รายงานสหกิจศึกษาฉบับสมบูรณ์

เครื่องมือเก็บข้อมูลและทดสอบประสิทธิภาพของระบบเก็บข้อมูล

(Data Collecting Tool and Cloud Cache Performance Testing)

นายวศิน นิ่มละออ

สาขาวิศวกรรมสารสนเทศ ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2562

ชื่อโครงงานสหกิจศึกษา เครื่องมือเก็บข้อมูลและทดสอบประสิทธิภาพของระบบเก็บข้อมูล

ชื่อ-สกุล นักศึกษา นายวศิน นิ่มละออ

คณะ วิศวกรรมศาสตร์ **ภาควิชา** วิศวกรรมคอมพิวเตอร์

ชื่อ-สกุล อาจารย์นิเทศ ผศ.ดร.เกล็ดดาว สัตย์เจริญ

ชื่อ-สกุล ผู้นิเทศงาน นายฤทธิ์ คันธพิศาล

สถานประกอบการ บริษัท รีฟินิทีฟ ซอฟต์แวร์(ประเทศไทย) จำกัด

บทคัดย่อ

เนื่องจากทีมไทม์ซีรี่ส์นั้นมีการทำงานที่เกี่ยวข้องกับการจัดการข้อมูลต่าง ๆ โดยจะมีการทำ Regression testing เพื่อทดสอบการทำงานของ Component ต่าง ๆ โดยการทำ Regression testing และ End-to-end testing นั้นจะทำการขอข้อมูลจาก Datastore ตัวหนึ่งที่เรียกว่า Query Engine โดยข้อมูลที่สร้างขึ้นเมื่อเวลาผ่านไป 30 วันแล้วจะหายไป จึงทำการสร้างเครื่องมือที่เก็บบันทึกข้อมูลที่ได้จาก Query Engine และทำการส่งข้อมูลที่บันทึกให้กับผู้ใช้งาน ซึ่งมีชื่อว่า Polly

นอกจากนี้ภายในทีมได้มีการพัฒนา component ใหม่ขึ้นมานั่นคือ Time Series Cloud Cache เพื่อเพิ่มประสิทธิภาพของเวลาในการรับ-ส่งข้อมูลให้ดียิ่งขึ้น โดยทางทีมต้องการจะทำการทดลองเพื่อพิสูจน์ว่า Time Series Cloud Cache นั้นเพิ่มประสิทธิภาพในการรับ-ส่งข้อมูลให้ดียิ่งขึ้นตามที่คาดไว้หรือไม่ โดยการทดลองนี้ชื่อว่า Single Response Time Test

Cooperative title: Data Collecting Tool and Cloud Cache Performance Testing

Student intern name: Mr. Wasin Nimlaor

Faculty: Engineering **Department:** Information Engineering

Advisor name: Asst. Prof. Dr. Kleddao Satcharoen

Mentor name: Mr. Rit Kantapisal

Company: Refinitiv Software (Thailand) Limited

ABSTRACT

Time Series team works with managing the data. Time Series uses Regression testing for testing components. In Regression testing client requests data from datastore called “Query Engine”. After 30 days from created date the data in Query Engine will no longer available, so we created the tool for recording the data from Query Engine and reply the data from recorded file to client. This tool is called “Polly”.

Besides, Time Series has developed the component called “Time Series Cloud Cache”. Time Series Cloud Cache’s purpose is to improve the performance of response time. To prove that we do the experiment called “Single Response Time Test”.

กิตติกรรมประกาศ

การทำสหกิจศึกษานี้สำเร็จลุล่วงไปได้ด้วยดี ด้วยคำแนะนำ และคำปรึกษาจากหลาย ๆ ท่าน ทั้งในเรื่องของการทำงาน และแนวคิดต่าง ๆ นอกจากนี้ยังมีพี่ ๆ ที่คอยสอนและให้คำแนะนำเกี่ยวกับงานต่าง ๆ ให้อีกด้วย

ขอขอบพระคุณอาจารย์ที่ปรึกษา คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่คอยให้คำแนะนำในเรื่องการทำสหกิจศึกษาในครั้งนี้

สุดท้ายนี้ขอขอบคุณรุ่นพี่และเพื่อน ๆ ทุกคนที่คอยให้คำปรึกษาในเรื่องการทำงาน การแก้ไขปัญหาต่าง ๆ ตั้งแต่เริ่มทำสหกิจศึกษาจนสำเร็จรายงานฉบับนี้ไปได้ด้วยดี

นายวศิน นิ่มละออ

สารบัญ

หน้า

สารบัญภาพ

หน้า

หน้า

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

บริษัท รีฟินิทีฟ ซอฟต์แวร์ (ประเทศไทย) จำกัด ได้พัฒนาซอฟต์แวร์ที่ให้บริการสาหรับผู้เชี่ยวชาญด้านการเงิน ในการแสดงผลและวิเคราะห์ข้อมูลทางการเงิน ช่วยให้สามารถเข้าถึงข้อมูลตลาดแบบเรียลไทม์ เช่น ข้อมูลข่าว ข้อมูลพื้นฐาน เป็นต้น มีเครื่องมือช่วยในการวิเคราะห์การซื้อขาย โดยทีม Time Series นั้นเป็นทีมที่ดูแลการพัฒนาซอฟต์แวร์ในส่วนของ Back-end ซึ่งต้องคอยจัดการกับข้อมูลเพื่อนำมาใช้งานในส่วนต่าง ๆ จึงต้องมีความแม่นยำ และสเถียรภาพของการรับและส่งข้อมูลอย่างถูกต้อง และรวดเร็ว เพื่อเป็นการทดสอบการทำงานของซอฟต์แวร์ที่ได้พัฒนาไปนั้นจะใช้การทำ Regression testing และ End-to-end testing โดยจะทำการส่ง request ต่าง ๆ ไปยัง Query Engine(QE) ซึ่งทำหน้าที่เป็น data store แต่ปัญหาที่เกิดขึ้นคือข้อมูลภายใน QE เมื่อเวลาผ่านไป 90 วันนับจากวันที่ถูกสร้างข้อมูลนั้นจะหายไป ทำให้ไม่สามารถเรียกใช้ข้อมูลนั้นในการทดสอบต่าง ๆได้ จึงได้ทำการสร้างเครื่องมือที่ชื่อ Polly ขึ้นมาเพื่อแก้ปัญหา โดยจะทำการบันทึกข้อมูลที่ได้จาก QE ทำให้ยังสามารถเรียกใช้ข้อมูลนั้นได้ แม้ว่าเวลาจะผ่านไปมากกว่า 90 วันแล้ว

นอกจากนี้ทางทีมยังได้ทำการพัฒนา component ที่ชื่อว่า Time Series Cloud-Cache(TSCC) ขึ้นมาเพื่อเพิ่มประสิทธิภาพในด้าน response time จึงต้องการทำการทดลองขึ้นมาเพื่อทดสอบว่า TSCC นั้นช่วยเพิ่มประสิทธิภาพในด้าน response time ตามสมมติฐานที่ได้ตั้งไว้หรือไม่ ซึ่งการทดลองนี้มีชื่อว่า Single Response Time Test

1.2 วัตถุประสงค์ของการทำวิจัย

1.2.1 เพื่อแก้ปัญหาของการเรียกใช้ข้อมูลใน QE ไม่ได้หลักจากเวลาผ่านไป 90 วัน

1.2.2 เพื่อทำการพิสูจน์ว่า TSCC นั้นช่วยเพิ่มประสิทธิภาพในด้าน response time ตามสมมติฐานที่ได้ตั้งไว้

1.3 ขอบเขตของการวิจัย

สามารถแบ่งขอบเขตของการวิจัยได้ดังนี้

1.3.1 Polly

1.3.1.1 ออกแบบและพัฒนาเครื่องมือเพื่อแก้ไขปัญหาของการเรียกใช้ข้อมูลไม่ได้หลังจากผ่านไป 90 วัน

1.3.1.2 นำเครื่องมือที่ได้ทำการพัฒนาไป deploy และ set up environment ให้ตรงกับที่ผู้ใช้งานต้องการ

1.3.2 Single Response Time Test

1.3.2.1 ทำการทดลองเพื่อพิสูจน์ว่า component ที่ได้ทำการสร้างขึ้นมาใหม่นั้นช่วยเพิ่มประสิทธิภาพในด้าน response time ตามสมมติฐานที่ได้ตั้งไว้

1.3.2.2 นำเอาผลที่ได้จากการทดลองมาแสดงผลเพื่อการวิเคราะห์

1.4 วิธีการดำเนินการวิจัย

1.4.1 Polly

1.4.1.1 กระบวนการเข้าใจปัญหาที่เกิดขึ้นของระบบการทำงานและออกแบบระบบ

1.4.1.1.1 ทำความเข้าใจระบบงานเดิม

1.4.1.1.2 ศึกษาปัญหาที่มีต่อความต้องการของผู้ใช้

1.4.1.2 กระบวนการวิเคราะห์ระบบและออกแบบระบบ

1.4.1.2.1 การเก็บรวบรวมความต้องการของผู้ใช้งาน

1.4.1.2.2 การออกแบบการทำงานของระบบ

1.4.1.2.2.1 Flow chart

1.4.1.2.2.2 Sequence Diagram

1.4.1.3 การพัฒนาระบบ

1.4.1.3.1 การรับ Request จากผู้ใช้งาน

1.4.1.3.2 การเลือก Feature ในการทำงาน

1.4.1.3.3 การทำงานของ Feature Replay

1.4.1.3.4 การทำงานของ Feature Proxy

1.4.1.3.5 การทำงานของ Feature Record

1.4.2 Single Response Time Test

1.4.2.1 ทำความเข้าใจจุดประสงค์ผู้ใช้งานและระบบ

1.4.2.2 ออกแบบและเขียน script

1.4.2.2.1 ออกแบบไฟล์ config

1.4.2.2.2 ออกแบบภาพรวมการทำงานของ script

1.4.2.2.3 ออกแบบ config.py

1.4.2.2.4 ออกแบบ model.py

1.4.2.2.5 ออกแบบ webService.py

1.4.2.2.6 เขียน dockerfile

1.4.2.2.7 เขียน shell script

1.4.2.2.8 แสดงผลและวิเคราะห์ผล

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่ได้รับจากการเข้าร่วมทำโปรเจคที่ได้รับมอบหมายตลอดระยะเวลาภายใต้โครงการสหกิจศึกษาสามารถแบ่งได้เป็น 2 ส่วน ดังต่อไปนี้

1.5.1 ประโยชน์ของ Polly

1.5.1.1 แก้ปัญหาเรื่องการใช้งานข้อมูลที่มีอายุมากกว่า 90 วันไม่ได้

1.5.1.2 ลดเวลาในการทำงานเมื่อใช้ Replay feature

1.5.1.3 สามารถนำไปต่อยอดในการใช้งานร่วมกับ Robot framework เพื่อทำเป็น Automation testing ได้

1.5.2 ประโยชน์ของ Single Response Time Test

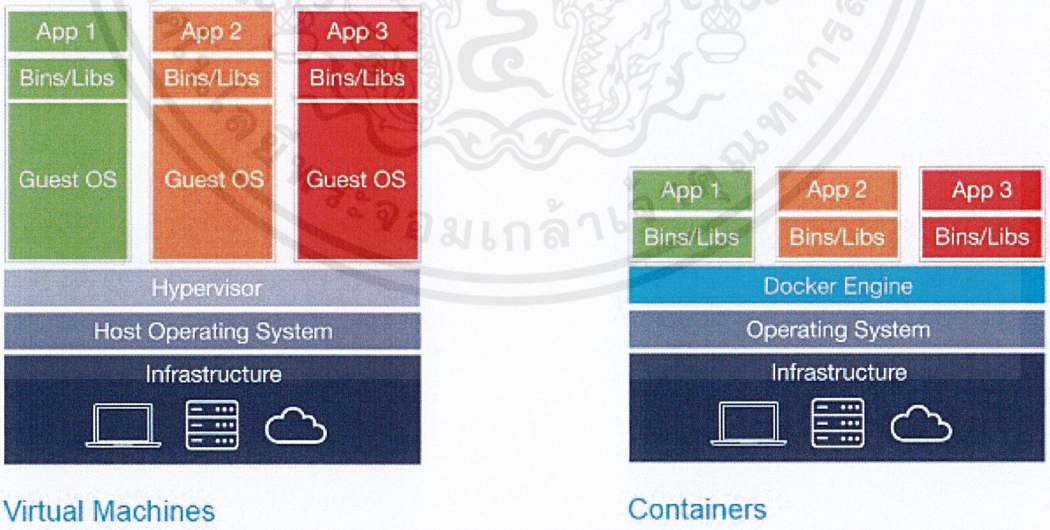
1.5.2.1 เพื่อทำการวัดผลและวิเคราะห์ว่า Component ที่สร้างมาใหม่นั้น ให้ผลลัพธ์เป็นไปตามจุดประสงค์ที่ตั้งไว้หรือไม่

บทที่ 2
แนวคิด ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะพูดถึงแนวคิด ทฤษฎี เทคโนโลยี และเครื่องมือที่ผู้วิจัยต้องทำการศึกษา เรียนรู้และใช้งานในการทำวิจัย โดยมีดังต่อไปนี้

2.1 Docker

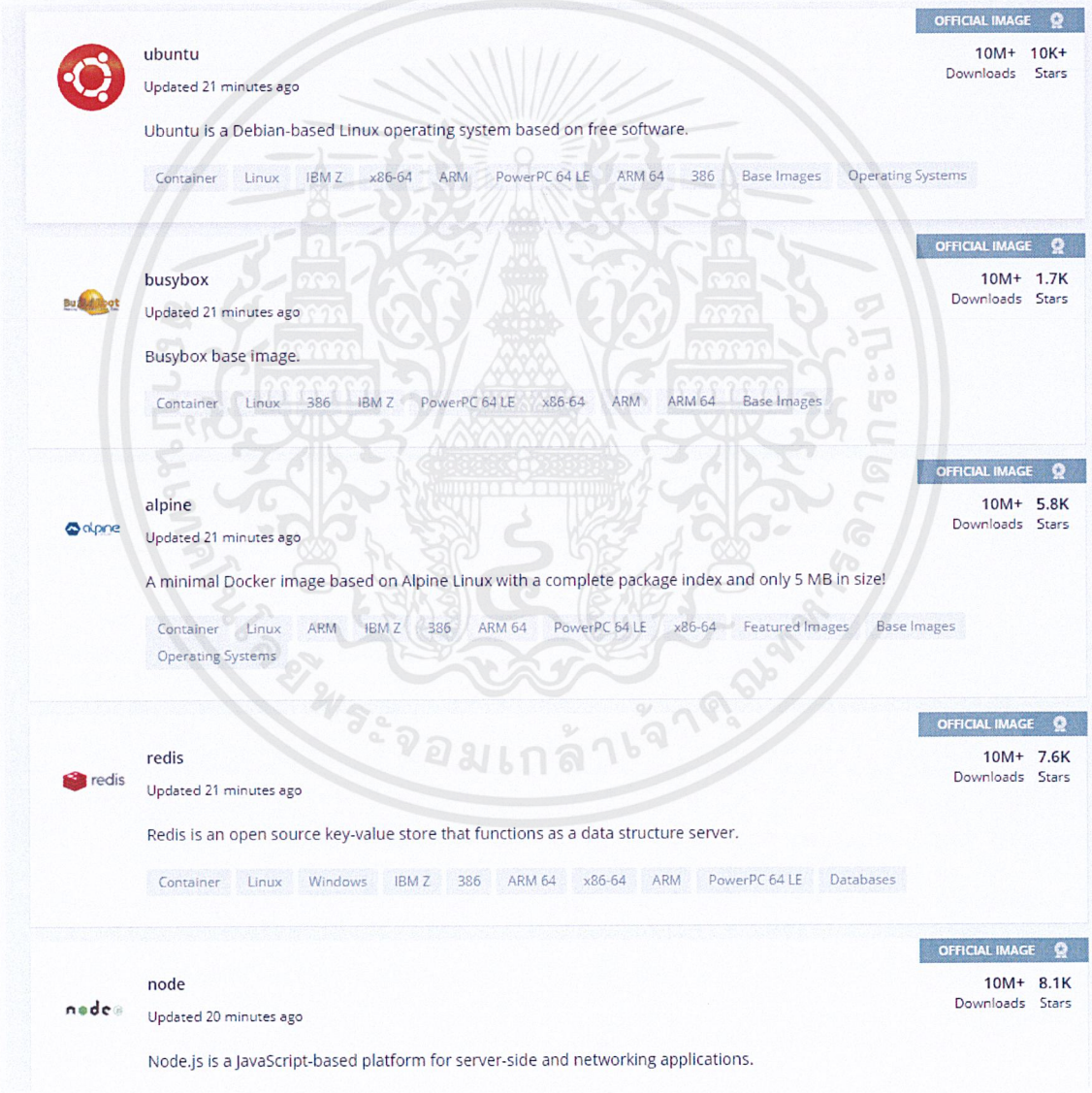
Docker เป็น engine ตัวหนึ่งที่มีลักษณะการทำงานโดยการจำลอง environment ขึ้นมาบนเครื่อง server เพื่อใช้ในการ run service โดยมีการทำงานที่คล้ายกับ Virtual Machine ตัวอย่างของ Virtual Machine เช่น VMWare, VirtualBox, XEN และ KVM เป็นต้น แต่สิ่งที่แตกต่างระหว่าง Docker และ Virtual Machine คือ Virtual Machine นั้นจะจำลองมาทั้ง Operating System(OS) เพื่อนำมาใช้งาน และเมื่อต้องการใช้งาน service ใด ๆ ก็จะติดตั้งเพิ่มเติมลงบน OS นั้น ๆ อีกที แต่ Docker นั้นจะมีการใช้ container ในการจำลอง environment เพื่อใช้งานสำหรับ 1 service ที่ต้องการจะใช้เท่านั้น โดยไม่มีความจำเป็นที่จะต้องติดตั้งส่วนของ OS ที่มีขนาดใหญ่



ภาพที่ 2.1 ภาพแสดงความแตกต่างระหว่าง Virtual Machine และ Docker

2.1.1 Docker Image

Docker Image นั้นเปรียบเสมือนกับเป็นพิมพ์เขียวของ container ซึ่งภายในนั้นก็จะประกอบไปด้วย Application และไฟล์ต่าง ๆ ที่ได้ทำการติดตั้งไว้เพื่อการใช้งานสำหรับ service นั้น ๆ นอกจากนี้ยังสามารถมีการ config ค่าต่าง ๆ สำหรับ environment และค่าอื่น ๆ โดย Docker Image นั้นสามารถ pull มาจาก Docker Hub ซึ่งจะมี image ที่พร้อมใช้งานมากมายหลายประเภท ตามแต่ผู้ใช้งานจะเลือกนำมาใช้ และผู้ใช้งานเองยังสามารถสร้าง image ใหม่สำหรับใช้งานเองได้อีกด้วย



ภาพที่ 2.2 ภาพตัวอย่างของ image ที่นิยมใช้งานบน Docker Hub

```
dockerfile > ...
1   FROM python:3.7.4-slim
2   WORKDIR /usr/src/app
3
4   COPY requirements.txt .
5   RUN pip install --no-cache-dir -r requirements.txt
6
7   COPY webService.py .
8   COPY config.py .
9   COPY Config.json .
10  COPY model ./model
11  COPY InterdaySummaries.csv .
12  COPY InterdaySummaries_GW_to_interday.csv .
13  COPY InterdaySummaries_GW_to_TSCC.csv .
14  COPY HttpRestUtils.py .
15
16  CMD [ "python", "webService.py" ]
17
```

ภาพที่ 2.3 ภาพตัวอย่างการใช้ dockerfile เพื่อสร้าง image มาใช้งานเอง

```
reltracker@pirate01:~$ docker images
REPOSITORY                                                                TAG                 IMAGE ID            CREATED             SIZE
bams-aws.refinitiv.com:5001/tsbkk-innovation/veriprez/veriprez_backend    20191025            ba080179d33c        3 weeks ago         1.06GB
bams-aws.refinitiv.com:5001/tsbkk-innovation/veriprez/veriprez_backend    dev_20191025        ba080179d33c        3 weeks ago         1.06GB
bams-aws.refinitiv.com:5001/tsbkk-innovation/veriprez/veriprez_backend    20190918            a2c0d49136c2        2 months ago        1.05GB
release_tracker_server_prod                                               latest              e74910eb1b34        2 months ago        1.15GB
release_tracker_webapp_prod                                               latest              5983d1089ac4        2 months ago        22.2MB
release_tracker_server_test                                               latest              9f2bc960e76c        2 months ago        1.15GB
release_tracker_webapp_test                                               latest              96bb92d09780        2 months ago        22.2MB
release_tracker_database_prod                                             latest              89e000c95836        2 months ago        372MB
release_tracker_database_test                                             latest              89e000c95836        2 months ago        372MB
bams-aws.refinitiv.com:5001/tsbkk-innovation/veriprez/veriprez_backend    refinitiv           af7151d55944        2 months ago        1.03GB
bams-aws.refinitiv.com:5001/tsbkk-innovation/veriprez/veriprez-mongodb    0                   39e2bd159751        4 months ago        364MB
ubuntu                                                                    16.04               a3551444fc85        6 months ago        119MB
```

ภาพที่ 2.4 ภาพตัวอย่างของ Docker Image ภายในเครื่อง

2.1.2 Docker Container

Docker Container เปรียบเสมือนกับตู้ container ที่ Docker Image นั้นเปรียบได้กับของที่จะนำไปบรรจุไว้อยู่ในตู้ container นั่นเอง โดยภายใน Docker Container นั้นจะมีการติดตั้งเพื่อให้สามารถใช้งาน service ที่ต้องการได้โดยมีพื้นฐานมาจาก image นั้น ๆ โดย Docker Container นั้นจะมีชื่อ, tag และ config ต่าง ๆ ที่จะแตกต่างกันออกไปแล้วแต่ผู้ใช้งานจะตั้งค่า โดยผู้ใช้งานสามารถสั่ง start container เมื่อต้องการจะใช้งาน และสั่ง stop เมื่อต้องการให้ container นั้นหยุดการทำงานได้

```
reltracker@pirate01:~$ docker ps
CONTAINER ID        IMAGE                                                                                    COMMAND                  CREATED             STATUS              PORTS
                    NAMES
c02a91e8b4f2        bams-aws.refinitiv.com:5001/tsbkk-innovation/veriprez/veriprez_backend:20191025          "npm start"              3 weeks ago         Up 3 weeks          0.0.0.0:8088->
081/tcp             veriprezbackend_prod
512b022466c9        bams-aws.refinitiv.com:5001/tsbkk-innovation/veriprez/veriprez_backend:dev_20191025      "npm start"              3 weeks ago         Up 3 weeks          0.0.0.0:8082->
081/tcp             veriprezbackend_dev
7d1044e92a5a        bams-aws.refinitiv.com:5001/tsbkk-innovation/veriprez/veriprez-mongodb:0                 "/usr/bin/mongod"        2 months ago        Up 5 weeks          0.0.0.0:27017-
27017/tcp           veriprezdb_prod
ea44f09e501b        release_tracker_database_prod                                                            "docker-entrypoint.s…"   2 months ago        Up 5 weeks          0.0.0.0:3306->
306/tcp, 33060/tcp  release_tracker_database_prod
```

ภาพที่ 2.5 ภาพตัวอย่างของ Docker Container ที่ทำงานอยู่ภายในเครื่อง

การจะสร้าง Docker Container นั้น ยังสามรถทำได้อีกวิธีหนึ่ง นั่นคือการใช้ Docker Compose โดย Docker Compose นั้นจะอยู่มีโครงสร้างแบบ yml ทำหน้าที่คล้ายกับ script ที่เอาไว้สั่งงานเพื่อสร้าง container โดยจะมีการ config ค่าต่าง ๆ เอาไว้ภายใน Docker Compose ก่อนแล้ว จึงทำให้มีความสะดวกกว่าการ start container แบบปกตินั่นเอง

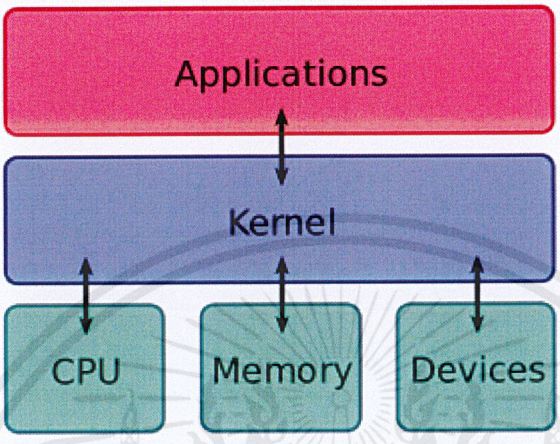
```
version: '3.1'
services:
  mysql:
    image: mysql:5.7
    ports:
      - 3308:3306
    volumes:
      - "./data:/var/lib/mysql"
    environment:
      MYSQL_ROOT_PASSWORD: 1234
      MYSQL_DATABASE: testhaha
  mongo:
    image: mongo
    restart: always
    volumes:
      - "./datamongo:/data/db"
    ports:
      - 27017:27017
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    ports:
      - 8000:80
    environment:
      PMA_PASSWORD: 1234
      PMA_USER: root
      PMA_HOSTS: mysql
```

ภาพที่ 2.6 ภาพตัวอย่างของ Docker Compose

2.2 Linux

Linux นั้นเป็น ระบบปฏิบัติการ (Operating System) หนึ่งที่มีความสามารถสูง มีลักษณะคล้ายการจำลองการทำงานมาจาก Unix แต่จะมีความยืดหยุ่นในการทำงานมากกว่า เป็น

ระบบปฏิบัติการที่ทางผู้พัฒนานั้นแจกให้ใช้งานได้ฟรี (Open Source) ตามความหมายของ Linux แล้วจริง ๆ หมายถึง Linux kernel หรือ operating system kernel ซึ่งทำหน้าที่เป็นตัวกลางเชื่อต่อระหว่าง hardware และ application เพื่อบริหารจัดการ resource ที่มีอยู่ให้เหมาะสม



ภาพที่ 2.7 ภาพ Linux kernel

ส่วนประกอบของ Linux operation system

- The Bootloader: ทำหน้าที่จัดการเรื่องการ boot ของ computer
- The kernel: เปรียบเสมือนคำเรียกของ “Linux” เพราะมันคือระบบส่วนกลางที่ทำหน้าจัดการทรัพยากรต่าง ๆ เช่น CPU memory และ อุปกรณ์ต่อเสริมต่าง เป็น layer ต่ำสุดที่อยู่ใกล้กับ OS
- Daemons: เป็นส่วนที่ทำงานอยู่เบื้องหลัง (background service) เริ่มทำงานตั้งแต่ระหว่างที่ boot และ เริ่ม login เข้าสู่ระบบ
- Shell: คือการทำงานของคำสั่งที่ทำให้สามารถควบคุมและสั่งการผ่านการพิมพ์ตัวอักษรเข้าไป
- Graphical Server: เป็นระบบที่ช่วยเสริมการแสดงผลบนจอ monitor
- Desktop Environment: คือส่วนที่ user ใช้งานจริง ซึ่งมีให้เลือกได้หลายที่โดยซึ่งก็คือชุดของ application ต่าง ๆ ที่ถูกจำมารวมกัน เช่น managers configuration tools web browsers games
- Applications: Linux มี software ที่มีคุณภาพที่ง่ายต้องการค้นหาแล้วติดตั้ง Linux ที่ได้รับความนิยมส่วนใหญ่มักจะมีเครื่องมือที่ใช้สำหรับค้นหาและติดตั้ง application ติดมาให้ เช่น

Ubuntu Linux ก็จะมี software center คือ apt ที่ใช้ในการ download และ ติดตั้ง application จากศูนย์กลาง

ภาพที่ 2.8 ภาพตัวอย่างการใช้งาน Ubuntu

2.3 Application Program Interface (API)

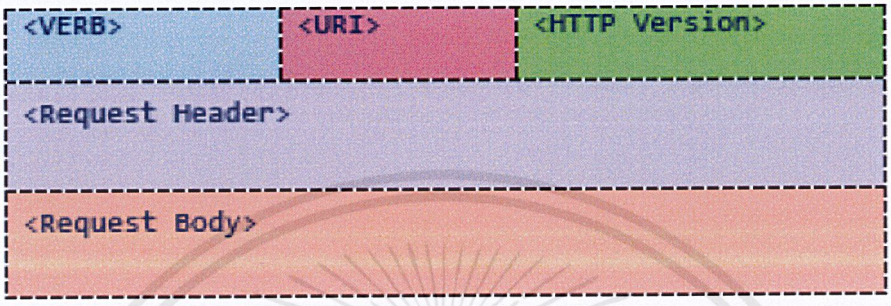
Application Program Interface (API) คือ ช่องทางการเชื่อมต่อระหว่างผู้ใช้งานกับ server โดยมีจุดประสงค์เพื่อให้ผู้ใช้งานรับและส่งข้อมูลกับ server ได้

2.3.1 Representational state transfer (REST)

Representational state transfer (REST) เป็นการสร้าง API แบบหนึ่งโดยใช้หลักการแบบ stateless นั่นคือจะไม่มี session ซึ่งต่างจาก Webservice แบบอื่น เช่น WSDL และ SOAP

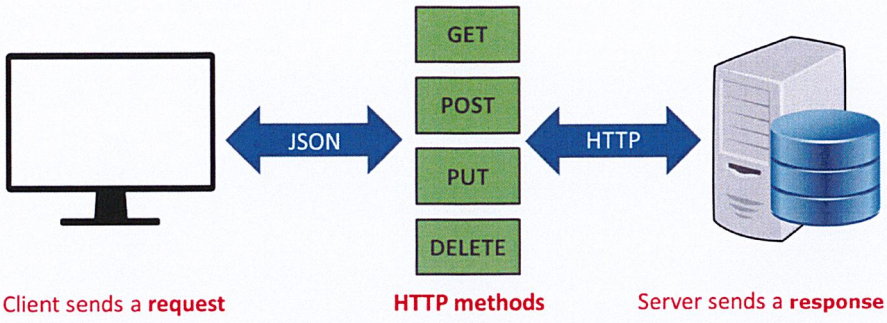
การทำงานของ RESTful Webservice นั้นจะอาศัย URI/URL ของ request สำหรับการค้นหาและประมวลผลแล้วตอบกลับไปให้ผู้ใช้งานในรูปแบบ XML HTML หรือ JSON โดยจะใช้ HTTP Protocol ในการรับและส่งข้อมูลระหว่างผู้ใช้งานและ RESTful API ดังนี้

HTTP Request



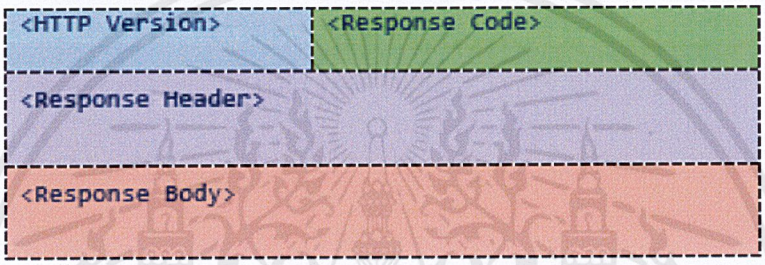
ภาพที่ 2.9 ภาพโครงสร้างของ HTTP Request

- Verb - เป็นส่วนของ HTTP Method โดยจะมีดังนี้
  HTTP Method
  - GET - ดึงข้อมูลจาก REST API
  - POST - สร้างข้อมูลใน REST API
  - PUT - แก้ไขข้อมูลใน REST API
  - DELETE - ลบข้อมูลใน REST API
- URI - เป็นที่อยู่ของข้อมูล
- HTTP Version - บอก version ของ HTTP ที่จะใช้งาน เช่น HTTP v1.1 เป็นต้น
- Request Header - ใช้เก็บค่า key-value ของ header เพื่อบอกข้อมูลผู้ส่ง เช่น format ของข้อมูล body
- Request Body - ส่วนข้อมูล content ที่ต้องการจะ request



ภาพที่ 2.10 ภาพการรับ-ส่งข้อมูลระหว่างผู้ใช้งานกับ REST API

HTTP Response



ภาพที่ 2.11 ภาพโครงสร้างของ HTTP Response

- Response Code - ผลลัพธ์การทำงานในระดับ HTTP โดยจะแสดงอยู่ในรูปของเลข 3 หลัก ดังตารางต่อไปนี้

Code	Message
200	OK
201	Created
204	No content
301	Moved permanently
302	Moved temporarily

400	Bad request
403	Forbidden
404	Not found
405	Method not allowed
500	Internal server error

ตารางที่ 2.1 ตารางแสดง HTTP Response Code

- Response Header ส่วนของ metadata ที่ใช้เก็บค่า key-value ของ header
- Request Body ส่วนข้อมูลผลลัพธ์ content ใน REST

JSON

```
{
 "ID": "1",
 "Name": "M Vaqqas",
 "Email": "m.vaqqas@gmail.com",
 "Country": "India"
}
```

XML

```
<Person>
<ID>1</ID>
<Name>M Vaqqas</Name>
<Email>m.vaqqas@gmail.com</Email>
<Country>India</Country>
</Person>
```

ภาพที่ 2.12 ภาพตัวอย่าง HTTP Response ในรูปแบบ JSON และ XML

2.4 ภาษา Java

ภาษาจาวา (Java programming language) เป็นภาษาที่ใช้เขียนโปรแกรมเชิงวัตถุ เรียกว่าโอโอพี (OOP) ย่อมาจาก Object-Oriented Programming โดยโปรแกรมที่เขียนขึ้นมานั้นจะถูกสร้างไว้ภายในคลาส (Class) ซึ่งในแต่ละคลาสจะเรียกส่วนที่เป็นโปรแกรมเหล่านั้นว่าเมธอด (Method) หรือพฤติกรรม (Behavior) โดยทั่วไปแล้วจะเปรียบเทียบคลาสว่าเป็นวัตถุ แต่ละวัตถุสามารถมีพฤติกรรมเกิดขึ้นได้มากมาย

การที่โปรแกรมประกอบด้วยหลายวัตถุหรือหลายคลาสมารวมกัน จะทำให้โปรแกรมนั้นมีความสมบูรณ์ นอกจากนี้ภาษาจาวายังถูกพัฒนาขึ้นมาให้สามารถรองรับรูปแบบของซอฟต์แวร์ที่มีการเชื่อมต่ออินเตอร์เน็ตได้อีกด้วย

แนวคิดการเขียนโปรแกรมเชิงวัตถุ

การเขียนโปรแกรมเป็นเหมือนการรวมกลุ่มของวัตถุ (Object) โดยแต่ละกลุ่มของวัตถุจะอยู่ในรูปของคลาส ซึ่งแต่ละคลาสอาจมีคุณสมบัติที่เหมือนหรือแตกต่างกันไป ดังต่อไปนี้

1) Encapsulation เป็นการรวมกลุ่มของข้อมูลและกลุ่มของโปรแกรม
เพื่อทำการปกป้องข้อมูล โดยใช้คีย์เวิร์ด ได้แก่ public private และ protected

2) Inheritance เป็นการนำโค้ดกลับมาใช้ใหม่ ในรูปแบบของการนำคลาส
ที่เคยทำการประกาศไว้มาทำการปรับปรุงหรือแก้ไขให้เกิดเป็นคลาสใหม่ ทำให้ไม่ต้องเริ่มต้นเขียนโค้ดใหม่ทั้งหมด คลาสที่เคยประกาศไว้แล้ว เรียกว่า คลาสแม่ (Superclass) คลาสใหม่ที่เกิดขึ้น เรียกว่า คลาสลูก (Subclass)

3) Polymorphism เป็นลักษณะของการทำงานของออปเจ็กต์ สามารถ
แบ่งออกได้เป็น 2 ลักษณะ

3.1) Overloading เป็นการกำหนดคุณสมบัติให้สามารถสร้างเมธอดชื่อเดียวกันให้อยู่ในคลาสเดียวกันได้ แต่เมธอดนั้นจะต้องมีจำนวนพารามิเตอร์ที่ต่างกัน หรือมีจำนวนเท่ากันแต่ต่างชนิดกันด้วย

3.2) Overriding เป็นการกำหนดคุณสมบัติที่มาพร้อมกับการสืบทอดกันของคลาส โดยกำหนดให้สามารถสร้างเมธอดของคลาสลูก ที่มีชื่อเดียวกันให้อยู่ในคลาสเดียวกันได้ แต่เมธอดนั้นจะต้องมีจำนวนพารามิเตอร์ที่ต่างกัน หรือมีจำนวนเท่ากันแต่ต่างชนิดกันด้วย

หลักการทำงานของภาษาจาวา

ภาษาจาวาเป็นภาษาที่ถูกเขียนขึ้นมาแค่ครั้งเดียว แต่สามารถนำไปใช้ได้กับระบบปฏิบัติการที่มีลักษณะที่แตกต่างกันออกไปได้ โดยกระบวนการทำงานจะเริ่มจากการเขียนซอร์ซโค้ด (Source code) เป็นไฟล์ที่มีนามสกุลชื่อ .java แล้วทำการคอมไพล์ (Compile) ให้กลายเป็นจาวาไบต์โค้ด (Java Byte Code) เก็บเป็นนามสกุล .class ในการใช้งานจะนำไฟล์ที่เป็นนามสกุล .class มาทำการ

คอมไพล์ให้กับอุปกรณ์หรือระบบนั้น ๆ เพื่อนำไปใช้งานได้ โดยการเขียนภาษาจาวานั้น จะต้องมีชุดพัฒนาโปรแกรมภาษาจาวา เรียกว่า เจดีเค (JDK) ย่อมาจาก Java Development Kit

ข้อดีของภาษาจาวา

1) ภาษาจาวาเป็นภาษาที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุ เหมาะสำหรับการพัฒนาระบบที่มีความซับซ้อน

2) ภาษาจาวาสามารถนำไปใช้ได้กับระบบปฏิบัติการที่แตกต่างกันได้ โดยไม่ต้องทำการเปลี่ยนแปลงแก้ไขหรือเพิ่มเติมส่วนใดเลย

3) ภาษาจาวาสามารถตรวจสอบความผิดพลาดในขณะทำการเขียนโปรแกรมอยู่ได้ ทำให้สามารถแก้ไขข้อผิดพลาดนั้นได้ทันที

4) ภาษาจาวามีความซับซ้อนน้อยกว่าภาษาอื่น ๆ เช่น C++ เมื่อเปรียบเทียบจำนวนโค้ดแล้วจะมีจำนวนที่น้อยกว่าภาษา C++ ส่งผลให้เกิดข้อผิดพลาดได้น้อยกว่า

5) ภาษาจาวาถูกออกแบบมาให้มีความปลอดภัยสูงมาตั้งแต่เริ่มแรก ทำให้มั่นใจได้ว่าการใช้ภาษาจาวาในการเขียนโปรแกรมจะมีความปลอดภัยสูงในการนำไปใช้งาน

6) ภาษาจาวาสามารถใช้งานได้โดยไม่ต้องเสียค่าใช้งาน เนื่องจากมีไอดีอี (IDE) แอปพลิเคชันเซิร์ฟเวอร์ (application server) และไลบรารี่ (library) ต่าง ๆ ให้สามารถใช้งานได้เลย ช่วยลดค่าใช้จ่ายได้มากขึ้น

ข้อเสียของภาษาจาวา

1) ภาษาจาวามีการทำงานที่ช้ากว่าภาษาอื่น ๆ ในเรื่องของการคอมไพล์โค้ดเพราะภาษาจาวามีการแปลงให้กลายเป็นภาษากลางก่อนที่จะแปลงเป็นภาษาของเครื่องอีกที ทำให้ขั้นตอนการทำงานมีมากกว่า ในขณะที่ภาษาอื่น ๆ เช่น C++ สามารถคอมไพล์แล้วได้เป็นภาษาเครื่องได้เลย ทำให้การใช้ภาษาจาวาอาจไม่เหมาะกับการทำงานที่ต้องการความรวดเร็ว

2) เครื่องมือที่ใช้ในการพัฒนาโปรแกรมจาวาอาจไม่สามารถทำงานได้ดีในบางการทำงาน ทำให้ผู้เขียนโปรแกรมต้องทำการคิดวิธีการขึ้นมาเอง ทำให้เสียเวลาในการทำงานมากขึ้น

2.5 ภาษา Python

ภาษา Python เป็นภาษาที่ใช้ในการเขียนโปรแกรมระดับสูง สร้างโดย Guido van Rossum และถูกเผยแพร่ครั้งแรกในปี 1991 Python นั้นถูกสร้างขึ้นมาจากภาษา C เป็นภาษาแบบ interpreter โดยออกแบบให้อ่านโค้ดได้ง่ายขึ้น และโครงสร้างของภาษานั้นทำให้ผู้เขียนเข้าใจโค้ดได้โดยใช้บรรทัดที่น้อยกว่าภาษา C++ และ Java ภาษา Python นั้นเป็น OpenSource เหมือนอย่าง PHP ทำให้ทุกคนสามารถที่จะนำ Python มาพัฒนาโปรแกรมของเราได้ฟรีโดยไม่ต้องเสียค่าใช้จ่าย และความเป็น Open Source ทำให้มีคนเข้ามาช่วยกันพัฒนาให้ Python มีความสามารถสูงขึ้น และใช้งานได้ครบคุมกับทุกลักษณะงาน

2.6 Scrum

Scrum (สกรัม) คือการนำแนวคิดในการทำงานแบบ Agile (อาไจล์) มาปฏิบัติตามขั้นตอนของสกรัม เพื่อระบุปัญหาที่มีความซับซ้อน เปลี่ยนแปลงบ่อย เพื่อให้สามารถส่งมอบผลิตภัณฑ์ที่ตอบสนองต่อการเปลี่ยนแปลงที่เกิดขึ้นได้อย่างรวดเร็ว

Scrum มีลักษณะ ดังนี้

- ไม่ซับซ้อน
- เข้าใจง่าย
- แต่ยากที่จะนำไปใช้ได้อย่างชำนาญ

วัตถุประสงค์ของ Scrum

- กำหนดทิศทางและความสามารถของผลิตภัณฑ์ อะไรควรมี อะไรไม่ควรมี
- เพิ่มคุณค่าให้ผลิตภัณฑ์
- ส่งมอบผลิตภัณฑ์ได้รวดเร็วและบ่อยมากขึ้น
- วงจรอายุการใช้งาน(Life cycle) ของผลิตภัณฑ์ที่ยั่งยืน

ทฤษฎี Scrum

Scrum เน้นการนำความรู้จากประสบการณ์เฉพาะที่เคยลงมือทำจริง (Empiricism) โดย Empiricism ชี้ชัดว่าความรู้นั้นได้ มาจากประสบการณ์ตรงและการตัดสินใจทำจากตัวแปรเท่าที่รู้ โดยที่ Scrum แบ่งการทำงานเป็นช่วง ๆ และมีผลงานเพิ่มขึ้นอย่างต่อเนื่อง ทำให้การคาดการณ์ มีความ

แม่นยำและยังควบคุมความเสี่ยงต่าง ๆ ได้อีกด้วย ในการควบคุมการทำงานแบบ Empiricism มีหลัก 3 ประการคือ

1. **ความโปร่งใส (Transparency)** คือทีมจะต้องเห็นภาพชัดเจนและเข้าใจตรงกัน มาตรฐานเดียวกัน ไม่ตีความหมายต่างกัน เช่น นิยามของคำว่างานเสร็จ หมายถึง การผลิตเสร็จ หรือผลิตและทดสอบเสร็จ หรือ ได้รับการเซ็นรับรอง หรือ ส่งมอบให้ผู้ใช้แล้ว ต้องนิยามและตกลงให้เข้าใจตรงกัน
2. **การตรวจสอบ (Inspection)** คือการนำผลลัพธ์การดำเนินกิจกรรมต่าง ๆ ของสกรัม (Scrum Artifact) มาตรวจสอบและวัดผลว่าบรรลุตามที่กำหนดไว้หรือไม่
3. **การปรับเปลี่ยน (Adaption)** คือหากผลลัพธ์ไม่เป็นไปตามที่กำหนด จะต้องปรับเปลี่ยนการดำเนินงาน หรือจำนวนทรัพยากรที่ใช้ เพื่อให้บรรลุผลตามที่กำหนดหรือใกล้เคียงได้มากที่สุด

ตำแหน่งในทีมสกรัม

สกรัมเหมาะสำหรับทีมขนาดเล็กที่พร้อมปรับตัว พัฒนา และเปลี่ยนแปลง สมาชิกในทีมสกรัมต้องมีความสามารถที่หลากหลาย สามารถบริหารและดำเนินงานกันเองได้ด้วยสมาชิกภายในทีม สามารถหาแก้ไขปัญหาได้เอง โดยไม่ต้องรอความช่วยเหลือจากนอกทีม โดยประกอบด้วย

1. Product Owner

- บริหารจัดการ Product Backlog ให้ชัดเจน
- จัดลำดับความสำคัญของงานใน Product Backlog และสามารถอธิบายเหตุผลได้
- ทำให้ทีมเข้าใจรายละเอียดของ Product Backlog ตรงกัน
- หาทางเพิ่มผลการดำเนินงานให้มีประสิทธิภาพมากขึ้น

2. Developer

- สมาชิกมีจำนวน 3–9 คน เพื่อให้สามารถรับปริมาณงานได้ไม่น้อยเกินไป และไม่เสียเวลาในการประสานงานมากเกินไป
- พัฒนางานตามที่วางแผนเอาไว้
- ไม่มีการแบ่งทีมย่อยภายใต้ทีมพัฒนาอีก เช่น ทีมออกแบบ ทีมทดสอบ
- สมาชิกทีมแต่ละคนจะมีความสามารถเฉพาะทางของตนเอง เช่น ความสามารถในการออกแบบ แต่ความรับผิดชอบงานจะเป็นของทั้งทีม หากงานออกแบบไม่เสร็จทั้งทีมต้องมาช่วยกันรับผิดชอบ

3. Scrum Master

- ทำให้ทีมเข้าในสกรัมและนำไปใช้ได้อย่างถูกต้อง

- ช่วยเหลือการดำเนินงานในขั้นตอนต่าง ๆ
- รู้ว่าการดำเนินงานแบบไหนดีหรือไม่ดีต่อทีม
- ทำให้ทีมเข้าใจขอบเขตของผลิตภัณฑ์

User Story

User Story ใช้สำหรับระบุความต้องการของผู้ใช้ให้ชัดเจน และระบุเกณฑ์การทดสอบว่าผลิตภัณฑ์ที่พัฒนาขึ้นมาตอบสนองความต้องการได้จริงหรือไม่ โดย User Story ที่ดีต้องทำให้ Developer เข้าใจสโคป (Scope) ของงานได้อย่างชัดเจนว่าอะไรที่เกี่ยวข้องแล้วต้องพัฒนาขึ้น และอะไรที่ไม่เกี่ยวข้อง เพื่อจะได้ไม่ต้องเสียเวลาในการพัฒนา

Product Backlog

งานทั้งหมดที่ต้องทำเพื่อพัฒนาผลิตภัณฑ์ ส่วนใหญ่นิยมเขียนในรูปแบบของ User Story โดยมีรายละเอียดของงาน เกณฑ์การทดสอบงาน การประเมินความซับซ้อนและเวลาที่ต้องใช้ในการพัฒนาด้วย เมื่อพัฒนาเสร็จและส่งมอบแล้ว อาจนำผลตอบรับจากผู้ใช้มาทบทวนและปรับปรุงผลิตภัณฑ์เพิ่มเติม โดยเขียนเป็น User Story ใหม่ เมื่อ Product Backlog มีการเปลี่ยนแปลง Product Owner จะต้องจัดเรียงความสำคัญใหม่ และแก้ไขรายละเอียดของ User Story ที่มีความสำคัญสูงให้พร้อมสำหรับนำไปพัฒนาได้ทันที การทำ Product Backlog จะมีการเปลี่ยนแปลงอยู่เรื่อย ๆ ตามความต้องการของผู้ใช้และตลาดเพื่อให้ผลิตภัณฑ์ของเราตอบสนองความต้องการได้มากที่สุด

กิจกรรมของสกรัม (Scrum Events)

การทำกิจกรรมของสกรัมเพื่อให้การดำเนินงานเป็นขั้นตอนชัดเจน ตรวจสอบ วัดผลได้ และลดการประชุมที่ไม่จำเป็น การทำสกรัมจะประกอบไปด้วยกิจกรรมต่าง ๆ ดังนี้

1. วางแผนสปรินท์ (Sprint Planning)

โดยจะแบ่งการวางแผนสปรินท์ออกเป็นสองส่วน

1.1 เลือกว่าจะทำงานอะไร

- Product Owner จะกำหนดเป้าหมายของสปรินท์ว่าจะส่งมอบอะไร เพราะอะไร และเพื่ออะไร ทำให้ทีมเข้าใจว่าต้องทำงานชิ้นไหนบ้างเพื่อให้เกิดอะไร

- Product Owner เลือก User Story จาก Product Backlog มาแจ้งให้ทีมทราบว่าต้องทำงานชิ้นไหนบ้างเพื่อที่จะบรรลุเป้าหมายของสปรินท์
- Developer จะเลือกงานที่ Product Owner เลือกไว้ใน Product Backlog เข้าสู่ Sprint Backlog เอง เพราะรู้ขีดจำกัดว่างานใดที่สามารถนำไปพัฒนาได้หรือต้องรองานส่วนอื่นเสร็จก่อน และปริมาณงานที่ทีมสามารถพัฒนาได้เสร็จในสปรินท์

1.2 ออกแบบวิธีว่าจะทำงานอย่างไร

- Developer ออกแบบแนวทางที่จะใช้ในการพัฒนางานแต่ละชิ้น
- ระบุส่วนที่ต้องรีบทำก่อนส่วนอื่น(ถ้ามี)
- แบ่งชิ้นงานขนาดใหญ่ให้เป็นชิ้นงานเล็ก ๆ
- อาจทำการประเมินความซับซ้อนและเวลาที่ใช้ในการพัฒนางานใหม่
- อาจนำผลการประเมินมาจัดเรียงความสำคัญของงานใหม่
- อาจมีการต่อรองปริมาณใน Sprint Backlog กับ Product Owner ใหม่ หากพบว่าการพัฒนามีความยากกว่าที่ประเมินไว้ในตอนแรก

2. สกรัมประจำวัน (Daily Scrum)

การประชุมประจำวัน (Daily Meeting) หรืออาจจะเรียกว่า Standup Meeting เพราะเป็นการล้อมวงยืนประชุมในเวลาสั้น ๆ ใช้เวลาไม่เกิน 15 นาที เน้นให้ Developer แจ้งความคืบหน้าในการพัฒนางานแก่กัน

- เพื่อตรวจสอบและแจ้งความคืบหน้าของงานในสปรินท์
- เพื่อเป็นการวางแผนทำงานในแต่ละวัน
- แต่ละคนแจ้งให้ทีมทราบว่า 1.ทำอะไรไปในเมื่อวาน 2.วันนี้จะทำอะไรเพิ่มเติม 3.ปัญหาที่เกิดขึ้นในการพัฒนา
- แจ้งให้ทีมทราบหากมีงานอื่นแทรกเข้ามา
- หากพบปัญหาหรือการเปลี่ยนแปลงที่ส่งผลต่อการบรรลุเป้าหมายของสปรินท์ อาจจัดประชุมหลังทำสกรัมประจำวันเพื่อวางแผนงานที่เหลือในสปรินท์ใหม่ โดยอาจเปลี่ยนแปลงสโคปงานหรือโยกย้ายงานที่จำเป็นน้อยกว่าออกจากสปรินท์

ภาพที่ 2.15 ภาพตัวอย่างการทำ Daily Scrum

3. ตรวจสอบผลลัพธ์ของสปรินท์ (Sprint Review)

ตรวจสอบงานที่พัฒนาและแสดงผลลัพธ์ของงานในสปรินท์ให้แก่ผู้เกี่ยวข้อง(Stakeholders) เพื่อรับฟังข้อเสนอแนะ และทบทวน Product Backlog ที่จะทำต่อไปให้สอดคล้องกับโอกาสและสถานการณ์ในปัจจุบัน

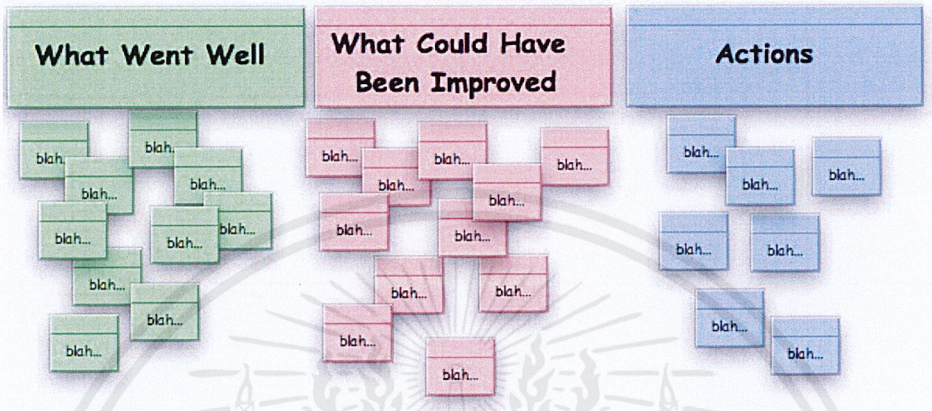
- Product Owner อธิบายว่ามีงานอะไรเสร็จหรือไม่เสร็จ
- Developer อธิบายการดำเนินงาน และปัญหาที่เกิดขึ้นรวมถึงวิธีที่แก้ไข
- Demo งานที่พัฒนาขึ้นในสปรินท์
- พิจารณาคุณค่าของงานที่ได้พัฒนาขึ้นมาและเลือกงานที่จะส่งมอบให้แก่ผู้ใช้
- ทบทวนทรัพยากรที่มี ทีมงาน เวลา เครื่องมือ ตลาด
- ทบทวนทิศทางตลาดและสถานการณ์ที่เปลี่ยนไป
- Product Owner อธิบายงานที่เหลือใน Product Backlog เกริ่นคร่าว ๆ ว่าจะทำอะไรเป็นลำดับต่อไปและจะเสร็จเมื่อไร โดยดูได้จาก Burndown Chart เป็นต้น

4. ตรวจสอบการดำเนินงานของสปรินท์ (Sprint Retrospective)

ตรวจสอบการดำเนินงานในสปรินท์ที่จบลง ทั้งในเรื่องของทีมงาน ความสัมพันธ์ภายในทีม ความรู้ เครื่องมือ สภาพแวดล้อมในการทำงาน เป็นต้น

- แต่ละคนแจ้งให้ทีมทราบว่าสปรินท์ที่จบลงมีอะไรที่ดีและไม่ดีบ้าง
- จัดลำดับความสำคัญของผลกระทบของสิ่งที่ไม่ดีและต้องการปรับปรุง
- ทีมเสนอแนวทางในการแก้ไขสิ่งที่ไม่ดี เพื่อเพิ่มประสิทธิภาพในการดำเนินงานและคุณภาพของผลิตภัณฑ์

- สรุปวิธีการดำเนินการที่จะใช้ในสปรินท์ถัดไป(ถ้ามี) โดยจะต้องสามารถวัดผลวิธีการดำเนินงานแบบใหม่ เพื่อนำมาเปรียบเทียบกับวิธีดำเนินงานแบบเก่าว่าแบบไหนมีประสิทธิภาพมากกว่ากัน
- Scrum Master เข้าร่วมในสถานะสมาชิกทีม และช่วยควบคุมเวลา



ภาพที่ 2.16 ภาพตัวอย่างการทำ Sprint Retrospective

5. ชี้แจงรายละเอียด Product Backlog (Product Backlog Refining)

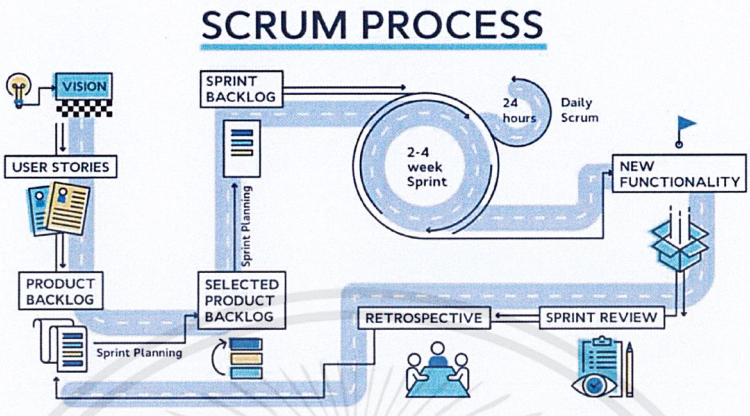
จัดเมื่อไรก็ได้ในสปรินท์โดยให้ทีมตกลงกันเอง ระหว่าง Product Owner และ Developer เพื่อชี้แจงรายละเอียดงาน ประเมินเวลาที่ใช้ในการพัฒนาชิ้นงาน และจัดลำดับความสำคัญของงาน

- Product Owner อธิบาย User Story ต่าง ๆ ใน Product Backlog
- งานที่มีลำดับความสำคัญสูงต้องมีรายละเอียดงานให้ครบ เพื่อให้ Developer สามารถประเมินความซับซ้อนและเวลาที่ต้องใช้ในการพัฒนางานได้อย่างแม่นยำ
- Developer ประเมินความซับซ้อนและเวลาที่ใช้ในการพัฒนางาน
- Product Owner นำผลการประเมินมาช่วยจัดลำดับความสำคัญใหม่
- Product Owner สรุปงานที่เหลือใน Product Backlog เทียบกับงานที่ทำในสปรินท์ปัจจุบัน เพื่อดูว่างานทั้งหมดจะพัฒนาเสร็จเมื่อไร (สามารถใช้ Burndown Chart ช่วยประเมินเวลาที่จะพัฒนาได้)

การทำสกรัมให้มีประสิทธิภาพ

- ทุกคนช่วยกันทำงานเพื่อบรรลุเป้าหมายของทีม ไม่ใช่เฉพาะของเรา
- ให้ความสำคัญกับงานในสปรินท์ก่อน
- พัฒนาผลิตภัณฑ์ในทางที่ถูกต้อง ไม่ซ่อนปัญหา

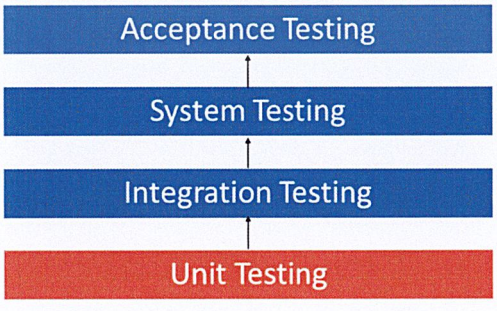
- เปิดใจกว้างต่องานทั้งหมดที่มี และความท้าทายในการทำงาน
- เคารพการทำงานของคนอื่น เพื่อให้แต่ละฝ่ายสามารถทำงานได้อิสระ



ภาพที่ 2.17 ภาพแสดงการทำงานแบบ Scrum

2.7 Unit test

Unit test คือการเขียน test ในส่วนเล็กของโปรแกรมเพื่อแสดงว่ามันทำงานได้อย่างที่ควรทำ โดยการเขียน Unit test นั้นควรทำง่ายได้ง่ายและไว เขียนสั้น กระชับ และเขียนเฉพาะส่วนที่ผู้พัฒนาเห็นว่าทำเป็นต้อง test จุดสำคัญของ Unit test คือ เขียน test เฉพาะส่วนที่โค้ดมี ส่วนไหนที่เป็นภายนอกของโค้ด ให้ทำการ Mock โดย Unit test ไม่ควรมี dependency กับส่วนอื่น ๆ ซึ่งมันเป็นการทดสอบว่าระบบทำงานภายในได้อย่างสมบูรณ์ และถูกต้อง จะแตกต่างจากเป้าหมายที่คนอื่นที่ไม่ใช่โปรแกรมเมอร์นั้นคาดหวังว่ามันควรทำงานกับระบบภายนอกได้



ภาพที่ 2.18 ภาพแสดง Testing Levels

บทที่ 3

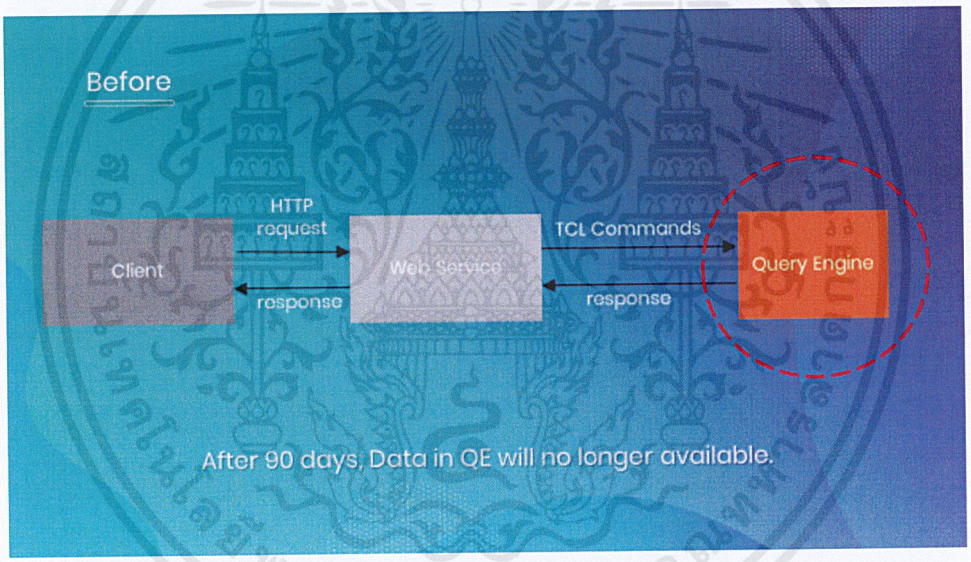
วิธีการดำเนินงาน

3.1 Polly

3.1.1 กระบวนการเข้าใจปัญหาที่เกิดขึ้นของระบบการทำงานและออกแบบระบบ

3.1.1.1 ทำความเข้าใจระบบงานเดิม

ทำความเข้าใจภาพรวม การรับ-ส่งข้อมูล ตัวแปรในการรับส่ง และ Component ต่าง ๆ ในระบบ เพื่อที่จะวางแผน และออกแบบแอปพลิเคชันได้อย่างถูกต้อง



ภาพที่ 3.1 ภาพแสดงการทำงานของระบบก่อนทำ Polly

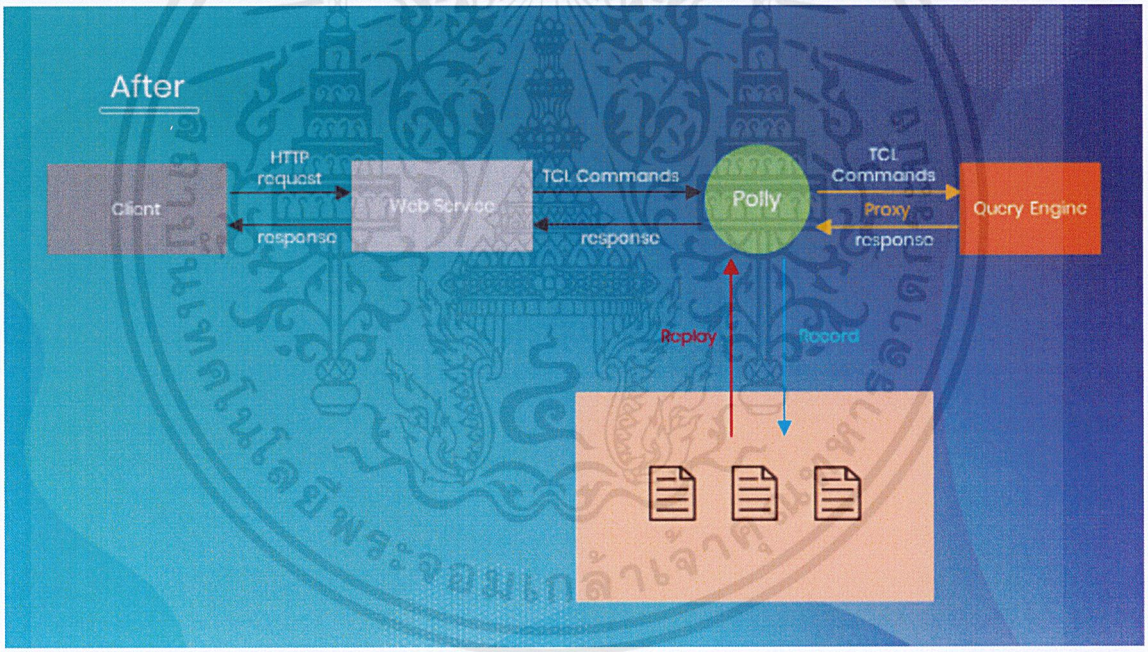
3.1.1.2 ศึกษาปัญหาที่มีต่อความต้องการของผู้ใช้

ทำความเข้าใจปัญหาของการทำ Regression testing โดยหลังจากที่สร้างข้อมูลเป็นเวลา 90 วัน ข้อมูลชุดดังกล่าวจะหายไป ทำให้เมื่อทำการขอข้อมูลย้อนหลังที่มีระยะเวลา 90 วันขึ้นไป จะทำให้ไม่สามารถนำข้อมูลมาทำ Regression testing ได้ จึงต้องทำการหาวิธีการบันทึกข้อมูล และสามารถส่งข้อมูลที่ถูกบันทึกไว้เมื่อมีการร้องขอข้อมูลโดยผู้ใช้ได้อย่างถูกต้อง เพื่อแก้ปัญหาเรื่องข้อมูลที่หายไปหลัง 90 วัน

3.1.2 กระบวนการวิเคราะห์ระบบและออกแบบระบบ

3.1.2.1 การเก็บรวบรวมความต้องการของผู้ใช้งาน

ทำการเก็บ Requirement กับ Project Owner โดยสอบถามถึงการใช้งานที่ต้องการ รูปแบบในการเก็บข้อมูลที่ได้จาก QE โครงสร้างของโฟลเดอร์ที่ทำการเก็บข้อมูลและภาษาที่ใช้ในการพัฒนาแอปพลิเคชัน หลังจากทำการเก็บ Requirement เสร็จก็ทำการออกแบบการทำงานของแอปพลิเคชัน ทางผู้ใช้งานต้องการที่จะให้มี feature อยู่ 3 แบบ คือ Replay Proxy และ Record โดยสามารถระบุ feature ที่ต้องการให้ application ทำผ่านใน request ที่ส่งเข้ามาในแต่ละ request หลังจากได้ทำการสอบถาม และหาความรู้จึงพบว่าการรับ HTTP Request แล้วแปลงเป็น TCL Commands โดย EDP Intraday Web Service นั้น ตัวแปรที่ชื่อ reqID นั้นยังคงมีค่าเหมือนเดิมใน TCL Commands จึงเลือกใช้ตัวแปรนี้ในการระบุว่า request ที่รับเข้าจะให้ application ของเราทำงานอย่างไร

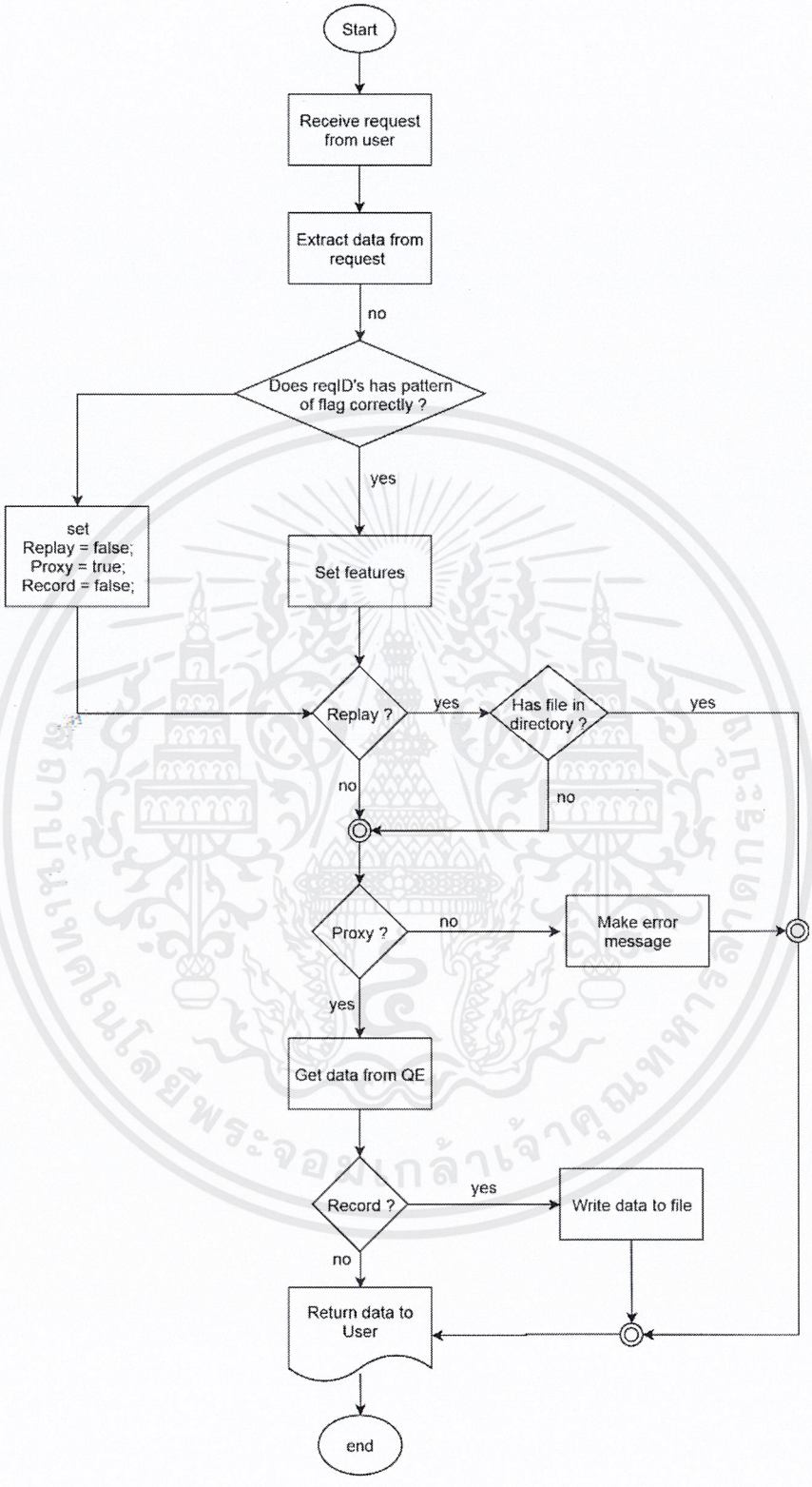


ภาพที่ 3.2 ภาพแสดงการทำงานของระบบหลังจากมี Polly

3.1.2.2 การออกแบบการทำงานของระบบ

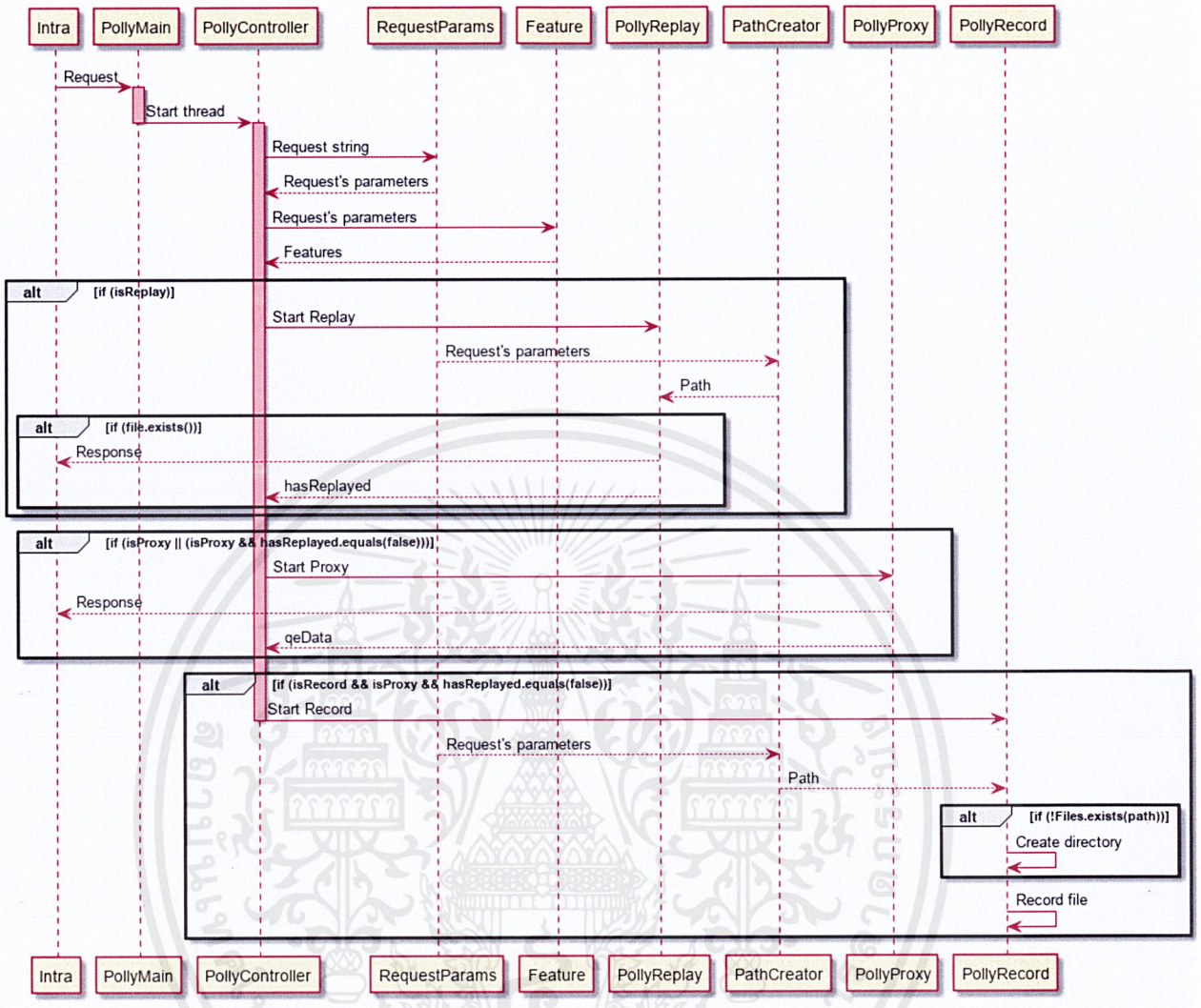
Polly นั้นจะมีรูปแบบการทำงานตาม Flow Chart และ Sequence-Diagram ดังภาพด้านล่างต่อไปนี้

3.1.2.2.1 Flow Chart



ภาพที่ 3.3 ภาพFlow chart แสดงการทำงานของ Polly

3.1.2.2.2 Sequence Diagram



ภาพที่ 3.4 ภาพ Sequence Diagram ของ Polly

3.1.3 การพัฒนาระบบ

3.1.3.1 การรับ Request จากผู้ใช้งาน

ผู้ใช้งานส่ง HTTP Request ไปให้กับ Webservice และ Webservice ก็ทำการแปลง HTTP Request เป็น TCL Commands ในคลาส PollyMain จะทำการสร้าง Server Socket ขึ้นมาเพื่อเป็นช่องทางในการแลกเปลี่ยนข้อมูลกับ Webservice ข้อมูลที่ได้จะอยู่ใน InputStream และข้อมูลที่ผู้ใช้งานส่งมาจะอยู่ใน Byte array และ DataReader object จะทำหน้าที่อ่านข้อมูลที่อยู่ใน Byte array

```
sSock = new ServerSocket( port: 7070);
sSock.setReuseAddress(true);
Socket clientSock = sSock.accept();
InputStream iStream = clientSock.getInputStream();
```

ภาพที่ 3.5 ภาพการสร้างตัวแปร Server Socket และ InputStream

```
/************************* Receive from client ****************************/
dataFromClient = new QeDataReceiver(iStream);
```

ภาพที่ 3.6 ภาพการสร้าง DataReader object เพื่ออ่านข้อมูลที่รับมาจากผู้ใช้งาน

```
public class DataReader {

    public static void readHeader(InputStream inputStream, byte[] headerBuffer) {
        readData(inputStream, headerBuffer,  dataSize: 4);
    }

    public static void readData(InputStream inputStream, byte[] dataBuffer, int dataSize) {
        try {
            int readSize = 0;
            int offset = 0;
            int expectedSize = dataSize;
            while (offset < dataSize) {
                readSize = inputStream.read(dataBuffer, offset, expectedSize);
                offset = offset + readSize;
                expectedSize = expectedSize - readSize;
            }
        } catch (IOException e) {

        }

    }

}
```

ภาพที่ 3.7 ภาพการอ่านข้อมูลที่อยู่ใน Byte stream โดยคลาส DataReader

ในคลาส PollyMain มีการใช้ multithreading ในการรับมือเหตุการณ์ที่มีผู้ใช้งานมากกว่า 1 คนส่ง request เข้ามาพร้อมกัน ทำให้การทำงานของ Polly ในแต่ละ request เกิดขึ้นพร้อมกันแบบ asynchronous ทำให้ใช้เวลาในการทำงานของ Polly เมื่อมีหลาย ๆ request ที่มาจากผู้ใช้งานหลายคนนั้นเร็วกว่าแบบ single thread เนื่องจากไม่ต้องรอให้ request ก่อนหน้าทำงานเสร็จก่อน แต่จะสร้าง thread ใหม่เพื่อมาทำงานในส่วนของอีก request

```
ExecutorService controllerExecutor = Executors.newFixedThreadPool(maxThread);
PollyController controller = new PollyController(firstQeAttrib, dataFromClient, oStream);
controllerExecutor.execute(controller);
```

ภาพที่ 3.8 ภาพการสร้างตัวแปร ExecutorService เพื่อกำหนดจำนวน thread สูงสุด

3.1.3.2 การเลือก Feature ในการทำงาน

การที่ Polly นั้นตัดสินใจว่าจะทำงานอย่างไรนั้น จะเป็นหน้าที่ของตัว Controller ที่เป็นตัวจัดการการทำงานของ Polly โดยใช้ตัวแปรต่าง ๆ ที่อยู่ใน TCL Commands จึงมีการสร้าง RequestParams object ขึ้นมาเพื่อเอาค่าของตัวแปรต่าง ๆ ที่อยู่ใน TCL-Commands มากำหนดเป็นตัวแปรต่าง ๆ เพื่อใช้ในการทำงานในระบบต่อไป และกำหนด Fetaure ของ Request นั้น ๆ ทำได้โดยการใช้ Regular Expression ในการตัดกลุ่มคำที่ตรงกับรูปแบบที่เราได้ทำการกำหนดเอาไว้ เพื่อให้ได้กลุ่มคำที่ต้องการ

```
public PollyController(HostAttributes firstQeAttrib, QeDataReceiver dataFromClient, OutputStream oStream) {
    this.firstQeAttrib = firstQeAttrib;
    this.dataFromClient = dataFromClient;
    this.oStream = oStream;
    clientTclRaw = dataFromClient.getDataRaw();
}

public void run() {
    try {
        String clientTclString = new String(clientTclRaw);
        logger.info( msg: "Request from Intra: " + clientTclString);
        RequestParams requestParams = new RequestParams(clientTclString);
        Feature feature = new Feature(requestParams);
        Boolean isReplay = feature.isReplayEnabled();
        Boolean isProxy = feature.isProxyEnabled();
        Boolean isRecord = feature.isRecordEnabled();

        /********************** Replay **************************/
        if (isReplay) {
            PollyReplay replay = new PollyReplay(oStream, requestParams);
            hasReplayed = replay.startReplay();
        }
        if (hasReplayed.equals(false)) {
            /************************ Proxy ************************/
            if (isProxy) {
                PollyProxy proxy = new PollyProxy(firstQeAttrib, dataFromClient, oStream);
                qeData = proxy.startProxy();
            }
            /************************ Record ************************/
            if (isRecord && isProxy) {
                PollyRecord record = new PollyRecord(requestParams, qeData);
                hasRecorded = record.startRecord();
            } else if(isRecord){
                logger.log(Level.WARNING, msg: "Record need \"Proxy\" flag");
            }
        }
    } catch (Error | Exception e){
        e.printStackTrace();
        System.exit( status: 1);
    }

}
```

ภาพที่ 3.9 ภาพคลาส Controller

```
RequestParams requestParams = new RequestParams(clientTclString);
```

ภาพที่ 3.10 ภาพ RequestParams object

```
static final Pattern flagRegEx = Pattern.compile("(?<=reqID \\{polly)(.*?)(?=_)");
static final Pattern tcRegEx = Pattern.compile("(?<=polly\\d\\d\\d_)(.*?)(?=})");
static final Pattern ricRegEx = Pattern.compile("(?<=instrument )(.*?)(?= /)");
static final Pattern interfaceRegEx = Pattern.compile("(?<=return \\[)(.*?)(?= /)");
static final Pattern policyNameRegEx = Pattern.compile("(?<=policyName )(.*?)(?= /)");
static final Pattern startTimeRegEx = Pattern.compile("(?<=startTime )(.*?)(?= /)");
static final Pattern endTimeRegEx = Pattern.compile("(?<=endTime )(.*?)(?= /)");
static final Pattern refTableRegEx = Pattern.compile("(?<=refTableListBestMatch \\[list )(.*?)(?=\\])");
```

ภาพที่ 3.11 ภาพตัวอย่างการใช้งาน Regular Expression

หลังจากที่ได้ตัวแปรแล้วจะนำมาตรวจสอบเงื่อนไขต่าง ๆ เพื่อกำหนด Feature ว่าจะให้ทำงานด้วย Feature อะไรบ้าง

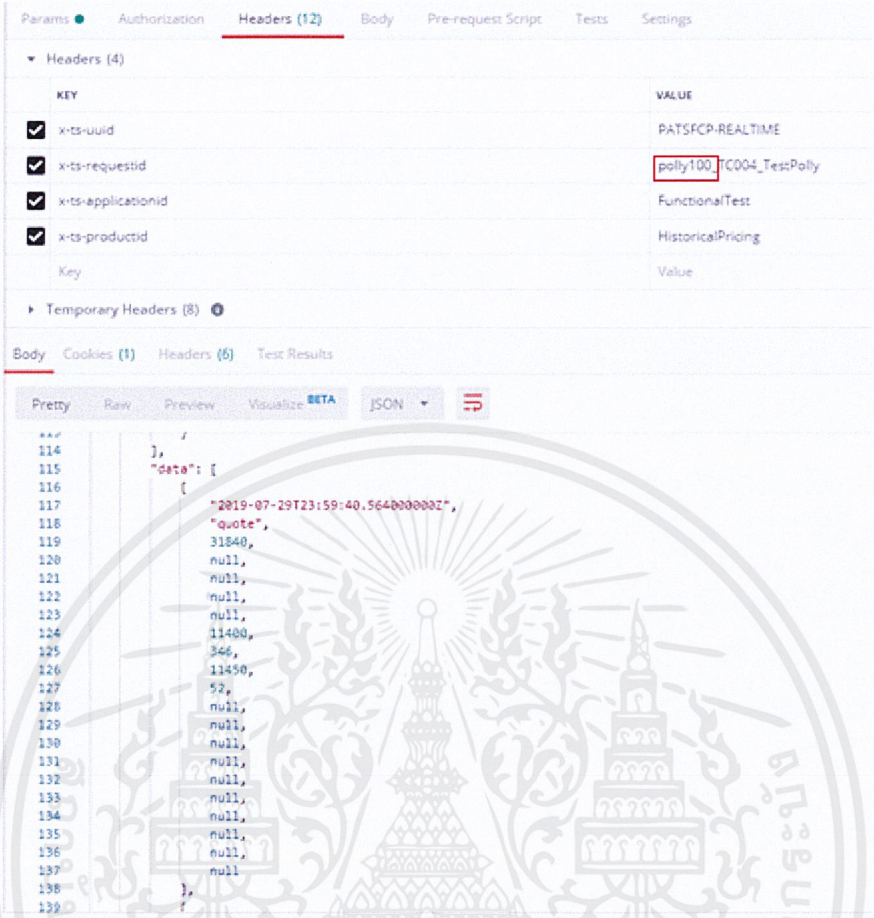
```
Feature feature = new Feature(requestParams);
Boolean isReplay = feature.isReplayEnabled();
Boolean isProxy = feature.isProxyEnabled();
Boolean isRecord = feature.isRecordEnabled();
```

ภาพที่ 3.12 ภาพการสร้าง Feature object โดยใช้ RequestParams object เป็น argument

```
public Feature(RequestParams reqParam) {
    Integer flag = reqParam.getFlag();
    String interfaceName = reqParam.getInterface();
    String testCase = reqParam.getTc();

    for (Velocity velocity : Velocity.values()) {
        if (velocity.name().equals(interfaceName)) {
            if (flag != null && !flag.equals("")) {
                if ((flag & replayEnum) == replayEnum) {
                    feature |= replayEnum;
                }

                if ((flag & proxyEnum) == proxyEnum) {
                    feature |= proxyEnum;
                }

                if ((flag & recordEnum) == recordEnum) {
                    feature |= recordEnum;
                }
            }

        } else if (flag == null || testCase == null) {
            feature = proxyEnum;

        } else if(flag == 0){
            logger.log(Level.WARNING, msg: "You're not select any features!(\"polly000\")");
        }
    }
}

/********************** Set feature *************************/

public Boolean isReplayEnabled() { return ((feature & replayEnum) == replayEnum); }

public Boolean isProxyEnabled() { return ((feature & proxyEnum) == proxyEnum); }

public Boolean isRecordEnabled() { return ((feature & recordEnum) == recordEnum); }
```

ภาพที่ 3.13 ภาพการตรวจสอบเงื่อนไขเพื่อกำหนด Feature ในการทำงาน

3.1.3.3 การทำงานของ Feature Replay



ภาพที่ 3.14 ภาพตัวอย่างการส่ง request โดยใช้ Feature Replay

เมื่อ controller ได้ตรวจสอบแล้วว่าสถานะของ Feature Replay เป็น True โดยจะตรวจสอบได้จากflag ที่อยู่ในตัวแปร x-ts-requestid ใน Header ของ request ที่ผู้ใช้งานนั้นส่งมา โดยเลขประจำหลักของ Feature Proxy จะอยู่หลักที่อยู่หน้าสุดของ flag ถ้าเป็น 1 จะทำให้สถานะของ Feature Replay เป็น True หลังจากตรวจสอบสถานะของ Feature Replay แล้วจะทำการสร้าง object ของคลาส PollyReplay ขึ้นมา และทำการ start ในขั้นแรกจะทำการตรวจสอบว่า path ที่ถูกสร้างจากตัวแปรต่าง ๆ ที่อยู่ใน TCL Commands นั้นมีอยู่ในเครื่องแล้วหรือไม่ ถ้ายังไม่มีจะทำการสร้าง directory ขึ้นมาใหม่ เพื่อเป็นที่ในการเก็บไฟล์ที่บันทึกข้อมูลที่ได้จากการขอข้อมูลจาก QE ถ้ามี path นั้นอยู่ในเครื่องก็จะทำการตรวจสอบต่อว่ามีไฟล์อยู่ในdirectory หรือไม่ ถ้าไม่มีจะทำการแสดงการแจ้งเตือนบน console แต่ถ้ามีแล้วจะทำการอ่านข้อมูลที่อยู่ในไฟล์นั้นโดยการอ่านไฟล์จะทำหน้าที่โดยคลาส FileDataReceiver

```
Nov 12, 2019 4:38:51 PM polly_qe.features.PollyReplay startReplay
WARNING: No file in directory
```

ภาพที่ 3.15 ภาพตัวอย่างการแจ้งเตือนเมื่อใช้ Replay แต่ไม่มีไฟล์

```
public Boolean startReplay() {

    /********************** Replay by recorded file ************************/
    path = PathCreator.getPath(reqParams);
    String fileName = "recordedFile.txt";

    String fileStorePath = path + "/" + fileName;
    File file = new File(fileStorePath);

    if (file.exists()) {
        int fileSize = (int) file.length();

        //Read from recorded file
        FileDataReceiver fileData = new FileDataReceiver(fileStorePath, fileSize);
        dataSizeRaw = fileData.getDataSizeRaw();
        readSize = fileData.getReadSize();
        dataRaw = fileData.getDataRaw();

        //Send data back to client
        DataWriter.writeData(oStream, dataSizeRaw, readSize, dataRaw);
        logger.info( msg: "Reply to client(Replay by file): ");

        hasReturnedData = true;

    } else {
        logger.log(Level.WARNING,  msg: "No file in directory");
    }

    return hasReturnedData;
}
```

ภาพที่ 3.16 ภาพคลาส PollyReplay

```
package polly_qe.data.Stream;

import ...

public class FileDataReceiver {
    private FileInputStream fis;
    private byte[] dataSizeRaw;
    private int readSize;
    private byte[] dataRaw;

    public FileDataReceiver(String fileStorePath, int fileSize){
        try {
            fis = new FileInputStream(fileStorePath);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        dataRaw = ByteBuffer.allocate(fileSize + 100).array();

        try {
            readSize = fis.read(dataRaw,  off: 0,  len: fileSize + 100);
            fis.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

        dataSizeRaw = ByteBuffer.allocate(4).putInt(readSize).array();

    }

    public byte[] getDataSizeRaw() { return dataSizeRaw; }

    public int getReadSize(){ return readSize; }

    public byte[] getDataRaw() { return dataRaw; }
}
```

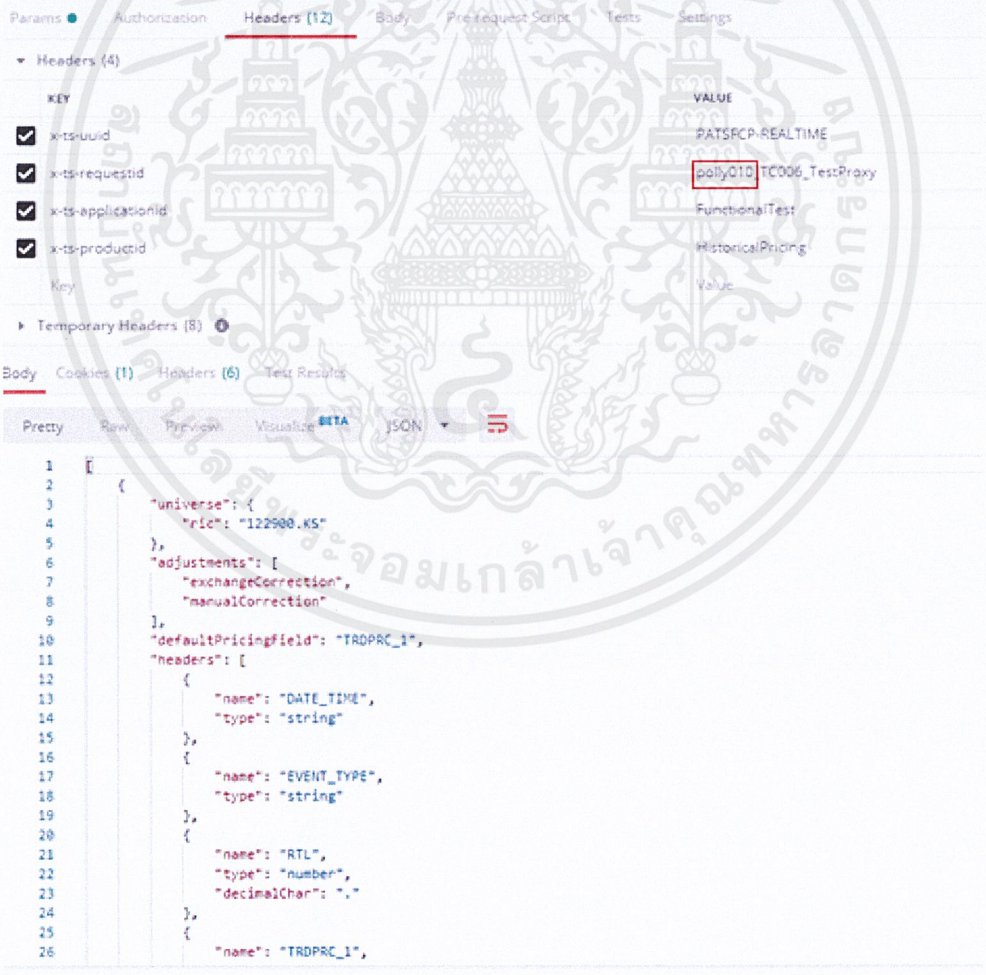
ภาพที่ 3.17 ภาพคลาส FileDataReceiver

หลังจากอ่านไฟล์เสร็จแล้ว จะนำข้อมูลที่อ่านส่งต่อไปให้ผู้ใช้งานโดยคลาส DataWriter โดยทำการส่งข้อมูลผ่าน OutputStream ไปยังผู้ใช้งาน

```
package polly_qe.data.stream;

import ...

public class DataWriter {
    static final  int sizeOfInteger = 4;

    public static void writeData(OutputStream oStream, byte[] recvSizeRaw, int dataSize, byte[] dataRaw) {
        try {
            oStream.write(recvSizeRaw,  off: 0, sizeOfInteger);
            oStream.write(dataRaw,  off: 0, dataSize);
        }catch (IOException e){

        }

    }
}
```

ภาพที่ 3.18 ภาพคลาส DataWriter

3.1.3.4 การทำงานของ Feature Proxy



ภาพที่ 3.19 ภาพตัวอย่างการส่ง request โดยใช้ Feature Proxy

เมื่อ controller ได้ตรวจสอบแล้วว่าสถานะของ Feature Proxy เป็น True โดยการที่สถานะของ Feature Proxy จะเป็น True ได้นั้น มีอยู่ 2 กรณี กรณีแรกดูได้จาก flag ที่อยู่ในตัวแปร x-ts-requestid ใน Header ของ request ที่ผู้ใช้งานนั้นส่งมา โดยเลขประจำหลักของ Feature Proxy จะอยู่หลักที่อยู่ถัดจากหลักหน้าสุดของ flag ถ้าเป็น 1 จะทำให้สถานะของ Feature Proxy เป็น True และกรณีที่สอง คือ flag เป็นอย่างอื่นที่ไม่ได้อยู่ในรูปแบบ “pollyXXX” หรือไม่ได้ใส่ flag ในตัวแปร x-ts-requestid ใน Header ของ request ก็จะทำให้สถานะของ Feature Proxy เป็น True ได้เช่นกัน เนื่องจากได้ระบุสถานะเป็นค่าเริ่มต้น หากเกิดกรณีที่ไม่ปกติเช่นนี้หลังจากตรวจสอบสถานะของ Feature Proxy แล้วจะทำการสร้าง object ของคลาส PollyProxy ขึ้นมา และทำการ start ในคลาส PollyProxy จะมีการสร้าง socket object ขึ้นมาสำหรับการส่งข้อมูลระหว่าง QE และ Polly และสร้าง InputStream และ OutputStream object ขึ้นมาเพื่อรับ และส่งข้อมูลระหว่าง QE และ Polly ตามลำดับ

```
//Create socket for connect QE
Socket qeSock = new Socket(firstQeAttrib.getHostname(), firstQeAttrib.getPortInt());
InputStream qeIStream = qeSock.getInputStream();
OutputStream qeOStream = qeSock.getOutputStream();
```

ภาพที่ 3.20 ภาพการสร้าง socket object InputStream object และ OutputStream object

เมื่อทำการสร้าง OutputStream object แล้ว จะทำการส่ง TCL Commands ไปยัง QE ผ่านทาง OutputStream object

```
/************************ Send to QE ****************************/
DataWriter.writeData(qeOStream, clientTclRawSizeRaw, clientTclRawSize, clientTclRaw);
```

ภาพที่ 3.21 ภาพการส่ง TCL Commands ไปยัง QE ผ่านทาง OutputStream object

หลังจากที่ทำการส่ง TCL Commands ไปยัง QE แล้วจะทำการรับ response ที่ได้ผ่าน QeDataReceiver object โดยภายในคลาส QeDataReceiver จะทำการอ่านข้อมูลจาก response ที่อยู่ใน InputStream แล้วแปลงเป็น Byte Array

เมื่อทำการอ่านข้อมูลที่ได้รับจาก QE แล้ว จะทำการส่งข้อมูลที่ได้ไปยังผู้ใช้งานผ่านทาง OutputStream และทำการสร้าง QeData objet สำหรับเป็นข้อมูลสำหรับคลาส PollyRecord ต่อไป

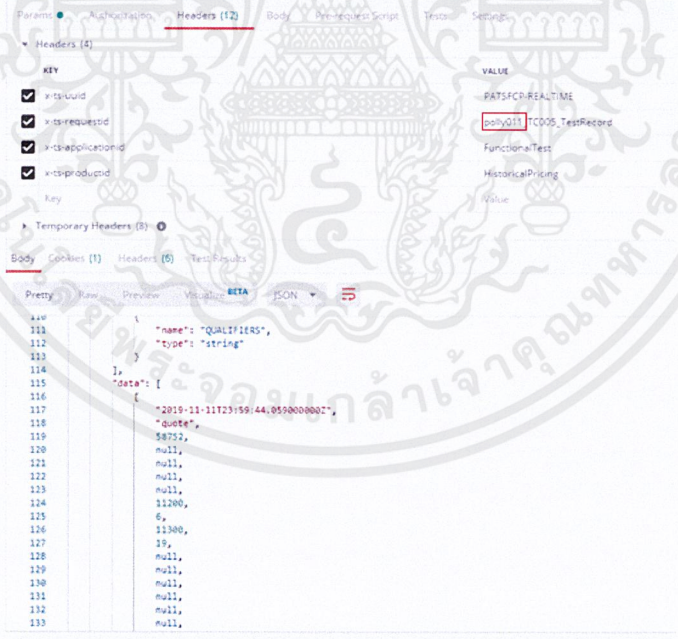
```java
public class QeDataReceiver {

    private byte[] recvSizeRaw = null;
    private int dataSize;
    private byte[] dataRaw = null;

    public QeDataReceiver(InputStream inputStream){
            recvSizeRaw = ByteBuffer.allocate(4).array();
            DataReader.readHeader(inputStream, recvSizeRaw);
            dataSize = ByteBuffer.wrap(recvSizeRaw).getInt();
            dataRaw = ByteBuffer.allocate(dataSize).array();
            DataReader.readData(inputStream, dataRaw, dataSize);
    }

    public byte[] getRecvSizeRaw() { return recvSizeRaw; }

    public int getDataSize() { return dataSize; }

    public byte[] getDataRaw() { return dataRaw; }
}
```

ภาพที่ 3.22 ภาพคลาส QeDataReceiver

```java
// Record Data
qeData = new QeData(dataRaw, recvSizeRaw);
```

ภาพที่ 3.23 ภาพการสร้าง QeData objet

3.1.3.5 การทำงานของ Feature Record



ภาพที่ 3.24 ภาพตัวอย่างการส่ง request โดยใช้ Feature Record และ Feature Proxy ด้วยกัน

ในคลาส PollyController จะมีการตรวจสอบก่อนว่าใน request นั้นมี flag ของ Feature Proxy และ Feature Record หรือไม่ เนื่องจากในการ Record นั้นจำเป็นต้องใช้

Feature Proxy เพื่อไปเอาข้อมูลมาจาก QE ก่อนจึงจะสามารถทำการบันทึกข้อมูลลงไปในไฟล์ได้ โดยจะตรวจสอบได้จาก flag ที่อยู่ในตัวแปร x-ts-requestid ใน Header ของ request ที่ผู้ใช้งานนั้นส่งมาโดยเลขประจำหลักของ Feature Record จะอยู่หลักที่อยู่หลังสุดของ flag ถ้าเป็น 1 จะทำให้สถานะของ Feature Record เป็น True

```
/************************ Record ****************************/
if (isRecord && isProxy) {
    PollyRecord record = new PollyRecord(requestParams, qeData);
    hasRecorded = record.startRecord();
} else if(isRecord){
    logger.log(Level.WARNING,  msg: "Record need \"Proxy\" flag");
}
```

ภาพที่ 3.25 ภาพการตรวจสอบเงื่อนไขก่อนทำการสร้าง PollyRecord เพื่อทำการบันทึกข้อมูลที่ได้จาก QE

ในคลาส PollyRecord นั้นในขั้นแรกจะทำการตรวจสอบก่อนว่ามี directory ที่จะทำการเก็บไฟล์ที่บันทึกข้อมูลที่ได้จาก QE แล้วหรือยัง ถ้ายังจะทำการสร้าง directory ขึ้นมาก่อน ถ้ามีแล้วก็จะใช้ directory นั้นในการบันทึกไฟล์เลยโดยการสร้าง directory นั้นจะนำตัวแปรต่าง ๆ ที่ได้มาจากการ Regular Expression ในคลาส RequestParams มาสร้างเป็น path ในการบันทึกไฟล์ โดยในคลาส PollyRecord จะมีการสร้าง PathCreator object ขึ้นมาเพื่อทำการสร้าง path ที่ได้มาจากคลาส RequestParams

```
public Path createDirectory() {
    path = PathCreator.getPath(reqParams);
    if (!Files.exists(path)) {
        try {
            Files.createDirectories(path);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    return path;
}
```

ภาพที่ 3.26 ภาพเมธอด createDirectory()

ภายในคลาส PathCreator นั้นจะทำการสร้าง path ตามตัวแปรต่าง ๆ ที่อยู่ใน TCL Commands ที่ได้รับมา โดยแต่ละ Request นั้นจะมีตัวแปรต่าง ๆ และค่าของตัวแปรที่แตกต่างกันออกไป จึงต้องออกแบบให้รองรับการสร้าง directory ตามตัวแปรแต่ละรูปแบบของ

request ที่ผู้ใช้งานนนั้นได้ส่งมาจึงทำให้เมื่อใช้งาน Feature Replay จะสามารถทำให้เรียกไฟล์ที่บันทึกข้อมูลจาก QE เพื่อส่งกลับไปยังผู้ใช้งานตามตัวแปรที่ระบุมาใน request ได้อย่างถูกต้อง

```java
public class PathCreator {

    public static Path getPath(RequestParams reqParams) {
        Path path = null;
        String testCase = reqParams.getTc();
        String ric = reqParams.getRic();
        String interfaceName = reqParams.getInterface();
        String refTable = reqParams.getRefTable();
        String policyName = reqParams.getPolicy();
        String startTime = reqParams.getStartTime();
        String endTime = reqParams.getEndTime();

        if (policyName == null && refTable != null) {
            path = Paths.get( first: "record/" + testCase + "/" + ric + "/" + interfaceName + "/" + refTable);
        } else if (refTable == null  && policyName != null) {
            path = Paths.get( first: "record/" + testCase + "/" + ric + "/" + interfaceName + "/" + policyName + "/" + startTime + "_" + endTime);
        } else if (ric != null){
            path = Paths.get( first: "record/" + testCase + "/" + ric + "/" + interfaceName);
        }

        return path;
    }
}
```

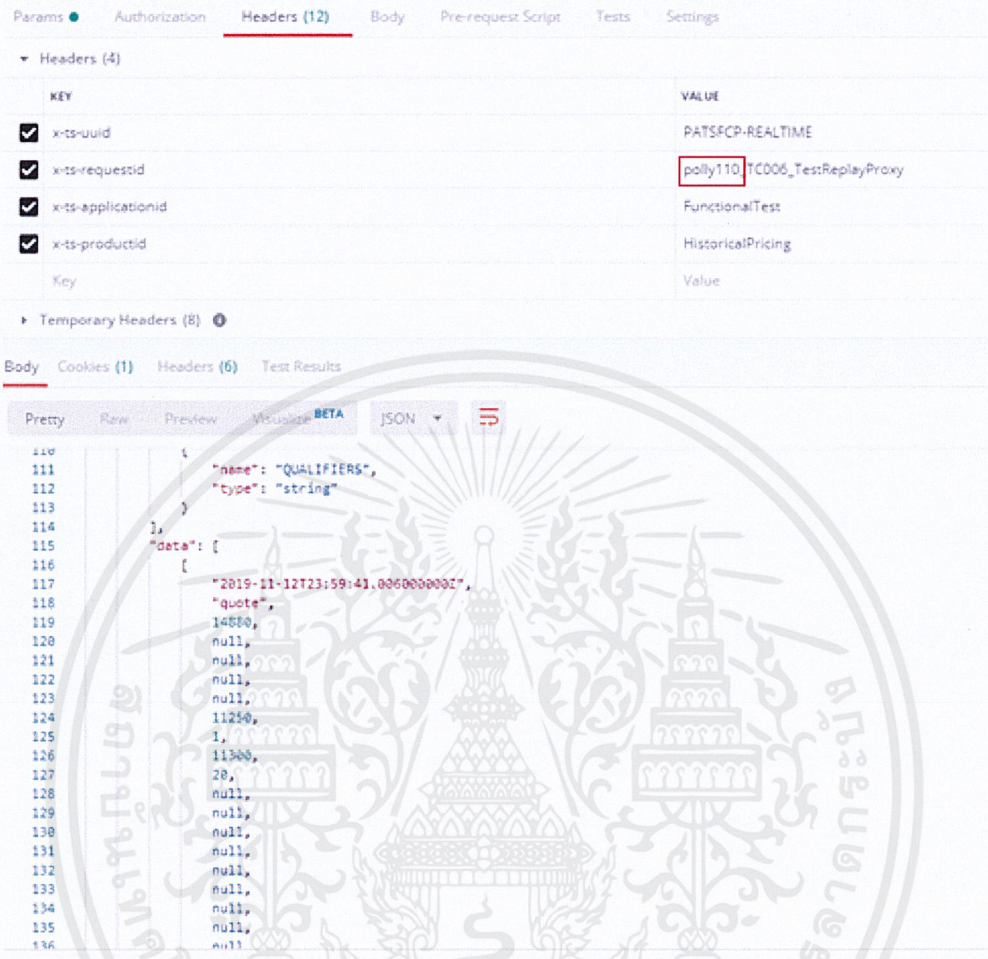
ภาพที่ 3.27 ภาพคลาส PathCreator

```
-TC003_testIntradayData
└──122900.KS
    ├──velocityGetChunkByDateUsingPolicy
    │   ├──TAQ_Cooked
    │   │   └──20190902000000_20190902235959
    │   └──TAS_Cooked
    │       └──20190902000000_20190902235959
    ├──velocityGetChunkUniqueKeysAndDataUsingPolicy
    │   ├──TAQ_Cooked
    │   │   └──20190902000000_20190902235959
    │   └──TAS_Cooked
    │       └──20190902000000_20190902235959
    ├──velocityGetFactList
    └──velocityGetMetaData
        └──[DefaultViewTable]
-TC004_TestPolly
└──122900.KS
    ├──velocityGetChunkByDateUsingPolicy
    │   ├──TAQ_Cooked
    │   │   └──20190729045600_20190729235959
    │   └──TAS_Cooked
    │       ├──20190729052900_20190729235900
    │       └──20190729052900_20190729235959
    ├──velocityGetChunkUniqueKeysAndDataUsingPolicy
    │   ├──TAQ_Cooked
    │   │   ├──20190729000000_20190729045559
    │   │   └──20190729000000_20190729235959
    │   └──TAS_Cooked
    │       ├──20190729000000_20190729052859
    │       └──20190729000000_20190729235959
    ├──velocityGetFactList
    └──velocityGetMetaData
        └──[DefaultViewTable]
```

ภาพที่ 3.28 ภาพตัวอย่าง path ที่ใช้บันทึกข้อมูลที่ได้จากการขอข้อมูลจาก QE

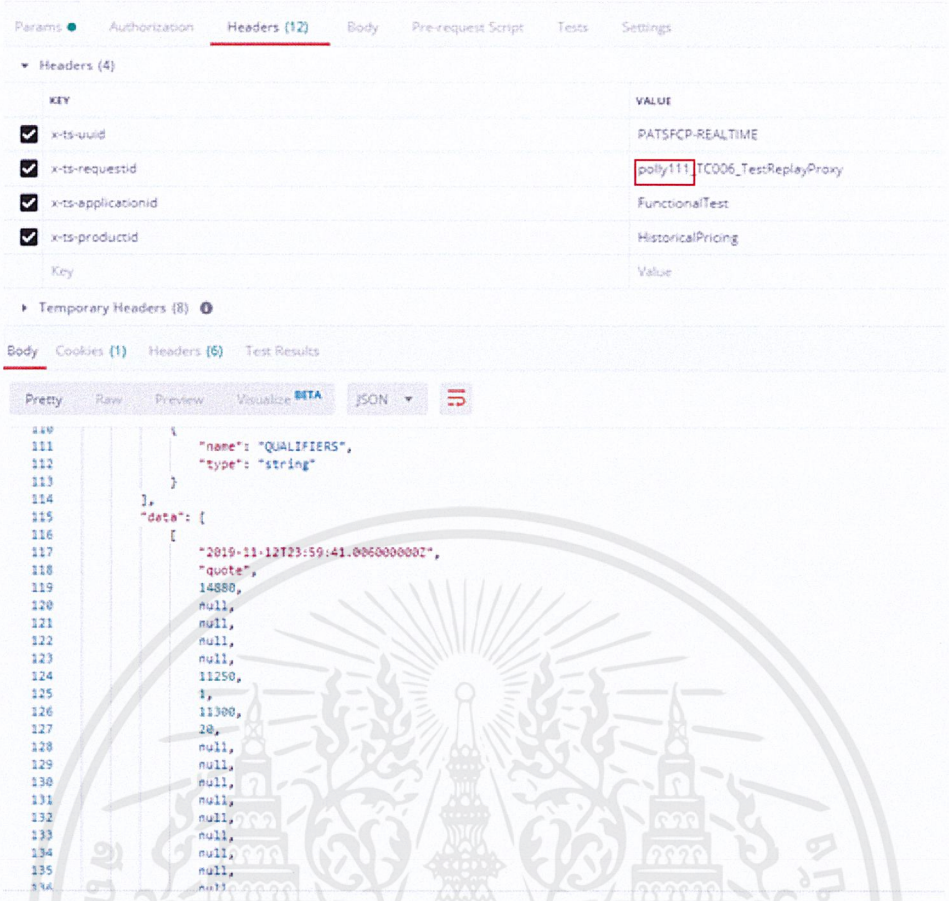
นอกจากจะใช้ Feature แต่ละอันตามที่กล่าวไปข้างต้นแล้ว ยังสามารถใช้งานทั้ง Feature ผสมกันได้ ยกตัวอย่าง เช่น ใช้ flag “polly110” จะเป็นการผสม Feature Replay และ Feature Proxy โดยถ้าหากมีไฟล์ที่บันทึกข้อมูลที่ได้จาก QE ตามตัวแปรต่าง ๆ ที่อยู่ใน request ที่ผู้ใช้งานได้ส่งมา จะทำการอ่านข้อมูลที่อยู่ในไฟล์นั้นแล้วส่งกลับไปให้ผู้ใช้งาน แต่ถ้าหากยังไม่มีไฟล์อยู่ภายในเครื่อง Feature Proxy จะทำงานโดยไป request ข้อมูลจาก QE แล้วนำข้อมูลนั้น

ส่งกลับไปให้ผู้ใช้งานแทน โดยการใช้ Feature Replay นั้นจะใช้เวลาในการทำงานน้อยกว่าการไปขอข้อมูลโดยตรงจาก QE อีกด้วย



ภาพที่ 3.29 ภาพตัวอย่างการส่ง request โดยใช้ Feature Replay และ Feature Proxy ด้วยกัน

นอกจากนี้ยังสามารถใช้ทั้ง 3 Feature ร่วมกันได้โดยใช้ flag “polly111” ถ้าหากมีไฟล์ที่บันทึกข้อมูลที่ได้จาก QE ตามตัวแปรต่าง ๆ ที่อยู่ใน request ที่ผู้ใช้งานได้ส่งมาจะทำการอ่านข้อมูลที่อยู่ในไฟล์นั้นแล้วส่งกลับไปให้ผู้ใช้งาน แต่ถ้าหากยังไม่มีไฟล์อยู่ภายในเครื่อง Feature Proxy จะทำงานโดยไป request ข้อมูลจาก QE แล้วนำข้อมูลนั้นส่งกลับไปให้ผู้ใช้งานแทน หลังจากส่งข้อมูลให้ผู้ใช้งานแล้วจะนำข้อมูลที่ได้มาบันทึกลงไปในไฟล์



ภาพที่ 3.30 ภาพตัวอย่างการส่ง request โดยใช้ทั้ง 3 feature ร่วมกัน

3.2 Single response testing

3.2.1 ทำความเข้าใจจุดประสงค์ผู้ใช้งานและระบบ

หลังจากที่ได้ทำการเก็บข้อมูล statistic ผู้ที่จะนำไปใช้งานลักษณะการใช้งานและตัวแปรต่าง ๆ ที่ใช้นำมาใช้ในการทดลอง จึงทำการเก็บ Requirement เพื่อทำความเข้าใจถึงความต้องการของผู้ใช้งานที่ต้องการจะทำการพิสูจน์ว่า component ใหม่ที่ได้พัฒนากันมาที่มีชื่อว่า TSCC ช่วยทำให้ระบบมี response time ที่ดีขึ้น โดย TSCC นั้นใช้ caching ในการทำให้ประสิทธิภาพดีขึ้น โดยการทดลองนั้นแบ่งออกเป็น 3 กรณี คือ

1. ทดลองส่ง request จาก Gateway ไปยัง Webservice
2. ทดลองส่ง request จาก Gateway ไปยัง TSCC(Cache miss)
3. ทดลองส่ง request จาก Gateway ไปยัง TSCC(Cache hit)

โดยได้ตั้งสมมติฐานไว้ว่าการส่ง request จาก Gateway ไปยัง TSCC แบบ Cache hit นั้นจะให้ response time ที่น้อยกว่าการส่ง request จาก Gateway ไปยัง Webservice

การทดสอบนั้นทำได้โดยการวัด response time ตาม test case test suite endpoint และจำนวนรอบที่จะเก็บผลที่ต่างกันออกไป โดยผู้ใช้งานนั้นต้องการจะให้เก็บเวลาของการตอบสนองของแต่ละ test case โดยจะมีการกำหนดชื่อ test case และตัวแปรต่าง ๆ ไว้ภายในไฟล์ csv จึงต้องออกแบบให้อ่านไฟล์ csv แล้วทำการดึงเอาตัวแปรต่าง ๆ มาเพื่อใช้ในการ request เพื่อการเก็บเวลาของการตอบสนองต่อไป

```
TestCase,RIC,Interval,Start,End,Adjustments,Fields,Count
EODHistoricalPricingEikonSingleRic_P1D_1Point_1,VOD.L,P1D,,,,,1
EODHistoricalPricingEikonSingleRic_P1D_1Point_2,EW.N,P1D,,,,,1
EODHistoricalPricingEikonSingleRic_P1D_1Point_3,TRFRBFVDc1,P1D,,,,,1
EODHistoricalPricingEikonSingleRic_P1D_1Point_4,BASFn.DE,P1D,,,,,1
EODHistoricalPricingEikonSingleRic_P1D_1Point_5,JUST.NS,P1D,,,,,1
EODHistoricalPricingEikonSingleRic_P1D_AllAdjustments_1,VOD.L,P1D,,,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1D_AllAdjustments_2,EW.N,P1D,,,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1D_AllAdjustments_3,TRFRBFVDc1,P1D,,,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1D_AllAdjustments_4,BASFn.DE,P1D,,,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1D_AllAdjustments_5,JUST.NS,P1D,,,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1Y_AllAdjustments_1,VOD.L,P1Y,,,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1Y_AllAdjustments_2,EW.N,P1Y,,,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1Y_AllAdjustments_3,TRFRBFVDc1,P1Y,,,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1Y_AllAdjustments_4,BASFn.DE,P1Y,,,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1Y_AllAdjustments_5,JUST.NS,P1Y,,,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1D_AllAdjustments_1YearData_1,VOD.L,P1D,now-360D,now,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1D_AllAdjustments_1YearData_2,EW.N,P1D,now-360D,now,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1D_AllAdjustments_1YearData_3,TRFRBFVDc1,P1D,now-360D,now,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1D_AllAdjustments_1YearData_4,BASFn.DE,P1D,now-360D,now,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1D_AllAdjustments_1YearData_5,JUST.NS,P1D,now-360D,now,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1Y_AllAdjustments_1YearData_1,VOD.L,P1Y,now-360D,now,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1Y_AllAdjustments_1YearData_2,EW.N,P1Y,now-360D,now,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1Y_AllAdjustments_1YearData_3,TRFRBFVDc1,P1Y,now-360D,now,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1Y_AllAdjustments_1YearData_4,BASFn.DE,P1Y,now-360D,now,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1Y_AllAdjustments_1YearData_5,JUST.NS,P1Y,now-360D,now,EXCHANGECORRECTION|MANUALCORRECTION|CCH|CRE|RPO|RTS,,
EODHistoricalPricingEikonSingleRic_P1D_UnAdjusted_1Point_1,VOD.L,P1D,,,UNADJUSTED,,1
EODHistoricalPricingEikonSingleRic_P1D_UnAdjusted_1Point_2,EW.N,P1D,,,UNADJUSTED,,1
EODHistoricalPricingEikonSingleRic_P1D_UnAdjusted_1Point_3,TRFRBFVDc1,P1D,,,UNADJUSTED,,1
EODHistoricalPricingEikonSingleRic_P1D_UnAdjusted_1Point_4,BASFn.DE,P1D,,,UNADJUSTED,,1
EODHistoricalPricingEikonSingleRic_P1D_UnAdjusted_1Point_5,JUST.NS,P1D,,,UNADJUSTED,,1
EODHistoricalPricingEikonSingleRic_P1W_UnAdjusted_1Point_1,VOD.L,P1W,,,UNADJUSTED,,1
EODHistoricalPricingEikonSingleRic_P1W_UnAdjusted_1Point_2,EW.N,P1W,,,UNADJUSTED,,1
```

ภาพที่ 3.31 ภาพตัวอย่างของไฟล์ csv ที่จะใช้เป็น test data

3.2.2 ออกแบบและเขียน script

3.2.2.1 ออกแบบไฟล์ config

ทำการออกแบบโครงสร้างไฟล์ config ที่ใช้สำหรับการตั้งค่าเพื่อทำการทดลองโดยสามารถตั้งค่าของตัวแปรต่าง ๆ เพื่อการทดสอบใน Environment ที่ต่างกันออกไปหรือชุดของข้อมูลที่ต้องการจะทำการทดลองโดยจะมีตัวแปรต่าง ๆ ดังนี้

- EndPoints

ตัวแปร EndPoints เก็บรวบรวมของข้อมูลต่าง ๆ ของ endpoints ต่าง ๆ ที่จะใช้ script ส่ง request ไป เช่น Hostname ของ endpoint ที่ต้องการจะให้ script ส่ง request ไปและเก็บค่า username password และค่าอื่น ๆ สำหรับการจะเข้าถึงเครื่องปลายทางได้

ภาพที่ 3.32 ภาพตัวแปร Endpoints เก็บค่าต่าง ๆ ที่จะใช้งานใน Endpoint นั้น ๆ

- Interfaces

ตัวแปร Interfaces เก็บค่าตัวแปรต่าง ๆ ของ Interface ที่เราต้องการจะทำการเก็บผลการจะเก็บเวลาของผลการตอบสนอง โดยภายในจะมีตัวแปร เช่น

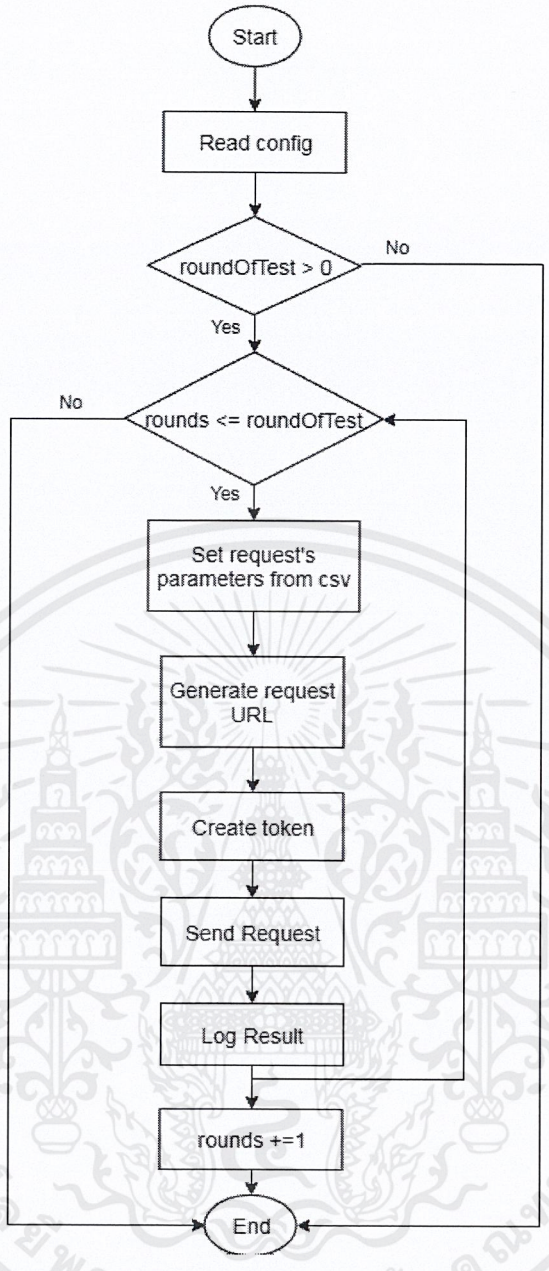
- RequestParams - มีเพื่อตรวจสอบ column ของไฟล์ test data ที่เป็น csv โดยแต่ละ interface จะมีไม่เหมือนกัน หากไม่เป็นไปตามที่กำหนดไว้จะมี error message แจ้งเตือน เพื่อให้ทำการแก้ไขไฟล์ csv
- TestSuites - เก็บ path ของไฟล์ test data endpoint ที่ต้องการจะส่ง request ไป และระบุจำนวนรอบที่ต้องการจะ run test โดยถ้าไม่ต้องการที่จะ run test suite ไหรก็ใส่ 0

ภาพที่ 3.33 ภาพตัวอย่างการใช้งานตัวแปร Interfaces

3.2.2.2 ออกแบบภาพรวมการทำงานของ script

หลักจากที่ได้ออกแบบโครงสร้างของไฟล์ config แล้ว ก็ทำการออกแบบโครงสร้างการทำงานของ Script โดยรวมดังนี้

- ดูข้อมูลต่าง ๆ เพื่อที่จะทำการ test ตามที่ได้กำหนดเอาไว้ในไฟล์ config
- อ่านไฟล์ test data ที่เป็น csv
- สร้าง request URL จากค่าของตัวแปรต่าง ๆ ในไฟล์ test data
- สร้าง token ในการใช้ส่ง request
- ส่ง request
- บันทึก response ลงไฟล์ csv



ภาพที่ 3.34 ภาพ Flow Chart แสดงการทำงานของ Script โดยรวม

3.2.2.3 ออกแบบ config.py

ทำหน้าที่เป็น getter เพื่อเป็นตัวกลางระหว่าง script ของเรากับไฟล์ Config.json เพื่อทำให้โคเดภายใน script นั้นสะอาด เรียบร้อย ใช้งานได้ง่าย อ่านง่ายและควบคุมการไหลของข้อมูลได้ดีกว่าการเรียกค่าต่าง ๆ ในไฟล์ Config.json โดยตรงภายใน script

```python
import json
import os
ROOT_DIR = os.path.dirname(os.path.abspath(__file__))

class Config:
    def __init__(self):
        self.fileConfig = os.path.join(ROOT_DIR, "log", "Config.json")
        self.config = self.loadFileConfig(self.fileConfig)

    def loadFileConfig(self,file):
        return json.loads(open(file).read())

    def getEndPointsData(self,EndPoint):
        return self.config['EndPoints'][EndPoint]

    def getEndPointsUsername(self,EndPoint):
        return self.config['EndPoints'][EndPoint]['User']

    def getEndPointsPassword(self, EndPoint):
        return self.config['EndPoints'][EndPoint]['Password']

    def getEndPointsHostname(self,EndPoint):
        return self.config['EndPoints'][EndPoint]['Hostname']

    def getEndPointsBC_Path(self,EndPoint):
        return self.config['EndPoints'][EndPoint]['Path_BC']

    def getInterfaces(self):
        return self.config['Interfaces']

    def getTestSuites(self, Interface):
        return self.config['Interfaces'][Interface]['TestSuites']

    def getTestSuiteEndPoint(self, Interface, TestSuite):
        return self.config['Interfaces'][Interface]['TestSuites'][TestSuite]['EndPoint']

    def getTestSuiteEndPointURL(self, Interface, TestSuite):
        endPoint = self.getTestSuiteEndPoint(Interface, TestSuite)
        return self.getEndPointsHostname(endPoint)
```

ภาพที่ 3.35 ภาพ config.py

3.2.2.4 ออกแบบ model.py

เนื่องจากในแต่ละ interface นั้น จะมีตัวแปรที่จะใช้ใน request URL ที่ต่างกันออกไป จึงทำการสร้าง model ของ interface A ขึ้นมาเพื่อเป็นต้นแบบของโครงสร้างเพื่อให้ interface อื่น ๆ ได้นำไป inheritance จากคลาส ของ model ที่เป็นต้นแบบ เพื่อลดการซ้ำซ้อนของการเขียนโค้ดในส่วนที่มีการใช้งานเหมือนกันได้

```python
class AModel:
    def __init__(self, testSuite):
        self.url = self.setUrl(endpoint=config.getTestSuiteEndPointURL('Summaries', testSuite))
        self.ric = ''
        self.queryString = {
            "interval":"",
            "start":"",
            "end":"",
            "adjustments":"",
            "fields":"",
            "count":""
        }
        self.path = "data/historical-pricing/v1/views/summaries"

    def setUrl(self,protocol='http',endpoint=None,port=8080):
        return "{protocol}://{endpoint}:{port}".format(protocol=protocol,endpoint=endpoint,port=port)

    def getURL(self):
        path = self.path
        keys = self.queryString.keys()
        for k in list(keys):
            if self.queryString[k]=="":
                self.queryString.pop(k)
        URL = "{url}/{path}/{ric}?{query}".format(url=self.url,path=path,ric=self.ric,query=urllib.parse.ur
        return URL

    def setRic(self, ric):
        self.ric = ric

    def setInterval(self, interval):
        self.queryString['interval'] = interval

    def setStartDate(self, startdate):
        self.queryString['start'] = startdate
```

ภาพที่ 3.36 ภาพคลาส Amodel

```python
def setDateByStartEnd(self,start,end):
    startDate = start
    endDate = end
    if start != "":
        if start != "now":
            start = start.split('-')[1]
            if start[-1] == "M":
                monthToDayCount = 0
                now  = datetime.datetime.now()
                for i in range(int(start[:-1]), 0, -1):
                    monthRange = calendar.monthrange(now.year-(i//now.month), now.month - (i%now.month))[1]
                    monthToDayCount += monthRange
                startDate = datetime.datetime.now() - datetime.timedelta(days=monthToDayCount)
            elif start[-1] == "W":
                startDate = datetime.datetime.now() - datetime.timedelta(weeks=int(start[:-1]))
            elif start[-1] == "D":
                startDate = datetime.datetime.now() - datetime.timedelta(days=int(start[:-1]))
            elif start[-1] == "h":
                startDate = datetime.datetime.now() - datetime.timedelta(hours=int(start[:-1]))
            elif start[-1] == "m":
                startDate = datetime.datetime.now() - datetime.timedelta(minutes=int(start[:-1]))
            elif start[-1] == 's':
                startDate = datetime.datetime.now() - datetime.timedelta(seconds=int(start[:-1]))
            else:
                startDate = datetime.datetime.now()
        elif start == "now":
            startDate = datetime.datetime.now()

        try:
            startDate = startDate.strftime('%Y-%m-%dT%H:%M:%SZ')
        except Exception as e:
            print(e)
```

ภาพที่ 3.37 ภาพ method ที่จะถูกนำไปใช้ใน interface อื่น ๆ โดยการ inheritance

```python
class BModel(AModel):
    def __init__(self, testSuite):
        self.url = self.setUrl(endpoint=config.getTestSuiteEndPointURL('Interday-Summaries', testSuite))
        self.ric = ''
        self.queryString = {
            "interval":"",
            "start":"",
            "end":"",
            "adjustments":"",
            "fields":"",
            "count":""
        }
        self.path = "data/historical-pricing/v1/views/interday-summaries"
```

ภาพที่ 3.38 ภาพคลาส BModel ที่ inheritance มาจากคลาส AModel

3.2.2.5 ออกแบบ webService.py

โดยแต่ละ interface จะมี method ของตนเองเพื่อทำการเก็บผล โดยในขั้นแรกจะอ่านไฟล์ config เพื่อทำการเอาค่าตัวแปรต่าง ๆ ของแต่ละ interface เช่น test suite ภายใน interface นั้น จำนวนรอบที่จะ run ของ test suite นั้น ชื่อของ column ในไฟล์ test data เพื่อจะนำมาตรวจสอบก่อนจะส่ง request และทำการอ่านไฟล์ test data ซึ่งเป็นไฟล์ csv ทีละบรรทัด เพื่อเอาข้อมูลของแต่ละ test case และค่าของตัวแปรต่าง ๆ ที่จะนำไปสร้างเป็น request URL เป็นต้น และทำการใช้ model ของ interface นั้น ๆ เพื่อการ set ค่าตัวแปรต่าง ๆ เพื่อทำการสร้าง request URL ขึ้นมา

```python
def B(config):
    interface = "Interday-Summaries"
    permission = 'PAISFCP-REALTIME'
    testSuites = config.getTestSuites(interface)
    for e in testSuites:
        testSuite = e
        rounds = 1
        roundOfTest = config.getRoundOfTest(interface, testSuite)
        if  roundOfTest > 0:
            testDataFile = config.getTestDataFile(interface, testSuite)
            testSuiteColumnName = getColumnName(testDataFile)
            configRequestParams = config.getRequestParams(interface)
            result = []
            print('{}: {}'.format(interface, testSuite))
            if testSuiteColumnName == configRequestParams:
                while(rounds <= roundOfTest):
                    print('rounds: {}'.format(rounds))
                    csvData = loadTestData(testDataFile)
                    for e in csvData:
                        data = e.split(',')
                        businessID = data[0]
                        ric = data[1]
                        interval = data[2]
                        start = data[3]
                        end = data[4]
                        adjustments = data[5].replace('|',',')
                        fields = data[6].replace('|',',')
                        count = data[7]
                        guid = str(uuid.uuid1()).upper()
                        interdaySummariesModel = InterdaySummariesModel(testSuite)
                        interdaySummariesModel.setRic(ric)
                        interdaySummariesModel.setInterval(interval)
                        interdaySummariesModel.setDateByStartEnd(start, end)
```

ภาพที่ 3.39 ภาพ method ของ interface B ทำการดึงค่าต่าง ๆ จาก config เพื่อสร้าง request URL

หลังจากที่ได้ทำการสร้าง request URL เรียบร้อยแล้ว ก็ทำการส่ง request นั้นไปยัง endpoint ที่ได้กำหนดไว้ใน config โดยการส่ง request นั้นจะสร้าง method ที่จะใช้ร่วมกันกับทุก method ที่จะทำการส่ง request โดยภายใน method นี้จะทำหน้ากำหนด header ของ request ส่ง request วัดเวลาของ response ที่ได้จาก request และเก็บค่าอื่น ๆ ที่ผู้ใช้งานต้องการ

```python
def sendRequest(interface, config, requestURL, businessID, guid, permission):
    headers = {'Accept-Encoding': 'gzip, deflate',
               'Content-Type': 'application/json',
               'E2EBusinessProcessID': businessID,
               'E2ETraceBC': 'True',
               'E2ETraceRequestID': guid,
               'x-ts-applicationID': 'SingleResponseTime',
               'x-ts-uuid': permission,
               'x-ts-requestID': guid,
               'x-ts-productID': 'SingleResponseTime'
               }

    start_time = time.time() * 1000
    response = requests.get(requestURL, headers=headers)
    diffTime = response.elapsed.total_seconds()
    end_time = time.time() * 1000  # millisec
    if str(response) != '<Response [404]>':
        content = response.json()
    else:
        content = '<Response [404]>'

    count = None
    Source = response.headers.get('Source')
    tsExpires = response.headers.get('Ts-Expires')

    try:
        for TS in content['timeseriesData']:
            count=len(TS['dataPoints'])
        print('Data point is ' + str(count))
        return diffTime, count, int(start_time), int(end_time), Source, tsExpires
    except:
        try:
            for TS in content:
                count=len(TS['data'])
            print('Historical Data point is ' + str(count))
            return diffTime, count, int(start_time), int(end_time), Source, tsExpires
        except KeyError as e:
            pprint.pprint(content)
            return 'KeyError',str(e),int(start_time), ''
        except TypeError as e:
            pprint.pprint(content)
            return 'TypeError', str(e), int(start_time), ''
        except Exception as e:
            return 'OtherException', str(e), int(start_time), ''

        pass
```

ภาพที่ 3.40 ภาพเมธอด sendRequest()

หลังจากที่ method ของ interface A ได้ทำการส่ง request ไปแล้วก็จะทำการเก็บผลที่ได้มาบันทึกลงในไฟล์ csv ต่อไป โดยมีการสร้าง path ตามชื่อ test suite ที่เป็น test data โดยไฟล์ csv นั้นจะนำไปใช้ในการตรวจสอบ แสดงผล และวิเคราะห์ข้อมูลต่าง ๆ ต่อไป

```
result_df = pandas.DataFrame(result, columns=columnName)
logPath = config.getTestDataFilePath(interface, testSuite)
logPath = logPath[:-4]
logPath = './log' + logPath [1:]
if not os.path.exists(logPath):
    os.makedirs(logPath)
result_df.to_csv(os.path.join(logPath, testSuite + '_' + datetime.datetime.now().strftime('%Y-%m-%dT%H_%M_%S') + '.csv'),index=False)
```

ภาพที่ 3.41 ภาพการเก็บผลที่ได้ลงไฟล์ csv

3.2.2.6 เขียน dockerfile

มีการใช้ docker เพื่อเป็นการจำลอง environment ขึ้นมา เพื่อที่เราจะสามารถนำ application นี้ไปใช้ในเครื่องที่มี OS และ environment ที่แตกต่างกันได้โดยไม่ต้องมีการ setup ใด ๆ เพื่อที่จะใช้งาน โดยใช้ dockerfile สำหรับการสร้าง docker image เพื่อเป็นพิมพ์เขียวของ docker container ซึ่งจะ run application ของเราภายในนั้น โดยภายใน dockerfile นั้นจะมีคำสั่งสำหรับการทำ command ต่าง ๆ และการ copy ไฟล์ที่ต้องการ

```
FROM python:3.7.4-slim
WORKDIR /usr/src/app

COPY requirements.txt .
RUN pip install --upgrade pip
RUN pip install --no-cache-dir -r requirements.txt

COPY webService.py .
COPY config.py .
COPY /log/Config.json .
COPY model ./model
COPY InterdaySummaries.csv .
COPY InterdaySummaries_GW_to_interday.csv .
COPY InterdaySummaries_GW_to_TSCC.csv .
COPY HttpRestUtils.py .

CMD [ "python", "webService.py" ]
```

ภาพที่ 3.42 ภาพการเขียน dockerfile

3.2.2.7 เขียน shell script

เมื่อเราต้องการจะ run application ก็จำเป็นต้องสร้าง image และ container ขึ้นมา โดยจะต้องพิมพ์คำสั่งต่าง ๆ ลงไป และต้องมีการ mount volume ระหว่าง

container กับเครื่อง local เพื่อที่จะทำการ share ไฟล์ระหว่างกันได้จึงได้ทำการเขียน script ขึ้นมาเพื่อความสะดวกของผู้ใช้งาน

```
docker stop tscc_single_response_time_test
docker rm -f tscc_single_response_time_test
docker rmi tscc_single_response_time_test
docker build -t tscc_single_response_time_test .
```

ภาพที่ 3.43 ภาพการเขียน script สำหรับ build Docker Image

```
docker stop tscc_single_response_time_test
docker rm -f tscc_single_response_time_test
docker run -it -v /$(pwd)/log:/usr/src/app/log --name tscc_single_response_time_test tscc_single_response_time_test
```

ภาพที่ 3.44 ภาพการเขียน script สำหรับ start Docker Container

3.2.2.8 แสดงผลและวิเคราะห์ผล

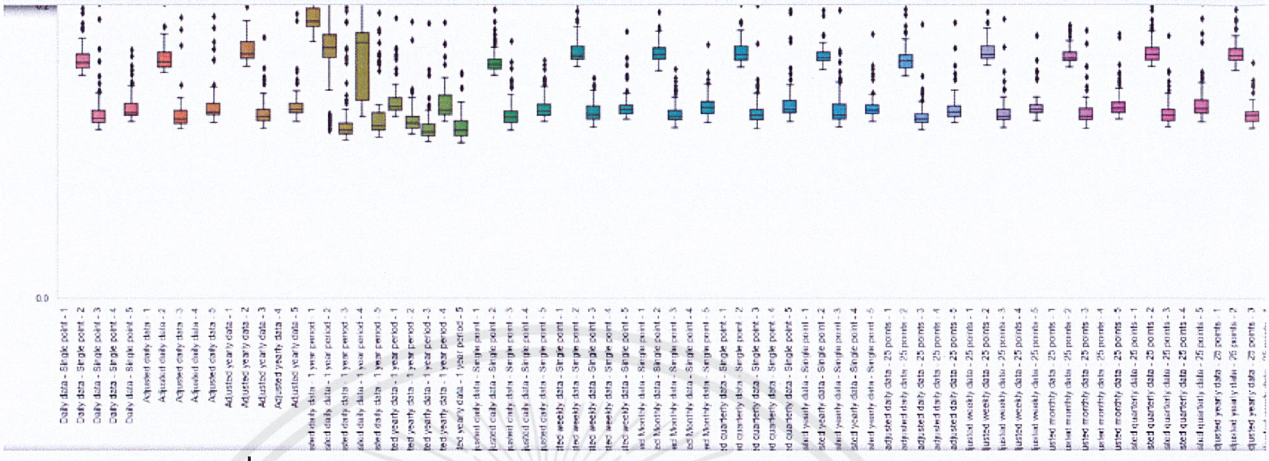
เมื่อได้ผลต่าง ๆ มาจากการส่ง request ในแต่ละ test case แล้ว ก็นำมาทำการ visualization เพื่อทำการวิเคราะห์ข้อมูลต่าง ๆ โดยใช้ Box graph Distribution Graph และ Bar chart ในการแสดงผล โดยนำเอาผลที่ได้จากการ request ที่อยู่ในรูปของไฟล์ csv มาแปลงเป็น DataFrame แล้วนำไปใช้แสดงผลค่าต่าง ๆ โดยในบาง test scenario นั้น ต้องแยกออกมาเปรียบเทียบกันในแต่ละ interface ออกเป็นกลุ่ม ๆ จึงต้องมีการสร้าง DataFrame ใหม่จากอันเดิมแยกออกไปตามกรณีอีกด้วย

Read CSV

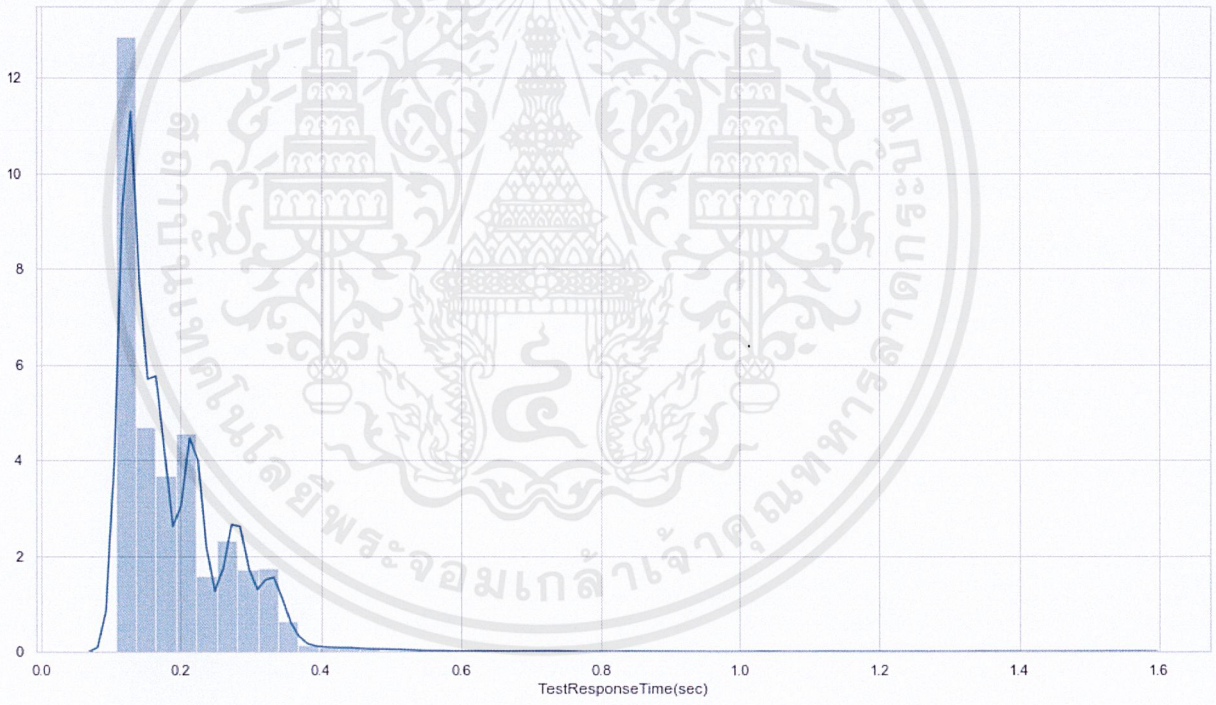
```
In [2]: sp1_df = pd.read_csv("GW-Interday_2019-11-29T10_46_45.csv", delimiter=',', header=0)
        sp2_df = pd.read_csv("GW-TSCC_miss.csv", delimiter=',', header=0)
        sp3_df = pd.read_csv("GW-TSCC_hit.csv", delimiter=',', header=0)
```

ภาพที่ 3.45 ภาพแสดงการอ่านไฟล์ csv แล้วแปลงเป็น DataFrame

```
plt.figure(figsize = (25, 50))
sns.set(style="whitegrid")
bp = sns.boxplot(data=sp1_df[['TestResponseTime(sec)', 'TestCaseName']], x='TestCaseName', y='TestResponseTime(sec)')
bp.set(ylim=(0, None))
bp.set_title('Gateway -> ETS Interday',fontsize=30)
bp.set_xlabel('TestCaseName', fontsize=20)
bp.set_ylabel('TestResponseTime(sec)', fontsize=20)
plt.xticks(rotation=90)
```



ภาพที่ 3.46 ภาพการใช้ข้อมูลใน DataFrame มา plot Box graph



ภาพที่ 3.47 ภาพการใช้ข้อมูลใน DataFrame มา plot Distribution graph

Median

```
sp1_med = sp1_df.reset_index().groupby(['TestCaseName'])['TestResponseTime(sec)'].median()
new_sp1_df = pd.DataFrame(sp1_med).reset_index()
new_sp1_df
```

	TestCaseName	TestResponseTime(sec)
0	Adjusted daily data - 1	0.316156
1	Adjusted daily data - 1 year period - 1	0.188749
2	Adjusted daily data - 1 year period - 2	0.170832
3	Adjusted daily data - 1 year period - 3	0.114638
4	Adjusted daily data - 1 year period - 4	0.174430
...	...	...
70	Unadjusted yearly data - Single point - 1	0.283030
71	Unadjusted yearly data - Single point - 2	0.163969
72	Unadjusted yearly data - Single point - 3	0.124539
73	Unadjusted yearly data - Single point - 4	0.221526
74	Unadjusted yearly data - Single point - 5	0.128014

75 rows × 2 columns

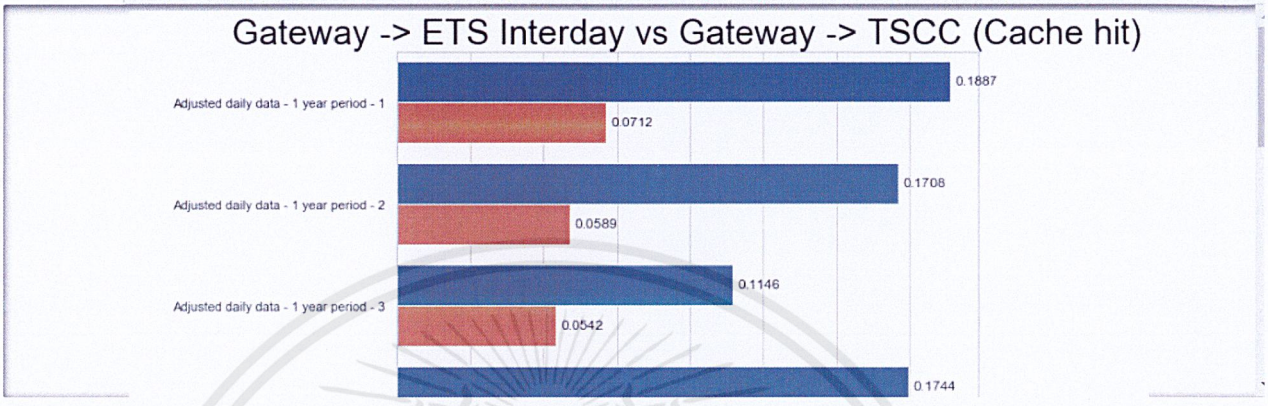
ภาพที่ 3.48 ภาพการนำ DataFrame มาหาค่า median ของแต่ละ test case แล้วสร้างเป็น DataFrame ใหม่

```
testcases_2 = ['Adjusted daily data - 1 year period - 1', 'Adjusted daily data - 1 year period - 2', 'Adjusted daily data
new_sp1_df_2 = new_sp1_df.loc[new_sp1_df['TestCaseName'].isin(testcases_2)]
new_sp1_df_2
```

	TestCaseName	TestResponseTime(sec)
1	Adjusted daily data - 1 year period - 1	0.188749
2	Adjusted daily data - 1 year period - 2	0.170832
3	Adjusted daily data - 1 year period - 3	0.114638
4	Adjusted daily data - 1 year period - 4	0.174430
5	Adjusted daily data - 1 year period - 5	0.117418
11	Adjusted yearly data - 1 year period - 1	0.130880
12	Adjusted yearly data - 1 year period - 2	0.119228
13	Adjusted yearly data - 1 year period - 3	0.113046
14	Adjusted yearly data - 1 year period - 4	0.128504
15	Adjusted yearly data - 1 year period - 5	0.114541

ภาพที่ 3.49 ภาพการนำ DataFrame มาเลือกบาง test case แล้วสร้างเป็น DataFrame ใหม่

```
In [26]: sns.set(style="whitegrid")
         plt.figure(figsize = (9, 16))
         df = pd.concat([new_sp1_df_2, new_sp3_df_2], sort=False)
         l1 = ['Gateway -> ETS Interday', 'Gateway -> TSCC (Cache hit)']*10
         df['label'] = sorted(l1)
         ax = sns.barplot(data = df, x="TestResponseTime(sec)", y='TestCaseName', hue="label")
         ax.set_title('Gateway -> ETS Interday vs Gateway -> TSCC (Cache hit)',fontsize=30)
         ax.set_ylabel('TestCaseName', fontsize=20)
         ax.set_xlabel('TestResponseTime(sec)', fontsize=20)

         for p in ax.patches:
             width = p.get_width()
             plt.text(p.get_width()+0.009, p.get_y()+0.5*p.get_height(),'{:1.4f}'.format(width),ha='center', va='center')
```



ภาพที่ 3.50 ภาพการนำ DataFrame ของ 2 interface มาเปรียบเทียบกันโดยใช้ Bar chart

บทที่ 4

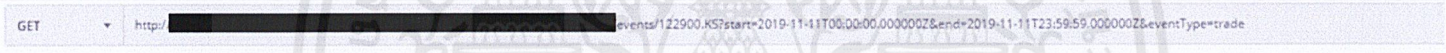
ผลการดำเนินงาน

ในบทนี้จะแบ่งผลการดำเนินงานออกเป็น 2 ส่วน โดยส่วนแรกจะเป็นของโพรเจกต์ที่ชื่อว่า Polly และในส่วนที่สองจะเป็นของโพรเจกต์ที่ชื่อว่า Single Response Testing

4.1 Polly

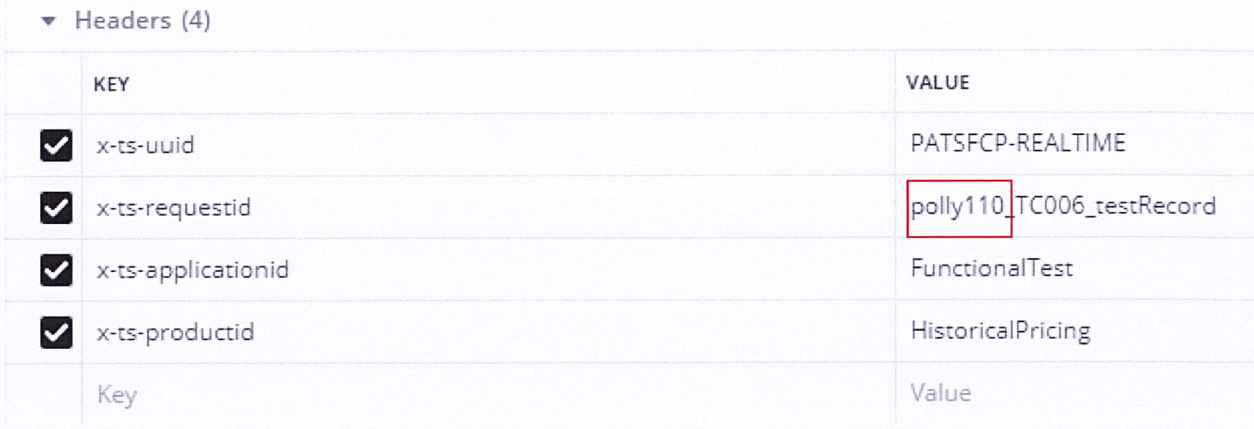
4.1.1 Replay Feature

ในส่วนการทำงานของ Replay Feature นั้นมีการทำงานได้ตามที่ได้คาดหวังเอาไว้ โดยเมื่อทำการ request โดยทดลองทำการขอข้อมูลที่ได้มีการบันทึกเอาไว้แล้ว ก็สามารถตอบกลับด้วยข้อมูลที่ถูกต้อง และหากยังไม่มีการบันทึกข้อมูลดังกล่าวไว้ภายในเครื่อง เมื่อใช้ Replay Feature ร่วมกับ Proxy Feature แล้วก็จะสามารถตอบกลับด้วยข้อมูลที่ถูกต้องตามที่ได้คาดเอาไว้ โดยได้ผลลัพธ์ตามภาพต่อไปนี้



ภาพที่ 4.1 ภาพตัวอย่าง URL ที่ใช้ในการ request ที่ใช้ใน Replay Feature

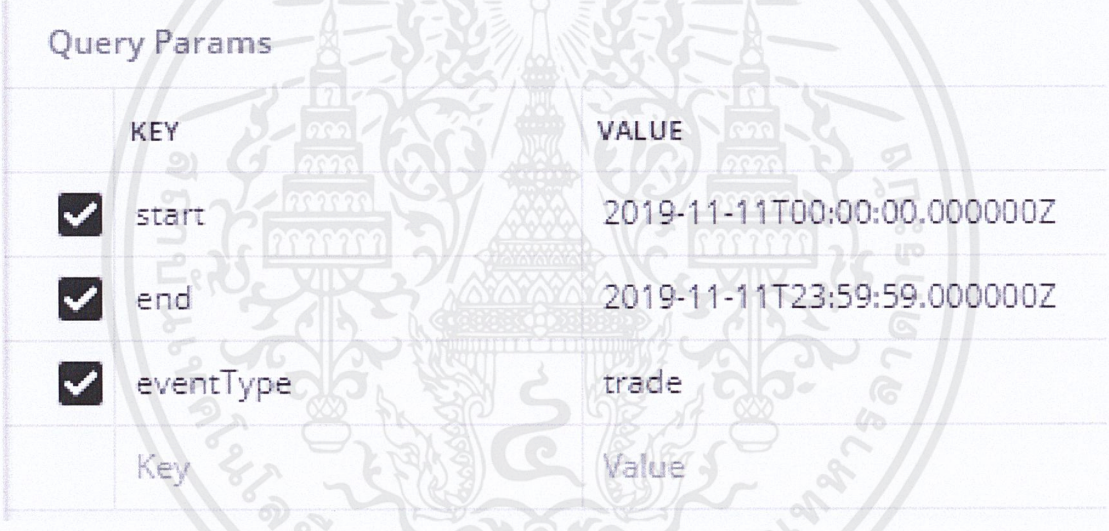
ภาพที่ 4.2 ภาพแสดง flag ของการใช้ Replay Feature อย่างเดียว



▼ Headers (4)

	KEY	VALUE
☑	x-ts-uuid	PATSFCP-REALTIME
☑	x-ts-requestid	polly110_TC006_testRecord
☑	x-ts-applicationid	FunctionalTest
☑	x-ts-productid	HistoricalPricing
	Key	Value

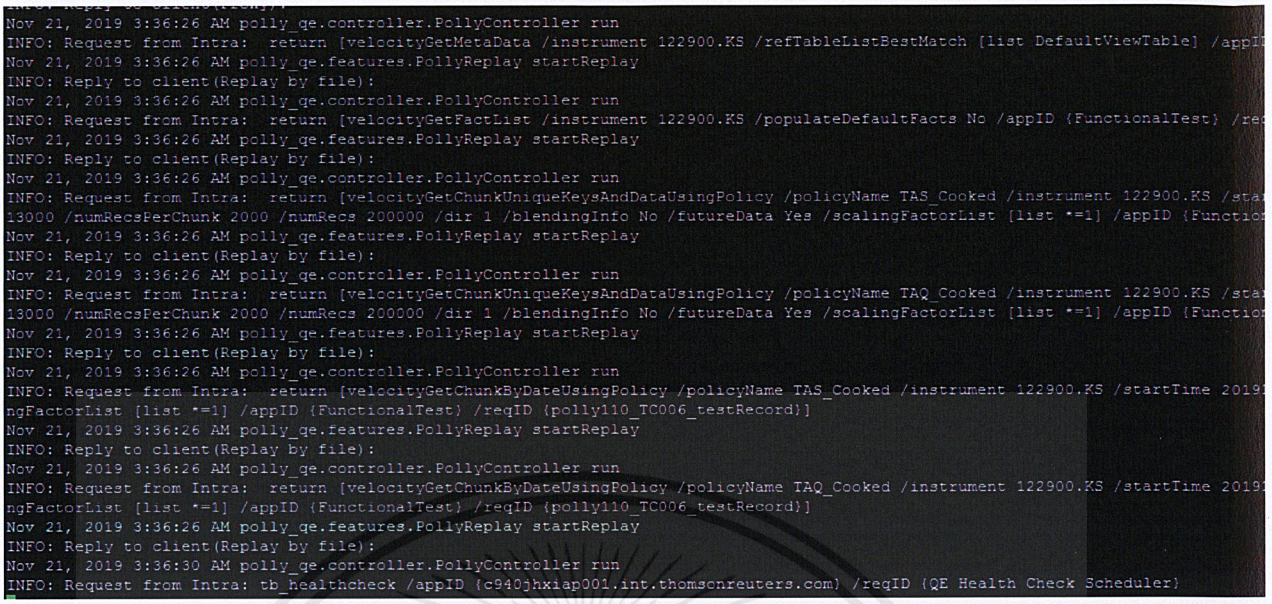
ภาพที่ 4.3 ภาพแสดง flag ของการใช้ Replay Feature ร่วมกับ Proxy Feature



Query Params

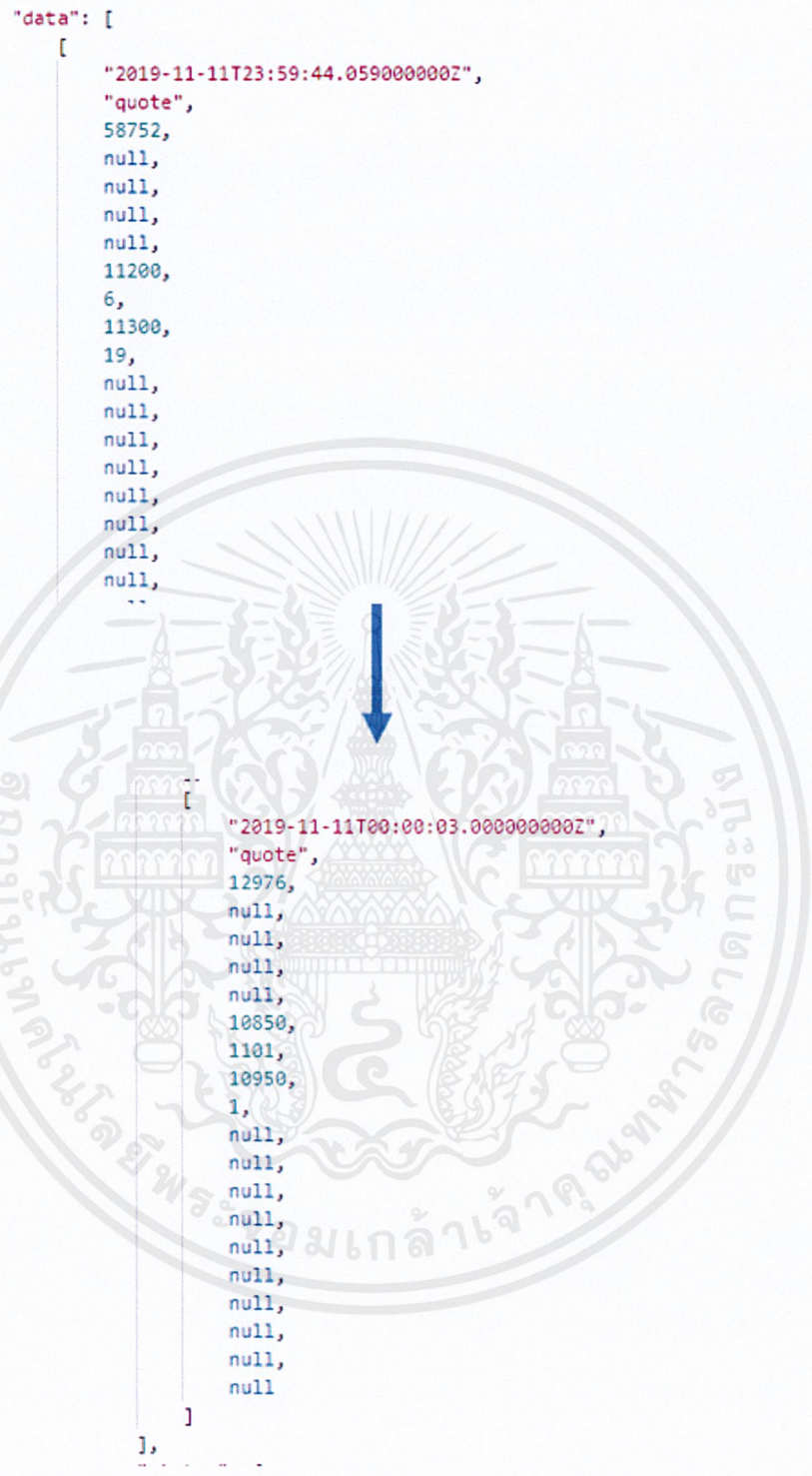
	KEY	VALUE
☑	start	2019-11-11T00:00:00.000000Z
☑	end	2019-11-11T23:59:59.000000Z
☑	eventType	trade
	Key	Value

ภาพที่ 4.4 ภาพแสดงค่าของตัวแปร start และ end ใน request URL ที่ใช้ใน Replay Feature



ภาพที่ 4.5 ภาพแสดงการ logging ของ Polly ที่ใช้ Replay Feature

ภาพที่ 4.6 ภาพแสดงค่าต่าง ๆ ที่มีความสัมพันธ์กับตัวแปรใน request URL ที่ใช้ใน Replay Feature

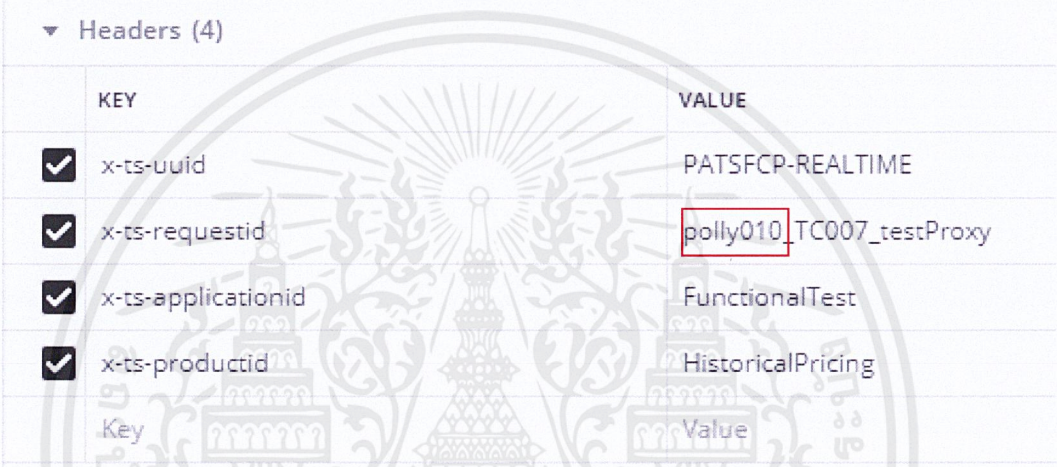


ภาพที่ 4.7 ภาพแสดงค่าของข้อมูลที่ได้รับมีเวลาสิ้นสุดและเริ่มต้นตรงกับค่าของตัวแปร start และ end ใน request URL ที่ใช้ใน Replay Feature

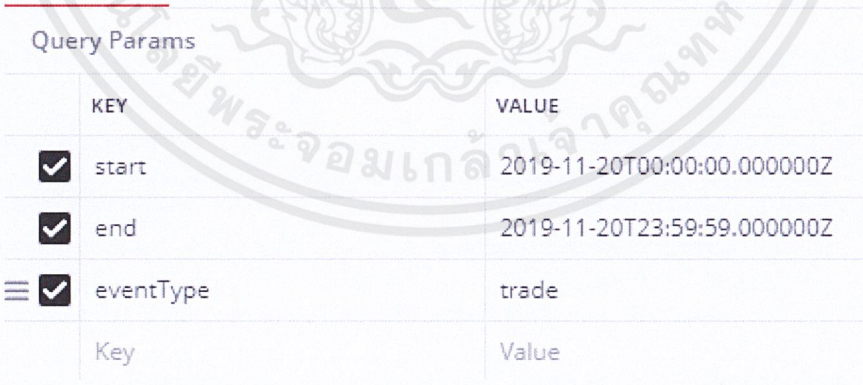
4.1.2 Proxy Feature

ในส่วนการทำงานของ Proxy Feature นั้นมีการทำงานได้ตามที่ได้คาดหวังเอาไว้ โดยเมื่อทำการ request โดยทดลองทำการขอข้อมูลในหลาย ๆ กรณีที่เป็นไปได้ ก็สามารถตอบกลับด้วยข้อมูลที่ถูกต้อง ตามที่ได้คาดเอาไว้ โดยได้ผลลัพธ์ตามภาพต่อไปนี้

ภาพที่ 4.8 ภาพตัวอย่าง URL ที่ใช้ในการ request ที่ใช้ใน Proxy Feature



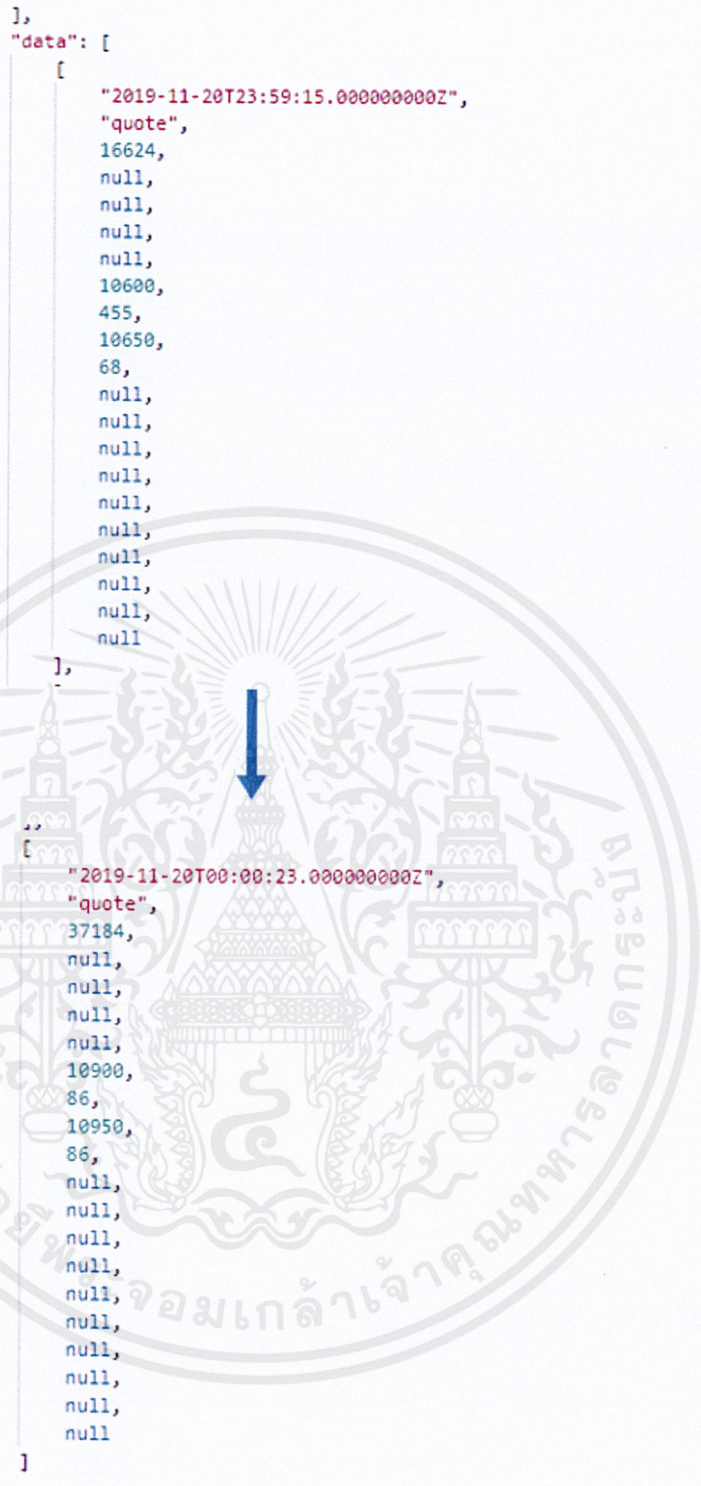
ภาพที่ 4.9 ภาพแสดง flag ของการใช้ Proxy Feature



ภาพที่ 4.10 ภาพแสดงค่าของตัวแปร start และ end ใน request URL ที่ใช้ใน Proxy Feature

ภาพที่ 4.11 ภาพแสดงการ logging ของ Polly ที่ใช้ Proxy Feature

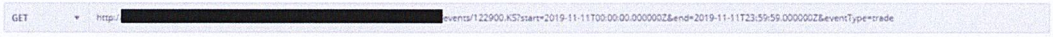
ภาพที่ 4.12 ภาพแสดงค่าต่าง ๆ ที่มีความสัมพันธ์กับตัวแปรใน request URL ที่ใช้ใน Proxy Feature



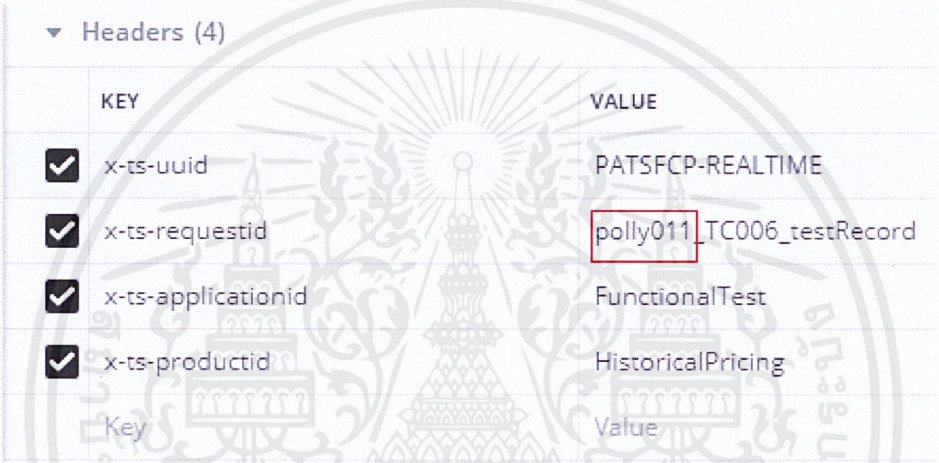
ภาพที่ 4.13 ภาพแสดงค่าของข้อมูลที่ได้รับมีเวลาสิ้นสุดและเริ่มต้นตรงกับค่าของตัวแปร start และ end ใน request URL ที่ใช้ใน Proxy Feature

4.1.3 Record Feature

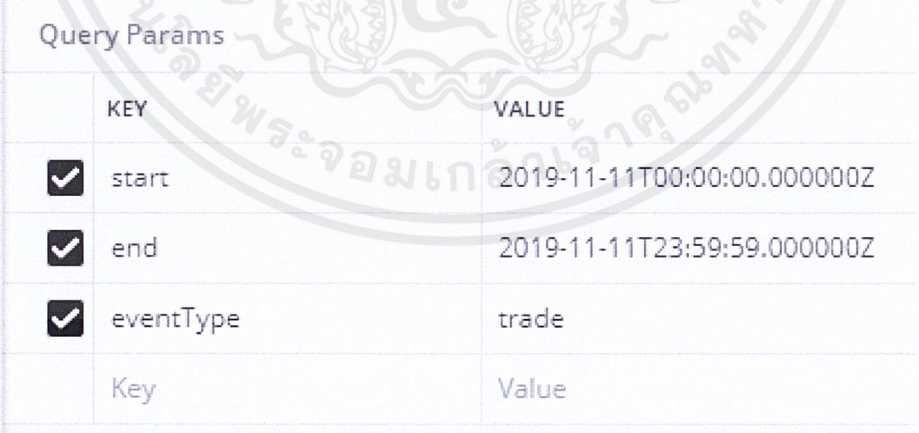
ในส่วนการทำงานของ Record Feature นั้นมีการทำงานได้ตามที่ได้คาดหวังเอาไว้ โดยเมื่อทำการ request โดยใช้ Record Feature ร่วมกับ Proxy Feature แล้วก็จะทำการบันทึก data ที่ได้จาก QE ลงในไฟล์ txt ตาม path ที่สร้างขึ้นจากตัวแปรต่าง ๆ ใน request ได้อย่างถูกต้อง



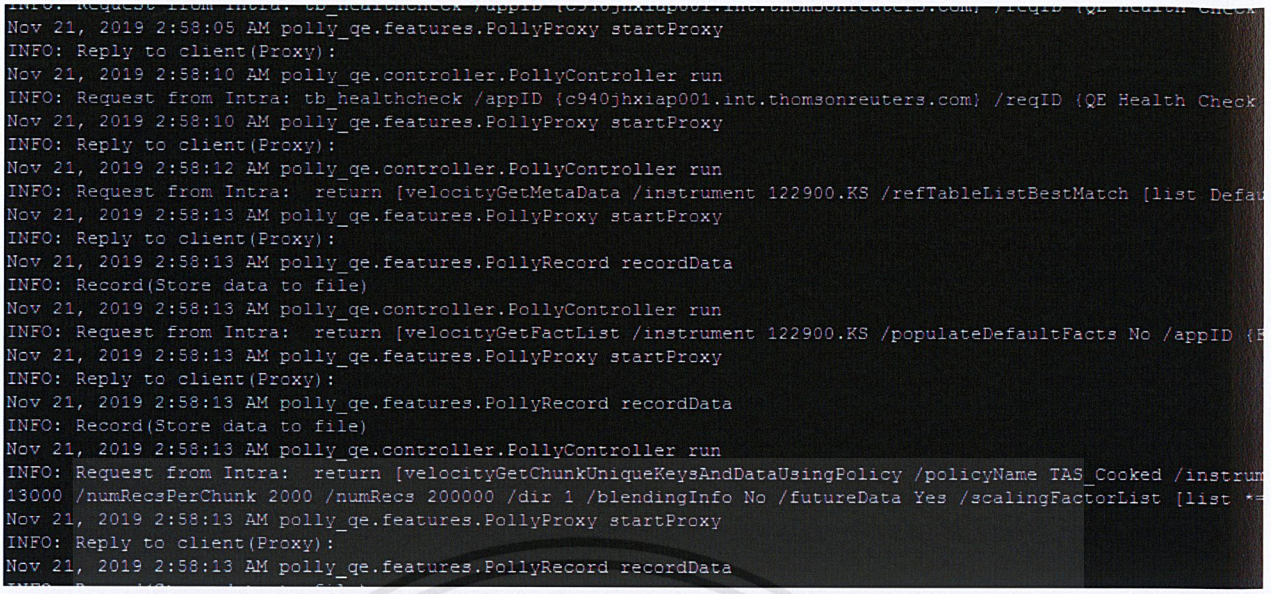
ภาพที่ 4.14 ภาพตัวอย่าง URL ที่ใช้ในการ request ที่ใช้ใน Record Feature



ภาพที่ 4.15 ภาพแสดง flag ของการใช้ Record Feature



ภาพที่ 4.16 ภาพแสดงค่าของตัวแปร start และ end ใน request URLที่ใช้ใน Record Feature



ภาพที่ 4.17 ภาพแสดงการ logging ของ Polly ที่ใช้ Record Feature

ภาพที่ 4.18 ภาพแสดงโฟลเดอร์ที่เก็บไฟล์ที่บันทึกข้อมูลที่ได้รับจาก QE

```
{<key>} {00000000000000000000000000000000000000000} {</key>} {<fieldName>} {VhBaseTime
RuleSetVersion RuleID RuleVersionID RuleClauseNo RecordStatus EditType SOURCE_DATETIME
{<resultSet>} {20191111000003.000 144119586122390784 20191111000003.000 20191111000003
10950 1 nil nil nil nil nil nil nil nil} {20191111000006.396 144119586122391041 201911
20191111000006.000 10950 2 11000 1 nil nil nil nil nil nil nil nil} {20191111000009.61
13633 33.0 23 nil nil 20191111000009.000 10850 1110 10950 1 nil nil nil nil nil nil ni
20191111000012.000 0 13024 0 6475.0 13633 33.0 23 nil nil 20191111000012.000 10850 110
144119586122393088 20191111000016.916 20191111000016.000 0 13120 0 6475.0 13633 33.0 2
20191111000017.356 0 49 0 0 144119586122393344 20191111000017.356 20191111000017.000 0
144119586122393600 20191111000017.000 20191111000017.978 0 49 0 0 144119586122393600 2
nil} {20191111000021.000 144119586122394624 20191111000021.000 20191111000021.094 0 49
nil nil nil nil nil nil nil} {20191111000033.000 144119586122395136 20191111000033.000
20191111000033.000 10900 2 10950 595 nil nil nil nil nil nil nil nil} {20191111000035.
6475.0 13633 33.0 23 nil nil 20191111000035.000 10850 1476 10900 314 nil nil nil nil n
20191111000035.000 0 13296 0 6475.0 13633 33.0 23 nil nil 20191111000035.000 10850 147
144119586122396160 20191111000049.107 20191111000049.000 0 13312 0 6475.0 13633 33.0 2
20191111000050.285 0 49 0 0 144119586122396416 20191111000050.285 20191111000050.000 0
144119586122396928 20191111000105.000 20191111000105.245 0 49 0 0 144119586122396928 2
nil} {20191111000106.000 144119586122397184 20191111000106.000 20191111000106.866 0 49
nil nil nil nil nil nil nil} {20191111000110.000 144119586122398208 20191111000110.000
20191111000110.000 10850 2 10950 596 nil nil nil nil nil nil nil nil} {20191111000111.
6475.0 13633 33.0 23 nil nil 20191111000111.000 10800 5361 10850 16 nil nil nil nil ni
20191111000111.000 0 13520 0 6475.0 13633 33.0 23 nil nil 20191111000111.000 10800 536
144119586122400256 20191111000112.844 20191111000112.000 0 13568 0 6475.0 13633 33.0 2
20191111000113.995 0 49 0 0 144119586122400512 20191111000113.995 20191111000114.000 0
144119586122400768 20191111000115.000 20191111000115.367 0 49 0 0 144119586122400768 2
nil nil} {20191111000118.000 144119586122401792 20191111000118.000 20191111000118.546
427 nil nil nil nil nil nil nil nil} {20191111000119.000 144119586122402304 2019111100
20191111000119.000 10800 7464 10950 411 nil nil nil nil nil nil nil nil} {201911110001
{<chunkSet>} {20191111062700 20191111235900 4C1751F8602F2566000000000FFFFFFFF00000000 48}
AFEB648CDE05D81300000000FFFFFFFF00000000 2007} {</chunkSet>} {<resultSet>} {</resultSet>}
```

```
<resultSet>
<ruleset_version>6491.0</ruleset_version>
<ts_reference_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="TSfCP_reference_
<ts_reference_mapping_data_table><ts_reference_identity TableName="DefaultViewTable" TableVersion="1.0"/><ts_reference_
            <value>156</value>
        </part><part key_pos="3">
            <value>38</value>
        </part><part key_pos="5">
            <value>113</value>
        </part></context_key><ts_reference_mapping_datum><pair>
            <from>TRDPRC_1</from>
            <to/>
        </pair></ts_reference_mapping_datum></ts_reference_mapping_data_row></ts_reference_mapping_data_table>
</ts_reference_file>
</resultSet>
```

ภาพที่ 4.19 ภาพตัวอย่างข้อมูลที่อยู่ในไฟล์ที่บันทึกข้อมูลที่ได้รับจาก QE

4.2 Single Response Testing

4.2.1 ผลที่ได้จากการส่ง request

ผลที่ได้จากการส่ง request จะถูกบันทึกในไฟล์ csv เพื่อนำไป visualize วิเคราะห์ผลข้อมูลต่าง ๆ และนำไปตรวจดูความผิดปกติของ response ที่ได้

ภาพที่ 4.20 ภาพผลที่ได้จากการส่ง request จะถูกบันทึกในไฟล์ csv

4.2.2 วิเคราะห์ผล

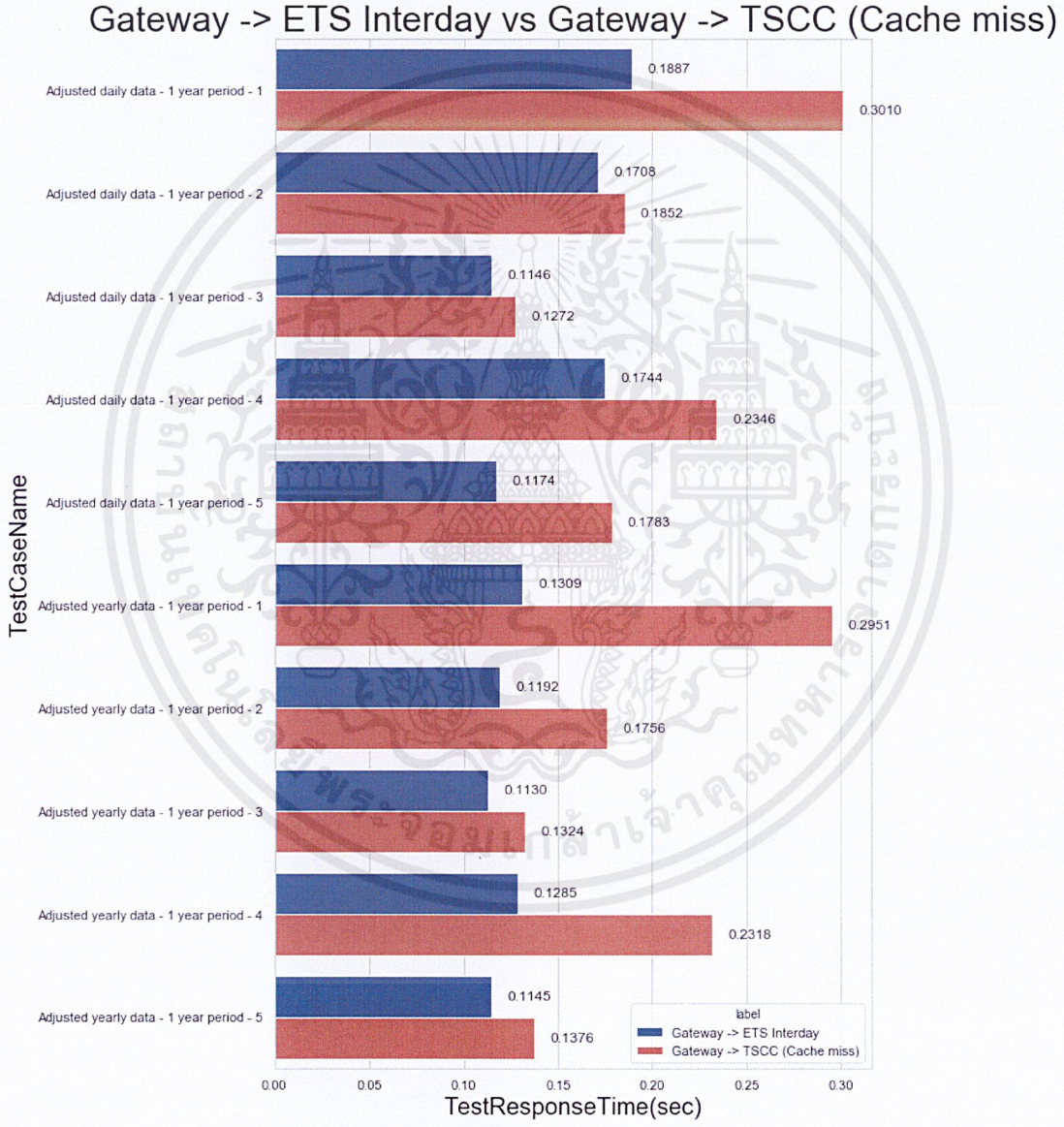
หลังจากที่ได้เขียน script ขึ้นมาเพื่อทำการส่ง request และเก็บค่า response time ที่ได้มาเพื่อการวิเคราะห์ข้อมูลต่าง ๆ พบว่าในกรณีแรก คือ ทดลองส่ง request จาก Gateway ไปยัง Webservice จากการ plot box graph เพื่อที่จะแสดงการกระจายตัวของข้อมูล พบว่า response time นั้นจะเกาะกลุ่มกันอยู่ที่ 0.1-0.2 วินาที และมี outlier ซึ่งแสดงถึงข้อมูลที่มีค่าแยกออกจากกลุ่มหรือผิดแผกแตกต่างไปจากข้อมูลค่าอื่น ๆ อยู่ประปราย กรณีที่สอง คือ ทดลองส่ง request จาก Gateway ไปยัง TSCC(Cache miss) จากการ plot box graph เพื่อที่จะแสดงการกระจายตัวของข้อมูล พบว่า response time นั้น จะเกาะกลุ่มกันอยู่ที่ 0.1-0.3 วินาที และมี outlier ซึ่งแสดงถึงข้อมูลที่มีค่าแยกออกจากกลุ่มหรือผิดแผกแตกต่างไปจากข้อมูลค่าอื่น ๆ อยู่ประปราย และกรณีที่สาม คือ ทดลองส่ง request จาก Gateway ไปยัง TSCC(Cache hit) จากการ plot box graph เพื่อที่จะแสดงการกระจายตัวของข้อมูล พบว่า response time นั้นจะเกาะกลุ่มกันอยู่ที่ 0-0.1 วินาที และมี outlier ซึ่งแสดงถึงข้อมูลที่มีค่าแยกออกจากกลุ่มหรือผิดแผกแตกต่างไปจากข้อมูลค่าอื่น ๆ อยู่ประปราย

ภาพที่ 4.21 ภาพแสดง Box graph ของผลการส่ง request จาก Gateway ไปยัง Webservice

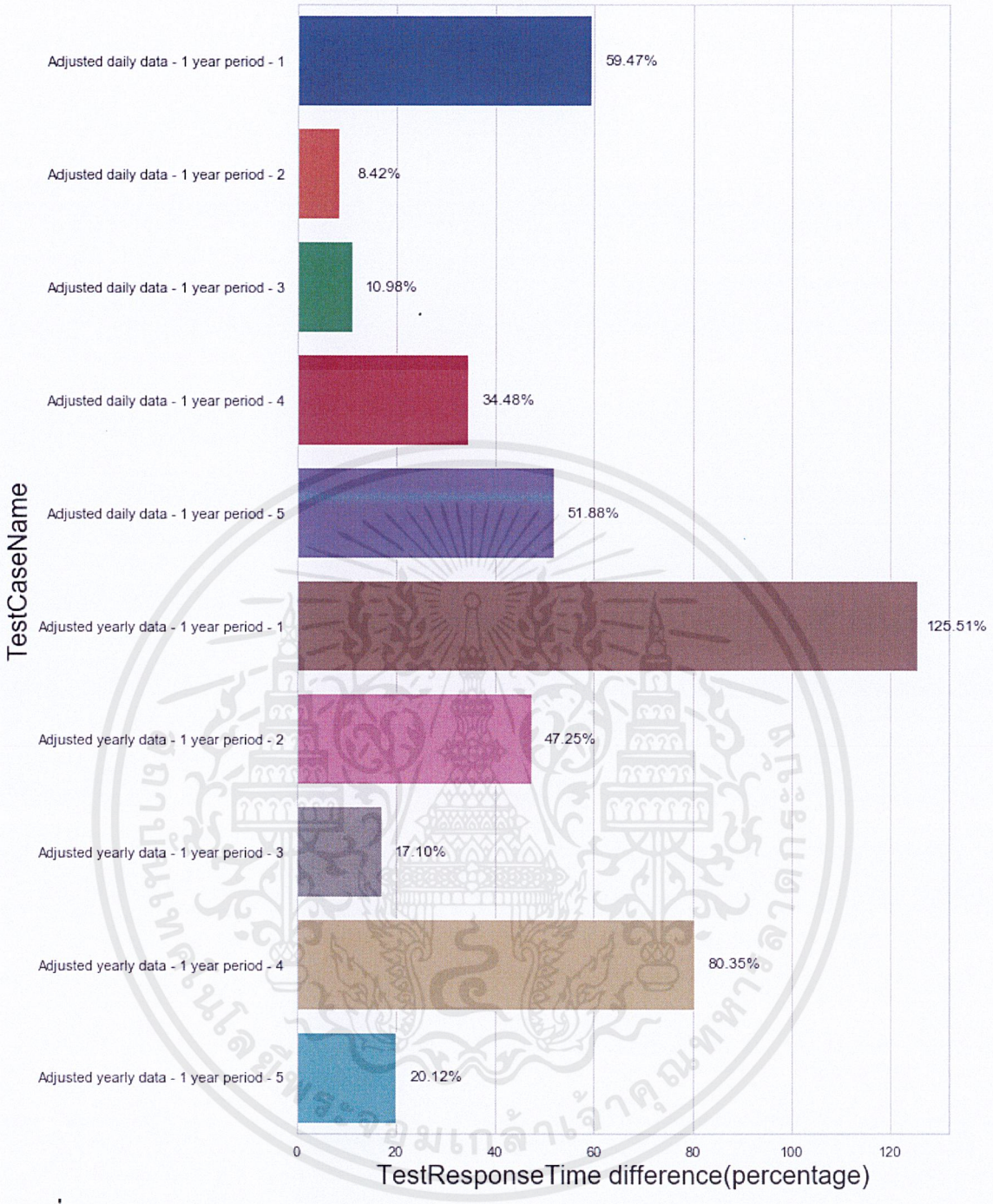
ภาพที่ 4.22 ภาพแสดง Box graph ซองผลการส่ง request จาก Gateway ไปยัง TSCC (Cache miss)

ภาพที่ 4.23 ภาพแสดง Box graph ซองผลการส่ง request จาก Gateway ไปยัง TSCC (Cache hit)

เมื่อใช้ Bar chart เพื่อแสดงการเปรียบเทียบระหว่าง response time ของการส่ง request จาก Gateway ไปยัง Webservice กับการส่ง request จาก Gateway ไปยัง TSCC (Cache miss) จะพบว่าการส่ง request จาก Gateway ไปยัง TSCC(Cache miss) นั้นมี response time มากกว่าการส่ง request จาก Gateway ไปยัง Webservice ในทุกกรณี เนื่องจาก TSCC นั้นไม่ได้เก็บ cache ในทุกกรณี และบาง test case นั้นจะมีบางตัวแปรที่มีวิธีการเรียกข้อมูลด้วยช่วงเวลาที่ต่างกัน และจะทำการส่ง request ไปยัง Backend เพื่อขอข้อมูลจึงทำให้ในบางกรณีได้ response time ที่มากกว่าการส่ง request จาก Gateway ไปยัง Webservice เป็นอย่างมาก



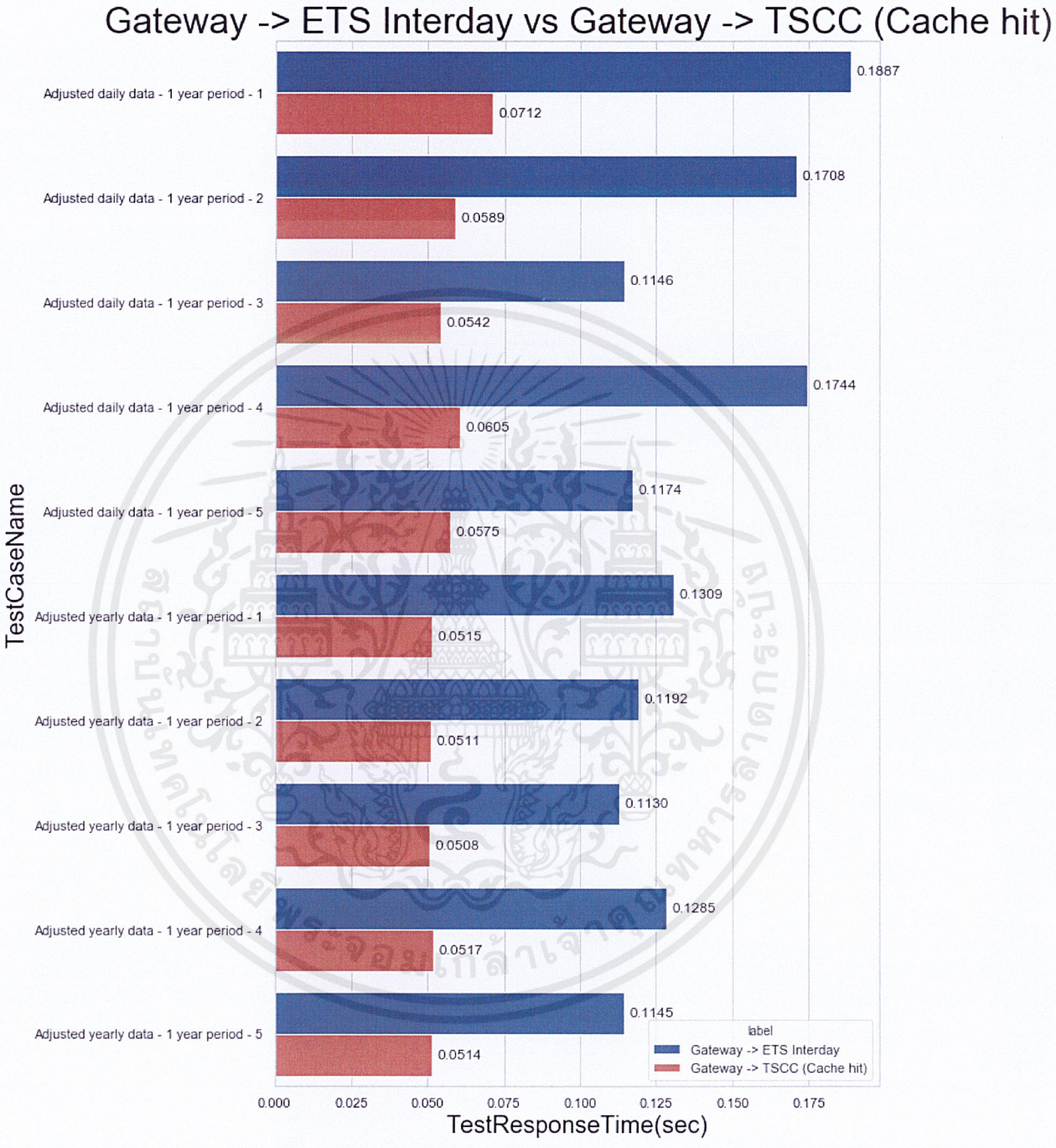
ภาพที่ 4.24 ภาพแสดง Bar chart เปรียบเทียบระหว่าง response time ของการส่ง request จาก Gateway ไปยัง Webservice กับการส่ง request จาก Gateway ไปยัง TSCC(Cache miss)



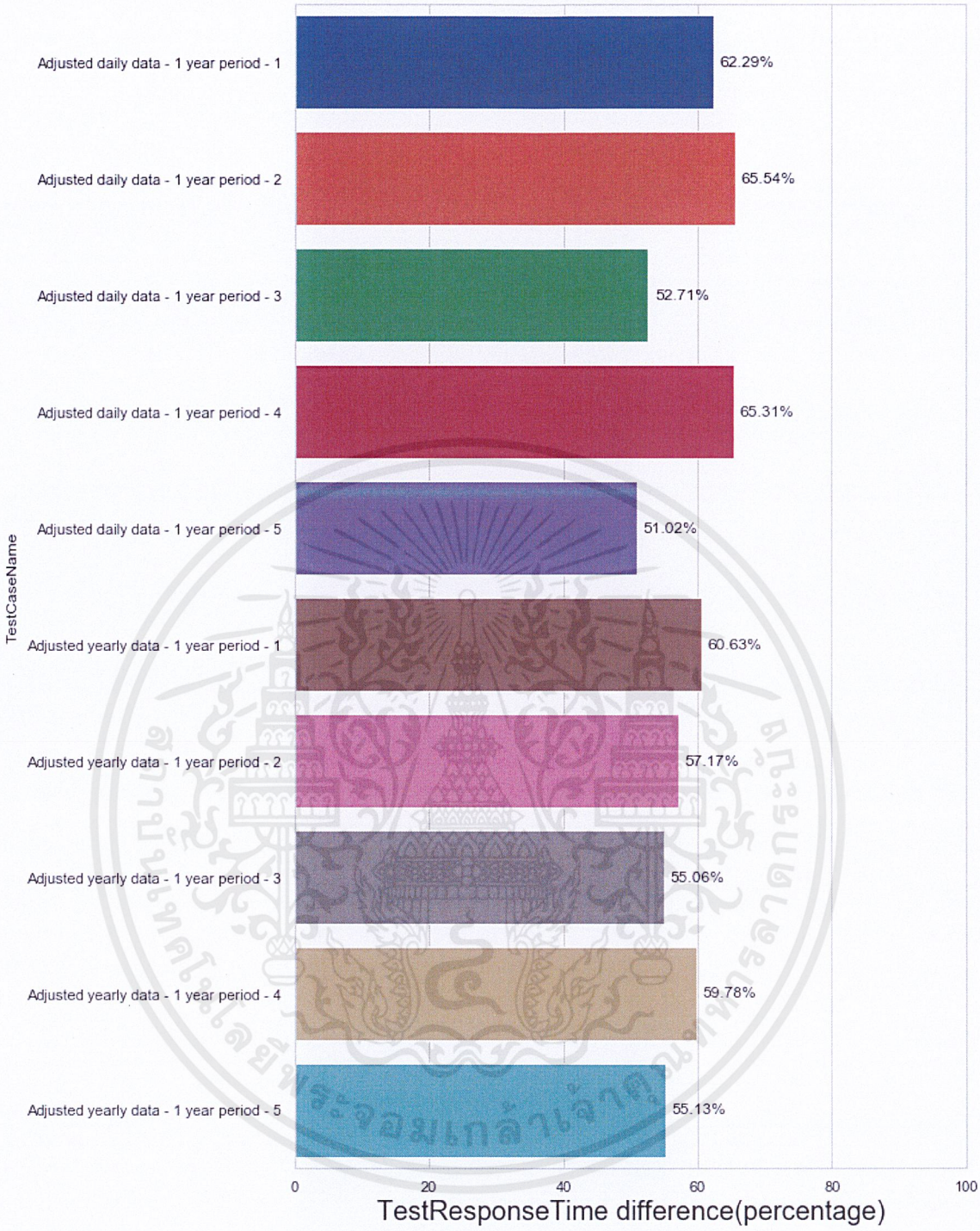
ภาพที่ 4.25 ภาพแสดง Bar chart แสดงผลต่างระหว่าง response time ของการส่ง request จาก Gateway ไปยัง Webservice กับการส่ง request จาก Gateway ไปยัง TSCC(Cache miss) ในหน่วย percentage

และเมื่อใช้ Bar chart เพื่อแสดงการเปรียบเทียบระหว่าง response time ของการส่ง request จาก Gateway ไปยัง Webservice กับการส่ง request จาก Gateway ไปยัง TSCC(Cache hit) จะพบว่าการส่ง request จาก Gateway ไปยัง TSCC(Cache hit) นั้นมี response time น้อยกว่าการส่ง request จาก Gateway ไปยัง Webservice ในทุกกรณีตาม

สมมติฐานที่ได้ตั้งไว้ในตอนต้น เนื่องจาก TSCC มีการใช้ caching ใน case ที่ผู้ใช้งานนั้นมีการเรียกใช้อยู่บ่อยครั้ง จึงทำให้ลดเวลาในการทำงานลงไป



ภาพที่ 4.26 ภาพแสดง Bar chart เปรียบเทียบระหว่าง response time ของการส่ง request จาก Gateway ไปยัง Webservice กับการส่ง request จาก Gateway ไปยัง TSCC(Cache hit)



ภาพที่ 4.27 ภาพแสดง Bar chart แสดงผลต่างระหว่าง response time ของการส่ง request จาก Gateway ไปยัง Webservice กับการส่ง request จาก Gateway ไปยัง TSCC(Cache hit) ในหน่วย percentage

จากรูปข้างต้นจะสรุปได้ว่า การส่ง request จาก Gateway ไปยัง TSCC (Cache-hit) นั้นให้ response time ที่น้อยกว่า การส่ง request จาก Gateway ไปยัง Webservice โดย

บทที่ 5
สรุปผลการดำเนินงานและข้อเสนอแนะ

ตลอดระยะเวลาการดำเนินงานวิเคราะห์รูปแบบการทำงานไปจนถึงการปรับปรุงกระบวนการต่าง ๆ จนกระทั่งสิ้นสุดโครงการสหกิจศึกษา สามารถสรุปผลการดำเนินงาน ปัญหาและอุปสรรค และข้อเสนอแนะแนวทางในอนาคตได้ดังต่อไปนี้

5.1 สรุปผลการดำเนินงาน

จากการแก้ไขปัญหาในการทำ Regression Testing และ End-to-end Testing โดยสร้างเครื่องมือขึ้นมาเพื่อแก้ไขปัญหานี้ที่มีชื่อว่า Polly นั้นสามารถแก้ปัญหาในเรื่องการไม่สามารถใช้งานข้อมูลที่อยู่ใน QE ได้หลังจากข้อมูลถูกสร้างมาแล้ว 90 วันได้ โดยการใช้ Record Feature บันทึกข้อมูลที่ได้จาก QE โดยใช้ Proxy Feature ลงไปในไฟล์ txt และใช้ Replay Feature เพื่ออ่านไฟล์ที่ถูกบันทึกแล้วส่งข้อมูลให้ผู้ใช้งานนอกผลพลอยได้ที่เกิดขึ้นคือทำให้ลดในการทำงานลงด้วย

การทดลอง Single Response Time Test ซึ่งทำเพื่อพิสูจน์ว่า Component ที่ทางทีมได้พัฒนาขึ้นมาที่มีชื่อว่า TSCC นั้นช่วยเพิ่มประสิทธิภาพในการทำงานของระบบ โดยผลการทดลองเป็นไปตามที่คาดหวังไว้ โดย response time ของการส่ง request จาก Gateway ไปยัง TSCC(Cache hit) นั้นน้อยกว่า response time ของการส่ง request จาก Gateway ไปยัง Webservice โดยเฉลี่ยถึงร้อยละ 58.56

5.2 ปัญหาและอุปสรรค

5.2.1 ขาดความรู้ความเข้าใจในด้านการออกแบบการเขียนแอปพลิเคชันแบบ MVC(Model–view–controller) จึงทำต้องการออกแบบและ refactor ใหม่

5.2.2 ต้องรอข้อมูลจากอีกทีมจึงทำให้บางครั้งเกิดความล่าช้าในการทำงาน

5.2.3 บางครั้งต้องแบ่งกันใช้เครื่องที่จะทำการทดลองกับอีกทีม จึงไม่สามารถทำการทดลองได้ในบางวัน

5.3 ข้อเสนอแนะและแนวทางในอนาคต

5.3.1 Polly สามารถนำไปใช้งานร่วมกับ Robot Framework เพื่อทำ Automation testing ได้โดยต้องเขียน test case ให้มีตัวแปรต่าง ๆ ของ request เป็นไปตามรูปแบบที่ได้ออกแบบเอาไว้สำหรับการใช้งาน Polly

เอกสารอ้างอิง

[1] Rachata Tongpagdee . 2561. “Docker คืออะไร ใช้งานอย่างไร” [Online].
แหล่งที่มา https://medium.com/@rachatatongpagdee/docker-คืออะไร-ใช้งานอย่างไร-7e77145967b6 (11 พฤศจิกายน 2562).

[2] Pattanapong Cherthong . 2559. “ทำความรู้จัก Docker และ Software Container” [Online].
แหล่งที่มา https://medium.com/thothzocial-engineering/ทำความรู้จัก-docker-และ-sofyware-container-c6338629da11 (11 พฤศจิกายน 2562).

[3] hub.docker . “Expolre - Docker Hub” [Online].
แหล่งที่มา https://hub.docker.com/search?q=&type=image (11 พฤศจิกายน 2562).

[4] Natthakarn Pruksapong . 2562 . “[Docker EP.011] ทำความรู้จัก Docker compose” [Online].
แหล่งที่มา https://medium.com/touch-technologies/ทำความรู้จัก–Docker-compose-b6688fc98c6f (11 พฤศจิกายน 2562).

[5] poundxi . 2559 . “Linux คืออะไร ?” [Online].
แหล่งที่มา https://poundxi.com/linux-คืออะไร/ (11 พฤศจิกายน 2562).

[6] Joseph Benharosh. 2561 . “REST API คืออะไร ?” [Online].
แหล่งที่มา https://phpenthusiast.com/blog/what-is-rest-api (12 พฤศจิกายน 2562).

[7] saixiii . 2559 . “RESTful หรือ REST คือ” [Online].
แหล่งที่มา https://saixiii.com/what-is-restful/ (12 พฤศจิกายน 2562).

[8] thai-amazon-cloud.blogspot. 2558 . “เกริ่นนำ: มารู้จักกับระบบ AMAZON AWS CLOUD (AMAZON WEB SERVICES)” [Online].
แหล่งที่มา http://thai-amazon-cloud.blogspot.com/2014/12/amazon-aws-cloud-amazon-web-services.html (12 พฤศจิกายน 2562).

[9] Sites.google. “ความหมายของภาษาจาวา.” [Online].
แหล่งที่มา https://sites.google.com/site/lllfejherh/phun-than-phasa-cawa
(13 พฤศจิกายน 2562).

[10] Eclipse4sl. “การพัฒนาภาษาจาวา.” [Online].
แหล่งที่มา https://www.eclipse4sl.org/ทำความเข้าใจกับภาษา-java-คือ/ (13 พฤศจิกายน 2562).

[11] Sites.google. “แนวคิดการเขียนโปรแกรมเชิงวัตถุ.” [Online].
แหล่งที่มา https://sites.google.com/site/lllfejherh/phun-than-phasa-cawa
(13 พฤศจิกายน 2562).

[12] อานนท์ หลงหัน. 2556. “การปกป้อง (Encapsulation).” [Online].
แหล่งที่มา https://arit.rmutsv.ac.th/th/blogs/80-การเขียนโปรแกรมเชิงวัตถุ-oop-object-oriented -programming-537 (13 พฤศจิกายน 2562).

[13] Marcuscode. 2561. “การสืบทอด (Inheritance).” [Online].
แหล่งที่มา http://marcuscode.com/lang/python/inheritance (13 พฤศจิกายน 2562).

[14] LordGift. 2555. “การพ้องรูป (Polymorphism).” [Online].
แหล่งที่มา https://www.lordgift.in.th/2012/11/overloading-overriding.html
(13 พฤศจิกายน 2562).

[15] จีระพงษ์ โพพันธุ์. 2560. “หลักการทำงานของภาษาจาวา.” [Online].
แหล่งที่มา https://www.krui3.com/content/knowledge-of-java/ (13 พฤศจิกายน 2562).

[16] Nongtha57. “ข้อดีของภาษาจาวา.” [Online].
แหล่งที่มา https://nongtha57.wordpress.com/ความเป็นมา-java/ (13 พฤศจิกายน 2562).

[17] Javastick. “ข้อเสียของภาษาจาวา.” [Online].
แหล่งที่มา http://javastick.web44.net/gbjava.html (13 พฤศจิกายน 2562).

[18] mindphp . 2562 . “Python คืออะไร ไพธอน คือภาษา สำหรับเขียนโปรแกรมคอมพิวเตอร์ภาษาหนึ่ง” [Online].
แหล่งที่มา https://www.mindphp.com/คู่มือ/73-คืออะไร/2417-python-คืออะไร.html (14 พฤศจิกายน 2562).

[19] Alex Kistenev . 2560 . “Daily Standup Meeting Excel Template for Your Scrum Team” [Online].
แหล่งที่มา https://standuply.com/blog/daily-standup-meeting-excel-template/ (14 พฤศจิกายน 2562).

[20] sunitarajain.wordpress . 2558 . “What is an Agile Sprint Retrospective” [Online].
แหล่งที่มา https://sunitarajain.wordpress.com/2014/09/27/what-is-an-agile-sprint-retrospective/ (14 พฤศจิกายน 2562).

[21] Thanyavuth Akarasomcheep . 2561 . “Scrum คืออะไร เริ่มใช้งานอย่างไร” [Online].
แหล่งที่มา https://medium.com/fastwork-engineering/scrum-คืออะไร-เริ่มใช้งานอย่างไร-2483e761a47e (14 พฤศจิกายน 2562).

[22] mimeo . 2561 . “Three Reasons to Become Scrum Master Certified” [Online]. แหล่งที่มา https://www.mimeo.com/blog/three-reasons-scrum-master-certified/ (14 พฤศจิกายน 2562).

[23] guru99 . “Unit Testing Tutorial: What is, Types, Tools, EXAMPLE” [Online]. แหล่งที่มา https://www.guru99.com/unit-testing-guide.html (14 พฤศจิกายน 2562).

ภาคผนวก

รายละเอียดสถานประกอบการ

REFINITIV

ชื่อสถานประกอบการ	บริษัท รีฟินิทีฟ ซอฟต์แวร์(ประเทศไทย) จำกัด
แผนก	Time Series
ระยะเวลาที่ปฏิบัติงาน	4 มิถุนายน – 30 ธันวาคม 2562
พนักงานที่ปรึกษา	นายฤทธิ์ คันธพิศาล

บทคัดย่อ

เนื่องจากทีมไทม์ซีรี่ส์นั้นมีการทำงานที่เกี่ยวข้องกับการจัดการข้อมูลต่าง ๆ โดยจะมีการทำ Regression testing เพื่อทดสอบการทำงานของ Component ต่าง ๆ โดยการทำ Regression testing และ End-to-end testing นั้นจะทำการขอข้อมูลจาก Datastore ตัวหนึ่งที่เรียกว่า Query Engine โดยข้อมูลที่สร้างขึ้นเมื่อเวลาผ่านไป 30 วันแล้วจะหายไป จึงทำการสร้างเครื่องมือที่เก็บบันทึกข้อมูลที่ได้จาก Query Engine และทำการส่งข้อมูลที่บันทึกให้กับผู้ใช้งาน จึงมีชื่อว่า Polly

นอกจากนี้ภายในทีมได้มีการพัฒนา component ใหม่ขึ้นมาหนึ่งคือ Time Series Cloud Cache เพื่อเพิ่มประสิทธิภาพของเวลาในการรับ-ส่งข้อมูลให้ดียิ่งขึ้น โดยทางทีมต้องการจะทำการทดลองเพื่อพิสูจน์ว่า Time Series Cloud Cache นั้นเพิ่มประสิทธิภาพในการรับ-ส่งข้อมูลให้ดีขึ้นตามที่คาดไว้หรือไม่ โดยการทดลองนี้เรียกว่า Single Response Time Test

วัตถุประสงค์

1. เพื่อแก้ปัญหาของการเรียกใช้ข้อมูลใน QE ไม่ได้หลังจากเวลาผ่านไป 90 วัน
2. เพื่อทำการพิสูจน์ว่า TSCC นั้นช่วยเพิ่มประสิทธิภาพในด้าน response time ตามสมมติฐานที่ได้ตั้งไว้

ขอบเขตการพัฒนา

สามารถแบ่งขอบเขตของการวิจัยได้ดังนี้ตรวจสอบสถานะของเว็บไซต์

1. Polly
   - 1.1. ออกแบบและพัฒนาเครื่องมือเพื่อแก้ไขปัญหาของการเรียกใช้ข้อมูลไม่ได้หลังจากผ่านไป 90 วัน
   - 1.2. นำเครื่องมือที่ได้ทำการพัฒนาไป deploy และ set up environment ให้ตรงกับที่ผู้ใช้งานต้องการ
2. Single Response Time Test
   - 2.1. ทำการทดลองเพื่อพิสูจน์ว่า componentที่ได้ทำการสร้างขึ้นมาใหม่นั้นช่วยเพิ่มประสิทธิภาพในด้าน response time ตามสมมติฐานที่ได้ตั้งไว้
   - 2.2. นำผลที่ได้จากการทดลองมาแสดงผลเพื่อการวิเคราะห์

ประโยชน์ที่ได้รับ

ประโยชน์ที่ได้รับจากการเข้าร่วมทำโปรเจคที่ได้รับมอบหมายตลอดระยะเวลาภายใต้โครงการ สหกิจศึกษาสามารถแบ่งได้เป็น 2 ส่วน ดังต่อไปนี้

1. ประโยชน์ของ Polly
   - 1.1. แก้ปัญหาเรื่องการใช้งานข้อมูลที่มีอายุมากกว่า 90 วันไม่ได้
   - 1.2. ลดเวลาในการทำงานเมื่อใช้ Replay feature
   - 1.3. สามารถนำไปต่อยอดในการใช้งานร่วมกับ Robot framework เพื่อทำเป็น Automation testing ได้
2. ประโยชน์ของ Single Response Time Test
   - 2.1. เพื่อทำการวัดผลและวิเคราะห์ว่า Component ที่สร้างมาใหม่นั้นให้ผลลัพธ์เป็นไปตามจุดประสงค์ที่ตั้งไว้หรือไม่

วิธีการดำเนินงาน

1. Polly
   - 1.1. กระบวนการเข้าใจปัญหาที่เกิดขึ้นของระบบการทำงานและออกแบบระบบ
     - 1.1.1. ทำความเข้าใจระบบงานเดิม
     - 1.1.2. ศึกษาปัญหาที่มีต่อความต้องการของผู้ใช้
   - 1.2 กระบวนการวิเคราะห์ระบบและออกแบบระบบ
     - 1.2.1. การเก็บรวบรวมความต้องการของผู้ใช้งาน
     - 1.2.2. การออกแบบการทำงานของระบบ
       - 1.2.2.1. Flow chart
       - 1.2.2.2. Sequence Diagram
   - 1.3. การพัฒนาระบบ
     - 1.3.1. การรับ Request จากผู้ใช้งาน
     - 1.3.2. การเลือก Feature ในการทำงาน
     - 1.3.3. การทำงานของ Feature Replay
     - 1.3.4. การทำงานของ Feature Proxy
     - 1.3.5. การทำงานของ Feature Record
2. Single Response Time Test
   - 2.1. ทำความเข้าใจจุดประสงค์ผู้ใช้งานและระบบ
   - 2.2. ออกแบบและเขียน script
     - 2.2.1. ออกแบบไฟล์ config
     - 2.2.2. ออกแบบการทวนการทำงานของ script
     - 2.2.3. ออกแบบ config.py
     - 2.2.4. ออกแบบ model.py
     - 2.2.5. ออกแบบ webService.py
     - 2.2.6. เขียน dockerfile
     - 2.2.7. เขียน shell script
     - 2.2.8. แสดงผลและวิเคราะห์ผล

เครื่องมือและโปรแกรมที่ใช้ในการพัฒนา

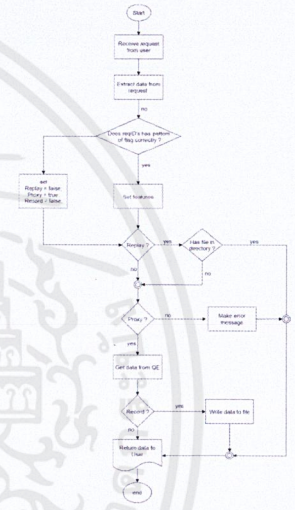
เทคโนโลยีที่ใช้ในการพัฒนา

Java 8 | amazon web services EC2 | REST API

ผลการพัฒนา

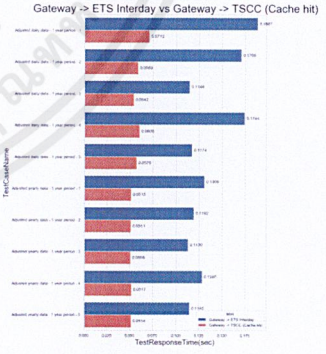
1. Polly

Polly นั้นสามารถแก้ปัญหาเรื่องการใช้งานข้อมูลที่มีอายุมากกว่า 90 วันไม่ได้ โดยมีการทำงานดัง Flow chart ดังภาพต่อไปนี้



2. Single Response Time Test

จากการทดลองพบว่าการส่ง request จาก Gateway ไปยัง TSCC (Cache-hit) นั้นให้ response time ที่น้อยกว่าการส่ง request จาก Gateway ไปยัง Webservice โดยเฉลี่ยยังร้อยละ 58.56



กิตติกรรมประกาศ

การทำสหกิจศึกษานี้สำเร็จลุล่วงไปได้ด้วยดี ด้วยคำแนะนำ และคำปรึกษาจากหลายๆท่าน ทั้งในเรื่องของการทำงาน และแนวคิดต่าง ๆ นอกจากนี้ยังมีพี่ ๆ ที่คอยสอนและให้คำแนะนำเกี่ยวกับงานต่าง ๆ ให้อีกด้วย

ขอขอบพระคุณอาจารย์ที่ปรึกษา คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่คอยให้คำแนะนำในเรื่องการทำสหกิจศึกษาในครั้งนี้

สุดท้ายนี้ขอขอบคุณรุ่นพี่และเพื่อน ๆ ทุกคนที่คอยให้คำปรึกษาในเรื่องการทำงาน การแก้ไขปัญหาต่าง ๆ ตั้งแต่เริ่มทำสหกิจศึกษาจนสำเร็จรายงานฉบับนี้ไปได้ด้วยดี

ภาพที่ 6.1 โปสเตอร์แสดงข้อมูลของโครงงานวิจัยสหกิจศึกษา