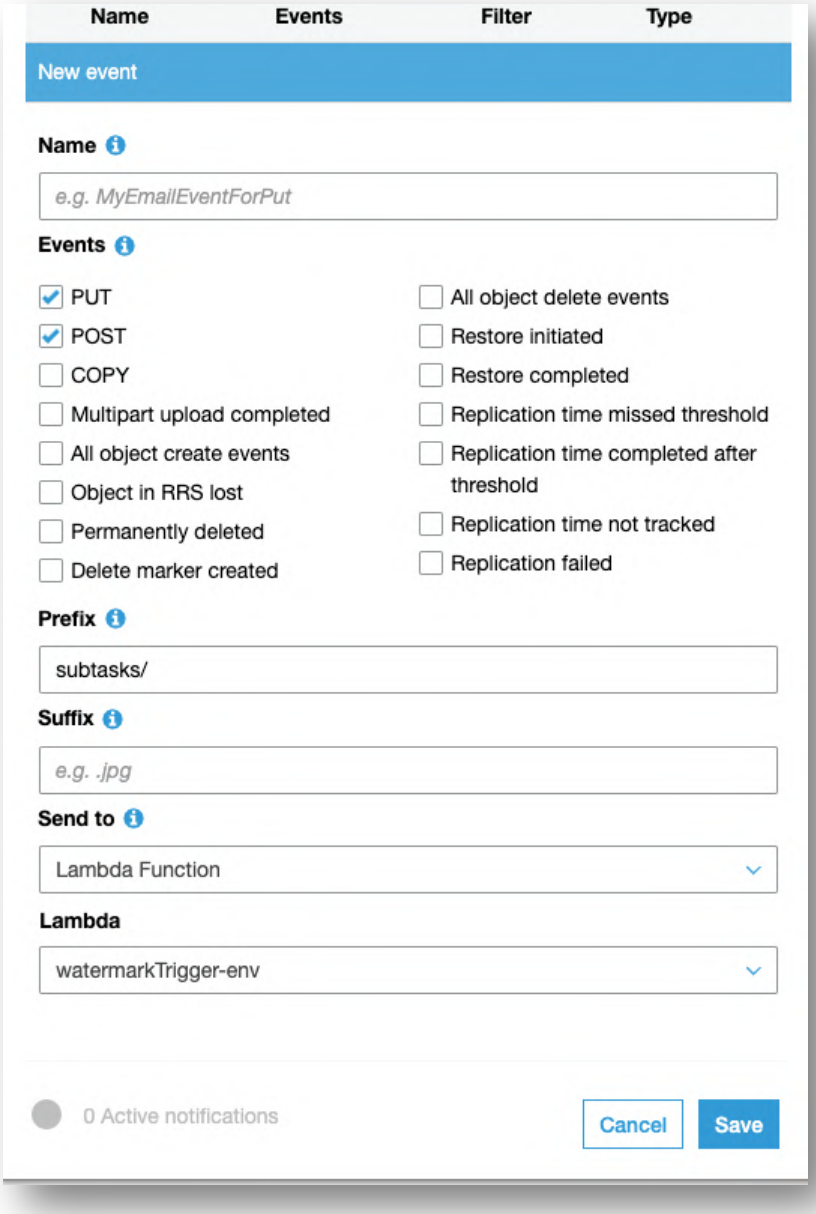
เมื่อพอใจแล้วให้ Deploy ขึ้นไปยัง AWS โดยพิมพ์คำสั่ง

$ amplify push

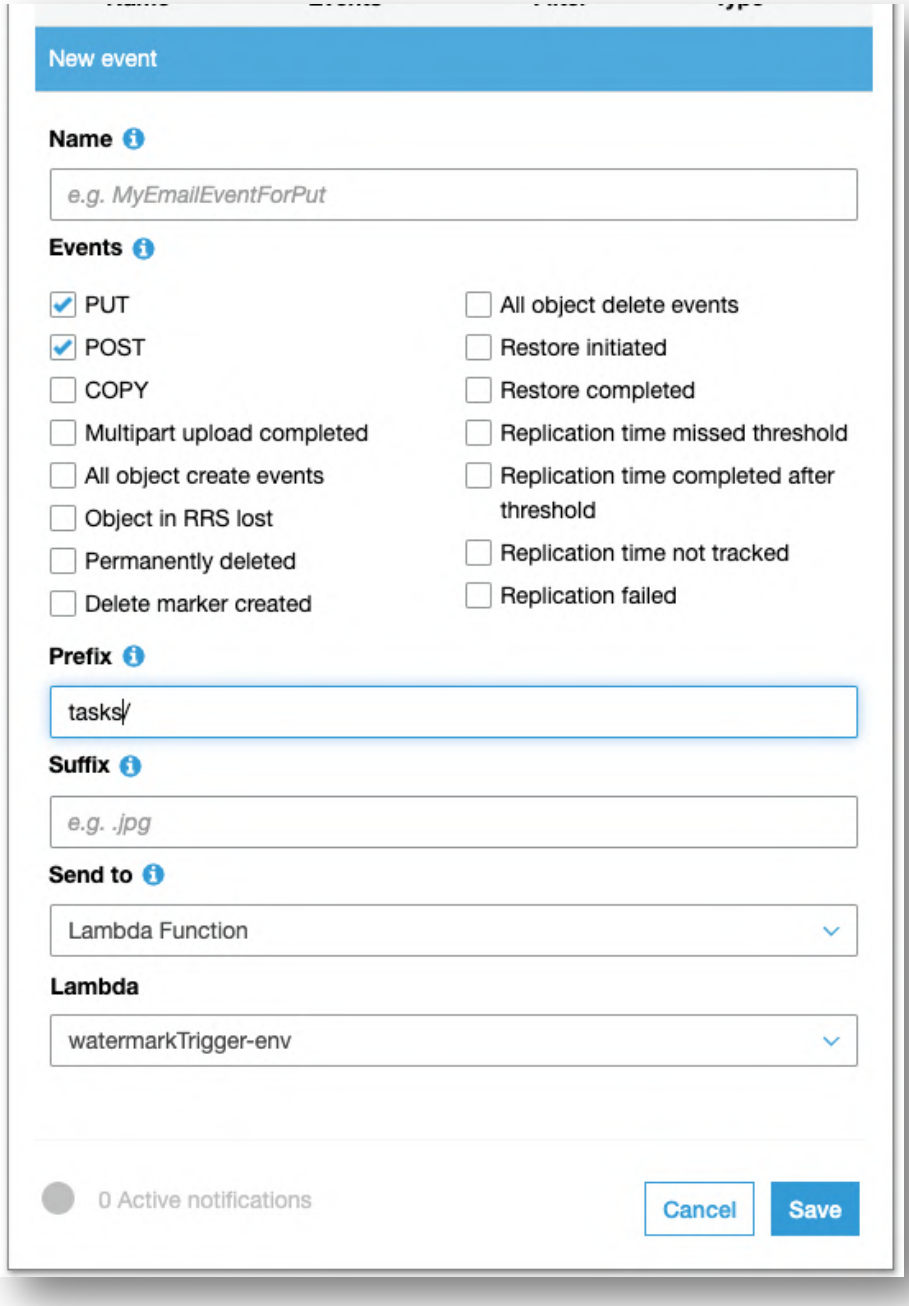
เมื่อเตรียมพร้อมทุกอย่างเสร็จสิ้นแล้ว นอกเหนือจากที่ทำมาทั้งหมด จำเป็นจะต้องผูก Trigger Function ไปยัง Services เพิ่มเติมได้แก่

- Amazon S3 Event Trigger
- AWS Cognito Post Registration

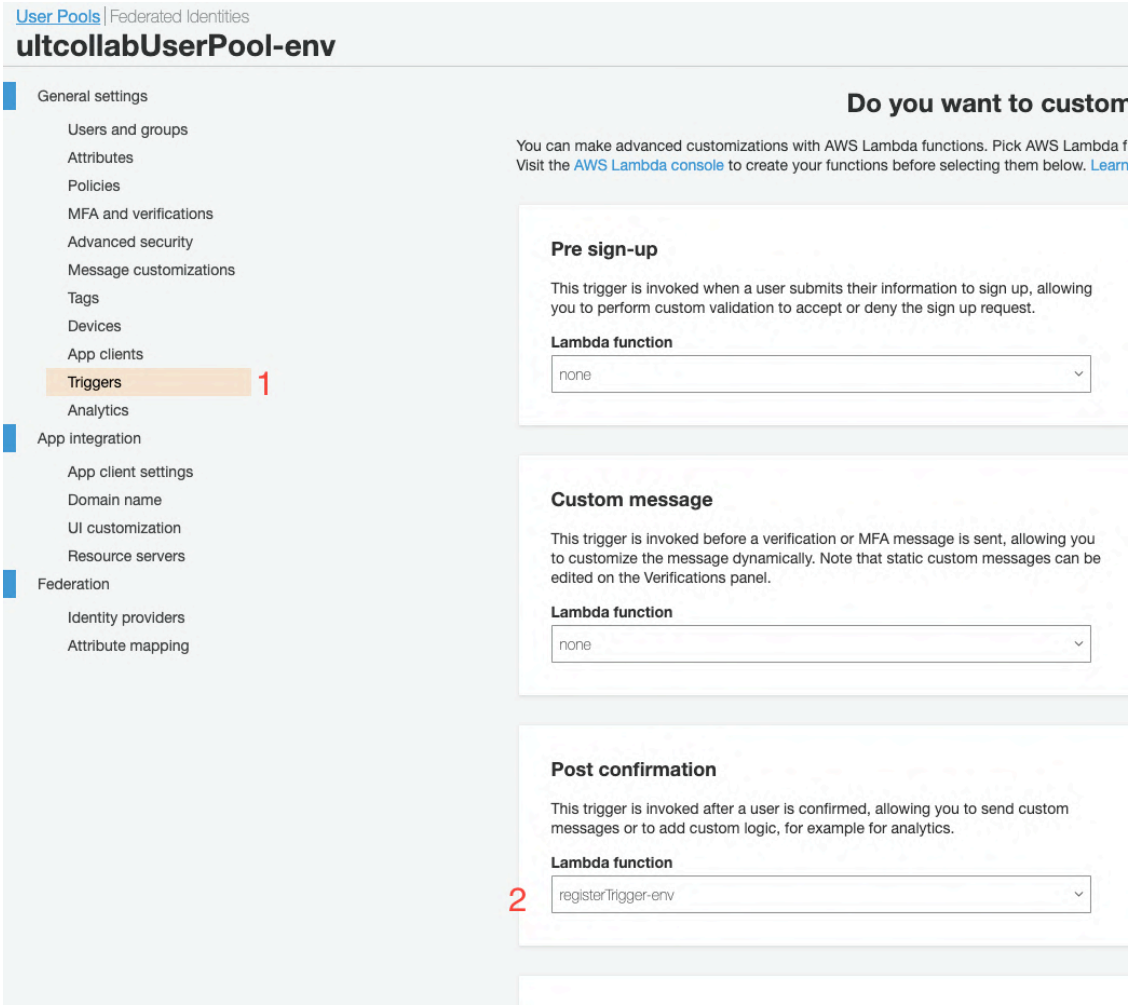
ในส่วนของ Amazon S3 นั้น ผู้พัฒนาได้เข้าไปที่ Bucket ที่ได้สร้างขึ้นในขั้นตอนข้างต้น จากนั้นสามารถนำ watermarkTrigger ไปผูกกับ Event ได้ดังภาพที่ 4.1.2.4 และ 4.1.2.5



ภาพที่ 4.1.2.4 การผูก Watermark Trigger ไปยัง S3 – Subtask



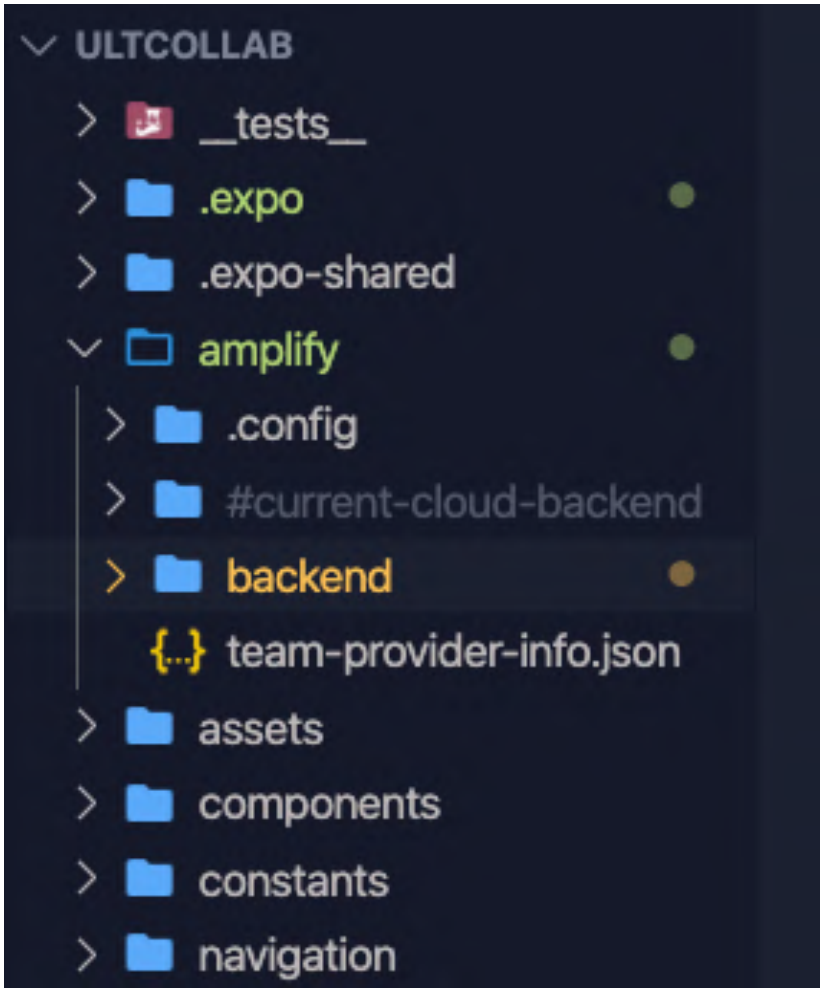
ภาพที่ 4.1.2.5 การผูก Watermark Trigger ไปยัง S3 - Task



ภาพที่ 4.1.2.6 การผูก Post Confirmation Trigger

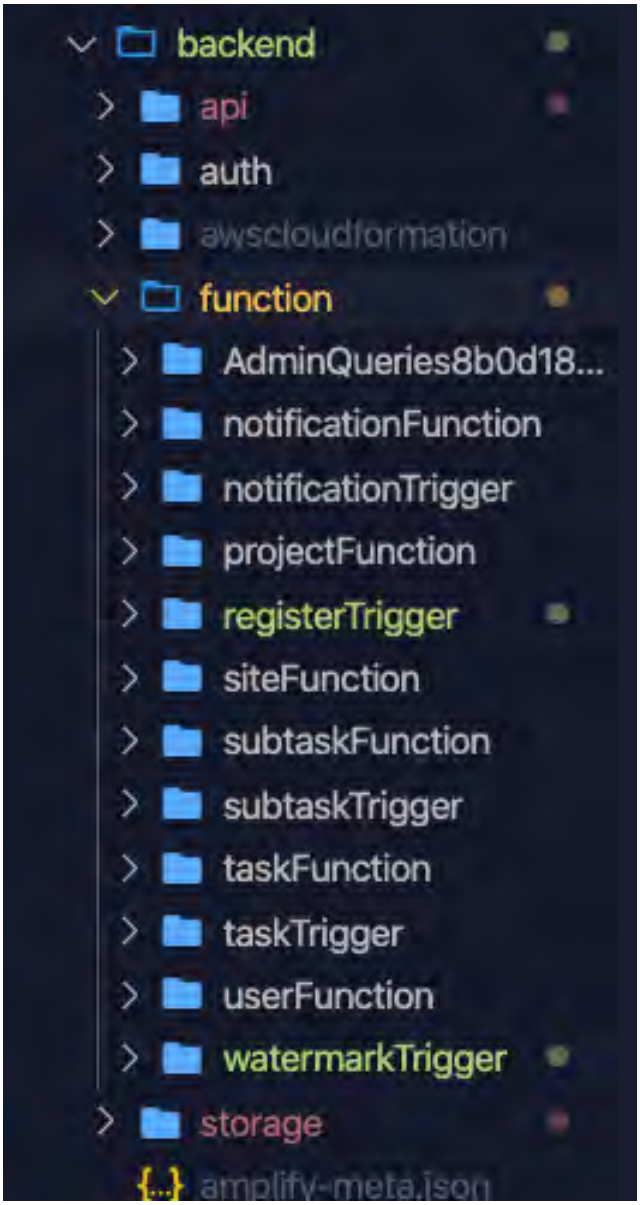
ในส่วนของ AWS Cognito ให้เข้าไปที่หน้า User Pool Console เลือกไปที่ User Pool ที่เราเพิ่งสร้างไปในขั้นตอนข้างต้น และสามารถเลือก registerTrigger ไปที่ Post Confirmation ตามรูปภาพที่ 4.1.2.6

หลังจากเตรียมการทุกอย่างเสร็จสิ้นก็พร้อมที่จะเริ่มเขียนโค้ดพัฒนาได้แล้วโดยการเปิดโฟลเดอร์ที่เรา สร้างขึ้นในตอนแรกที่เรา Clone Git มา เมื่อเข้าไปจะพบกับโฟลเดอร์ชื่อ /amplify/backend ดังรูปที่ 4.1.2.7

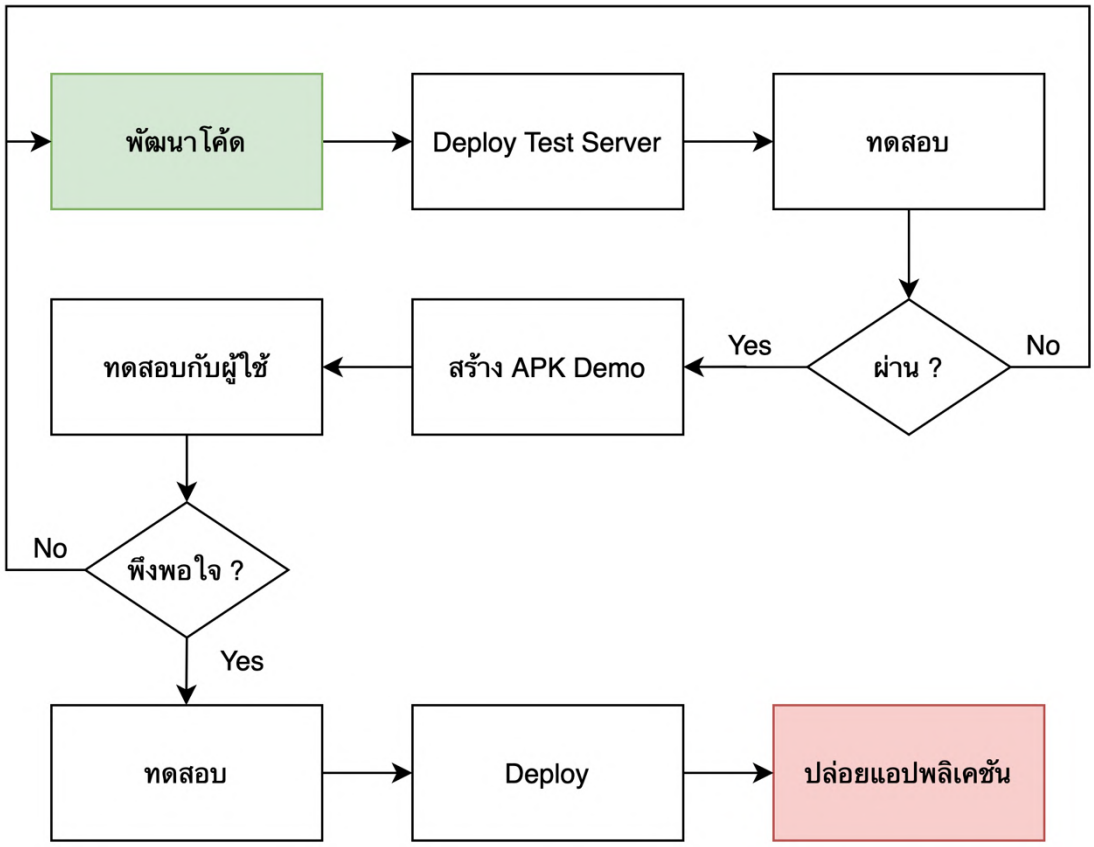


ภาพที่ 4.1.2.7 แสดงโฟลเดอร์ amplify

เมื่อเข้าไปในโฟลเดอร์ backend จะพบกับรายชื่อฟังก์ชันที่เราสร้างขึ้นมาในข้างต้นทั้งหมด เราสามารถเข้าไปแก้ไขโค้ดของฟังก์ชันได้เลย ดังรูปที่ 4.1.2.8



ภาพที่ 4.1.2.8 แสดงโฟลเดอร์ function

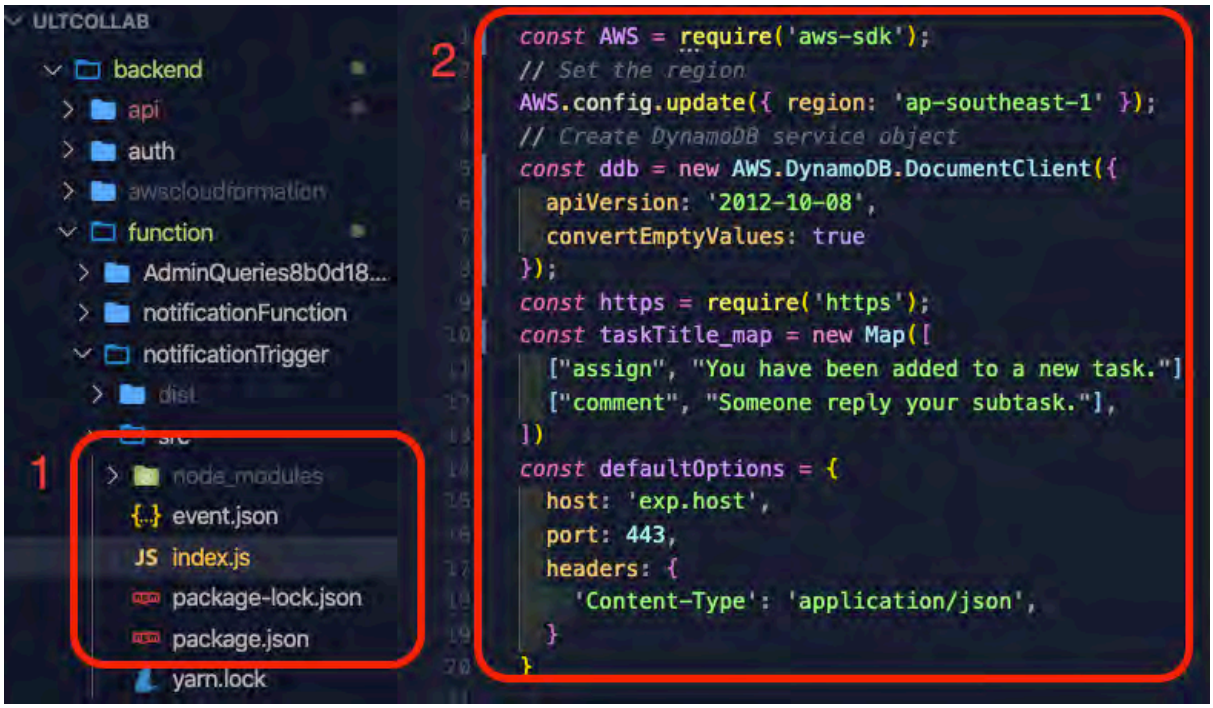


ภาพที่ 4.1.2.9 แผนการพัฒนาแอปพลิเคชัน

จากแผนภาพการไหล (Flowchart Diagram) สามารถอธิบายได้ว่า ระบบหลังบ้านสามารถ Deploy Test Server กี่ครั้งก็ได้เพื่อทดสอบจนกว่าจะผ่านครบทุก Functional Test หากทดสอบไม่ผ่านสามารถกลับมาแก้ไขโค้ดและ Deploy ใหม่อีกครั้ง ผู้พัฒนาจะใช้ Flow นี้ในการพัฒนาระบบหลังบ้านก่อนถึงมือลูกค้า โดยในส่วนถัดไปจะกล่าวถึงการเขียนโค้ดเบื้องต้น

4.1.3 เริ่มต้นการพัฒนาโค้ดหลังบ้าน

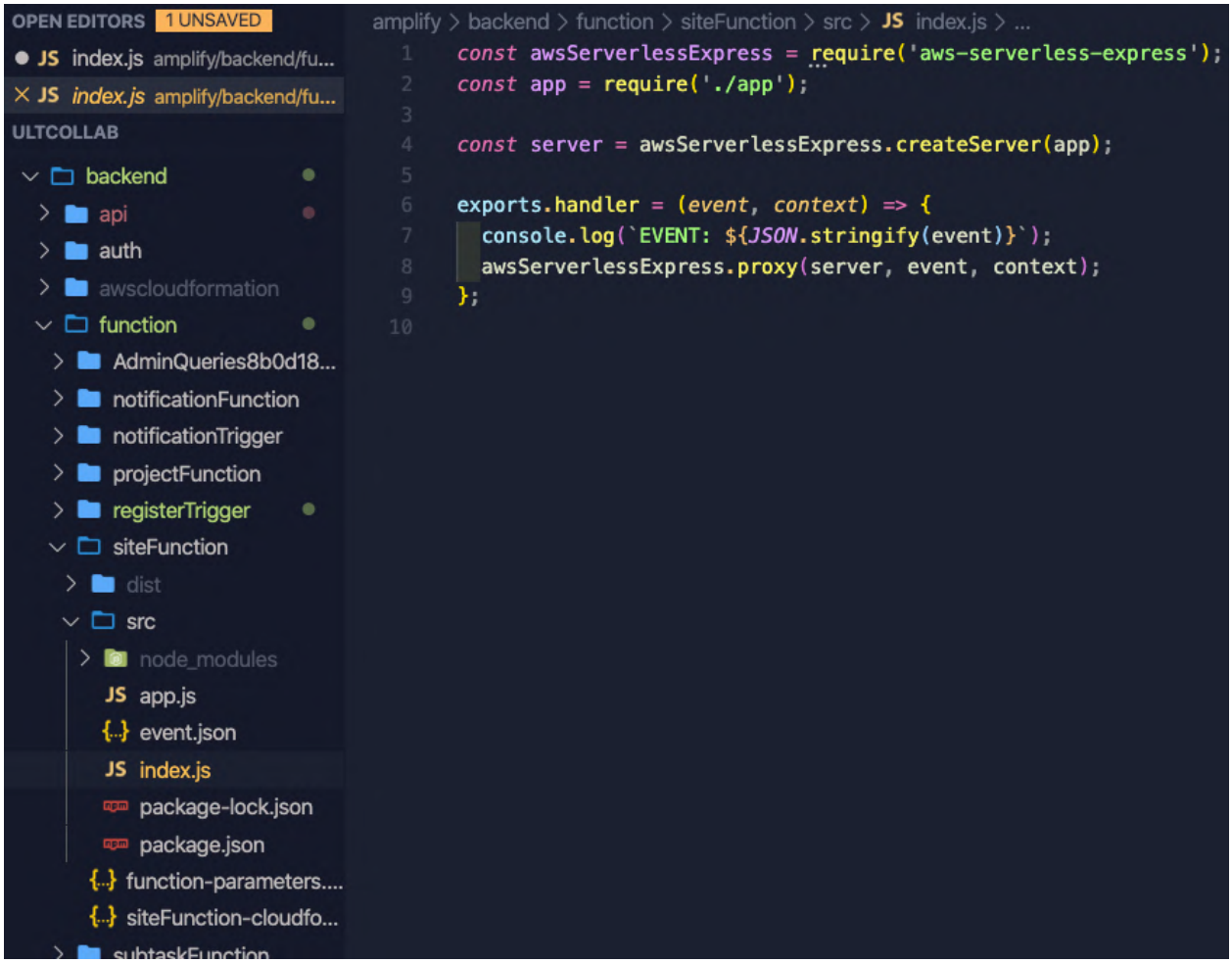
สามารถเข้าไปเขียนโค้ดได้เลยจากคอมพิวเตอร์ตนเอง ผู้พัฒนาขอยกตัวอย่างโค้ดจากฟังก์ชัน notificationTrigger ดังภาพที่ 4.1.3.1 ด้านล่าง



ภาพที่ 4.1.3.1 ตัวอย่างโค้ดจากฟังก์ชัน notificationTrigger

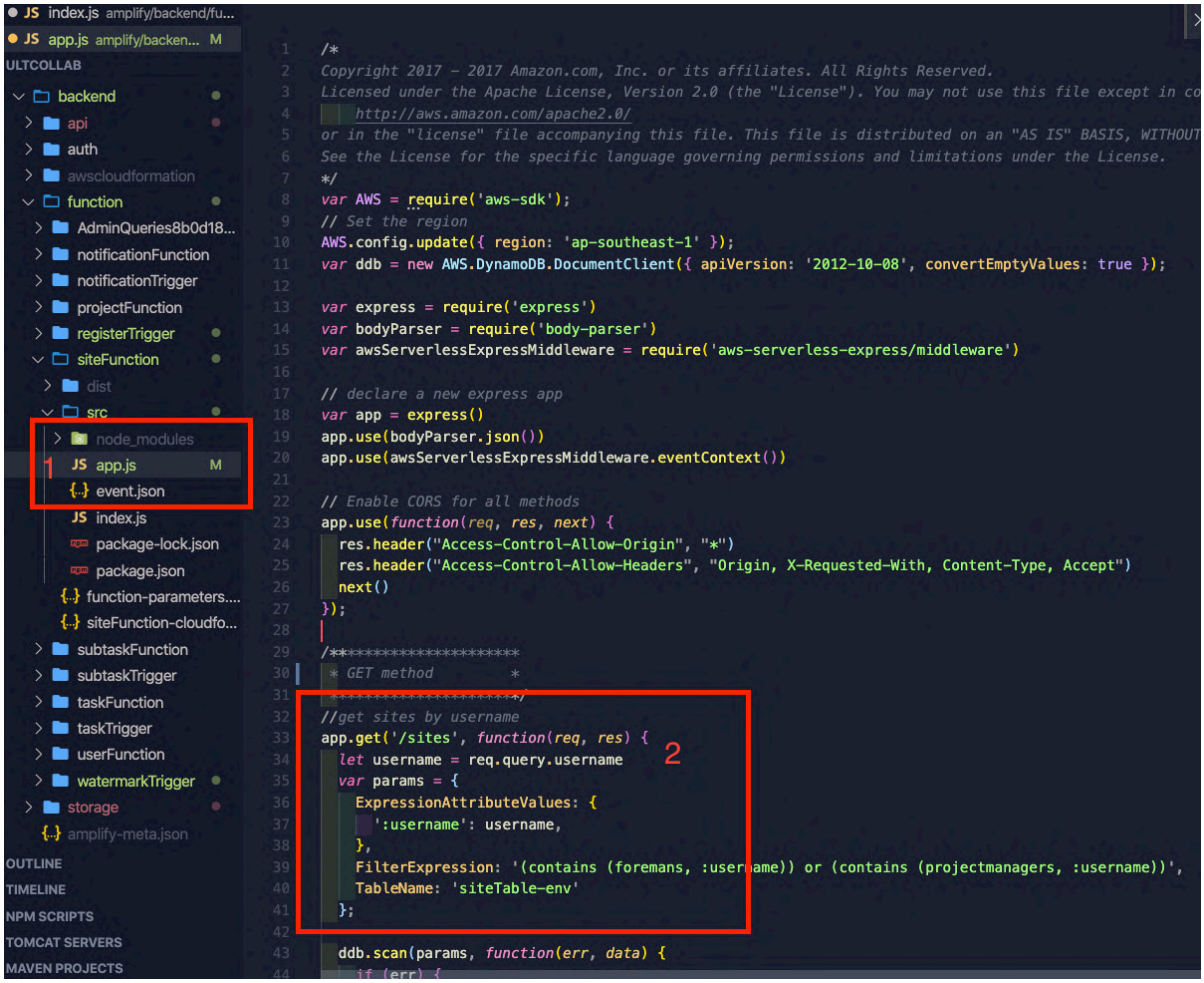
อธิบายได้ว่าในโฟลเดอร์ของฟังก์ชัน ผู้พัฒนาสามารถเข้าไปที่โฟลเดอร์ src จะพบกับ index.js ตามสัญลักษณ์ 1 ในภาพข้างต้น และสามารถแก้ไขโค้ดภายในได้ดังสัญลักษณ์ 2 ในภาพข้างต้น

ในการทำงานของ AWS Lambda Function นั้น ปกติจะเรียนใช้งานจากไฟล์ index.js ก่อนเสมอ แต่สำหรับฟังก์ชัน API ที่เราได้ออกแบบมา ผู้พัฒนาได้ใช้ Express ในการ Route และเป็น Middleware ของทุก ๆ ฟังก์ชัน API ผู้พัฒนาจึงขอยกตัวอย่างโค้ดการใช้ Express โดยจะยกตัวอย่างจาก siteFunction ก่อน ดังภาพที่ 4.1.3.2 ด้านล่าง



ภาพที่ 4.1.3.2 ยกตัวอย่าง Express

จากภาพข้างต้น อธิบายได้ว่าเมื่อระบบหลังบ้านได้รับข้อมูล API จาก API Gateway ก็จะเข้ามาสู่ฟังก์ชัน exports.handler ในบรรทัดที่ 6 เมื่อเข้าฟังก์ชันแล้วจะเข้าสู่ Middleware ของ Express นั้นเอง ดังบรรทัดที่ 8 ต่อจากนี้ไปข้อมูล API จะเข้าไปสู่ Path ต่าง ๆ ตาม Method ที่เราได้ตั้งค่าเอาไว้ในไฟล์ app.js เรียบร้อย ดังภาพ 4.1.3.3



ภาพที่ 4.1.3.3 ตัวอย่าง app.js

จากภาพข้างต้น อธิบายได้ว่าเมื่อเข้าไปยังไฟล์ app.js ในสัญลักษณ์ 1 แล้ว ข้อมูล API ก็จะมาสู่ส่วนของสัญลักษณ์ 2 ต่อไป เป็นอันเสร็จสิ้นการอธิบาย Express เบื้องต้น

ผู้พัฒนาจะขอตัดภาพโค้ดในแต่บางส่วน นำมาอธิบายด้วยบรรทัดที่สำคัญ โดยจะอธิบายตั้งแต่การเขียนโค้ดรับ API โดยฟังก์ชันที่นำมายกตัวอย่างได้แก่ taskFunction

```
app.get('/tasks', function(req, res) {
  // Add your code here
  let username = req.query.username
  let projectId = req.query.id
  if (projectId == null) {
    var params = {
      ExpressionAttributeValues: {
        ':username': username,
      },
      FilterExpression: 'contains (foremans, :username)',
      TableName: 'taskTable-env'
    };

    ddb.scan(params, function(err, data) {
      if (err) {
        res.json({ error: 'Could not load items: ' + err });
      }
      else {
        console.log("Success", data.Items);
        res.json(data.Items);
      }
    })
  }
```

ภาพที่ 4.1.3.4taskFunction/src/app.js – ตัวอย่าง GET Method

เมื่อได้รับ API มาจากผู้พัฒนาส่วนหน้าบ้านแล้วพบว่าเป็น GET Method และ Path = ‘/tasks’ Express จะทำการ Route มาที่ฟังก์ชันดังภาพข้างต้นทันที โค้ดสามารถเข้าถึง username, projectId ที่ผู้พัฒนาส่วนหน้าบ้านส่งมาดังงบรรทัดที่ 41, 42 และหากจะเข้าถึง DynamoDB จำเป็นจะต้องสร้าง params ทุกครั้ง ดังบรรทัดที่ 44 โดยสร้างเป็น JSON ดังภาพ (จำเป็นจะต้องเปิด Document ของ AWS ประกอบ)

ผู้พัฒนาจะอธิบาย Key ได้สังเขปดังนี้

ตารางที่ 4.1.3.1 Key ของ DynamoDB params (GET Method)

Key	คำอธิบาย
ExpressionAttributeValues	ประกาศตัวแปร เพื่อนำไปใช้เขียน FilterExpression
FilterExpression	เงื่อนไขในการ Query, Scan
TableName	ชื่อของ Table ที่ต้องการเข้าถึง

เมื่อสร้าง params สำเร็จแล้วก็สามารถนำไปใช้กับ Object ddb ได้ทันที โดยในภาพจะใช้เป็นฟังก์ชัน scan() ดูได้ที่บรรทัด 52 แล้วหลังจากเรียกใช้ฟังก์ชันก็จะได้ข้อมูลที่ต้องการ Query กลับมาเสร็จแล้ว จะใช้ Arrow Function มี Parameters เป็น err, data

ในบรรทัดที่ 58 ฟังก์ชันได้เรียก Callback เป็น res.json(data.Items) เพื่อส่งข้อมูลย้อนไปหาผู้พัฒนาส่วนหน้าบ้าน

```
// add new task
app.post('/tasks', function(req, res) {
  let id = uuidv4()
  let title = req.body.title
  let startDate = req.body.startDate
  let dueDate = req.body.dueDate
  let image = req.body.image
  let projectmanager = req.body.projectmanager
  let foremans = req.body.foremans
  let description = req.body.description
  let timestamp = Date.now().toString()
  let siteId = req.body.siteId
  let projectId = req.body.projectId
  let finish = false
  let blueprints = req.body.blueprints

  let params = {
    TableName: "taskTable-env",
    Item: {
      "id": id,
      "title": title,
      "startDate": startDate,
      "dueDate": dueDate,
      "projectmanager": projectmanager,
      "foremans": ddb.createSet(foremans),
      "description": description,
      "siteId": siteId,
      "projectId": projectId,
      "image": image,
      "timestamp": timestamp,
      "blueprints": blueprints,
      "finish": finish
    }
  };

  ddb.put(params, function(err) {
    if (err) {
      res.json({ error: 'Could not load items: ' + err });
    }
    res.json({ message: "Item put success." });
  });
});
```

ภาพที่ 4.1.3.4 taskFunction/src/app.js – ตัวอย่าง POST Method

เมื่อได้รับ API มาจากผู้พัฒนาส่วนหน้าบ้านแล้วพบว่าเป็น POST Method และ Path = ‘/tasks’ Express จะทำการ Route มาที่ฟังก์ชันดังภาพข้างต้นทันที โดยสามารถรับข้อมูล Request Body ได้ดังบรรทัดที่ 133 – 145 หลังจากนั้นก็สร้าง params มี Key เป็น TableName, Item โดย JSON Key ที่ชื่อว่า Item นี่คือ Atrributes ของ Item นั่นเอง

บรรทัดที่ 166, 170 เมื่อเตรียมทุกอย่างเสร็จสิ้นก็นำเข้าสู่ ddb.put() หากไม่มี err ก็จะ Callback res.json() เป็นข้อความว่าระบบได้เพิ่ม Item สำเร็จแล้ว

```
app.put('/tasks/:id', function(req, res) {
  let id = req.params.id
  let title = req.body.title
  let startDate = req.body.startDate
  let dueDate = req.body.dueDate
  let description = req.body.description

  let params = {
    TableName: "taskTable-env",
    ExpressionAttributeValues: {
      ':title': title,
      ":startDate": startDate,
      ":dueDate": dueDate,
      ":description": description,
    },
    Key: {
      "id": id
    },
    UpdateExpression: 'set title = :title, startDate=:startDate, dueDate=:dueDate, description=:description',
    ReturnValues: "UPDATED_NEW"
  };

  ddb.update(params, function(err, data) {
    if (err) {
      res.json({ error: 'Could not update items: ' + err });
    }
    res.json({ message: "Item update success.", updated: data });
  });
});
```

ภาพที่ 4.1.3.5 taskFunction/src/app.js – ตัวอย่าง PUT Method

เมื่อได้รับ API มาจากผู้พัฒนาส่วนหน้าบ้านแล้วพบว่าเป็น PUT Method และ Path = ‘/tasks’ Express จะทำการ Route มาที่ฟังก์ชันดังภาพข้างต้นทันที โดยสามารถรับข้อมูล Request Body และ params ได้ดังบรรทัดที่ 180 – 184

หลังจากนั้นก็สร้าง params มี Key ดังบรรทัดที่ 187 – 198 สามารถอธิบายได้ดังตาราง

ตารางที่ 4.1.3.2 Key ของ DynamoDB params (PUT Method)

Key	คำอธิบาย
ExpressionAttributeValues	ประกาศตัวแปร เพื่อนำไปใช้เขียน FilterExpression
TableName	ชื่อของ Table ที่ต้องการเข้าถึง
Key	ระบุ Partition Key และ Sort Key ของ Table ดังกล่าว
UpdateExpression	เงื่อนไขในการ Update
ReturnValues	กำหนดค่าที่จะคืนมาหลัง Update สำเร็จ

บรรทัดที่ 201, 205 เมื่อเตรียมทุกอย่างเสร็จสิ้นก็นำเข้าสู่ ddb.update() หากไม่มี err ก็จะ Callback res.json() เป็นข้อความว่าระบบได้เพิ่ม Item สำเร็จและแนบข้อมูล Item ที่อัพเดทแล้วกลับไป

```
278
279 app.delete('/tasks/:id', function(req, res) {
280   // Add your code here
281   let id = req.params.id
282   var params = {
283     TableName: "taskTable-env",
284     Key: {
285       "id": id,
286     }
287   }
288   ddb.delete(params, function(err, data) {
289     if (err) {
290       res.json({ error: 'Could not load items: ' + err })
291     }
292     else {
293       res.json({ message: 'Delete item succes' })
294     }
295   })
296 })
297
```

ภาพที่ 4.1.3.6 taskFunction/src/app.js – ตัวอย่าง DEL Method

เมื่อได้รับ API มาจากผู้พัฒนาส่วนหน้าบ้านแล้วพบว่าเป็น PUT Method และ Path = ‘/tasks’ Express จะทำการ Route มาที่ฟังก์ชันดังภาพข้างต้นทันที โดยสามารถรับข้อมูล params ได้ดังบรรทัดที่ 281

บรรทัดที่ 282 – 293 สร้าง params และนำเข้าสู่ฟังก์ชัน ddb.delete() หากไม่มี err ก็จะ Callback ส่งข้อความลบสำเร็จกลับไปยังผู้พัฒนาส่วนหน้าบ้าน

```
exports.handler = async(event, context) => {
    console.log(event);
    let date = new Date();
    // If the required parameters are present, proceed
    if (event.userName) {
        var params_group = {
            'GroupName': event.request.userAttributes["custom:role"],
            'UserPoolId': 'ap-southeast-1_IMveYmaeb',
            'Username': event.userName.toString(),
        };
        try {
            await cognitoidentityserviceprovider.adminAddUserToGroup(params_group).promise();
        }
        catch (error) {
            console.error(error);
        }

        // -- Write data to DDB
        let params = {
            Item: {
                'username': event.userName.toString(),
                'phone_number': event.request.userAttributes.phone_number,
                'role': event.request.userAttributes["custom:role"],
                'email': event.request.userAttributes.email,
                'createdAt': date.toISOString().substring(0, 10),
                'image': 'default/profile_pic.png'
            },
            TableName: tableName
        };
        // Call DynamoDB
        try {
            await ddb.put(params).promise()
            console.log("Success");
        }
        catch (err) {
            console.log("Error", err);
        }

        console.log("Success: Everything executed correctly");
        context.done(null, event);
    }
    else {
        // Nothing to do, the user's email ID is unknown
        console.log("Error: Nothing was written to DDB or SQS");
        context.done(null, event);
    }
```

ภาพที่ 4.1.3.7 registerTrigger/src/index.js – Post Cofirmation Trigger Method

จากภาพข้างต้น สามารถอธิบายได้ว่าเมื่อเกิดการ Trigger ใด ๆ ฟังก์ชันที่ทำหน้าที่เป็นตัว Listener จะได้รับ Event เข้าสู่ ฟังก์ชัน exports.handler เสมอ

ภาพข้างต้นจะเกิดขึ้นเมื่อผู้ใช้สมัครสำเร็จและได้ยืนยันตัวตนสำเร็จ เมื่อฟังก์ชันได้รับ event มาแล้ว เราสามารถ console.log(event) ออกมาเช็ค event body ได้เลย หรือจะเปิดจาก AWS Document อ่านประกอบด้วยก็ได้เช่นกัน ผู้พัฒนาได้สร้าง params ของ Cognito ดังบรรทัดที่ 13 และได้เพิ่มผู้ใช้เข้าสู่ Authentication Groups ตามที่ได้สมัครมาเบื้องต้น ดังบรรทัดที่ 19

บรรทัดที่ 26 เมื่อเพิ่มผู้ใช้เข้าสู่ Groups สำเร็จ ก็จะเพิ่มผู้ใช้เข้าสู่ฐานข้อมูลโดยใช้ params ของ DynamoDB

บรรทัดที่ 39 เป็นการเรียกใช้ params ด้วยฟังก์ชัน ddb.put() หากสำเร็จจะขึ้น “Success” บน Console Log

บรรทัดที่ 47 เป็นการจบการทำงานของฟังก์ชัน - ปิด Severless ฟังก์ชัน

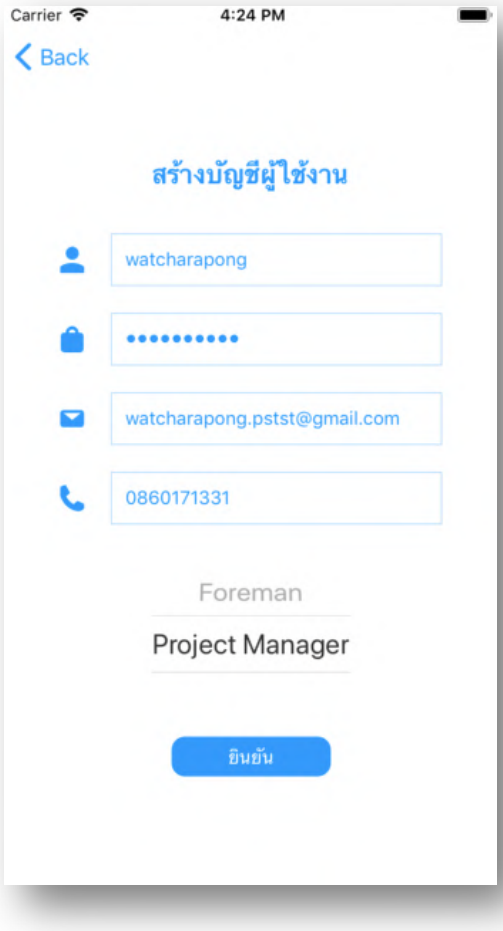
4.2 การทดสอบระบบ

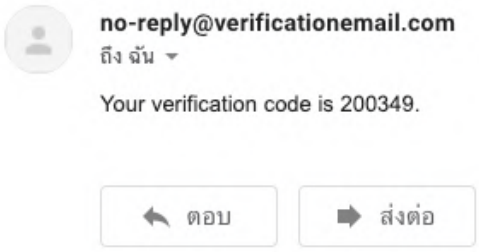
4.2.1 การทดสอบโปรแกรม

ในขั้นตอนการทดสอบระบบนี้ ผู้พัฒนาจะทำการทดสอบความสามารถของระบบจากที่ได้กล่าวไว้ในข้อ 3.2 และ 3.3 นั่นคือ

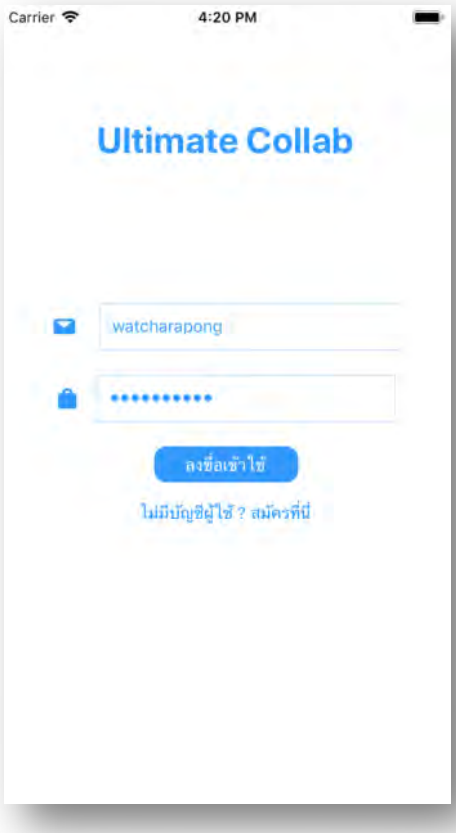
- ผู้ใช้สามารถสมัครสมาชิกเพื่อใช้งานระบบได้
- ผู้ใช้สามารถลงชื่อเข้าใช้งานระบบได้
- ผู้ใช้สามารถ เพิ่ม/แก้ไข รูปภาพส่วนตัวได้
- ผู้ใช้สามารถรับสารแจ้งเตือนขณะใช้แอปพลิเคชันได้
- การเข้าถึงข้อมูลจะขึ้นอยู่กับผู้ใช้และตำแหน่งของผู้ใช้ (Role)
- ผู้ใช้สามารถเรียกข้อมูลจากเซิฟเวอร์ได้
- ผู้ใช้สามารถแก้ไขข้อมูลในเซิฟเวอร์ได้
- ผู้ใช้สามารถเพิ่มข้อมูลเข้าสู่เซิฟเวอร์ได้
- ผู้ใช้สามารถลบข้อมูลออกจากเซิฟเวอร์ได้
- หลังผู้ใช้อัพโหลดภาพแล้ว ภาพจะมีการเขียนวันที่ทับลงไปโดยอัตโนมัติ
- ระบบบีบภาพอัตโนมัติ เพื่อไม่ให้ภาพขนาดใหญ่จนเกินไป และช่วยให้การโหลดรูปเป็นไปได้รวดเร็ว

ตารางที่ 4.2.1.1 แสดงรายละเอียดการทดสอบระบบสมัครสมาชิก

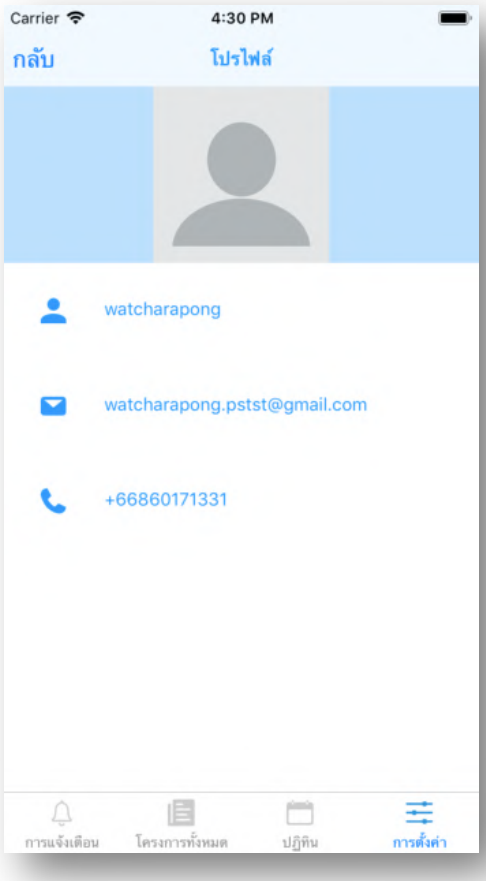
ชื่อการทดสอบ	ทดสอบการสมัครสมาชิก
วัตถุประสงค์	• ผู้ใช้สามารถสมัครสมาชิกได้สำเร็จ • มี Item ของผู้ใช้ปรากฏขึ้นบน userTable • ผู้ใช้ได้รับ Email เป็น Verification Code • ผู้ใช้สามารถยืนยันตัวตนได้
วิธีการทดสอบ	1. เปิดแอปพลิเคชัน 2. กดปุ่ม สมัครสมาชิก 3. กรอกข้อมูลดังนี้ - username: “watcharapong” - password: “adminadmin” - email: “watcharapong.pstst@gmail.com” - phone number: “0860171331” - role: “projectmanager” 

	4. เปิด Email box – Gmail 5. ลงชื่อเข้าใช้ด้วย - username: “watcharapong” - password: “adminadmin” 6. นำ Verification Code ที่ได้รับจาก Email ไปกรอก  7. เช็คตาราง userTable ว่ามี User ที่ชื่อว่า watcharapong ไหม
ผลการทดสอบ	ทำงานได้ตามวัตถุประสงค์

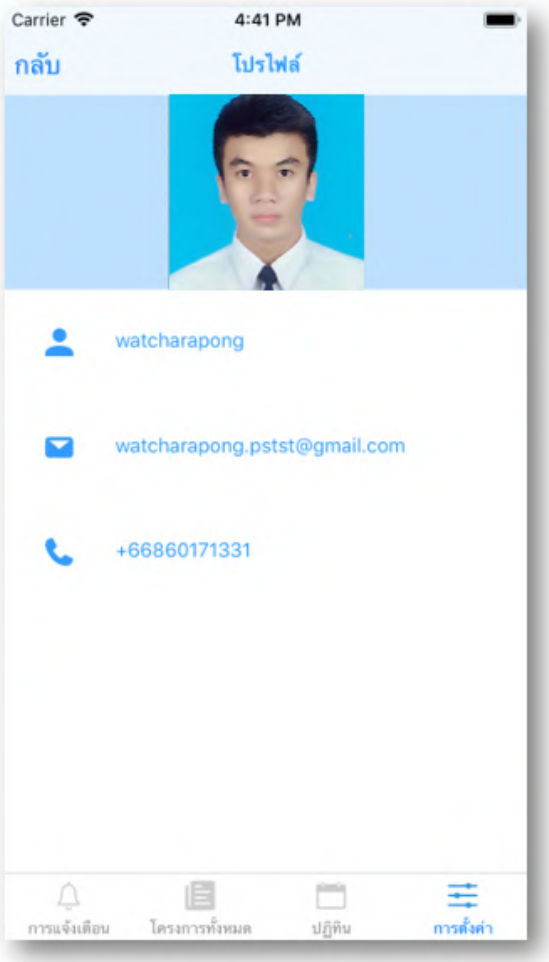
ตารางที่ 4.2.1.2 แสดงรายละเอียดการทดสอบระบบลงชื่อเข้าใช้

ชื่อการทดสอบ	ทดสอบระบบลงชื่อเข้าใช้
วัตถุประสงค์	• ผู้ใช้สามารถสมัครสมาชิกได้สำเร็จ • ผู้ใช้สามารถเรียก Credentials ได้
วิธีการทดสอบ	1. เปิดแอปพลิเคชัน 2. กรอกข้อมูล - username: “watcharapong” - password: “adminadmin”  3. กดปุ่มลงชื่อเข้าใช้ 4. เช็ค Console ว่าข้อมูล Credentials ของผู้ใช้ watcharapong ขึ้นไหม ?
ผลการทดสอบ	ทำงานได้ตามวัตถุประสงค์

ตารางที่ 4.2.1.3 แสดงรายละเอียดการทดสอบระบบสมัครสมาชิก

ชื่อการทดสอบ	ทดสอบการสมัครสมาชิก
วัตถุประสงค์	• ผู้ใช้สามารถ เพิ่ม/แก้ไข รูปภาพส่วนตัวได้
วิธีการทดสอบ	1. เปิดแอปพลิเคชัน 2. ไปที่หน้า “ตั้งค่า” 3. กดปุ่ม “โปรไฟล์” 4. กดที่รูปภาพด้านบน  5. เลือกภาพจากอุปกรณ์

6. กดปุ่มยืนยัน
7. เช็คภาพด้านบนว่าได้เปลี่ยนเป็นภาพล่าสุดหรือยัง

	
ผลการทดสอบ	ทำงานได้ตามวัตถุประสงค์

ตารางที่ 4.2.1.4 แสดงรายละเอียดการทดสอบการรับสารแจ้งเตือนขณะใช้แอปพลิเคชัน

ชื่อการทดสอบ	ทดสอบการรับสารแจ้งเตือนขณะใช้แอปพลิเคชัน
วัตถุประสงค์	• ผู้ใช้สามารถรับสารแจ้งเตือน • สารแจ้งเตือนถูกต้อง
วิธีการทดสอบ	1. เปิดแอปพลิเคชันบนอุปกรณ์มือถือจริง ๆ (ไม่ใช่ Simulator) 2. เปิด Notification ให้กับแอปพลิเคชันในระบบปฏิบัติการ 3. ทดลองสร้าง Notification โดยการเพิ่มไปในระบบ Console หลังบ้าน 4. เช็ค Notification ที่หน้าจออุปกรณ์มือถือ
ผลการทดสอบ	ทำงานได้ตามวัตถุประสงค์

ตารางที่ 4.2.1.5 แสดงรายละเอียดการทดสอบการเข้าถึงข้อมูลของตำแหน่งของผู้ใช้

ชื่อการทดสอบ	ทดสอบการเข้าถึงข้อมูลของตำแหน่งของผู้ใช้
วัตถุประสงค์	● ผู้ใช้ตำแหน่ง Foreman ไม่สามารถสร้างงานได้ เว้นแต่ Subtask ● ผู้ใช้ตำแหน่ง Project Manager สามารถใช้งานแอปพลิเคชันได้ครบทุกส่วน ● ผู้ใช้สามารถมองเห็นข้อมูลของตนเองได้ ● ผู้ใช้ไม่สามารถมองเห็นข้อมูลที่ตนไม่เกี่ยวข้อง
วิธีการทดสอบ	<u>การทดสอบสิทธิ์</u> 1. เปิดแอปพลิเคชัน 2. ลงชื่อเข้าใช้ username: “watcharapong” password: “adminadmin” (โดย Username “watcharapong” อยู่ในตำแหน่ง Project Manager) 3. ทดลองสร้าง - Project --> จะต้องสร้างได้ - Task --> จะต้องสร้างได้ - Subtask --> จะต้องสร้างได้ - Comment --> จะต้องสร้างได้ 4. ลงชื่อออก 5. ลงชื่อเข้าใช้ username: “kuptipong” password: “adminadmin” (โดย Username “kuptipong” อยู่ในตำแหน่ง Foreman) 6. ทดลองสร้าง - Project --> จะต้องสร้างไม่ได้ - Task --> จะต้องสร้างไม่ได้ - Subtask --> จะต้องสร้างได้ - Comment --> จะต้องสร้างได้ 7. ลงชื่อออก <u>การทดสอบการมองเห็นข้อมูล</u> 1. เพิ่มผู้ใช้ kuptipong ไปยัง Project “ทดสอบ 1” โดยใช้ Amazon DynamoDB Console 2. เปิดแอปพลิเคชัน

	3. ลงชื่อเข้าใช้ username: “kuptipong” password: “adminadmin” 4. เข้าไปยังหน้าโปรเจค ตรวจสอบว่าพบ Project “ทดสอบ 1” ไหม ? 5. ลบ kuptipong ออกจาก Project “ทดสอบ 1” โดยใช้ Amazon DynamoDB Console 6. รีเฟรชหน้า Project อีกครั้ง ตรวจสอบว่าพบ Project “ทดสอบ 1” ไหม ?
ผลการทดสอบ	ทำงานได้ตามวัตถุประสงค์

ตารางที่ 4.2.1.6 แสดงรายละเอียดการทดสอบการเรียกข้อมูลจากเซิฟเวอร์

ชื่อการทดสอบ	ทดสอบการเรียกข้อมูลจากเซิฟเวอร์
วัตถุประสงค์	● ผู้ใช้สามารถรับข้อมูลจากเซิฟเวอร์ได้
วิธีการทดสอบ	1. เปิดแอปพลิเคชัน 2. ทดสอบได้สองแบบ - ไล่ทำตาม Scenario โดยเริ่มจาก Project Manager เพิ่มผู้ใช้ที่ต้องการทดสอบ ไปยัง Site, Project, Task, Subtask แล้วค่อย ๆ เปิดเช็คความถูกต้องทีละหน้า - ทดสอบโดยการนำโค้ด API.get() ไปวางในฟังก์ชัน componentDidMount() ของ React Application กรอกค่า resource, path และ params จากนั้น console.log() ค่าออกมา 3. ตรวจสอบข้อมูลจาก Console ของแอปพลิเคชัน
ผลการทดสอบ	ทำงานได้ตามวัตถุประสงค์

ตารางที่ 4.2.1.7 แสดงรายละเอียดการทดสอบการแก้ไขข้อมูลบนเซิฟเวอร์

ชื่อการทดสอบ	ทดสอบการแก้ไขข้อมูลบนเซิฟเวอร์
วัตถุประสงค์	● ผู้ใช้สามารถแก้ไขข้อมูลบนเซิฟเวอร์ได้
วิธีการทดสอบ	1. เปิดแอปพลิเคชัน 2. ทดสอบได้สองแบบ - ไล่ทำตาม Scenario โดยเริ่มจาก Project Manager แก้ไขข้อมูลของ Project, Task, Subtask - ทดสอบโดยการนำโค้ด API.put() ไปวางในฟังก์ชัน componentDidMount() ของ React Application กรอกค่า resource, path, params และ body จากนั้น console.log() ค่าออกมา 3. ตรวจสอบข้อมูลจาก Console ของแอปพลิเคชัน 4. ตรวจสอบข้อมูลจาก Amazon DynamoDB Console
ผลการทดสอบ	ทำงานได้ตามวัตถุประสงค์

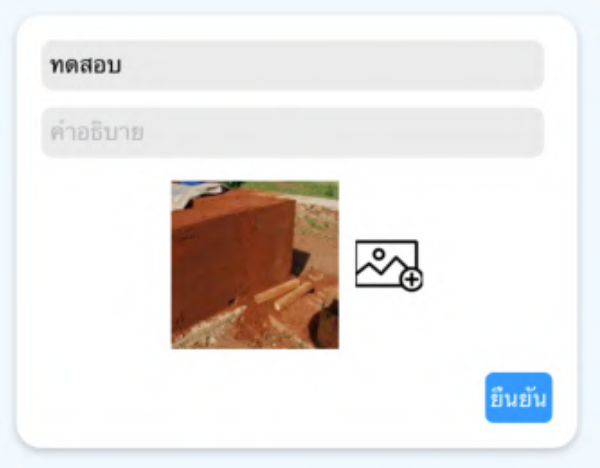
ตารางที่ 4.2.1.8 แสดงรายละเอียดการทดสอบการเพิ่มมูลไปยังเซิฟเวอร์

ชื่อการทดสอบ	ทดสอบการเพิ่มข้อมูลไปยังเซิฟเวอร์
วัตถุประสงค์	● ผู้ใช้สามารถเพิ่มข้อมูลไปยังเซิฟเวอร์ได้
วิธีการทดสอบ	1. เปิดแอปพลิเคชัน 2. ทดสอบได้สองแบบ - ไล่ทำตาม Scenario โดยเริ่มจาก Project Manager เพิ่มข้อมูลที่ต้องการทดสอบ ไปยัง Site, Project, Task, Subtask - ทดสอบโดยการนำโค้ด API.post() ไปวางในฟังก์ชัน componentDidMount() ของ React Application กรอกค่า resource, path, params และ body จากนั้น console.log() ค่าออกมา 3. ตรวจสอบข้อมูลจาก Console ของแอปพลิเคชัน 4. ตรวจสอบข้อมูลจาก Amazon DynamoDB Console
ผลการทดสอบ	ทำงานได้ตามวัตถุประสงค์

ตารางที่ 4.2.1.9 แสดงรายละเอียดการทดสอบการลบข้อมูลออกจากเซิฟเวอร์

ชื่อการทดสอบ	ทดสอบการลบข้อมูลออกจากเซิฟเวอร์
วัตถุประสงค์	● ผู้ใช้สามารถลบข้อมูลออกจากเซิฟเวอร์ได้
วิธีการทดสอบ	1. เปิดแอปพลิเคชัน 2. ทดสอบได้สองแบบ - ไล่ทำตาม Scenario โดยเริ่มจาก Project Manager เพิ่มข้อมูลที่ต้องการทดสอบการลบ ไปยัง Site, Project, Task, Subtask แล้วทดลองลบทีละข้อมูลที่เพิ่มเข้าไป - ทดสอบโดยการนำโค้ด API.del() ไปวางในฟังก์ชัน componentDidMount() ของ React Application กรอกค่า resource, path และ params จากนั้น console.log() ค่าออกมา 3. ตรวจสอบข้อมูลจาก Console ของแอปพลิเคชัน 4. ตรวจสอบข้อมูลจาก Amazon DynamoDB Console
ผลการทดสอบ	ทำงานได้ตามวัตถุประสงค์

ตารางที่ 4.2.1.10 แสดงรายละเอียดการทดสอบระบบลายน้ำ

ชื่อการทดสอบ	ทดสอบระบบลายน้ำ
วัตถุประสงค์	● รูปภาพมีลายน้ำวันที่ หลังจากอัพโหลดไปยังเซิฟเวอร์
วิธีการทดสอบ	1. เปิดแอปพลิเคชัน 2. ตรวจสอบภาพที่ต้องการทดสอบ 3. เพิ่มรูปภาพที่ต้องการทดสอบไปยัง Subtask  4. เปิด Amazon S3 แล้วไปยัง Path ของภาพต้นแบบ

	5. ดาวน์โหลดภาพ 6. ตรวจสอบลายน้ำของภาพ 
ผลการทดสอบ	ทำงานได้ตามวัตถุประสงค์

ตารางที่ 4.2.1.11 แสดงรายละเอียดการทดสอบระบบบีบภาพ

ชื่อการทดสอบ	ทดสอบระบบบีบภาพ
วัตถุประสงค์	● รูปภาพมีขนาดเล็กไฟล์ลง หลังจากอัพโหลดไปยังเซิฟเวอร์
วิธีการทดสอบ	1. เปิดแอปพลิเคชัน 2. ตรวจสอบขนาดภาพที่ต้องการทดสอบ บันทึกไว้ 3. เพิ่มรูปภาพที่ต้องการทดสอบไปยัง Subtask 4. เปิด Amazon S3 แล้วไปยัง Path ของภาพต้นแบบ 5. ดาวน์โหลดภาพ 6. ตรวจสอบขนาดภาพ 7. เปรียบเทียบขนาดและคุณภาพจากภาพต้นแบบ
ผลการทดสอบ	ทำงานได้ตามวัตถุประสงค์

4.2.2 การทดสอบเชิงประสิทธิภาพ

ผู้พัฒนาต้องการที่จะทดสอบ Performance Testing เพื่อตอบโจทย์ Non-Functional Requirement ที่ลูกค้าต้องการระบบที่มีความเสถียรภาพ โดยอ้างอิงจากหัวข้อที่ 3.3 ดังนี้

- สามารถรองรับการประมวลผลที่มากขึ้นได้ในอนาคต เมื่อมีผู้ใช้และจำนวนงานมากขึ้น
- ระบบสามารถประมวลผลได้รวดเร็วและตอบสนองได้ทันท่วงทีต่อการใช้งาน
- ระบบสามารถรองรับผู้ใช้จำนวนมากในเวลาเดียวกันต่องานเดียวกันได้
- ระบบสามารถตอบสนองต่องานที่ผู้ใช้หลาย ๆ คนแก้ไขงานเดียวกันได้ถูกต้อง

4.2.2.1 ทดสอบด้วยการสร้างปริมาณการใช้งานเข้าไปและวัดการตอบสนองของระบบ (Load Testing)

เนื่องจาก AWS เป็นระบบหลังบ้านที่ใช้จ่ายตามจริง ผู้พัฒนาจึงไม่สามารถสร้างปริมาณการใช้งานจำนวนมหาศาลได้มากนัก เนื่องจากค่าใช้จ่ายสูง ผู้พัฒนาได้ออกแบบ Code สำหรับการเรียกชุดข้อมูลเป็นจำนวน Requests ได้แก่ 10, 100, 1,000, 10,000 จำนวนการใช้งาน โดยเป็นการทำงานแบบ Asynchronous

```
multiuserTestGet = async () => {
  var i;
  let id = "f9775719-5630-49b0-aba3-ab9e83ca0b9c"
  for (i = 1; i < 5001; i++) {
    let count = i
    API.get("taskAPI", "/tasks/byid?id=" + id)
      .then(async data => {
        var date = new Date()
        var hour = date.getHours();
        var min = date.getMinutes();
        var sec = date.getSeconds();
        var msec = date.getMilliseconds();
        var time = hour + ":" + min + ":" + sec + ":" + msec
        console.log("ครั้งที่:", count, "เวลาจบ:", time, "ข้อมูล:", data.description)
      })
  }
}
```

ภาพที่ 4.2.2.1.1 โค้ดการทำงาน Load Testing

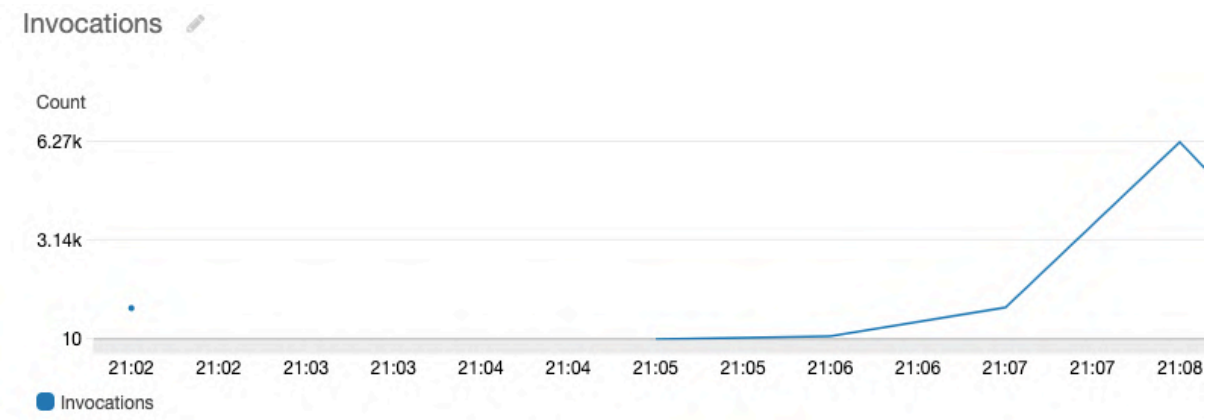
สามารถอธิบายโค้ดได้ดังนี้

บรรทัดที่ 205 ประกาศตัวแปร id ของงานที่ต้องการ GET

บรรทัดที่ 206 วนลูปเป็นจำนวน 5,000 ครั้ง

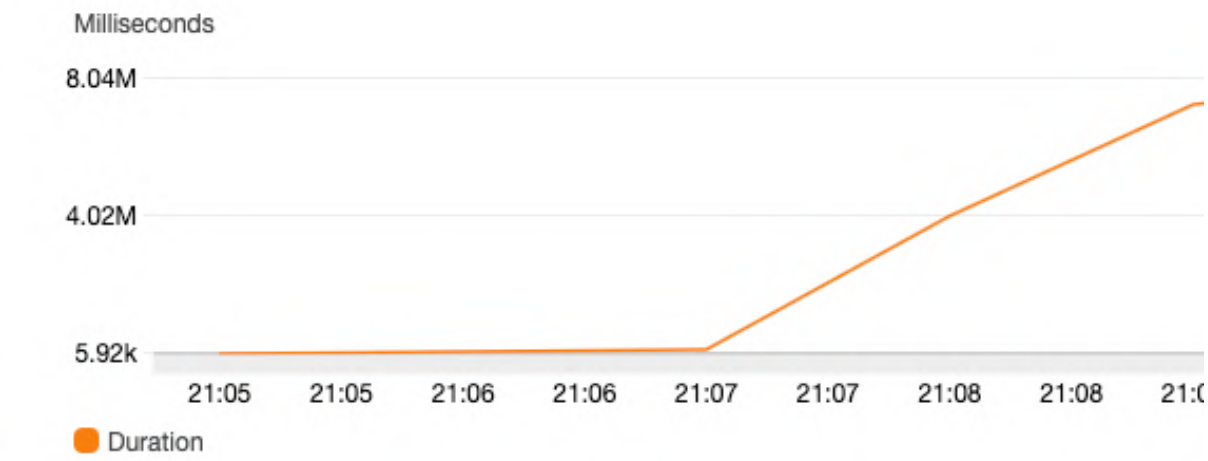
บรรทัดที่ 208 เรียก GET API

เมื่อการทดสอบเสร็จสิ้น สามารถวาดกราฟออกมาได้ดังนี้



ภาพที่ 4.2.2.1.2 กราฟจำนวนการเรียกใช้ Task Function

จากกราฟ การทำงานที่ 10, 100, 1000 การเรียกใช้ สามารถทำงานได้ปกติ แต่จะเห็นได้ว่ากราฟจะเริ่มคอขวดที่จำนวนการเรียกใช้ที่ 6,000 ครั้ง (เวลา 21:08 นาฬิกา)



ภาพที่ 4.2.2.1.3 กราฟระยะเวลาการทำงานของ Task Function

ระยะเวลาการทำงานเริ่มทำงานนานสูงถึง 4,000,000 milliseconds ในช่วงการเรียกใช้ที่ 6,000 ครั้ง (เวลา 21:07 – 21:08 นาฬิกา)

- การทำงานที่ 10 ครั้ง ใช้เวลา 5,920 milliseconds

- การทำงานที่ 100 ครั้ง ใช้เวลา 58,300 milliseconds

- การทำงานที่ 1,000 ครั้ง ใช้เวลา 158,000 milliseconds

ภาพที่ 4.2.2.1.4 กราฟจำนวน Error ของ Task Function

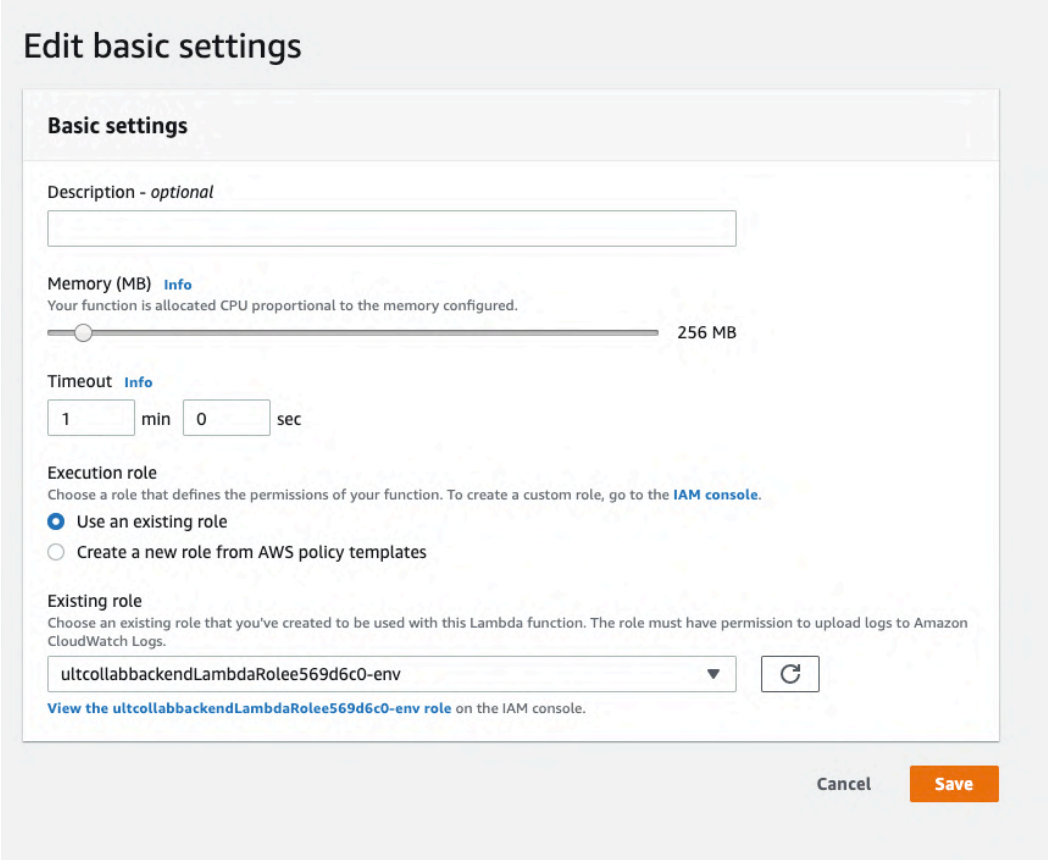
จะพบได้ว่าช่วงการทำงานที่ 6,000 การเรียกใช้ เกิด Error จำนวน 69 ครั้ง และเกิดขึ้นสูงถึง 171 ครั้ง ในช่วงเวลาถัดไป (เวลา 21:08 เป็นต้นไป)

ผู้พัฒนาจึงตรวจสอบต้นตอและสาเหตุพบว่าปัญหาคอขวดเกิดจากสาเหตุหลักดังนี้

- เกิดจากคอขวดที่ฐานข้อมูล DynamoDB
- เกิดจากคอขวดที่การตั้งค่า Lambda
- เกิดจากคอขวดที่จำนวน Concurrency

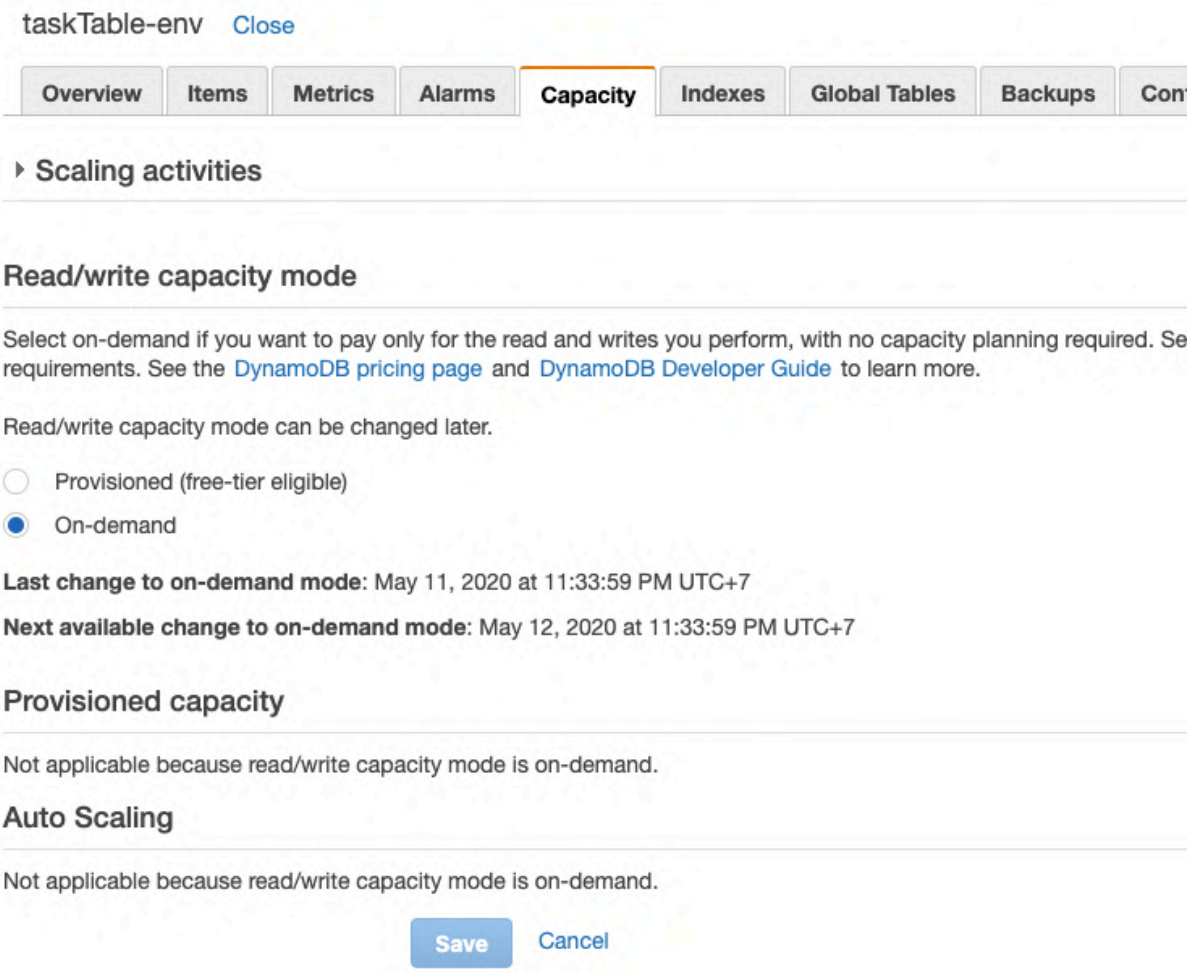
ในการเพิ่ม Concurrency จำเป็นจะต้องติดต่อทาง Amazon ไปเพื่อเปลี่ยนแผนการใช้งาน ผู้พัฒนาไม่สามารถทำในส่วนนี้ได้เพราะใช้บัญชี AWS ที่เป็นงบประมาณของภาควิชา

การแก้ปัญหาคอขวดที่การตั้งค่า Lambda ซึ่งจะเป็นการตั้งค่า Memory และ Timeout โดยผู้พัฒนาเพิ่ม Memory จาก 128 เป็น 256 และเพิ่มระยะเวลา Timeout จาก 30 วินาที เป็น 1 นาที สามารถดำเนินการแก้ไขได้ดังภาพที่ 4.2.2.1.5



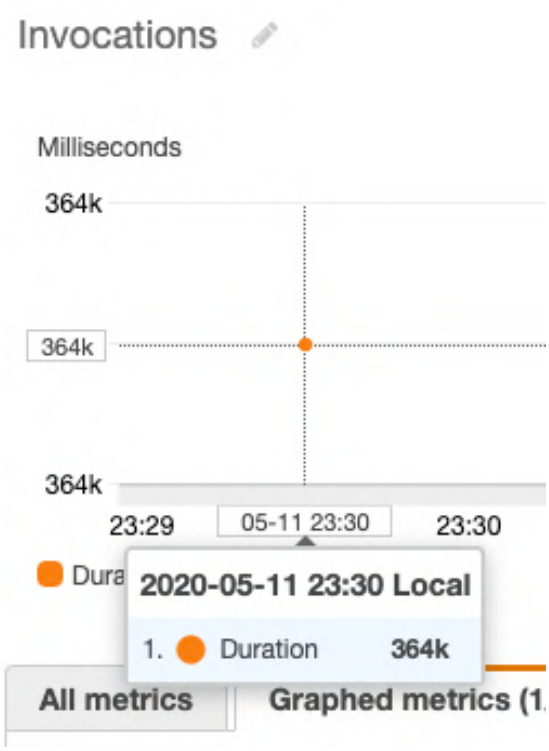
ภาพที่ 4.2.2.1.5 การตั้งค่า Lambda

ในส่วนการแก้ไขฐานข้อมูล DynamoDB นั้นผู้พัฒนาได้เพิ่มลิมิต Read / Write Capacity จาก 5 unit เป็น On-demand (หรือก็คือตามการใช้งาน ยิ่งใช้งานมาก็ยิ่งจ่ายแพง ไม่มีลิมิตการใช้งาน) ดังภาพที่ 4.2.2.1.6



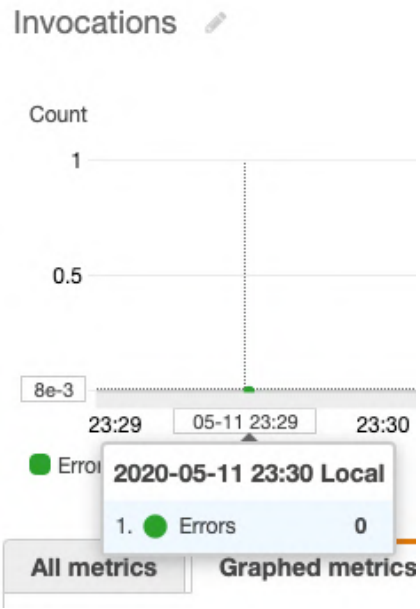
ภาพที่ 4.2.2.1.6 การตั้งค่า DynamoDB

หลักจากปรับแก้ไขทั้งสองปัญหาคอขวดแล้ว ก็ลองทดสอบเรียกใช้ฟังก์ชันจำนวน 10,000 ครั้ง ได้ผลทดสอบดังภาพที่ 4.2.2.1.7



ภาพที่ 4.2.2.1.7 กราฟระยะเวลาการทำงานของ Task Function หลังปรับแก้ไขระบบ

จะเห็นได้ว่า ระยะการทำงานต่อ 10,000 การเรียกใช้ ลดลงเหลือ 364,000 milliseconds จาก ตอนแรกที่ยังไม่ได้แก้ไขระบบที่ใช้ระยะการทำงานประมาณ 10,000,000 milliseconds ซึ่งสามารถทำงานได้มีประสิทธิภาพยิ่งขึ้นเมื่อมีผู้ใช้มากขึ้น (ผู้ใช้มีการเรียกใช้มากกว่า 6,000 ครั้งพร้อมกันใน 1 เวลา) ประมาณ 27 เท่าตัว และจำนวน Error ลดลงเหลือ 0 ดังภาพที่ 4.2.2.1.8



ภาพที่ 4.2.2.1.8 กราฟจำนวน Error ของ Task Function หลังปรับแก้ไขระบบ

ดังนั้นจึงสามารถสรุปได้ว่า ระบบทำงานเสถียรและไม่มีปัญหาหลังจากการแก้ไขในข้างต้น สามารถเขียนตารางสรุปได้ดังนี้

ตารางที่ 4.2.2.1.1 ตารางสรุปผลการทำงานของ Load Testing หลังแก้ไขระบบ

จำนวนการเรียกใช้ (ครั้ง)	ระยะเวลาการทำงาน (milliseconds)	จำนวน Error
10	5,920	0
100	58,300	0
1,000	158,000	0
10,000	364,000	0

4.2.2.2 ทดสอบการจำลองสถานการณ์ผู้ใช้จำนวนมากแก้ไขข้อมูลพร้อมกัน (Multi-user Testing)

ผู้พัฒนาได้ออกแบบการทดสอบเพื่อตรวจสอบประสิทธิภาพของระบบ เมื่อผู้ใช้หลาย ๆ คน แก้ไขฐานข้อมูลเดียวกันพร้อมกันใน 1 เวลา โดยผู้พัฒนาได้ใช้ผู้ใช้จำนวน 40 คน เข้ามาแก้ไขงานจำนวน 1 งาน และเรียกค่าเพื่อเช็คผล สามารถเขียนโค้ดตัวอย่างได้ดังนี้

```
multiuserTestPut = async (username) => {
  let id = "f9775719-5630-49b0-aba3-ab9e83ca0b9c"
  Auth.signIn(username, "12345678").then(user => {
    let req = {
      body: {
        title: "Multi-users Testing",
        startDate: "2020-05-02",
        dueDate: "2020-06-02",
        description: "Hello, " + username,
      }
    }
    API.put("taskAPI", "/tasks/" + id, req).then(data => {
      if (data.title != '') {
        var date = new Date()
        var time = date.getHours() + ":" + date.getMinutes() + ":"
          + date.getSeconds() + ":" + date.getMilliseconds()
        console.log("PUT:", "เวลา:", time, "ผู้ใช้:", user.username,
          "ข้อมูลที่เปลี่ยนแปลง:", data.updated.Attributes.description)
        API.get("taskAPI", "/tasks/byid?id=" + id)
          .then(async data => {
            var date = new Date()
            var time = date.getHours() + ":" + date.getMinutes() + ":"
              + date.getSeconds() + ":" + date.getMilliseconds()
            console.log("GET:", "เวลา:", time, "ข้อมูล:", data.description)
          })
      } else {
        console.log("ERROR", "ผู้ใช้ที่ ERROR:", username)
      }
    })
  })
}
```

ภาพที่ 4.2.2.2.1 โค้ดตัวอย่างการทำงาน Multi-user Testing

จากภาพที่ 4.2.2.2.1 สามารถอธิบายโค้ดได้ดังนี้

บรรทัดที่ 167 ประกาศตัวแปร id ของงานที่ต้องการแก้ไข

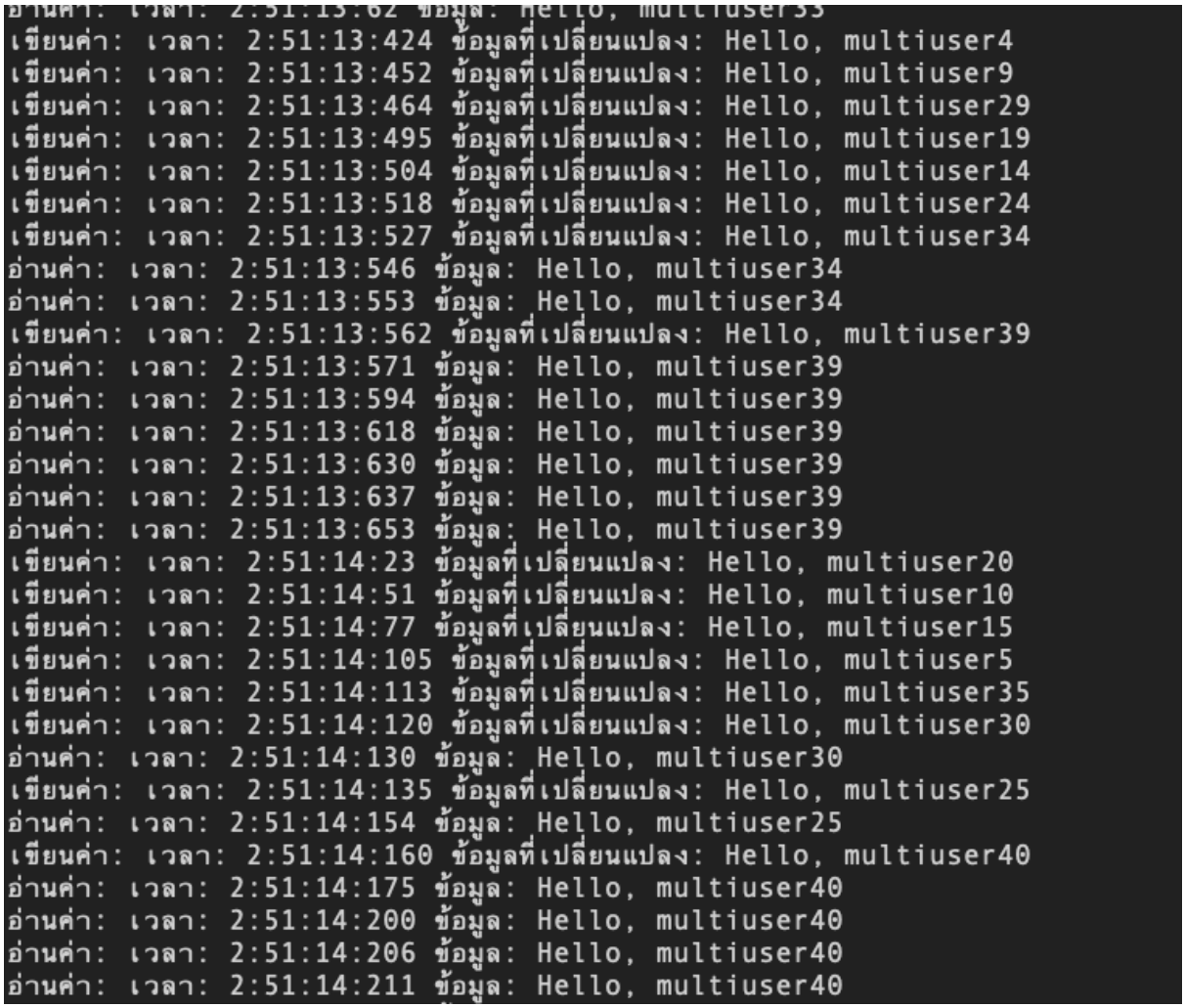
บรรทัดที่ 168 เป็นการลงชื่อเข้าใช้ ผู้ใช้ที่ระบุ

บรรทัดที่ 169 ประกาศ Parameter ที่จะใช้ PUT

บรรทัดที่ 177 เรียกใช้ PUT เพื่อแก้ไขงาน

บรรทัดที่ 184 เรียกใช้ GET เพื่อเช็คค่าของงาน

โดยทำการรันโค้ดบนอุปกรณ์จริงจำนวน 8 เครื่อง แบบสุ่มเวลา หลังจากทดสอบเสร็จสิ้น สามารถเช็ค Log ออกมาได้ดังรูปที่ 4.2.2.2.2

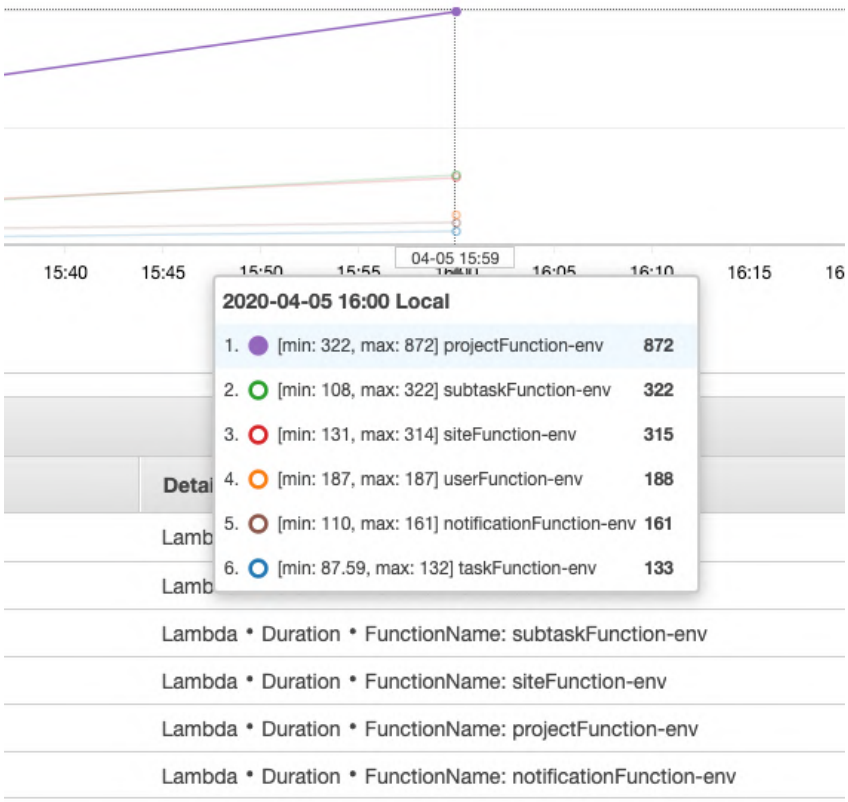


ภาพที่ 4.2.2.2.2 Log การทำงานของ User 1 - 40

จากภาพที่ 4.2.2.2.2 จะสามารถอธิบายได้ว่า เมื่อผู้ใช้เปลี่ยนแปลงค่า และเรียกค่าพร้อม ๆ กันเป็นจำนวน 40 คน ระบบสามารถตอบสนองได้ทันและถูกต้องในหน่วย millisecond

4.2.2.3 ทดสอบกับผู้ใช้จริง (User Testing)

การทดสอบผู้พัฒนาได้ Deploy ระบบหลังบ้านและ Build ไฟล์ APK ให้ลูกค้าจากบริษัท ปริญสิริ และ บริษัท Team Built จำนวนทั้งหมด 13 คน ที่ใช้อุปกรณ์ที่มีระบบปฏิบัติการ Android ดาวน์โหลดและติดตั้งเพื่อทดลองใช้แบบอิสระเป็นเวลา 1 วัน สามารถสรุปผลได้ดังนี้



ภาพที่ 4.2.2.3.1 กราฟแสดงระยะเวลาประมวลผลเฉลี่ยของ API ช่วงเวลา 16.00 นาฬิกา

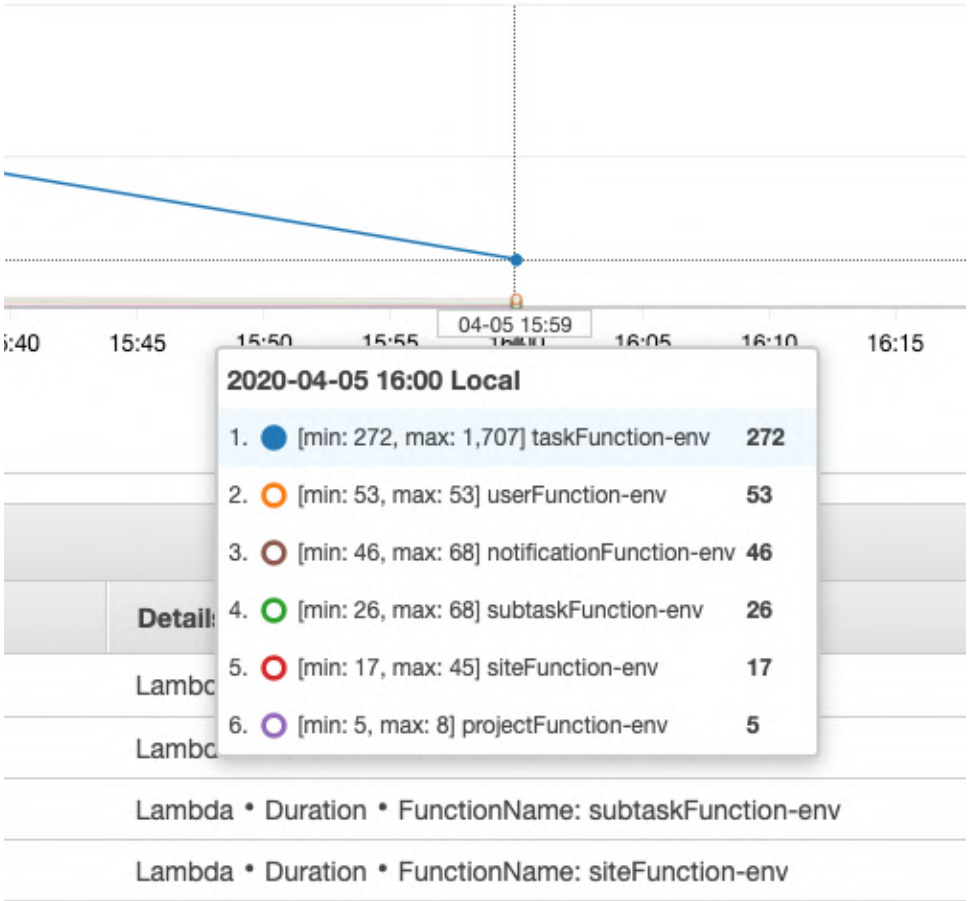
จากภาพข้างต้นผู้พัฒนาวัดประสิทธิภาพจาก API ทั้งหมด โดยวัดจากระยะเวลาประมวลผลเฉลี่ยทั้งหมด ในหน่วย millisecond

โดยฟังก์ชันที่วัดผล ได้แก่ siteFunction, projectFunction, taskFunction, subtaskFunction, userFunction และ notificationFunction โดยผู้พัฒนาต้องการวัดประสิทธิภาพจากช่วงชั่วโมงที่มีการเรียกใช้เยอะที่สุด นั่นคือช่วง 16.00 นาฬิกา

จะสามารถสรุปได้ว่า projectFunction มีระยะเวลาการทำงานเฉลี่ยสูงถึง 872 ms ต่อการเรียกใช้ ผู้พัฒนาพบว่า หากผู้ใช้มีจำนวน Project สูง ก็มีโอกาสที่จะ Query เป็นเวลานาน ผู้พัฒนาจำเป็นจะต้องแก้ไข Optimization ในส่วนของการ Query และการเก็บข้อมูลให้ใช้เวลาเร็วขึ้น

อีกทั้งยังเกิดจากผู้พัฒนาหน้าบ้านที่ออกแบบการเรียกบางข้อมูลที่ไม่จำเป็นทำให้ระบบหลังบ้านใช้เวลานานในการค้นหาข้อมูลเหล่านี้ ยกตัวอย่างเช่น หน้า siteScreen ที่เรียกแค่ชื่อของ Project, Description, Due Date, Start Date ก็เพียงพอต่อการแสดงผล

และด้วยอุปสรรคโรคระบาด Covid-19 ที่ทำให้ผู้พัฒนาไม่สามารถเข้าไปดูแลการทดสอบได้อย่างเต็มที่ ทำให้ผู้ใช้สงสัยและไม่เข้าการใช้งานบางหน้าได้ เช่น หน้าเพิ่ม Task ที่ลืมเพิ่ม Foreman ทำให้เกิดการเรียกใช้ซ้ำ ๆ (เพื่อหางาน) จากทางฝั่ง Foreman ได้ จึงเกิดการเรียกใช้สูงถึง 272 ครั้ง ดังรูปที่ 4.2.2.3.2



ภาพที่ 4.2.2.3.2 กราฟแสดงจำนวนการเรียกใช้ API ช่วงเวลา 16.00 นาฬิกา

4.2.3 การทดสอบโดยผู้ใช้บริการ

เนื่องจากการพัฒนาครั้งนี้เป็นการพัฒนาระบบหลังบ้านทั้งสิ้น โดยผู้ใช้บริการหลักของเราจะเป็นนักพัฒนาในส่วนหน้าบ้าน จึงได้เรียนเชิญนักพัฒนาในส่วนหน้าบ้านจำนวน 2 คน จากทั้งหมด 2 คน มาทดสอบ ได้ผลการทดสอบดังนี้

ตารางที่ 4.2.3.1 ผลการทำสอบโดยผู้ใช้บริการ

ประเด็น / หัวข้อ การพิจารณา	ใช่	ไม่
1. การทดสอบฟังก์ชันการทำงาน		
1.1 การดึงข้อมูลจากระบบหลังบ้านไปแสดงผล สามารถแสดงผลได้อย่างถูกต้อง	2 คน	
1.2 คำร้องขอสามารถส่งมายังระบบหลังบ้านได้อย่างถูกต้อง	2 คน	
1.3 ระบบ Login สามารถทำงานได้อย่างถูกต้อง	2 คน	
1.4 ระบบ Register สามารถทำงานได้อย่างถูกต้อง	2 คน	
1.5 Verification Code สามารถส่งไปยัง Email ได้ถูกต้อง	2 คน	
1.6 ผู้พัฒนาสามารถเรียก Credentials ของผู้ใช้ได้อย่างถูกต้อง	2 คน	
1.7 อีเมลยันยันตัวตนมีเนื้อหาถูกต้อง	2 คน	
1.8 ผู้พัฒนาสามารถอัพโหลดรูปภาพได้อย่างถูกต้อง	2 คน	
1.9 ผู้พัฒนาสามารถเรียก URL ของรูปภาพได้อย่างถูกต้อง	2 คน	
1.10 แอปพลิเคชันสามารถรับ Notification ได้อย่างถูกต้อง	2 คน	
1.11 ผู้พัฒนาสามารถเรียกใช้ API ได้ครับทุกฟังก์ชัน		
1.11.1 GET API	2 คน	
1.11.2 POST API	2 คน	
1.11.3 PUT API	2 คน	
1.11.4 DELETE API	2 คน	
1.12 ระบบสามารถแสดงผลข้อความ Error ได้อย่างถูกต้อง	2 คน	

ตารางที่ 4.2.3.2 ผลการทำสอบโดยผู้ใช้บริการ (ต่อ)

ประเด็น / หัวข้อ การพิจารณา	ดีเยี่ยม (5)	ดี (4)	ปานกลาง (3)	พอใช้ (2)	ควรปรับปรุง (1)	ค่าเฉลี่ย (คะแนน)
2. การทดสอบการใช้งานภาพรวมของระบบ						
2.1 ความเร็วในการเรียกใช้ / ตอบสนอง	1 คน	1 คน				4.5
2.2 ความเสถียรของระบบ		1 คน	1 คน			3.5
2.3 ความง่ายในการเรียกใช้		1 คน	1 คน			3.5
2.4 ความง่ายในการเข้าใจระบบ		2 คน				4
2.5 API, ฐานข้อมูล						
2.5.1 ความเหมาะสมต่อระบบ		2 คน				4
2.5.2 เข้าใจง่าย	1 คน		1 คน			4
2.5.3 แก้ไข / เพิ่มเติมง่าย	1 คน	1 คน				4.5
2.5.4 ความครบของข้อมูล		2 คน				4
2.6 คุณภาพในการเก็บรูปภาพ	1 คน	1 คน				4.5
2.7 ความแม่นยำของ Notification		1 คน	1 คน			3.5

บทที่ 5
ข้อสรุปและข้อเสนอแนะ

5.1 ข้อสรุป

ผู้พัฒนาได้เพื่อสร้างแอปพลิเคชันช่วยในการตรวจสอบคุณภาพสำหรับกระบวนการก่อสร้าง ซึ่งแอปพลิเคชันนี้ช่วยในการติดต่อสื่อสารภายในองค์กรและบริษัท และลดข้อผิดพลาดในการติดต่อสื่อสาร สร้างความเข้าใจที่ตรงกัน สามารถติดตามงานได้ทันท่วงที ซึ่งทำให้วิศวะกรระดับสูงไม่ต้องลงพื้นที่หน้างานด้วยตัวเองบ่อยครั้ง ทางผู้พัฒนาได้มีการทดสอบเพื่อตรวจสอบให้ตรงกับความต้องการของผู้ใช้งาน ซึ่งผลการทดสอบนั้นแอปพลิเคชันสามารถทำงานได้ตรงความต้องการของผู้ใช้งานทั้งด้าน Functional Requirement และ Non-Functional Requirement สำหรับสิ่งที่ผู้พัฒนาได้เรียนรู้จากการทำแอปพลิเคชันนี้คือ ได้รับทักษะการออกแบบฐานข้อมูลแบบ NoSQL ได้เรียนรู้ Tools ใหม่ ๆ ในการทำแอปพลิเคชัน และความรู้เรื่องของ API, Cloud Storage และ Cloud Computing ที่กำลังถูกใช้อย่างแพร่หลายในปัจจุบัน

5.2 ปัญหาและอุปสรรค

1. ปัญหาด้านความรู้ในการพัฒนาซอฟต์แวร์ที่ AWS นั้นมี Services มากกว่า 200 Service และผู้พัฒนาไม่เคยใช้ AWS มาก่อนพร้อมทั้ง Document ของ AWS ยังไม่ค่อยสมบูรณ์ จึงทำให้แผนการดำเนินงานเมื่อทำตามจริงนั้นเกิดความล่าช้า เพราะต้องใช้เวลาในการศึกษานานกว่าแผนการดำเนินการ
2. AWS เป็นอะไรที่ใหม่สำหรับนักพัฒนาในประเทศไทย ทำให้หาข้อมูลศึกษายาก หรือหาผู้ปรึกษายากและไม่ตรงตามความต้องการ
3. ปัญหาเวลาไม่ตรงกันเนื่องจากผู้พัฒนามีเวลาว่างที่ไม่ตรงกันมากเท่าที่ควรจึงส่งผลให้การทำงานเกิดความล่าช้า
4. ปัญหาโรคระบาด COVID-19 เนื่องจากช่วงเดือนกุมภาพันธ์เป็นต้นมามีโรคระบาดเกิดขึ้นซึ่งเป็นช่วงที่ต้องติดต่อกับฝั่งบริษัทผู้ใช้งาน ที่บ่อยครั้งนั้นต้องถูกเลื่อน และการติดต่อกับฝั่งผู้พัฒนาด้วยกันเองก็เป็นไปด้วยความยากลำบาก

5.3 วิธีการแก้ปัญหา

1. ผู้พัฒนาได้ศึกษาจากเว็บไซต์อื่น ๆ พร้อมทั้งลองผิดลองถูกเองเนื่องจาก Document ของ AWS นั้นยังไม่สมบูรณ์
2. ผู้พัฒนาได้ทดลองสร้างแอปพลิเคชันมาทดสอบระบบหลังบ้านเองเพื่อทดสอบคำสั่งกันเองโดยไม่เกี่ยวข้องกับระบบหน้าบ้าน

3. ผู้พัฒนาวางแผนตารางการพูดคุยใหม่โดยเพิ่มเวลาในการคุยกันมากขึ้นแทนเวลาที่เสียไปจากเวลาว่างที่ไม่ตรงกัน
4. ใช้การประชุมแบบออนไลน์แทนซึ่งมีทั้งการ ประชุมปรึกษาหารือ ประชุมเพื่อตรวจสอบและประเมินผล และประชุมเพื่อทำงานร่วมกันระหว่างผู้พัฒนา

5.4 ข้อเสนอแนะในการพัฒนาระบบ

จากการทดสอบพบว่ายังมีอีกหลาย ๆ อย่างที่จำเป็นจะต้องแก้ไขเพิ่มเติม อีกทั้งยังจำเป็นจะต้องศึกษาเพิ่มอีกมาก ผู้พัฒนาจะขอระบุข้อเสนอแนะต่าง ๆ เป็นหัวข้อ ดังนี้

1. ศึกษาระบบของ AWS เพิ่มเติม โดยเรียน AWS ออนไลน์ หรือเข้าคอร์สต่าง ๆ จะช่วยให้การทำงานเร็วมากขึ้นกว่านี้ และยังออกแบบระบบได้ถูกต้อง
2. ปรับปรุงแก้ไขโครงสร้างฐานข้อมูลใหม่ เนื่องจากปัจจุบันมีการเรียกใช้เยอะทำให้ใช้งบประมาณสูงจำเป็นจะต้องลดการเรียกข้อมูล
3. พัฒนาระบบ AI วิเคราะห์ชนิดของรูปภาพ, แก้ไขรูปภาพ
4. ใช้ Services ของ AWS ให้เต็มประสิทธิภาพมากขึ้น
5. เพิ่ม API รองรับการใช้งานมากขึ้น
6. ออกแบบ Authentication ใหม่ ให้รองรับผู้ใช้มากขึ้น
7. แก้ไข S3 Bucket ให้มีโครงสร้างที่เป็นสากลมากยิ่งขึ้น
8. เพิ่มระบบ Notification ให้รองรับ iPhone และ iPad

บรรณานุกรม

[1] “อัพเดตพฤติกรรมและข้อมูลผู้ใช้งาน LINE ประเทศไทย”. [ระบบออนไลน์]. Available from: https://www.twfdigital.com/blog/2018/09/thailandline-statsand-behaviour/ [10 ตุลาคม 2562]

[2] “การศึกษาพฤติกรรมการสื่อสารภายในองค์กรธุรกิจอุตสาหกรรม”. [ระบบออนไลน์]. AvailableFrom: https://www.tcithaijo.org/index.php/rpu/article/download/112383/87577/ [11 ตุลาคม 2562].

[3] “AWS AppSync ขับเคลื่อนแอปพลิเคชันของคุณด้วยข้อมูลที่ถูกต้อง จากแหล่งข้อมูลหนึ่งรายการขึ้นไปในระดับสากล”. [ระบบออนไลน์]. Available from: https://aws.amazon.com/th/appsync/ [10 ตุลาคม 2562]

[4] “Amazon DynamoDB Examples”. [ระบบออนไลน์]. Available from: https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/dynamodb-examples.html [11 ตุลาคม 2562]

[5] “เริ่มต้นสร้างบน AWS วันนี้”. [ระบบออนไลน์]. Available from: https://aws.amazon.com/th/ [13 ตุลาคม 2562]

[6] “What is GraphQL?”. [ระบบออนไลน์]. Available from: https://aws.amazon.com/graphql/ [13 ตุลาคม 2562]

[7] “Using AWS with React Native”. [ระบบออนไลน์]. Available from: https://facebook.github.io/react-native/blog/2018/03/05/AWS-app-sync/ [14 ตุลาคม 2562]

[8] “รู้จักกับ Visual Studio Code”. [ระบบออนไลน์]. Available from: https://www.mindphp.com/บทความ/microsoft/4829-visual-studio-code.html [3 พฤศจิกายน 2562].

[9] “พื้นฐาน JavaScript” .[ระบบออนไลน์]. Avaliable from: https://medium.com/open-source-technology/สรุปพื้นฐาน-javascript-ec02f18cfe47 [3 พฤศจิกายน 2562].

[10] “มาใช้ NodeJS กันเถอะ”. [ระบบออนไลน์]. Available from: https://medium.com/@mosprogramer/มาใช้-nodejs-กันเถอะ-bf77845e60bf/ [3 พฤศจิกายน 2562]

[11] “สร้าง RESTful API ด้วย express”. [ระบบออนไลน์]. Avaliable from: http://medium.com/@aofleejay/สร้าง-restful-api-ด้วย-express-express-101-ee37cc4952b4/ [7 ธันวาคม 2562]

[12] “Serverless คืออะไรกัน”. [ระบบออนไลน์]. Available from: http://teerapuch.com/technology/what-serverless/ [7 ธันวาคม 2562]

[13]“NoSQL คืออะไร? ต่างจาก RDBMS หรือ SQL Database อย่างไร?”. [ระบบออนไลน์]. Available from: http://medium.com/@every.phu/nosql-คืออะไร-ต่างจาก-rdbms-หรือ-sql-database-อย่างไร-dd8ac91a4197/ [7 ธันวาคม 2562]

[14] “Microservice Architecture”. [ระบบออนไลน์]. Available from: https://www.softnix.co.th/2018/08/09/microservices-in-10-minutes/ [7 ธันวาคม 2562]

[15] “[AWS] Amazon DynamoDB”. [ระบบออนไลน์]. Available from: http://cjrequena.com/2019-07-29/aws-dynamodb/ [10 ธันวาคม 2562]

[16] “AWS Lambda”. [ระบบออนไลน์]. Available from: https://aws.amazon.com/th/lambda/ [10 ธันวาคม 2562]

[17] “AWS CloudFormation #Chapter1”. Available from: http://medium.com/@Krucamper/aws-cloudformation-chapter1-912c612a29bb/ [3 ธันวาคม 2562]

ภาคผนวก

ภาคผนวก ก

แบบเสนอหัวข้อโครงงาน รายวิชา 2301399 Project Proposal ปีการศึกษา 2562

ชื่อโครงงาน (ภาษาไทย)	แอปพลิเคชันช่วยในการตรวจสอบคุณภาพสำหรับกระบวนการก่อสร้าง (ระบบส่วนหลังบ้าน)
ชื่อโครงงาน (ภาษาอังกฤษ)	Construction process quality control assisted application (Back-end)
อาจารย์ที่ปรึกษา	รองศาสตราจารย์ ดร.นกุล คูหะโรจนานนท์
ผู้ดำเนินการ	1. นายวัชรพงศ์ พงศ์สุทธิศรัทธา 2. นายคุปติพงศ์ สุวรรณไตรย์ สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

หลักการและเหตุผล

ในปัจจุบันมีผู้ใช้หลากหลายประเภทใช้แอปพลิเคชันสื่อสารที่มีชื่อเสียงอย่างเช่น LINE [1] แต่แอปพลิเคชันดังกล่าวกลับไม่สามารถตอบโจทย์ผู้ใช้ได้ครบ ทำให้เกิดปัญหาในการสื่อสารในองค์กร โดยส่งผลกระทบต่อเวลา, ค่าใช้จ่าย และบุคลากร [2]

หากเราจัดทำแอปพลิเคชันที่จะมาช่วยเหลือในการตรวจสอบคุณภาพสำหรับกระบวนการก่อสร้าง จะทำให้การสื่อสารระหว่างทีมก่อสร้างเป็นไปในแบบแผนเดียวกันและมีมาตรฐานเดียวกัน แต่ระบบฐานข้อมูลและการเก็บข้อมูลที่ใช้ในปัจจุบันนั้นจำเป็นจะต้องวางเครื่องเซิฟเวอร์ที่มีค่าใช้จ่ายเกินขนาดของแอปพลิเคชันเกินไป และเพื่อศึกษาระบบฐานข้อมูลใหม่ ๆ , บริการใหม่ ๆ ที่เพิ่งกำเนิดขึ้นมาในยุคปัจจุบัน [3][4] ซึ่งให้บริการในรูปแบบ Cloud Services ยกอย่างเช่น Amazon Web Services (AWS) [5], Google Cloud ซึ่งไม่จำเป็นจะต้องใช้เครื่องเซิฟเวอร์ขนาดใหญ่ และจ่ายเงินตามที่ใช้งานจริง

ผู้พัฒนาเห็นว่าหากมีช่องทางที่จะนำบริการเหล่านี้มาใช้กับแอปพลิเคชันที่ผู้พัฒนาจะพัฒนาซึ่งกล่าวมาในข้างต้น จะทำให้ระบบส่วนหน้าบ้านสามารถใช้บริการที่รวดเร็วกว่าเดิม, มีค่าใช้จ่ายในการพัฒนาน้อยลง และสามารถนำไปผสานรวบรวมกับบริการอื่น ๆ ใน Amazon Web Services [6][7] ได้อย่างครบครัน

วัตถุประสงค์

1. เพื่อสร้างนวัตกรรมที่สามารถตอบโจทย์โครงการก่อสร้างต่าง ๆ ให้มีการติดต่อสื่อสารกันที่ง่ายรวดเร็วและมีประสิทธิภาพ และเป็นแบบแผนมากยิ่งขึ้น
2. สามารถตรวจสอบการดำเนินงานในหน้างานได้โดยที่วิศวกรระดับสูงไม่จำเป็นต้องลงพื้นที่หน้างาน
3. สามารถดูเข้าถึงข้อมูลของโครงการผ่านทางระบบออนไลน์แบบ Realtime

ขอบเขตของโครงงาน

1. บริษัทก่อสร้างที่ให้เก็บความต้องการของผู้ใช้งาน
3. บริษัทก่อสร้างที่ให้เก็บความต้องการของผู้ใช้งานTeam built CO.,LTD.
   3.1. บริษัท Team built CO.,LTD.
   3.2. สำนักวิจัยและพัฒนาทาง กรมทางหลวง
   3.3. บริษัท ปริญสิริ จำกัด (มหาชน)Apple iPhone 6
4. แอปพลิเคชันสามารถใช้ได้กับอุปกรณ์ Apple iPhone 6

วิธีการดำเนินงาน

ขั้นตอนการวิจัย

1. ศึกษาการใช้งาน Amazon Web Services (AWS)
   1.1 AWS Amplify
   1.2 AWS AppSync
   1.3 Amazon DynamoDB
   1.4 AWS Lambda
   1.5 Amazon S3 (Storage)
2. เก็บรวมรวบความต้องการของผู้ใช้งาน
3. ปรึกษากับระบบส่วนหน้าบ้านเพื่อออกแบบฐานข้อมูลเบื้องต้น
4. จัดการสร้าง Services ต่าง ๆ สำหรับเรียกใช้งาน
5. นำระบบหลังบ้านมาเชื่อมต่อกับระบบส่วนหน้าบ้าน
6. ทดสอบประสิทธิภาพของระบบ
7. สรุปผลการดำเนินงานและจัดทำเอกสารประกอบโครงการ

ระยะเวลาการดำเนินงาน

ขั้นตอนการดำเนินงาน	ปี พ.ศ. 2562				ปี พ.ศ. 2563		
	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
1. ศึกษาการใช้งาน Amazon Web Services (AWS)							
2. เก็บรวมรวบความต้องการของผู้ใช้งาน							
3. ปรึกษากับระบบส่วนหน้าบ้านเพื่อออกแบบฐานข้อมูลเบื้องต้น							
4. จัดการสร้าง Services ต่าง ๆ สำหรับเรียกใช้งาน							
5. นำระบบหลังบ้านมาเชื่อมต่อกับระบบส่วนหน้าบ้าน							
6. ทดสอบประสิทธิภาพของระบบ							
7. สรุปผลการดำเนินงานและจัดทำเอกสารประกอบโครงการ							

ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ต่อผู้พัฒนาโครงงาน

1. ได้รับทักษะการออกแบบฐานข้อมูล โดยสามารถนำไปใช้ได้จริงในภาคอุตสากรรม
2. ได้พัฒนาความรู้ในด้าน API, Cloud Database and Storage และ Cloud Computing
3. ได้พัฒนาความรู้เกี่ยวกับ Tools ใหม่ ๆ ที่ใช้กันอย่างแพร่หลายในปัจจุบัน
4. ได้พัฒนาทักษะการคิด วิเคราะห์ และการทำงานเป็นทีม

ประโยชน์ที่ได้จากโครงงานที่พัฒนาขึ้น

1. เพิ่มประสิทธิภาพด้านการสื่อสารของโครงการก่อสร้าง
2. เพิ่มประสิทธิภาพด้านการสื่อสารของโครงการก่อสร้าง
3. สามารถดูข้อมูลของโครงการผ่านทางระบบออนไลน์แบบประมวลแบบทันท่วงที
4. ลดปริมาณการใช้วิศวกรระดับสูงลงไปที่หน้างาน

อุปกรณ์และเครื่องมือที่ใช้

1. ฮาร์ดแวร์
    1.1. เครื่องคอมพิวเตอร์ส่วนบุคคล หรือคอมพิวเตอร์พกพาที่มีคุณสมบัติดังนี้
        Macbook Pro 13-inch Processor 3.1 GHz intel core i5
    1.2. Apple iPhone 6
    1.3. อุปกรณ์จัดเก็บแบบพกพา (External Harddisk)
2. ซอฟแวร์
    2.1. Google Chrome
    2.2. Vscode
    2.3. Lucid Chart
    2.4. Docker
    2.5. Team Viewer
    2.6. Adobe Photoshop
3. อื่น ๆ
    3.1. Amazon Web Services
    3.2. Google Drive
    3.3. Google Colab
    3.4. Github

งบประมาณ

1. SSD		5,000 บาท
2. อุปกรณ์จัดเก็บข้อมูลชนิด HDD		2,500 บาท
3. หน่วยความจำเข้าถึงโดยสุ่ม (RAM) ขนาด 16 GB		2,500 บาท
	รวม	10,000บาท

หมายเหตุ ค่าใช้จ่ายทั้งหมดถัวเฉลี่ยกันทุกรายการ

บรรณานุกรม

[18]“อัพเดตพฤติกรรมและข้อมูลผู้ใช้งาน LINE ประเทศไทย”. [ระบบออนไลน์]. Available from: https://www.twfdigital.com/blog/2018/09/thailandline-statsand-behaviour/ [10 ตุลาคม 2562]

[19]“การศึกษาพฤติกรรมการสื่อสารภายในองค์กรธุรกิจอุตสาหกรรม”. [ระบบออนไลน์]. AvailableFrom: https://www.tcithaijo.org/index.php/rpu/article/download/112383/87577/ [11 ตุลาคม 2562].

[20]“AWS AppSync ขับเคลื่อนแอปพลิเคชันของคุณด้วยข้อมูลที่ถูกต้อง จากแหล่งข้อมูลหนึ่งรายการขึ้นไปในระดับสากล”. [ระบบออนไลน์]. Available from: https://aws.amazon.com/th/appsync/ [10 ตุลาคม 2562]

[21]“Amazon DynamoDB Examples”. [ระบบออนไลน์]. Available from: https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/dynamodb-examples.html [11 ตุลาคม 2562]

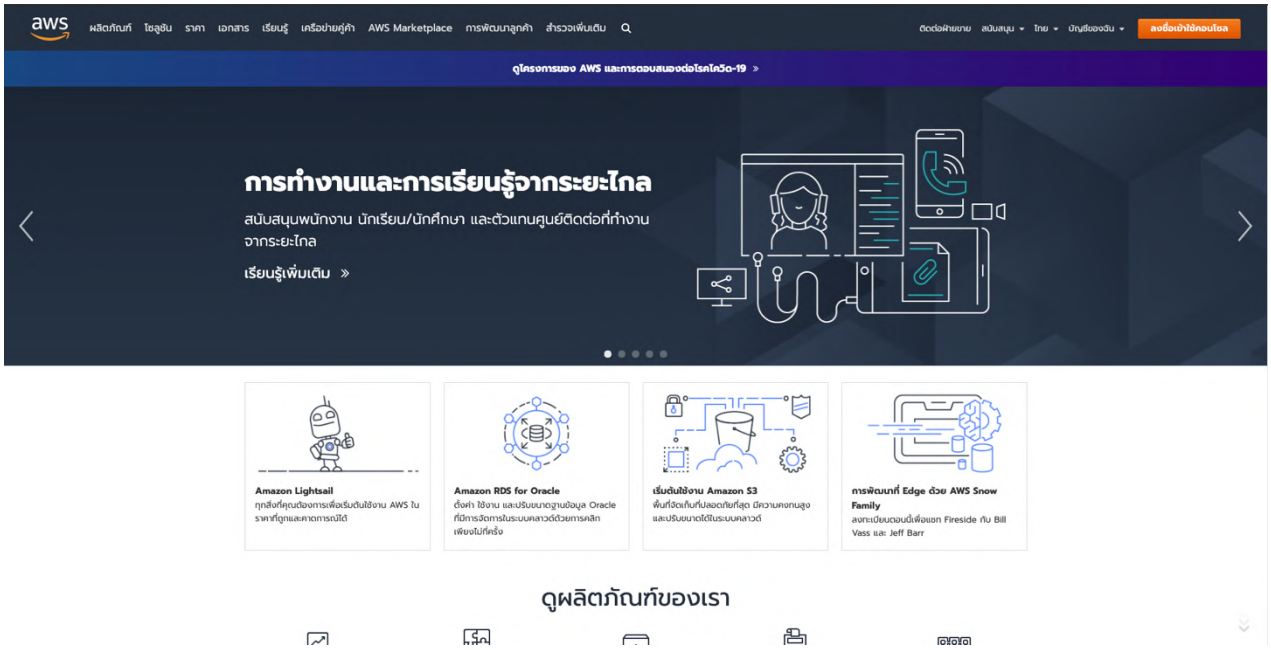
[22]“เริ่มต้นสร้างบน AWS วันนี้”. [ระบบออนไลน์]. Available from: https://aws.amazon.com/th/ [13 ตุลาคม 2562]

[23]“What is GraphQL?”. [ระบบออนไลน์]. Available from: https://aws.amazon.com/graphql/ [13 ตุลาคม 2562]

ภาคผนวก ข

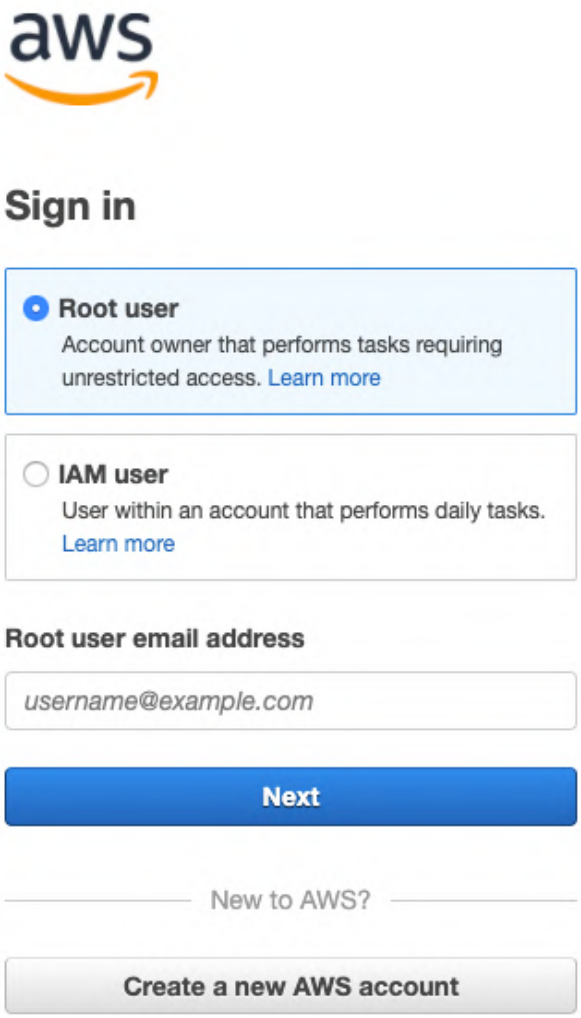
คู่มือการใช้งาน AWS

เมื่อเข้าสู่เว็บไซต์ https://aws.amazon.com/th/ แล้ว จะพบกับหน้าแนะนำ Services ต่าง ๆ ดังภาพ ข-1



ภาพ ข-1 หน้าหลักของ AWS

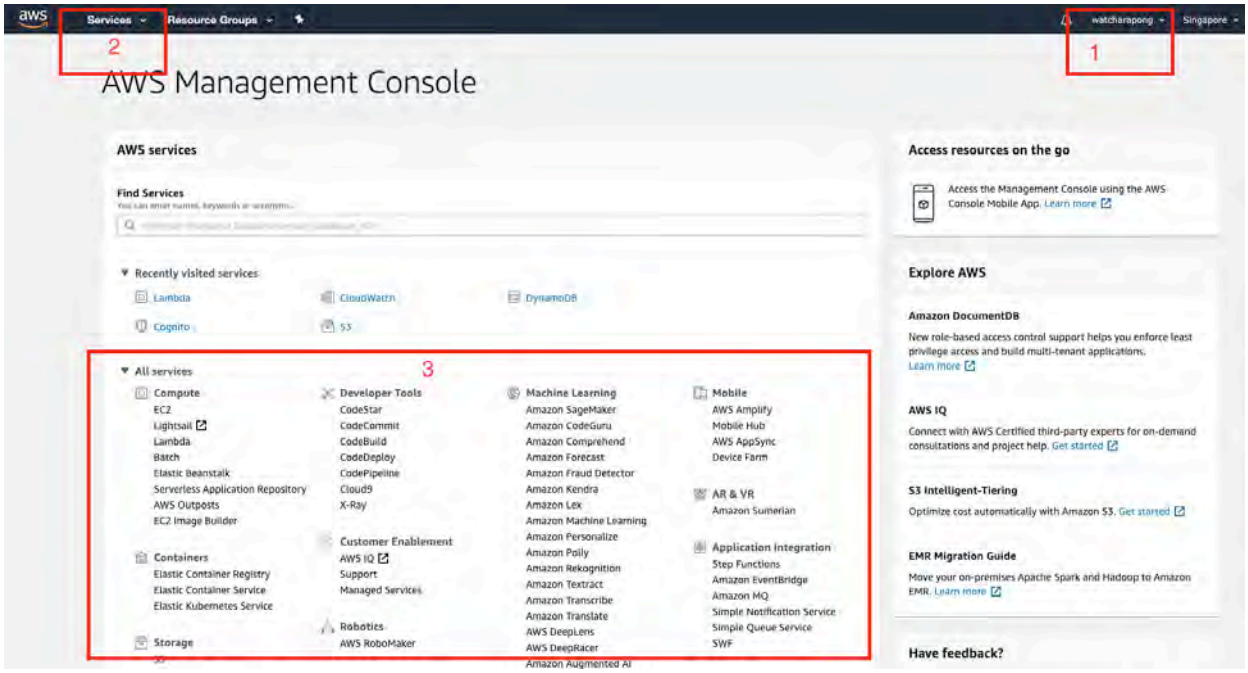
เมื่อเข้าสู่หน้า Sign In ให้กรอกข้อมูลผู้ใช้ โดยจะมีให้เลือกเป็น Root user คือบัญชีหลัก ส่วน IAM user เป็น ผู้ใช้ที่แตกแขนงจากบัญชีหลักอีกที (สมาชิกทีม)



ภาพ ข-2 หน้า Login AWS Console

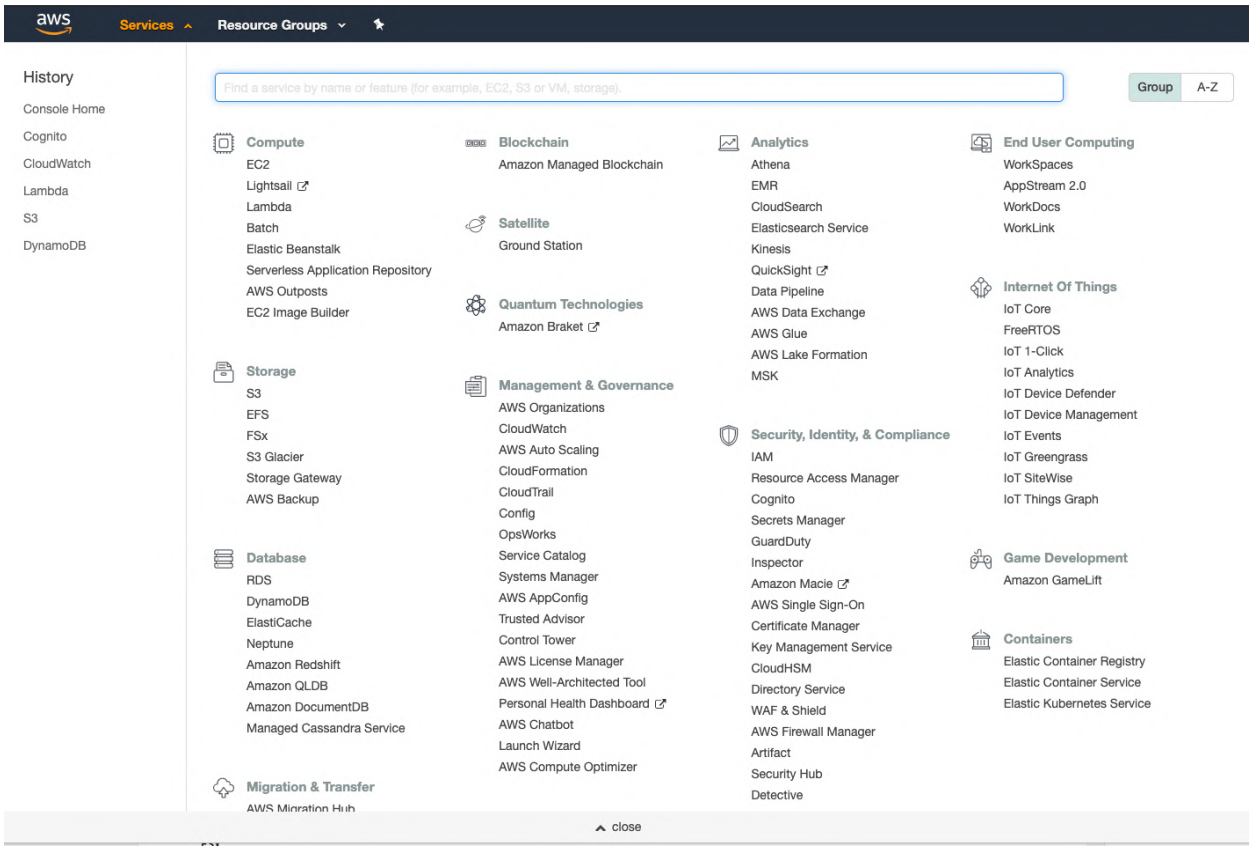
หน้าหลักหลังจาก Login แล้วจะเป็น AWS Management Console ผู้พัฒนาขออธิบายเป็นสามส่วนดังภาพ ข-3

- สัญลักษณ์ 1 จะเป็นการจัดการ Account และข้างขวาเยื้องไปจะเป็น Region ใช้สำหรับเปลี่ยนภูมิประเทศ (มีผลต่อ Services ไม่ควรเปลี่ยนหลังเริ่มพัฒนาแล้ว)
- สัญลักษณ์ 2 เป็น Services ต่าง ๆ ปรากฏอยู่ในทุกหน้าของเว็บไซต์
- สัญลักษณ์ 3 เป็นภาพรวม Services ทั้งหมด



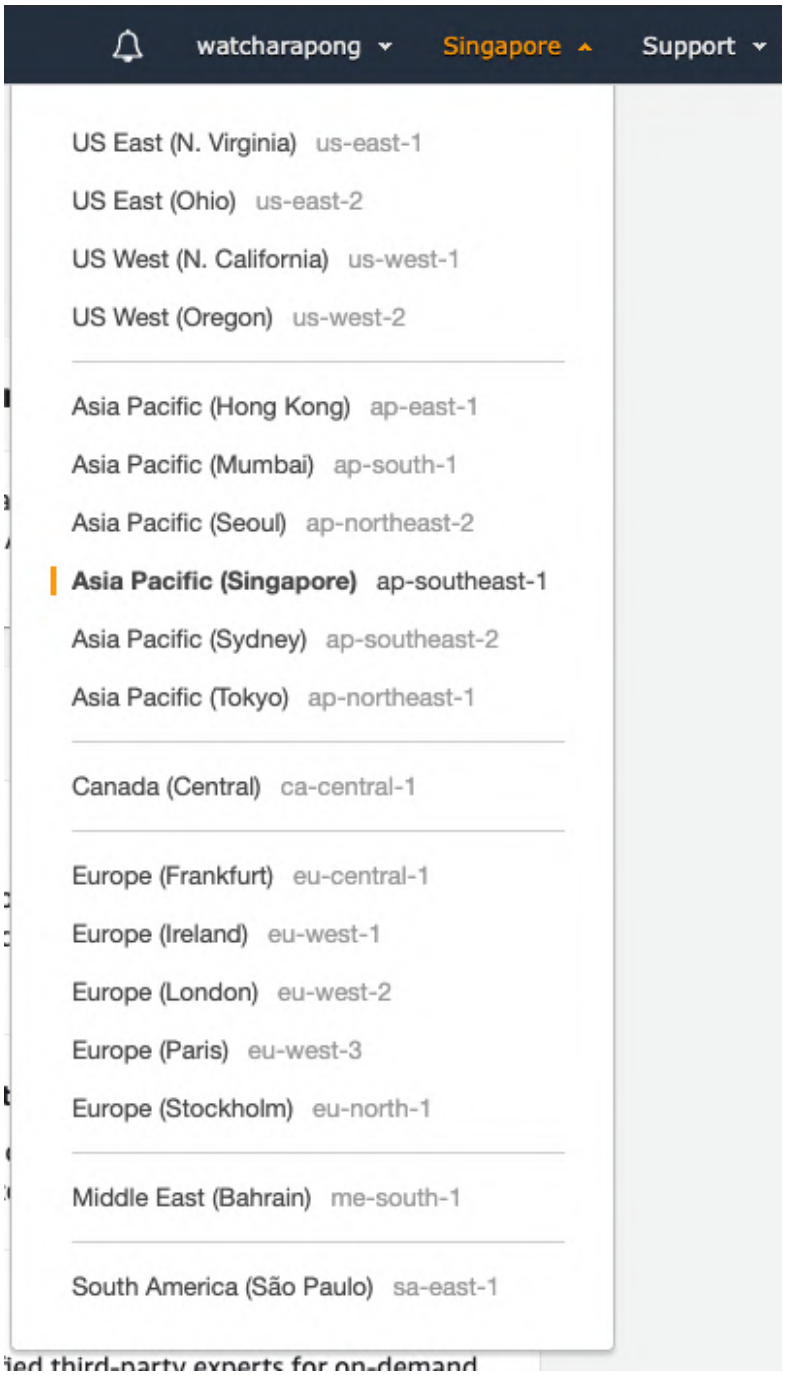
ภาพ ข-3 หน้า AWS Management Console

เมื่อกดสัญลักษณ์ 2 จะแสดงหน้า ข-4



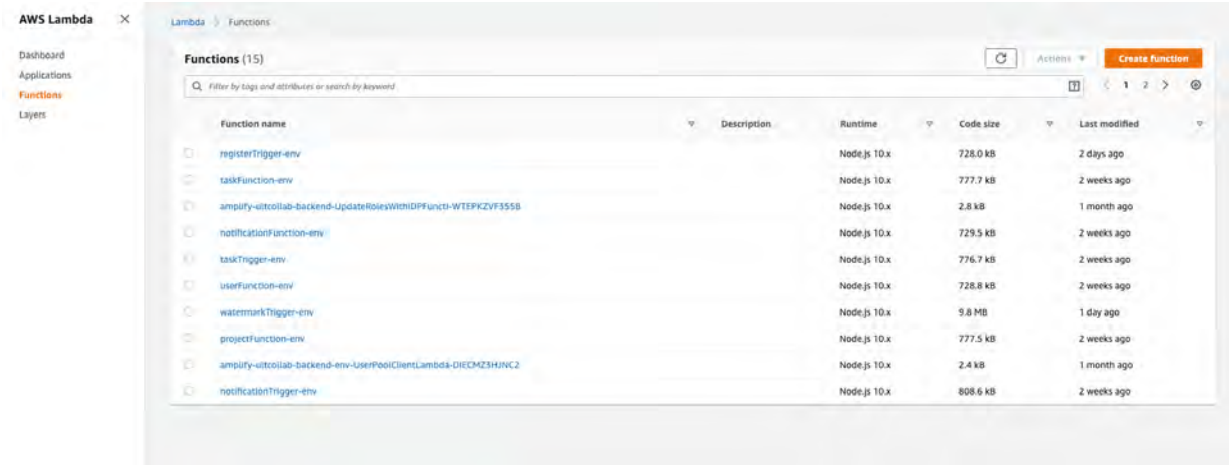
ภาพ ข-4 หน้า AWS Services Tab

เมื่อกดสัญลักษณ์ 1 จะแสดงหน้า ข-5



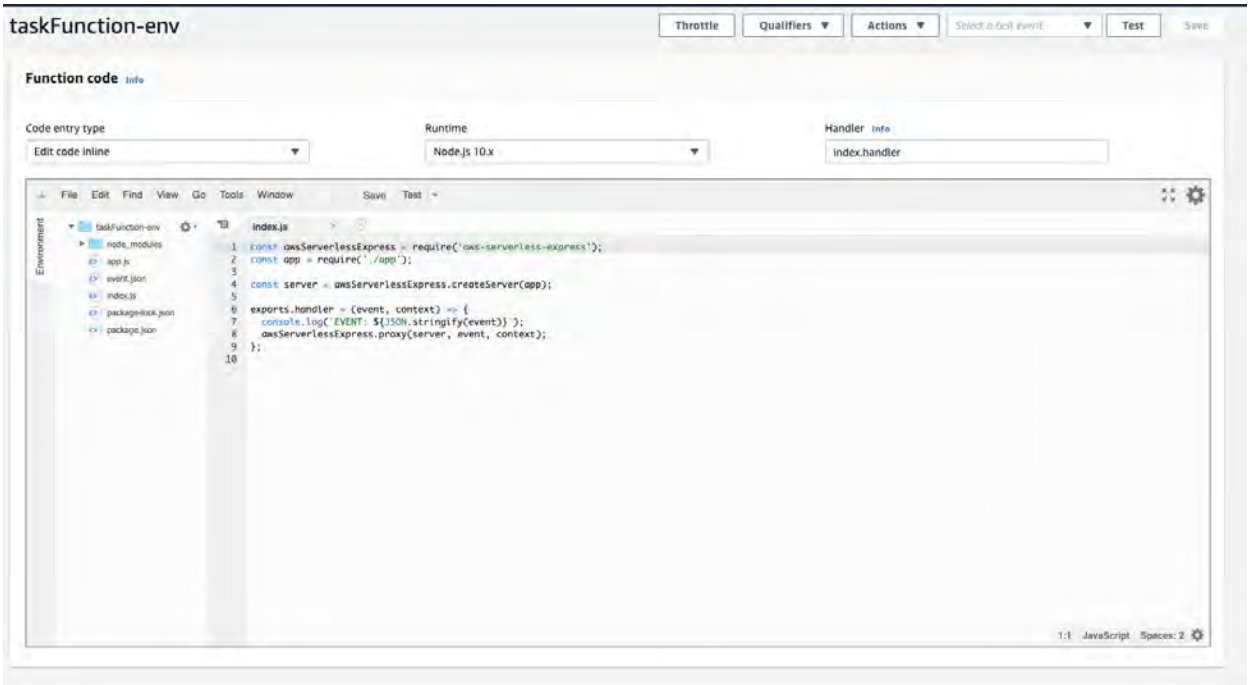
ภาพ ข-4 หน้า AWS Services Tab

เมื่อเข้ามาที่ AWS Lambda Service แล้วจะพบกับหน้า ข-5 ในหน้านี้จะพบกับ Function ทั้งหมดที่ผู้พัฒนาสร้างขึ้นมา



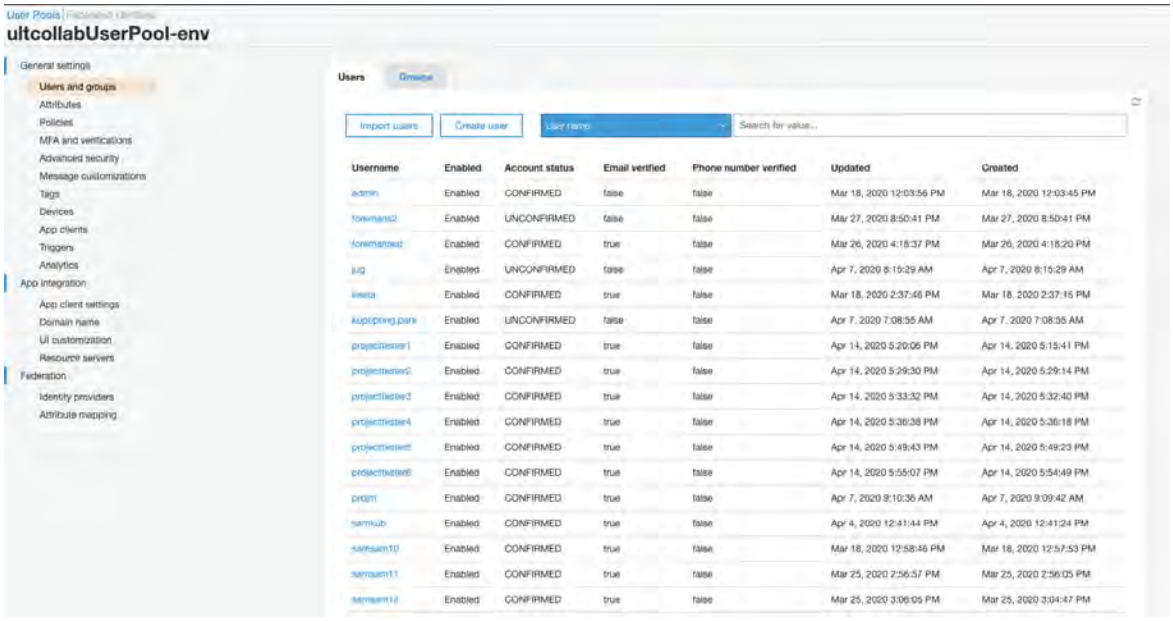
ภาพ ข-5 หน้า AWS Lambda

เมื่อเข้าฟังก์ชันมาแล้วจะพบกับหน้า ข-6 ผู้พัฒนาสามารถแก้ไขฟังก์ชันในหน้าได้เลย โดยไม่ต้องแก้ไขใน Local แต่หาก Function มีขนาดใหญ่จะไม่สามารถใช้งานหน้านี้ได้



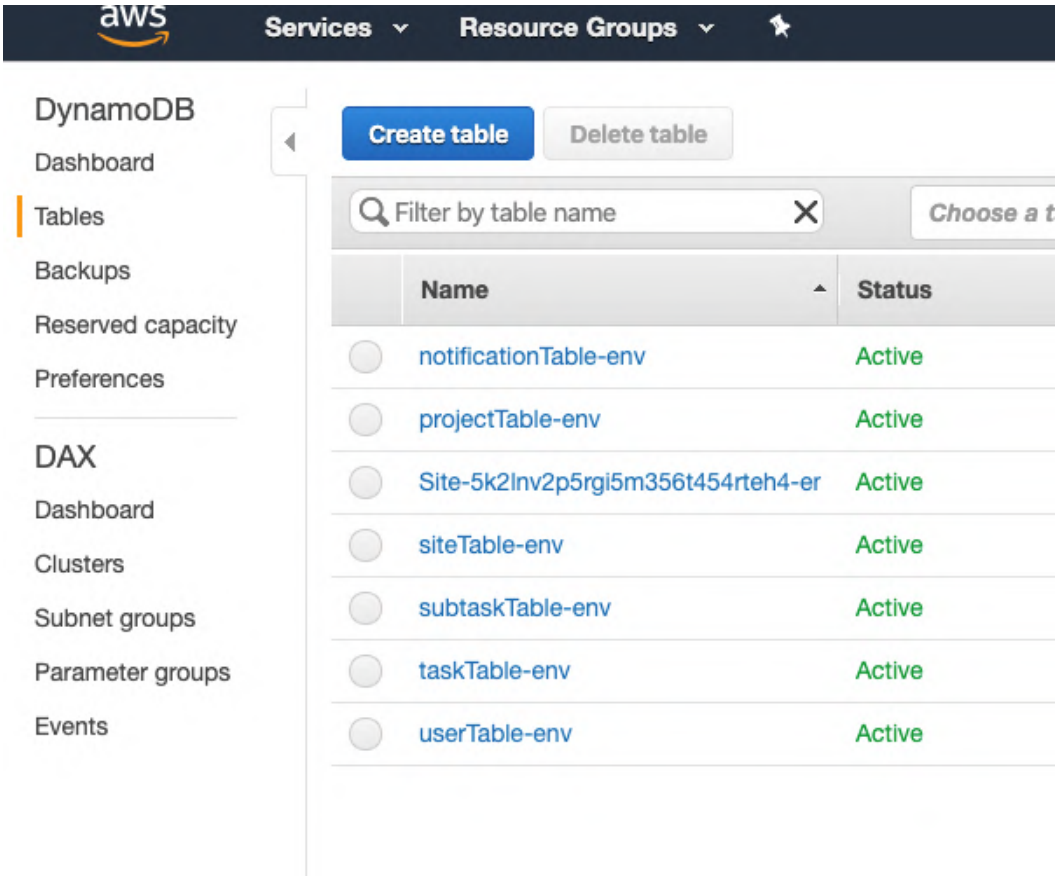
ภาพ ข-6 หน้า AWS Lambda - Function

เมื่อเข้า User Pool แล้วจะพบกับหน้า ข-7 ผู้พัฒนาสามารถเพิ่ม / ลบผู้ใช้ของแอปพลิเคชันได้



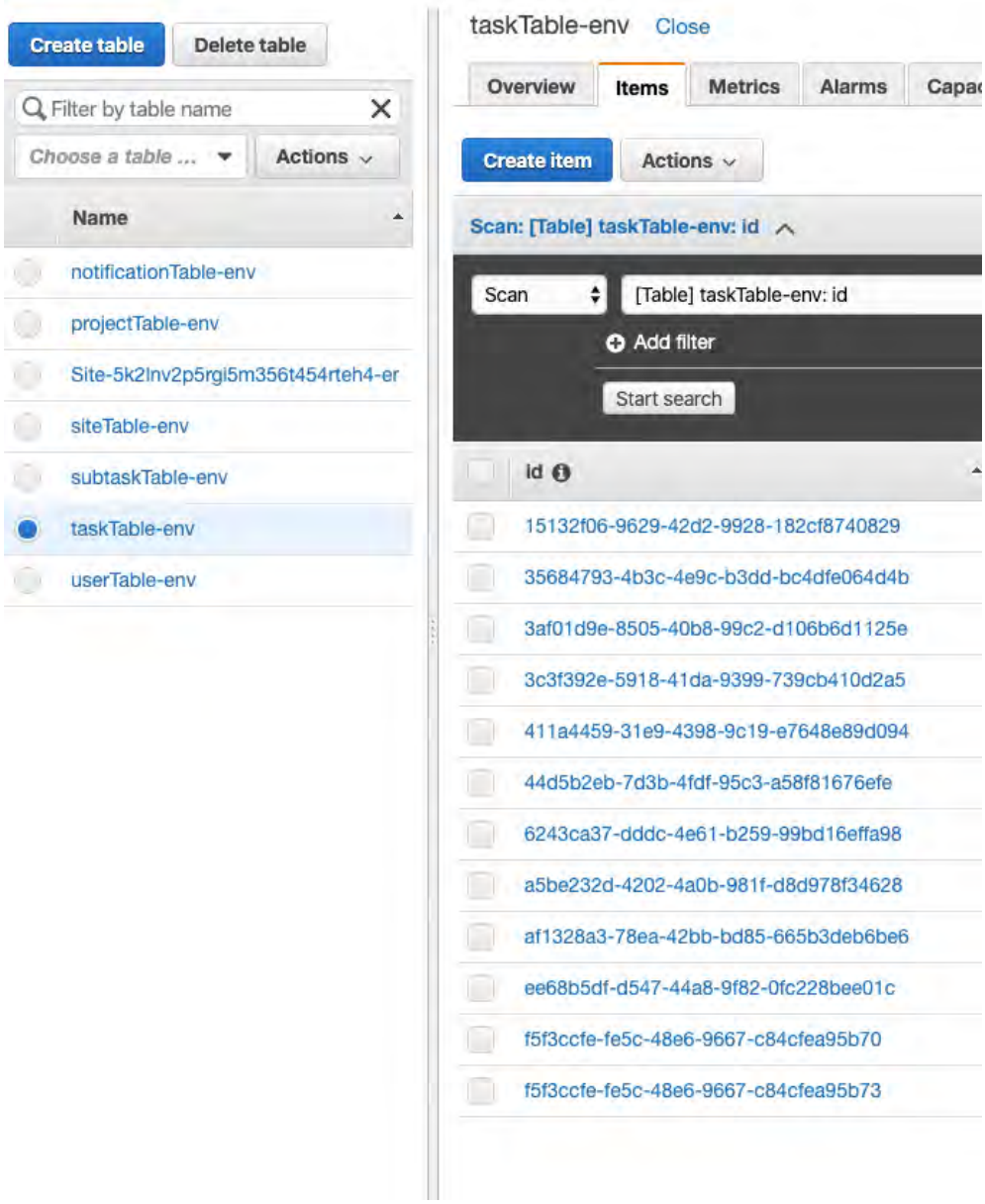
ภาพ ข-7 หน้า Amazon Cognito - User Pool

เมื่อเข้า DynamoDB แล้วจะพบกับหน้า ข-8 ผู้พัฒนาสามารถเพิ่ม, ลบ, แก้ไข ตารางได้



ภาพ ข-8 หน้า Amazon DynamoDB – Tables

เมื่อคลิ๊กที่ Table แล้วจะพบกับหน้า ข-9 ผู้พัฒนาสามารถเพิ่ม, ลบ, แก้ไขข้อมูลในตารางได้



ภาพ ข-9 หน้า Amazon DynamoDB – Tables

ประวัติผู้เขียน

นายคุปติพงศ์ สุวรรณไตรย์
เกิดวันที่: 13 มิถุนายน 2540
อีเมล: kuptipong.s@student.chula.ac.th
อีเมลสำรอง: kuptipong.park@gmail.com
วุฒิการศึกษา: กำลังศึกษาหลักสูตรวิทยาศาสตร์บัณฑิต คณะวิทยาศาสตร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย

นายวัชรพงศ์ พงศ์สุทธิศรัทธา
เกิดวันที่: 7 กรกฎาคม 2540
อีเมล: watcharapong.po@student.chula.ac.th
อีเมลสำรอง: watcharapong.pstst@gmail.com
วุฒิการศึกษา: กำลังศึกษาหลักสูตรวิทยาศาสตร์บัณฑิต คณะวิทยาศาสตร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย