รายงานสหกิจศึกษาฉบับสมบูรณ์

ระบบช่วยในการตรวจสอบประสิทธิภาพของระบบ
Performance Analysis System (PerfWatch)

นายอรรถ อัสโสรัตน์กุล

ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

รายงานสหกิจศึกษาฉบับสมบูรณ์

ระบบช่วยในการตรวจสอบประสิทธิภาพของระบบ
Performance Analysis System (PerfWatch)

นายอรรถ อัสโสรัตน์กุล

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ชื่อโครงงานสหกิจศึกษา ระบบช่วยในการตรวจสอบประสิทธิภาพของระบบ
ผู้จัดทำ นายอรรถ อัสโสรัตน์กุล 58011424
คณะ วิศวกรรมศาสตร์
ภาควิชา คอมพิวเตอร์
อาจารย์ผู้นิเทศ อาจารย์บัณฑิต พัสยา
อาจารย์จิระศักดิ์ สิทธิกร
สถานที่ประกอบการ บริษัท รอยเตอร์ ซอฟท์แวร์ จำกัด (มหาชน)

บทคัดย่อ

เนื่องจากในปัจจุบันเครื่องมือที่ใช้ในการตรวจสอบประสิทธิภาพของระบบที่ทีม Time Series ใช้อยู่นั้น ไม่สามารถบอกความผิดปกติที่เกิดขึ้นในระบบได้อย่างชัดเจน และไม่มีความสามารถในการหาสาเหตุของความผิดปกติที่เกิดขึ้นภายในระบบ ทำให้ผู้ที่มีหน้าที่ตรวจสอบประสิทธิภาพของระบบ ต้องทำการรวบรวมข้อมูลของการทำงานที่อยู่ภายในองค์ประกอบต่าง ๆ ของระบบ และนำข้อมูลเหล่านั้นมาวิเคราะห์ด้วยตัวเอง ซึ่งเป็นกระบวนการที่ใช้เวลานานและอาจจะใช้เวลาถึง 4 วันในการดำเนินงาน

ผู้พัฒนาจึงพัฒนาเครื่องมือที่ช่วยอำนวยความสะดวกให้กับผู้ที่มีหน้าที่ตรวจสอบประสิทธิภาพของระบบ เพื่อให้สามารถตรวจสอบและหาสาเหตุของความผิดปกติที่เกิดขึ้นในระบบได้อย่างรวดเร็วและแม่นยำ โดยเครื่องมือที่ผู้พัฒนา พัฒนาขึ้นนั้นจะทำการรวบรวมข้อมูลการทำงานที่อยู่ภายในองค์ประกอบต่าง ๆ ของระบบและนำข้อมูลเหล่านั้นมาวิเคราะห์เพื่อหาความผิดปกติของข้อมูล จากนั้นจะนำเอาข้อมูลที่ทำการวิเคราะห์มาแสดงผลในรูปแบบ Dashboard เพื่อให้ผู้ที่มีหน้าที่ตรวจสอบประสิทธิภาพของระบบสามารถตรวจสอบข้อมูลที่มีจำนวนมากได้สะดวกและรวดเร็วมากยิ่งขึ้น

จากการที่ให้ผู้มีหน้าที่ตรวจสอบประสิทธิภาพของระบบทดลองใช้งานเครื่องมือที่ผู้พัฒนาพัฒนาขึ้นนั้น พบว่าสามารถช่วยให้กระบวนการการตรวจสอบประสิทธิภาพของระบบทำได้สะดวกและรวดเร็วขึ้นจริง แต่เนื่องจากระบบแต่ละระบบมีข้อมูลที่เกิดจากการทำงานแตกต่างกันมากทำให้เครื่องมือนี้ยังไม่สามารถนำไปประยุกต์ใช้กับระบบอื่นได้สะดวกอย่างที่ควร

คำสำคัญ: การตรวจสอบประสิทธิภาพของระบบ การวิเคราะห์ข้อมูล เครื่องมืออำนวยความสะดวก

Title	Performance Analysis System (PerfWatch)
Student	Mr. Utt Assoratgoon 58011424
Faculty	Engineering
Department	Computer
Advisor	Mr. Bundit Pasaya
	Mr. Jirasak Sittigorn
Company	Reuters Software (Thailand) Ltd

ABSTRACT

At present, the tools used to monitor the performance of the system used by the Time Series team cannot tell the abnormalities in the system clearly. And no ability to find the cause of the disorder within the system. So, it is necessary for system performance examiner to collect the data of the work within the system components and analyze the data manually. This is a time-consuming process and may take up to 4 days to complete.

The developer has developed a tool that facilitates system performance examiner. In order to be able to quickly and accurately detect and diagnose causes of faults in the system. The developed tool will collect the data in the various components of the system and analyze the data for finding abnormalities then visualize the analyzed data in form of dashboard, so system performance examiner can quickly and easily check the large amount of data.

By giving the system performance examiner to try out the developed tools. It can help to speed up the process of system performance examination. But because each system has a very different data structure, it cannot be easily applied to other systems.

Keywords: System performance examination Data analysis Facilitated tool

กิตติกรรมประกาศ

โครงงานสหกิจครั้งนี้สามารถดำเนินงานสำเร็จได้ด้วยดีเนื่องจากได้รับความช่วยเหลือ การสนับสนุน และความกรุณาจากองค์กรและบุคคลหลายท่าน คณะผู้จัดทำขอกล่าวคำขอบคุณองค์กรและบุคคลดังต่อไปนี้

ขอขอบพระคุณ อาจารย์บัณฑิต พัสยา และอาจารย์จิระศักดิ์ สิทธิกร อาจารย์ที่ปรึกษาสหกิจศึกษา ที่ได้ให้คำปรึกษาพร้อมทั้งแนวทางแก้ปัญหา รวมทั้งตรวจแก้สหกิจศึกษาฉบับนี้ให้มีความสมบูรณ์เพิ่มมากขึ้น

ขอขอบพระคุณอาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้ความรู้และประสบการณ์ตลอดระยะเวลาที่ได้รับการศึกษา

ขอขอบพระคุณ นายชิษณุ ทองฉิม นายฤทธิ์ คันธพิศาล และนายวีรศักดิ์ เจนวิทยาวุธ ผู้ดูแลและให้คำปรึกษาข้าพเจ้าขณะฝึกงานอยู่ที่บริษัทรอยเตอร์ซอฟต์แวร์ (ประเทศไทย) จำกัด

ขอขอบพระคุณ บริษัทรอยเตอร์ซอฟต์แวร์ (ประเทศไทย) จำกัด ที่ให้โอกาสในการเข้ารับการฝึกงานและได้รับประสบการณ์ในการทำงานในสภาพแวดล้อมจริง

อรรถ อัสโสรัตน์กุล

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญภาพ

หน้า

สารบัญภาพ (ต่อ)

หน้า

สารบัญภาพ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1. ความเป็นมาและความสำคัญ

เนื่องจากการพัฒนาผลิตภัณฑ์ Software ที่มีขนาดใหญ่และมีระบบที่ซับซ้อน จะต้องมีการทดสอบระบบเพื่อตรวจสอบประสิทธิภาพของระบบในแต่ละเวอร์ชั่น ก่อนที่จะนำผลิตภัณฑ์ไป Deploy ให้ลูกค้าใช้งานจริง เพื่อให้มั่นใจว่าระบบมีความสมบูรณ์ และจะไม่เกิดข้อผิดพลาดเมื่อลูกค้านำผลิตภัณฑ์ไปใช้งาน

โดยปกติแล้วกระบวนการและขั้นตอนการตรวจสอบประสิทธิภาพของระบบจะมีทีมที่รับผิดชอบโดยเฉพาะ แต่เนื่องจากทางบริษัทมีการเปลี่ยนแปลงนโยบาย จึงทำให้เกิดการโยกย้ายเปลี่ยนแปลงตำแหน่งของพนักงานภายในบริษัท ผู้พัฒนาระบบจึงต้องทำการรับหน้าที่ เพื่อตรวจสอบประสิทธิภาพของระบบ ซึ่งการตรวจสอบประสิทธิภาพของระบบนี้ มีขั้นตอนและกระบวนการที่ซับซ้อน อีกทั้งเครื่องมือที่ใช้ในการตรวจสอบประสิทธิภาพของระบบในปัจจุบัน ไม่มีประสิทธิภาพมากพอ ทำให้สิ้นเปลืองเวลาและสิ้นเปลืองทรัพยากรบุคคล เป็นอย่างมาก

ทางทีม Time Series เล็งเห็นถึงปัญหานี้ จึงทำให้เกิดโครงงาน PerWatch ขึ้น โดยจุดประสงค์หลักเพื่อที่จะลดระยะเวลาในการตรวจสอบประสิทธิภาพของระบบและสืบหาความผิดพลาดในระบบที่ทำให้ระบบมีประสิทธิภาพลดลง ให้เหลือเพียงครึ่งหนึ่งหรือน้อยกว่าเวลาที่เคยใช้ และสามารถหาต้นต่อของความผิดพลาดนั้นได้อย่างรวดเร็วและแม่นยำ

1.2. วัตถุประสงค์ของการวิจัย

1.2.1. เพื่อสร้างระบบที่ช่วยอำนวยความสะดวกให้กับผู้ตรวจสอบประสิทธิภาพของระบบ
1.2.2. เพื่อลดระยะเวลาที่ใช้ในการตรวจสอบประสิทธิภาพของระบบ
1.2.3. เพื่อเก็บผลการทดสอบของระบบและนำมาวิเคราะห์

1.3. วิธีการดำเนินงานวิจัย

1.3.1. ใช้กระบวนการพัฒนา Software แบบ Agile

1.4. ขอบเขตของงานวิจัย

1.4.1. พัฒนา Scripts เพื่อรวบรวมผลการทดสอบและคำนวณระยะเวลาที่ใช้ในแต่ละองค์ประกอบของระบบใน Use Case ของ ETS Interday, ETS Intraday

1.4.2. พัฒนา Scripts เพื่อรวบรวมผลการทดสอบและคำนวณระยะเวลาที่ใช้ในแต่ละองค์ประกอบของระบบใน Use Case ของ TSI6

1.4.3. พัฒนา Scripts เพื่อรวบรวมผลการทดสอบและคำนวณระยะเวลาที่ใช้ในแต่ละองค์ประกอบของระบบใน Use Case ของ TSI7

1.4.4. จัดเก็บผลการทดสอบและผลที่ได้จากการคำนวณเข้าสู่ Elasticsearch

1.4.5. พัฒนา Scripts เพื่อวิเคราะห์ผลการทดสอบ

1.4.6. พัฒนาแอพพลิเคชั่น Dashboard เพื่อนำเสนอผลการทดสอบให้ออกมาในรูปแบบที่สามารถช่วยให้การตรวจสอบประสิทธิภาพของระบบทำได้ง่ายยิ่งขึ้น สำหรับแต่ละ Use Case

1.4.7. Deploy เพื่อให้สมาชิกในทีม Time series สามารถใช้งาน PerfWatch Dashboard ได้

1.5. ประโยชน์ที่คาดว่าจะได้รับ

1.5.1. ประโยชน์ต่อตนเอง

1.5.1.1. ได้ศึกษาและใช้ซอฟต์แวร์ที่ตนเองไม่เคยใช้

1.5.1.2. ได้ประสบการณ์ในการทำงานจริง

1.5.1.3. ได้เรียนรู้กระบวนการการพัฒนา Software ที่มีมาตรฐาน

1.5.2. ประโยชน์ต่อองค์กร

1.5.2.1. ช่วยทำให้กระบวนการตรวจสอบประสิทธิภาพของระบบรวดเร็วขึ้น

1.5.2.2. ช่วยให้การระบุต้นเหตุที่ทำให้ประสิทธิภาพของระบบมีความผิดปกติได้แม่นยำยิ่งขึ้น

1.5.2.3. ช่วยทำให้มีระบบสำหรับจัดเก็บผลการทดสอบของระบบ เพื่อนำไปทำการวิเคราะห์ต่อในอนาคต

1.6. เครื่องมือและภาษาที่ใช้ในโครงงานสหกิจศึกษา

1.6.1. ฮาร์ดแวร์

1.6.1.1. เครื่องคอมพิวเตอร์

1) ตัวประมวลผล(Processor) Intel(R) Core (TM) i5-3320M CPU @ 2.60GHz

2) หน่วยความจำหลัก(RAM) 8.00 GB

3) ชนิดของระบบ (System type) 64-bit Operating System

4) ระบบปฏิบัติการ(Operating System) Windows 7 Enterprise

1.6.1.2. เครื่องเซิร์ฟเวอร์

1) ตัวประมวลผล(Processor) Dual 2.40 GHz Intel Xeon(R)E5-2680 v4

2) หน่วยความจำหลัก(RAM) 8.00 GB

3) ชนิดของระบบ(System type) 64-bit Operating System, x64-based processor

4) ระบบปฏิบัติการ(Operating System) Windows Server 2012 R Standard

1.6.2. ซอฟต์แวร์

1.6.2.1. Microsoft Visual Studio Code ใช้ในการเขียนโปรแกรม

1.6.2.2. mRemote NG ใช้ในการเข้าถึงเครื่องเซิร์ฟเวอร์

1.6.2.3. Postman ใช้เพื่อติดต่อกับ Elasticsearch ด้วย HTTP Request

1.6.2.4. Microsoft Office Excel 2007 ใช้ในการอ่านไฟล์ csv และ ไฟล์ excel

1.6.2.5. draw.io ใช้ออกแบบระบบในรูปแบบของ Diagram

1.6.2.6. Trello ใช้ในการวางแผนการทำงาน

บทที่ 2
ทฤษฎีและเทคโนโลยีที่ใช้ในการพัฒนาระบบ

2.1. Agile

Agile Methodology คือแนวคิดในการทำงาน ไม่ใช่ขั้นตอนหรือกระบวนการการทำงานและไม่จำกัดว่าจะต้องใช้กับการพัฒนา Software เท่านั้น แนวคิด Agile จะให้ความสำคัญกับการสื่อสารกับผู้ที่เกี่ยวข้องทุกฝ่ายและเน้นการนำ Feedback ของผู้ที่เกี่ยวข้องมาปรับปรุงในทั้งผลิตภัณฑ์และกระบวนการการทำงาน เพื่อให้ผลิตภัณฑ์ที่พัฒนาขึ้นมาสามารถตอบสนองความต้องการของผู้ใช้ได้มากที่สุด

2.1.1. หลักการทำงานของ Agile

2.1.1.1. Individuals and interactions over processes and tools

ให้ความสำคัญกับการสื่อสารและปฏิสัมพันธ์กันระหว่างคน มากกว่าเครื่องมือและขั้นตอนในการพัฒนา

2.1.1.2. Working software over comprehensive documentation

ให้ความสำคัญกับการพัฒนาผลิตภัณฑ์มากกว่าการทำเอกสาร

2.1.1.3. Customer collaboration over contract negotiation

ให้ความสำคัญในการร่วมมือกับลูกค้าในการทำงานให้สำเร็จ มากกว่าการเจรจาต่อรองเพื่อทำสัญญา

2.1.1.4. Responding to change over following a plan

ให้ความสำคัญกับการเปลี่ยนแปลงของสิ่งต่างๆ เช่น ความต้องการของผู้ใช้และลำดับความสำคัญของงานที่อาจจะมีการเปลี่ยนแปลง มากกว่าการทำตามแผนที่วางเอาไว้

2.1.2. ตำแหน่งและหน้าที่ของผู้ที่อยู่ในระบบ Agile

2.1.2.1. Stakeholders

ผู้ที่เกี่ยวข้องกับผลิตภัณฑ์ เช่น ผู้ใช้งาน หรือผู้ที่มีส่วนได้ส่วนเสียกับผลิตภัณฑ์

2.1.2.2. Product Owner

ผู้ที่ต้องการสร้างผลิตภัณฑ์บางอย่างเพื่อตอบสนองความต้องการของ Stakeholders และค่อยประสานงานระหว่าง Developer กับ Stakeholders โดยมีหน้าที่หลักๆดังนี้

1) เปลี่ยนความต้องการของ Stakeholders (Requirement) ให้เป็น Use Story หรือปฏิเสธบาง ความต้องการของลูกค้า เพื่อไม่ให้มีงานเข้ามาเยอะเกินความสามารถที่จะรับงานได้ของ Developer

2) แปลง User Story ที่มีขนาดใหญ่ให้มีขนาดเล็กมากที่สุดโดยที่ยังมี Business value อยู่ เพื่อให้การพัฒนามีระยะเวลาไม่มากเกินไป และ สามารถรับ Feedback จาก Stakeholders เพื่อนำมาปรับปรุงผลิตภัณฑ์ได้อย่างรวดเร็ว

3) รักษาสมดุลระหว่างการพัฒนาผลิตภัณฑ์ทั้งในด้าน การพัฒนาฟังก์ชั่นใหม่ การแก้ปัญหาข้อผิดพลาดระหว่างการพัฒนา (Bug) และการเพิ่มประสิทธิภาพให้กับผลิตภัณฑ์ (Optimization)

4) ค่อยพิจารณาและจัดการเรื่องความเสี่ยงและต้นทุนในการพัฒนาผลิตภัณฑ์

5) จัดลำดับความสำคัญของงานโดยคำนึงถึงผลกระทบที่มีต่อผู้ใช้และ ธุรกิจ

2.1.2.3. Developer

ผู้ที่ทำการพัฒนาผลิตภัณฑ์ โดยประกอบไปด้วยตำแหน่งย่อย ดังนี้ Business Analyst, UX Designer, UI Designer, Developer, Quality Assurance โดยหนึ่งคนอาจมีบทบาทได้มากกว่า 1 ตำแหน่งและมีหน้าที่หลักๆดังนี้

1) ประเมินขนาดและประมาณเวลาของงานแต่ละงาน เพื่อประกอบการตัดสินใจเลือก Use Story ที่จะทำ

2) ออกแบบโครงสร้างของระบบ

3) ทำ Proof of Concept (POC) ของสิ่งที่จะทำ ก่อนจะนำไปใช้ในการพัฒนาผลิตภัณฑ์จริง

4) พัฒนาผลิตภัณฑ์โดยยึดตาม User Story และจบสามารถปิด User Story นั้นๆได้ก็ต่อเมื่องานที่ทำนั้นผ่าน Acceptance Criteria ที่ Product Owner ให้ไว้

5) ทำการทดสอบผลิตภัณฑ์ (Test) เพื่อให้ทำการทดสอบผลิตภัณฑ์ได้อย่างสมบูรณ์และง่าย จะมีการทำ Test check list ของผลิตภัณฑ์ให้ครบถ้วนและครอบคลุมการใช้งานของผู้ใช้ และทำการทดสอบผลิตภัณฑ์ตาม Test checklist นั้น

6) รวบรวมงานที่แต่ละทีมช่วยกันพัฒนา และทำการ Deploy เพื่อให้ผู้ใช้สามารถใช้งานได้

2.1.3. วิธีการทำงานแบบ Agile

2.1.3.1. Stakeholders มีความต้องการหรือปัญหาบางอย่างที่ต้องการแก้ไข

2.1.3.2. Product Owner ที่ต้องการแก้ปัญหาของ Stakeholders ทำการเก็บรวบรวม Requirement

2.1.3.3. Product Owner จะนำ Requirement ที่ได้มาแปลงเป็น User Story เพื่อให้ Developer นำไปพัฒนา โดย User Story แต่ละอัน Product Owner จะต้องกำหมด Acceptance Criteria เพื่อเป็นเงื่อนไขในการปิดงานของ User Story นั้นๆ

2.1.3.4. Developer ออกแบบและพัฒนาระบบตาม User Story ที่ได้รับ และต้องทำให้ผ่านเงื่อนไขตาม Acceptance Criteria

2.1.3.5. Developer จะส่งผลิตภัณฑ์ให้กับ Stakeholders ใช้งาน

2.1.3.6. Stakeholders จะลองนำผลิตภัณฑ์ไปใช้งานและให้ Feedback กลับมา

2.1.3.7. Product Owner จะนำ Feedback ของ Stakeholders มาทำเป็น User Story และนำไปให้ Developer พัฒนาต่อไป

2.2. Scrum

Scrum คือการนำแนวคิดในการทำงานแบบ Agile มาปฏิบัติด้วยกิจกรรมต่างๆ เพื่อระบุปัญหาที่มีความซับซ้อนและ เปลี่ยนแปลงบ่อย เพื่อให้สามารถส่งมอบผลิตภัณฑ์ที่ตอบสนองต่อการเปลี่ยนแปลงที่เกิดขึ้นได้อย่างรวดเร็ว

ช่วยให้การพัฒนาผลิตภัณฑ์แบบ Agile มีขั้นตอนการการดำเนินงานและผลลัพธ์ที่ชัดเจน โปร่งใส สามารถตรวจสอบประสิทธิภาพของแต่ละขั้นตอนการดำเนินงาน ปรับปรุง และวัดผลของการปรับปรุงที่เกิดขึ้นได้

2.2.1. ตำแหน่งและหน้าที่ในทีม Scrum

Scrum เหมาะสำหรับทีมขนาดเล็กที่พร้อมปรับตัว พัฒนา และเปลี่ยนแปลง สมาชิกในทีมสกรัมต้องมีความสามารถที่หลากหลาย สามารถบริหารและดำเนินงานกันเองได้ด้วยสมาชิกภายในทีม สามารถหาแก้ไขปัญหาได้เอง โดยไม่ต้องรอความช่วยเหลือจากนอกทีม (Self-Organizing Teams) ซึ่งประกอบด้วย

2.2.1.1. Product Owner

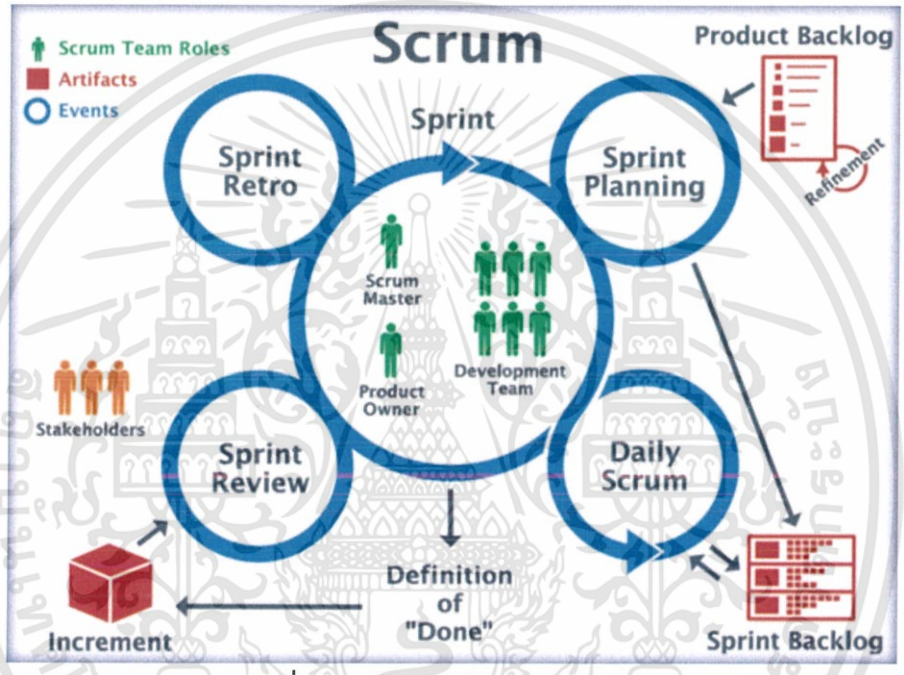
คือ คนที่ประเมินค่าของงาน(Value) และความสำคัญของแต่ละงาน(Priority) ให้กับทีม Product Owner ยังมีหน้าที่ทำให้คนในทีมเข้าใจรายละเอียดของงานตรงกันอีกด้วย

2.2.1.2. Scrum Master

คือ คนที่ทำให้การทำงานของคนในทีมเป็นไปอย่างราบรื่น ไม่ใช่ผู้นำทีม แต่เป็นคนที่คอยกำจัดอุปสรรคที่ขัดขวางไม่ให้ทีมบรรลุเป้าหมาย

2.2.1.3. Team

ประกอบไปด้วยคนประมาณ 3-6 คน โดยแต่ละคนนั้นต้องสามารถทำงานตั้งแต่ต้นจนจบได้ด้วยตัวเอง ซึ่งหมายความว่าทุกคนสามารถสลับ หรือช่วยงานกันได้ทั้งหมด



ภาพที่ 2.1 หลักการทำงานของ Scrum

2.2.2. หลักการทำงานของ Scrum

การทำงาน ด้วย Scrum จะแบ่งรอบของการทำงานเป็นหลายๆรอบโดยมีระยะเวลารอบละ 1 อาทิตย์ โดยแต่ละรอบการทำงานจะมีขั้นตอนดังนี้

2.2.2.1. Planning

ทุกๆวันแรกของรอบการทำงานจะมีการประชุมเพื่อวางแผนว่าในรอบการทำงานนี้ จะต้องทำงานอะไรบ้างโดยจะมี Product Owner เป็นคนประเมินงานและจัดลำดับความสำคัญให้ เมื่อกำหมดงานทั้งหมดที่ต้องทำในรอบการทำงานได้แล้ว จึงแบ่งงานให้กับทุกคนในทีม

2.2.2.2. Standup Meeting

ทุกวันในรอบการทำงาน จะมีการยืนประชุมกันเป็นเวลาสั้นๆเพื่อให้คนในทีมได้มีการสื่อสารกัน โดยทุกคนต้องบอกถึงการทำงานของวันที่ผ่านมาว่าได้ทำอะไรไปบ้าง แล้ววันนี้จะทำอะไร และเจออุปสรรคในการทำงานที่ผ่านมาหรือไม่ หากเห็นว่างานที่ทำจะไม่ทำให้เสร็จได้ในรอบการทำงานก็สามารถขอความช่วยเหลือของคนในทีมได้

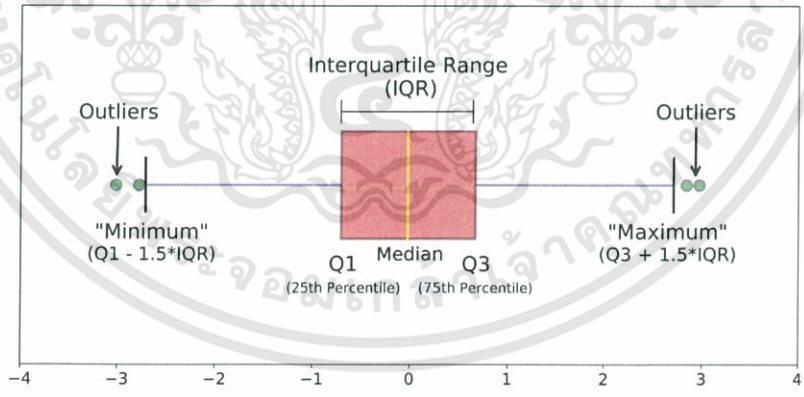
2.2.2.3. Review (Demo)

วันสุดท้ายของรอบการทำงานจะมีการนำเสนองาน (Demo) ที่ได้ทำมาตลอดรอบการทำงาน ให้กับผู้ใช้เพื่อให้ผู้ใช้แสดงความคิดเห็น และนำความเห็นของผู้ใช้ไปปรับเปลี่ยนในรอบการทำงานต่อไป

2.2.2.4. Retrospective

หลังจากจบรอบการทำงานจะมีการประชุมเพื่อให้ทุกคน ได้บอกเล่าถึงเรื่องราวที่เกิดขึ้นระหว่างรอบการทำงาน ไม่ว่าทำเป็นเรื่องดีๆที่เกิดขึ้นและต้องการให้คงไว้ หรือเรื่องที่ต้องปรับปรุง หากมีเรื่องที่ต้องปรับปรุงคนในทีมจะต้องช่วยกันหาวิธีที่แก้ไขเรื่องนั้น

2.3. Boxplot



ภาพที่ 2.2 องค์ประกอบของ Boxplot

Boxplot คือ กราฟที่แสดงถึงการกระจายของข้อมูลและสามารถใช้ในการระบุค่า Outlier ของข้อมูลได้อีกด้วย โดยส่วนใหญ่แล้วจะนิยมใช้ Boxplot ในการเปรียบเทียบการกระจายของข้อมูลหลายๆ

ชุด เนื่องจาก Boxplot มีขนาดที่เล็กและแสดงผลได้ชัดเจนในเรื่องกระจายของข้อมูลมากกว่า Histogram ซึ่งสามารถบอกการกระจากของข้อมูลได้เหมือนกันแต่มีขนาดใหญ่และไม่สามารถใช้เปรียบเทียบกับข้อมูลชุดอื่นๆได้อย่างชัดเจน

ส่วนประกอบของ Boxplot

2.3.1. Median (Percentile ที่ 50) คือ ค่าที่อยู่ตำแหน่งตรงกลางของชุดข้อมูล

2.3.2. Percentile ที่ 25(Q1) คือ ค่าที่อยู่ตำแหน่งตรงกลางระหว่างค่าที่อยู่ตำแหน่งต่ำสุดและค่าMedian

2.3.3. Percentile ที่ 75(Q2) คือ ค่าที่อยู่ตำแหน่งตรงกลางระหว่างค่าMedianและค่าที่อยู่ตำแหน่งสูงสุด

2.3.4. Interquartile (IQR) คือ ค่าผลต่างของ Percentile ที่ 25 กับ Percentile ที่ 75 (Q1 - Q2)

2.3.5. Minimum คือ Q1 - 1.5*IQR

2.3.6. Maximum คือ Q3 + 1.5*IQR

2.3.7. Lower Whisker คือ ระยะระหว่าง Minimum ถึง Percentile ที่ 25

2.3.8. Upper Whisker คือ ระยะระหว่าง Percentile ที่ 75 ถึง Maximum

2.3.9. Outliers คือ ข้อมูลที่มีค่าน้อยกว่า Lower Whisker หรือมากกว่า Upper Whisker ซึ่งข้อมูลเหล่านี้จะสามรถทำให้ภาพรวมของชุดข้อมูลนั้น ผิดเพี้ยนไปจากความเป็นจริงได้

2.4. Elasticsearch

Elasticseach ถูกสร้างขึ้นครั้งแรกในปี ค.ศ. 2010 โดย Shay Banon ซึ่งได้พัฒนาต่อยอดมาจาก Apache Lucene (ซอฟต์แวร์สำหรับจัดการลำดับ (Indexing) และ ค้นหาข้อมูลพัฒนาด้วยภาษาJava) เป็นแหล่งเก็บข้อมูลแบบกระจายตัว (Distributed data store) และเก็บข้อมูลในรูปแบบ JSON ซึ่งใช้ RESTful API ในการติดต่อกับส่วนของ Data store

โดยจุดเด่นของ Elasticsearch คือความเร็วในการค้นหาข้อมูล ซึ่งเป็นการค้นหาแบบ distributed search ที่ทุกๆ ฟิลด์ข้อมูลที่เก็บจะถูกทำ index ไว้ทำให้ความเร็วในการค้นหากับข้อมูลที่มีปริมาณมหาศาล ใกล้เคียง Real-time

2.4.1. หลักการทำงานของ Elasticsearch

2.4.1.1. การทำดัชนีค้นหา(Index)

การทำดัชนีค้นหาเผื่อทำให้ผู้ใช้สามารถค้นหาข้อมูลได้ โดยจะใช้ระบบที่เรียกว่า Inverted Index

ภาพที่ 2.3 ตัวอย่างการทำ Inverted Index

จากตัวอย่างข้างต้น ทางด้านซ้ายมือคือข้อมูลที่ต้องการจัดเก็บรวมทั้งสิ้น3 ฉบับ แต่ข้อมูลนี้ยังไม่สามารถใช้ในการค้นหาได้ ระบบ Inverted Index จะทำการจัดเก็บข้อมูลโดยนำข้อมูลมาจัดเรียงใหม่โดยมีลักษณะคล้ายกับการทำสารบัญ เพื่อระบุว่าคำแต่ละคำนั้นอยู่ในเอกสารฉบับไหน

หากค้นหาคำว่า “fury” Elasticsearch จะเข้าไปค้นหาใน Inverted Index เพื่อดูว่า “fury” อยู่ในเอกสารฉบับไหนบ้าง และจะทราบได้ทันทีว่าอยู่ในเอกสารฉบับที่2 ตามรูปตัวอย่างข้างต้น

2.4.2. ระบบคะแนนผลลัพธ์ (Relevancy)

Elasticsearch จะใช้ระบบ TF-IDF (Term Frequency-Inverse Document Frequency) ในการคำนวณคะแนนเพื่อแสดงผลลัพธ์

2.4.2.1. Term Frequency

คำนวนคะแนนจากความถี่ของคำที่ค้นหาในเอกสารนั้นๆ หากเอกสารใดมีจำนวนคำที่ค้นหามากจะได้คะแนนมาก

2.4.2.2. Inverse Document Frequency

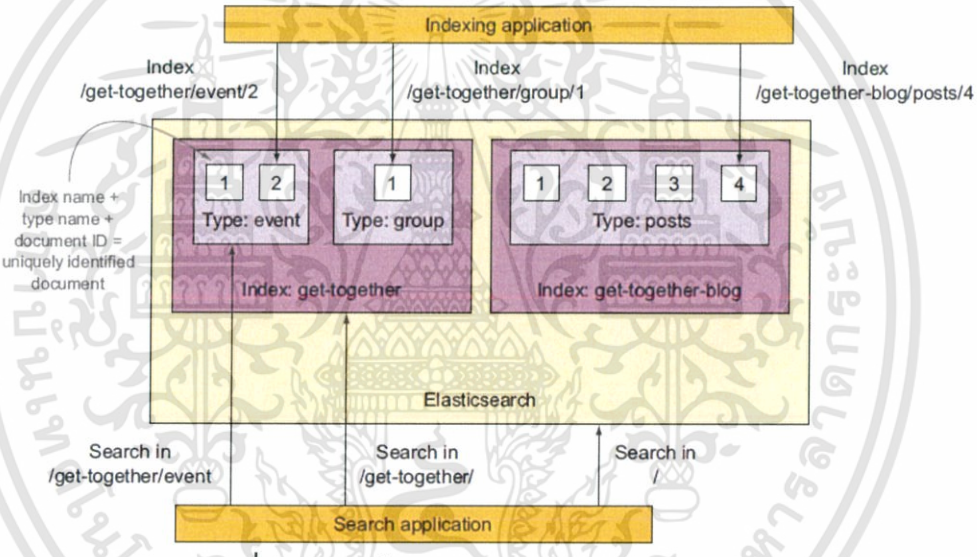
หากคำๆนั้น ไปปรากฏในเอกสารจำนวนน้อยจะได้คะแนนมาก แต่หากคำใดที่ไปปรากฏอยู่ในเอกสารหลายฉบับจะได้คะแนนน้อย และระบบจะให้ความสำคัญกับคำนั้นน้อย ตัวอย่างเช่น “is”, “the” ซึ่งเป็นคำเชื่อมและสามารถปรากฏได้ในเอกสารทุกฉบับ

2.4.3. สถาปัตยกรรม หรือ Architecture ของ Elasticearch

สถาปัตยกรรมแบ่งเป็นสองระดับคือ ระดับ Logical และระดับ Physical

2.4.3.1. Logical

เป็นส่วนที่ใช้สำหรับการค้นหา เหมือนกับฐานข้อมูลโดย Elasticsearchจะเรียกส่วนนี้ว่า Index



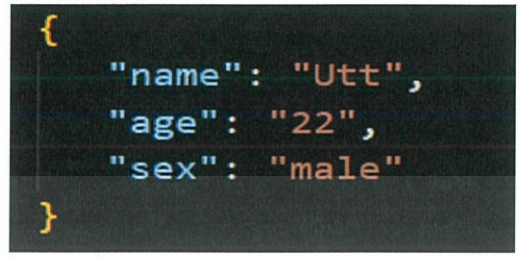
ภาพที่ 2.4 แผนผัง Logical ของ Elasticsearch

จากภาพข้างต้นส่วน Logical ของ Elasticsearch จะแบ่งออกเป็น 3 ส่วนคือ

1) Index ซึ่งเปรียบเหมือนฐานข้อมูล ดังนั้นจากภาพข้างต้นจะมี Index 2 ตัวคือ get-together กับ get-together-blog

2) Type เป็นประเภทสำหรับการจัดระเบียบใน Index จากภาพข้างต้นมี 3 type คือ event, group, posts

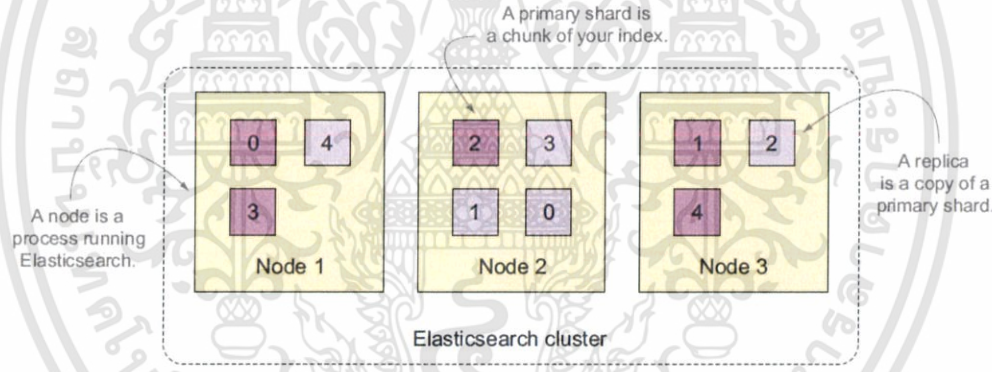
3) Document คือข้อมูลที่ถูกเก็บจริงๆในระบบและเป็นส่วนที่เล็กที่สุดของ Elasticsearch โดย Document จะถูกเก็บในรูปแบบ JSON



ภาพที่ 2.5 ตัวอย่างข้อมูลรูปแบบ JSON

2.4.3.2. Physical

เป็นส่วน Hardware ที่ Elasticsearch ใช้ในการเก็บข้อมูลจริงๆ มีองค์ประกอบ 4 อย่างคือ Node, Clusters, Shard, Replica



ภาพที่ 2.6 แผนผัง Physical ของ Elasticsearch

1) Node

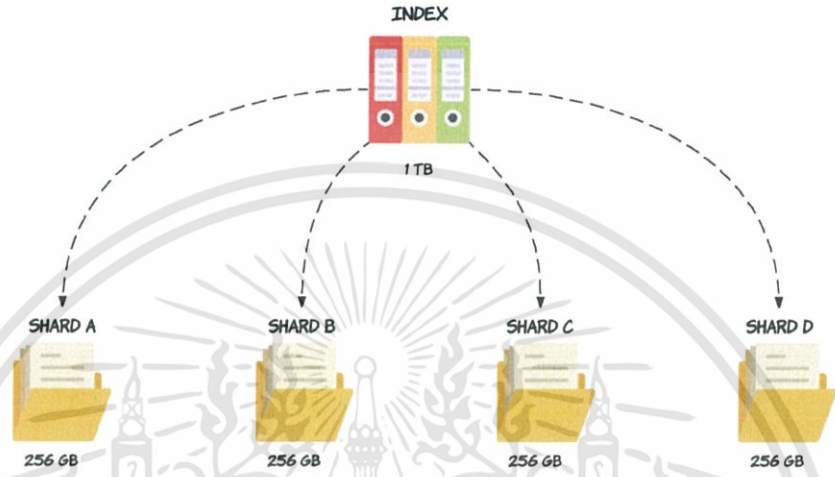
เป็นเครื่อง server ที่ใช้เก็บข้อมูล ในภาพข้างต้นจะมี 3 Node

2) Cluster

คือกลุ่มของ Node ที่อยู่ด้วยกัน ดังนั้น ภาพด้านบนคือภาพของ Cluster ที่มี 3 Node

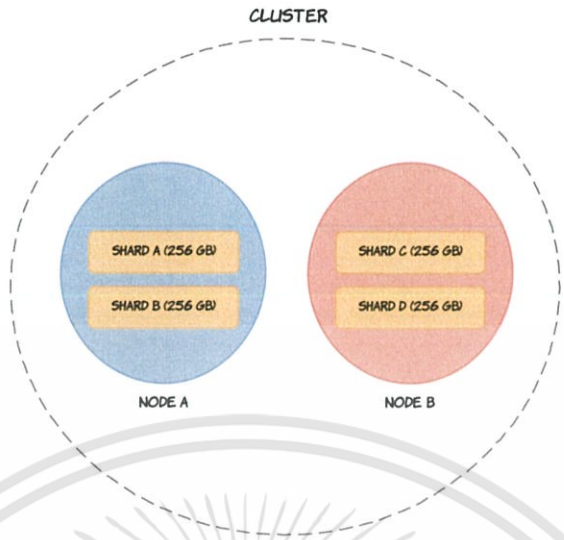
3) Shard

เป็นการแบ่งข้อมูลใน Index ออกเป็นส่วนๆ เพื่อเพิ่มประสิทธิภาพในการค้นหาและการเก็บข้อมูล ในกรณีที่ข้อมูลใน Index มีขนาดใหญ่มากๆ เช่น 1TB ขึ้นไป Elasticsearch จะทำการแบ่ง Index ออกเป็น Shard ดังภาพด้านล่าง



ภาพที่ 2.7 การแบ่ง Index ออกเป็น Shard

การแบ่ง Index ออกเป็น Shard มีข้อดีคือ Node ไม่ต้องรับภาระในการเก็บข้อมูลเยอะๆ ไว้เพียง Node เดียวโดย Elasticsearch จะกระจาย Shard ไปเก็บใน Node อื่นๆที่อยู่ใน Cluster

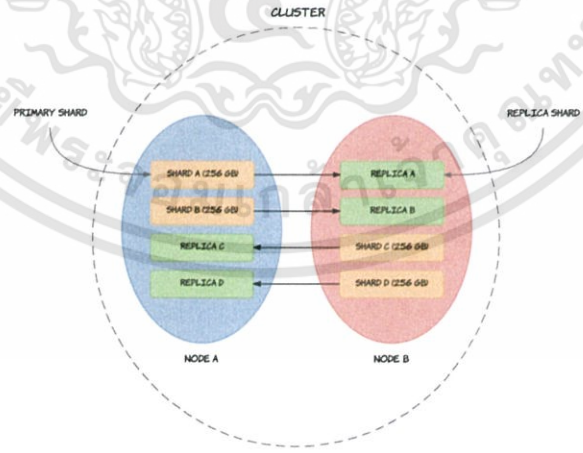


ภาพที่ 2.8 การกระจาย Shard ไปเก็บใน Node ต่างๆใน Cluster

นอกจากการทำ Sharding จะทำให้การเก็บข้อมูลมีประสิทธิภาพมากขึ้นแล้ว ยังทำให้การค้นหาทำได้อย่างรวดเร็วมากขึ้นด้วย เนื่องจากสามารถค้นหาข้อมูลใน 1 Index โดยใช้ 2 Node ค้นหาพร้อมกันได้

4) Replica

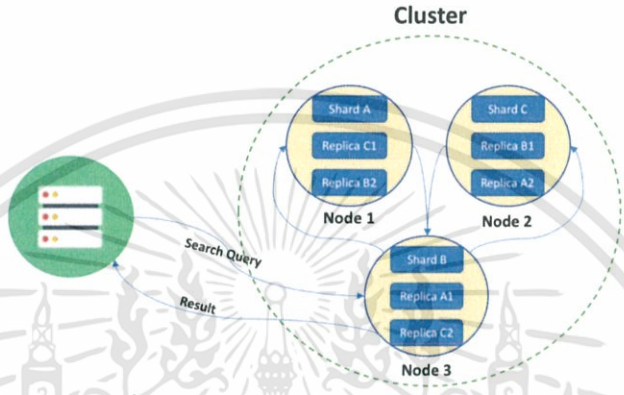
เพื่อให้ระบบมีความพร้อมในการใช้งานได้ตลอดเวลา Elasticsearch จึงมีการทำ Replica เพื่อทดแทนในกรณีที่ Primary Shard เสียหายหรือไม่สามารถเข้าถึงได้ โดย Replica จะมีข้อมูล เหมือน Primary Shard แต่จะถูกนำไปเก็บใน Node ที่ต่างกัน



ภาพที่ 2.9 การแบ่งเก็บ Replica

2.4.4. การค้นหาข้อมูล

เริ่มจาก Client ส่งคำค้นหา (Search query) เข้าไปที่ Elasticsearch จากนั้น Elasticsearch จะค้นหาข้อมูลตามคำค้นหาที่ได้รับใน Index นั้นๆ แต่เนื่องจาก Elasticsearch ได้แบ่งข้อมูลใน Index ออกเป็นส่วนๆที่เรียกว่า Shard กระจายออกไปอยู่ตาม Node ต่างๆใน Cluster ทำให้มีกระบวนการค้นหาดังนี้



ภาพที่ 2.10 การค้นหาของ Elasticsearch

โดยคำค้นหา (Search Query) จะเข้าไปที่ Node ใด Node หนึ่งใน Cluster และ Node ที่ได้รับคำค้นหาจะกลายเป็น Node ประสานงานซึ่งจะมีข้อมูลทั้งหมดว่า Node ไหนมี Index ที่ต้องการค้นหาอยู่บ้าง จากนั้น Node ประสานงานจะกระจายคำค้นหาไปยัง Node อื่นๆเพื่อนำไปค้นหาใน Node ของตัวเอง แล้วจึงนำผลลัพธ์ที่ได้ส่งไปรวมที่ Node ประสานงาน Node ประสานงานก็จะนำผลลัพธ์ที่ได้ส่งกลับไปให้ Client

2.4.5. การใช้งาน Elasticsearch

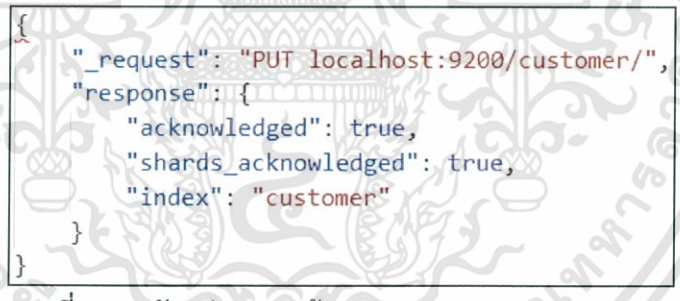
2.4.5.1. การติดตั้ง

การติดตั้ง Elasticsearch สามารถทำได้โดยง่าย ด้วยการทำตามคำแนะนำในเว็บไซต์ Elasticsearch (www.elastic.co) ในระหว่างการติดตั้ง สามารถเลือกโฟลเดอร์สำหรับการจัดเก็บข้อมูล และสำหรับไฟล์ configuration ได้ และยังสามารถเลือก Port ที่จะให้ Elasticsearch ใช้งานได้อีกด้วยหลังจากการติดตั้งลองทำการติดต่อกับ Elasticsearch ด้วย RESTful API ดังนี้

ภาพที่ 2.11 ตัวอย่างการเรียกดูสถานะของ Elasticsearch

2.4.5.2. การสร้าง Index

การสร้าง Index จะใช้คำสั่ง PUT ดังตัวอย่างการสร้าง Index ชื่อ customer โดยใช้ HTTP method “PUT localhost:9200/customer” ดังต่อไปนี้



ภาพที่ 2.12 ตัวอย่างการสร้าง Index ของ Elasticsearch

และสามารถเรียกดู Index ทั้งหมดที่อยู่ Elasticsearch ได้โดยใช้ HTTP method “GET localhost:9200/_cat/indices” ดังต่อไปนี้

ภาพที่ 2.13 ตัวอย่างการเรียกดู Index ทั้งหมดใน Elasticsearch

2.4.5.3. การเพิ่ม Document

ตัวอย่างการเพิ่ม Document เข้าไปใน Index “customer” โดยระบุให้มี ID เป็น 1 ด้วยการ POST โดยใช้ HTTP method “POST localhost:9200/customer /_doc/1” และตามด้วย Body “{ “name”: Utt Assoratgoon }” ดังต่อไปนี้

ภาพที่ 2.14 ตัวอย่างการเพิ่ม Document เข้าไปยัง Index ของ Elasticsearch

2.4.5.4. การอัพเดต document

ตัวอย่างการเพิ่ม key “age” ใน Document ที่ทำการเพิ่มเข้าไปก่อนหน้าด้วยการระบุ ID ของ Document นั้น โดยใช้ HTTP method “POST localhost:9200 /customer/_doc/1/update” และตามด้วย Body “{ “doc”: { “age”: 22 }}” ดังต่อไปนี้

ภาพที่ 2.15 ตัวอย่างการอัพเดต Document ของ Elasticsearch

2.4.5.5. การค้นหาข้อมูลใน Elasticsearch

ตัวอย่างการค้นหา Document ทั้งหมดใน Index “customer” โดยใช้ HTTP method “GET localhost:9200/customer/_search”และตามด้วย Body “{ “quary”: { “match_all”: { } } }”

```
{
    "_request": "GET localhost:9200/customer/_search",
    "body": {
        "query": { "match_all": {} }
    },
    "response": {
        "took": 5,
        "timed_out": false,
        "_shards": {
            "total": 5,
            "successful": 5,
            "skipped": 0,
            "failed": 0
        },
        "hits": {
            "total": 1,
            "max_score": 1,
            "hits": [
                {
                    "_index": "customer",
                    "_type": "_doc",
                    "_id": "1",
                    "_score": 1,
                    "_source": {
                        "name": "Utt Assoratgoon",
                        "age": 22
                    }
                }
            ]
        }
    }
}
```


ภาพที่ 2.16 ตัวอย่างการเรียกดูข้อมูลทั้งหมดใน Index ของ Elasticsearch

2.5. Logstash

คือเครื่องมือที่ช่วยในการนำเอาข้อมูลเข้าสู่ Elasticsearch โดยการทำงานของ Logstash จะทำงานตามไฟล์ Configuration ที่กำหนดไว้

ไฟล์ Configuration ของ Logstash ประกอบด้วย 3 ส่วนคือ Input, Filter และ Output

2.5.1. Input

เป็นส่วนที่ระบุว่าจะให้ Logstash นำข้อมูลจาก Folder ไหนและไฟล์ชนิดใดเข้าสู่ Elasticsearch โดยหากมีข้อมูลใหม่เข้ามาใน Folder ที่ระบุ Logstash จะนำข้อมูลนั้นเข้าสู่ Elasticsearch ให้โดยอัตโนมัติ

2.5.2. Filter

คือส่วนที่ใช้จัดการกับข้อมูลก่อนที่จะนำเข้าสู่ Elasticsearch อย่างเช่น เปลี่ยนชื่อ Field, ระบุชนิดของข้อมูล, เปลี่ยนรูปแบบ (Format) เวลา ฯลฯ

2.5.3. Output

ใช้เพื่อบอกให้ Logstash รู้ว่าต้องนำข้อมูลเข้าสู่ Index ไหนใน Elasticsearch

2.6. Python

ภาพที่ 2.17 Logo ของ Python

Python เป็นภาษา Dynamic Object-Oriented Programming ที่ใช้กันอย่างกว้างขวาง ถูกพัฒนาขึ้นโดย Guido van Rossum ที่ Centrum Wiskunde & Informatica (CWI) ในประเทศเนเธอร์แลนด์และในปัจจุบันดูแลโดยมูลนิธิซอฟต์แวร์ไพธอน (Python Software Foundation : PSF) โดยเวอร์ชันล่าสุดคือ Python 3.5 และ Python 2.7 ซึ่ง Python ถูกพัฒนาขึ้นมาจากภาษา C มีตัวแปรภาษาคือไพธอนอิเทอร์พรีเตอร์ (Python Interpreter) การประมวลผลจะทำในรูปแบบ Interpreter คือจะประมวลผลไปทีละบรรทัดและปฏิบัติตามคำสั่งที่ได้รับ

2.6.1. คุณลักษณะเด่นของภาษาไพธอน

2.6.1.1. พัฒนาให้ไม่ยึดติดกับแพลตฟอร์ม กล่าวคือ สามารถทำงานได้หลายระบบ เช่น ระบบ Unix, Linux, Windows เป็นต้น

2.6.1.2. ไวยากรณ์ของภาษา Python ถูกออกแบบมาเพื่อให้เข้าใจได้ง่าย ไม่ซับซ้อน โดยจะใช้คำในภาษาอังกฤษมาเป็นคำสั่ง

2.6.1.3. เป็นซอฟต์แวร์ที่เปิดเผยรหัสต้นฉบับ (Open Source) ซึ่งทำให้สามารถพัฒนาโปรแกรมได้โดยไม่เสียค่าใช้จ่ายใดๆ

2.6.1.4. สนับสนุนแนวคิดแบบเชิงวัตถุ หรือ Object Oriented Programming (OOP)

2.6.1.5. รองรับการเขียนส่วนติดต่อกับผู้ใช้ (User Interface)

2.6.1.6. เป็น Dynamic typing คือ สามารถเปลี่ยนชนิดข้อมูลได้ง่ายและสะดวก

2.6.1.7. มี Buil-in Object Types คือ โครงสร้างของข้อมูลที่สามารถใช้ได้ใน Python ประกอบไปด้วย ลิสต์ (List), ดิกชันนารี (Dictionary), สตริง (String) ที่ง่ายต่อการใช้งานและมีประสิทธิภาพสูง

2.6.1.8. มีโมดูล (Module) สำหรับจัดการ Regular Expresion

2.6.1.9. มีโมดูลที่สร้างขึ้นจากนักพัฒนาที่สนับสนุนมากมาย

2.6.1.10. สามารถจัดการหน่วยความจาอย่างอัตโนมัติและมีประสิทธิภาพ

2.6.1.11. มีไลบรารีสนับสนุนด้านปัญญาประดิษฐ์

2.6.2. ชนิดของตัวแปร

ชนิดของตัวแปรในภาษาไพธอน แบ่งออกเป็น 5 ชนิดใหญ่ ๆ ได้แก่ number, string, list, tuple, dictionary ซึ่งเป็นตัวแปรทั่ว ๆ ไปแต่ภาษาไพธอนยอมให้มีตัวแปร list, tuple, dictionary ที่ผสมกันได้ เรียกว่าชนิด complex ถ้าหากต้องการทราบว่าตัวแปรที่ประกาศใช้นั้นเป็นชนิดใด เราสามารถสอบถามชนิดตัวแปรได้จากคาสั่ง type(var)

2.6.3. ตัวอย่างไลบรารีของ Python

2.6.3.1. Pandas

ภาพที่ 2.18 Logo ของ Pandas

Pandas เป็นเครื่องมือที่ช่วยในการจัดการข้อมูล โดย Pandas จะนำข้อมูลที่ต้องการ ดำเนินงาน มาจัดอยู่ในรูปแบบตาราง หรือ Pandas Data Frame จากนั้นเราสามารถนำ Data Frame ที่ได้นี้ไปทำกระบวนการต่างๆตามที่เราต้องการได้อย่างง่ายดาย เช่น การเข้าถึงข้อมูลในแต่ละเซลล์, คำนวณค่าทางสถิติ, เปลี่ยนค่าของเซลล์ตามเงื่อนไขที่กำหนด ฯลฯ Pandas ยังมีความสามารถรองรับข้อมูลที่มาจากแหล่งข้อมูลที่หลากหลายได้อีกด้วย เช่น ข้อมูลที่มาจาก Python เองเช่น List, Dictionary หรือ Object จากไลบรารีอื่นเช่น Numpy และข้อมูลที่เป็นไฟล์ เช่น CSV, JSON หรือแม่งกระทั่ง Excel

ตัวอย่าง Pandas

```
import pandas as pd

data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'])
print(df)
```

ภาพที่ 2.19 ตัวอย่างการสร้าง DataFrame ของ Pandas

```
     Name  Age
0    Alex   10
1     Bob   12
2  Clarke   13
```

ภาพที่ 2.20 ตัวอย่าง DataFrame ของ Pandas

2.6.3.2. Bokeh

ภาพที่ 2.21 Logo ของ Bokeh

Bokeh เป็นไลบรารีสำหรับการแสดงผล (Visualization) ที่สามารถตอบสนอง (Interactive) กับผู้ใช้ได้บนเบราเซอร์ โดยจุดประสงค์หลักคือ เพื่อสร้างกราฟที่มีความหลากหลายและสวยงามโดยสามารถสร้างกราฟของข้อมูลมหาศาลหรือกระแสข้อมูล (Streaming data) ได้อย่างมีประสิทธิภาพ นอกจากจะสามารถสร้างกราฟที่ตอบสนองกับผู้ใช้ได้แล้ว Bokeh ยังสามารถช่วยให้เราสร้าง Dashboard และแอพพลิเคชันข้อมูล (Data application) ได้อย่างรวดเร็วและง่ายดาย และ

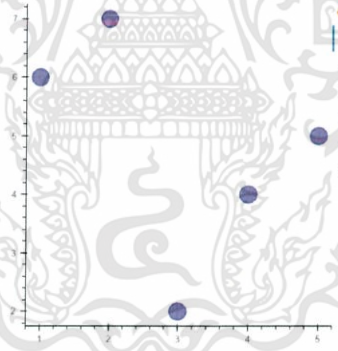
เนื่องจากกราฟที่สร้างจาก Bokeh สามารถตอบสนองกับผู้ใช้ได้จึงเป็นเครื่องมือที่เหมาะแก่การวิเคราะห์ข้อมูลเป็นอย่างมาก เพราะผู้ใช้จะเห็นความสัมพันธ์ของข้อมูลได้อย่างชัดเจน

ตัวอย่าง โค้ดBokeh

1) Scatter plot

```
from bokeh.plotting import figure, output_file, show

# output to static HTML file
output_file("line.html")

p = figure(plot_width=400, plot_height=400)

# add a circle renderer with a size, color, and alpha
p.circle([1, 2, 3, 4, 5], [6, 7, 2, 4, 5], size=20, color="navy", alpha=0.5)

# show the results
show(p)
```

ภาพที่ 2.22 ตัวอย่างการเขียนโค้ดเพื่อสร้างกราฟ Scatter ของ Bokeh



ภาพที่ 2.23 ตัวอย่างกราฟ Scatter ของ Bokeh

2) Lines graph

```
from bokeh.plotting import figure, output_file, show

output_file("line.html")

p = figure(plot_width=400, plot_height=400)

# add a line renderer
p.line([1, 2, 3, 4, 5], [6, 7, 2, 4, 5], line_width=2)

show(p)
```

ภาพที่ 2.24 ตัวอย่างการสร้างกราฟเส้นของ Bokeh

ภาพที่ 2.25 ตัวอย่างกราฟเส้นของ Bokeh

2.7. mRemote NG

ภาพที่ 2.26 Logo ของ mRemote NG

mRemoteNG เป็นโปรแกรมที่ใช้ในการเชื่อมต่อและควบคุมคอมพิวเตอร์ระยะไกล โดยสามารถ login เข้าไปควบคุมเครื่องคอมพิวเตอร์ที่ต้องการได้ โดยสามารถมองเห็นหน้าจอ เสมือนว่ากาลังนั่งอยู่หน้าคอมพิวเตอร์เครื่องนั้น สามารถติดตั้งและใช้งานได้เฉพาะระบบปฏิบัติการ Windows เท่านั้นและรองรับการคัดลอกไฟล์ระหว่างเครื่องคอมพิวเตอร์ที่ถูกควบคุมกับคอมพิวเตอร์ที่ใช้ควบคุม

2.8. Microsoft Visual Studio Code

ภาพที่ 2.27 Logo ของ Visual Studio Code

Visual Studio Code (VS Code) คือ Editor ที่ผู้พัฒนา Software นิยมใช้ในการเขียนโปรแกรม เนื่องจากสามารถรองรับการเขียนโปรแกรมได้หลายภาษาอย่างเช่น JavaScript, TypeScript, C++, C#,

Java, Python, PHPและ ฯลฯ โดย VS Code จะมีฟีเจอร์หลักอยู่ 4 อย่างที่ช่วยอำนวยความสะดวกให้กับผู้ใช้ในการพัฒนา Software

2.8.1. IntelliSense

IntelliSense คือฟีเจอร์ที่ช่วยผู้ใช้เรื่องไวยากรณ์ (Syntax) ไม่ว่าจะเป็นการเน้นข้อผิดพลาดหรือแก้ไขคำผิดอัตโนมัติ (autocomplete) และ IntelliSense จะแก้ไขคำที่ผิดพลาดโดยมีพื้นฐานมาจากชนิดของตัวแปร (Variable types), ฟังชั่นที่ใช้และ modules ที่ทำการ Import มา

2.8.2. Debugging

ผู้ใช้สามารถทำการ Debug โปรแกรมได้โดยการตั้ง Break point และสามารถเรียกดู Call stacks ได้อีกด้วย

2.8.3. Git commands built-in

เพื่อความสะดวกในการใช้ Version control อย่าง Git VS Code จึงมีฟีเจอร์ที่สามารถใช้งาน Git commands ได้ โดยอยู่ในรูปแบบ User Interface(UI) ซึ่งช่วยให้สามารถข้ามขั้นตอนในการพิมพ์ Git commands ที่ยุ่งยากและซับซ้อนได้

2.8.4. Extension

VS Code มีส่วนเสริม (Extension) มากมายเพื่อให้ผู้ใช้ได้ปรับแต่งตามความเหมาะสมของการใช้งาน ไม่ว่าจะเป็นส่วนเสริมที่ช่วยในการเขียนโปรแกรมในภาษานั้น ๆ รวมไปถึงธีมของโปรแกรมที่ต้องการและ การเชื่อมต่อกับบริการอื่น ๆ อย่างเช่น Docker เพื่อช่วยให้ผู้พัฒนาสามารถเขียนโปรแกรมได้รวดเร็วและมีประสิทธิภาพมากขึ้น

2.9. Git

ภาพที่ 2.28 Logo ของ Git

Git คือ Version Control ที่ผู้พัฒนา Software ใช้กันอย่างแผร่หลาย โดย Git มีความสามารถในการจัดเก็บความเปลี่ยนแปลงของ Source code ทำให้ผู้พัฒนาสามารถย้อนกลับไปเวอร์ชั่นเก่าได้ ในกรณีที่เวอร์ชั่นใหม่มีปัญหา และยังเป็นเครื่องมือที่ช่วยให้การพัฒนา Software แบบเป็นทีมทำได้สะดวกมากขึ้นด้วยเพราะ ผู้พัฒนาสามารถทำงานแยกกันได้โดยอิสระโดยที่โค้ดของแต่ละคนไม่มีผลกระทบซึ่งกันและกัน ประโยชน์ที่สำคัญอีกอย่างของ Git คือ ผู้พัฒนาสามารถลดความเสี่ยงที่โค้ดจะสูญหายในกรณีที่เครื่องคอมพิวเตอร์มีปัญหาหรือเกิดการลบโค้ดที่ทำงานอยู่ไปโดยอุบัติเหตุ เนื่องจากผู้พัฒนาสามารถดึงโค้ดที่กำลังพัฒนาอยู่ในเวอร์ชั่นล่าสุดได้จาก Git

2.10. CI/CD (Continuous integration and Continuous Deployment)

2.10.1. CI (Continuous integration)

คือวิธีปฏิบัติเพื่อให้ผู้พัฒนา Software แก้ไข source code ให้น้อยที่สุดเท่าที่จำเป็นแล้วทำการ check in source code เข้าสู่ Version Control บ่อยๆ โดยมีวัตถุประสงค์เพื่อให้มั่นใจว่าชิ้นส่วนของ Software ที่พัฒนาแยกกันจากหลายผู้พัฒนาสามารถรวมกันได้โดยไม่เกิดปัญหา ซึ่งอาจหมายรวมไปถึงการทดสอบในระดับเล็กของโปรแกรม (Unit Test) และครอบคลุมถึงการทำให้การทำงานดังกล่าวเป็นไปแบบอัตโนมัติอีกด้วย

2.10.2. CD (Continuous Deployment)

คือการนำ Software ที่ผ่านการทดสอบแล้วไป Deploy ในระดับ Production เพื่อให้ลูกค้าหรือผู้ใช้สามารถใช้งานได้อย่างอัตโนมัติ

2.11. GitLab CI/CD

ภาพที่ 2.29 Logo ของ Git

GitLab คือ เครื่องมือที่ใช้จัดการกับ Git Repository และมีความสามารถในการทำ CI/CD (Continuous integration and Continuous Deployment) ซึ่งข้อดีของการทำ CI/CD บน GitLab คือ งานที่ทำนั้นถูกเก็บไว้ใน Git Repository ของ GitLab อยู่แล้วทำให้ไม่ต้องตั้งค่าหรือติดตั้งเครื่องมืออื่นๆ

GitLab CI/CD ทำงานเป็น Pipeline โดยมีลำดับดังนี้ Build, Test, Deploy ซึ่งในแต่ละขั้นตอนของ Pipeline สามารถกำหนดได้ด้วย ไฟล์ชื่อ .gitlab-ci.yml ใน Git Repository

```
stages:
  - build
  - deploy

process1:
  stage: build
  tags:
    - deployed machine
  script:
    - task1
    - task2
  only:
    - master

process2:
  stage: deploy
  tags:
    - deployed machine
  script:
    - task3
    - task4
  only:
    - master
```

ภาพที่ 2.30 ตัวอย่างไฟล์ .gitlab-ci.yml

2.12. PM2

PM2 คือ เครื่องมือที่ช่วยในการจัดการ Process (Process Manager) โดยจะทำให้โปรแกรมที่ทำการ Deploy รันอยู่ใน Backgroud และสามารถดู Logs ของ Process ที่รันอยู่ได้ ซึ่งเป็นเครื่องมือที่นิยมให้กันมากในการทำงานระดับ Production

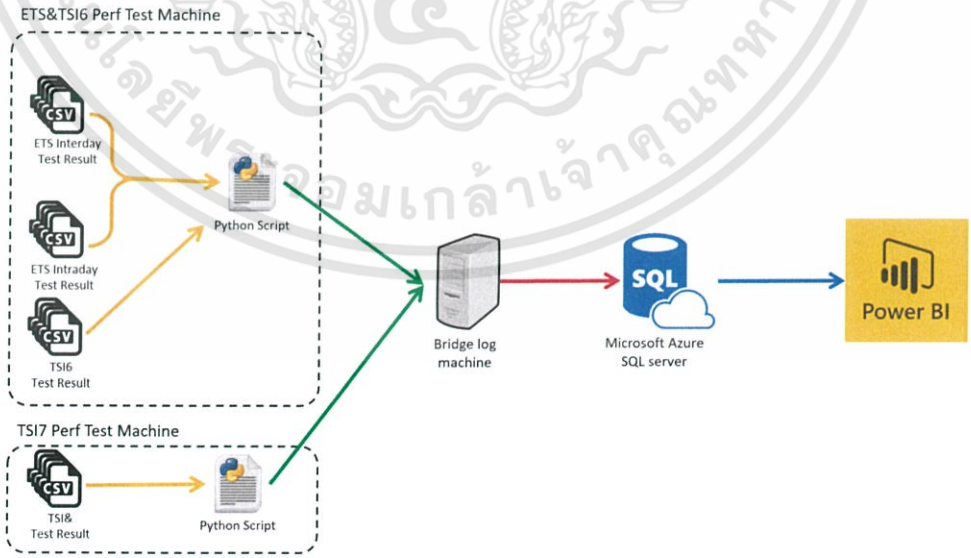
Name	mode	status	?	cpu	memory
Bokeh_App	fork	online	0	0%	50.2 MB
Bokeh_App_Dev	fork	online	0	0%	71.8 MB
PerfWatchDashboard	fork	online	0	0%	46.2 MB
PerfWatchDashboard_Dev	fork	online	0	0%	27.5 MB

ภาพที่ 2.31 ตัวอย่างการเรียกดูสถานะของ Process ของ PM2

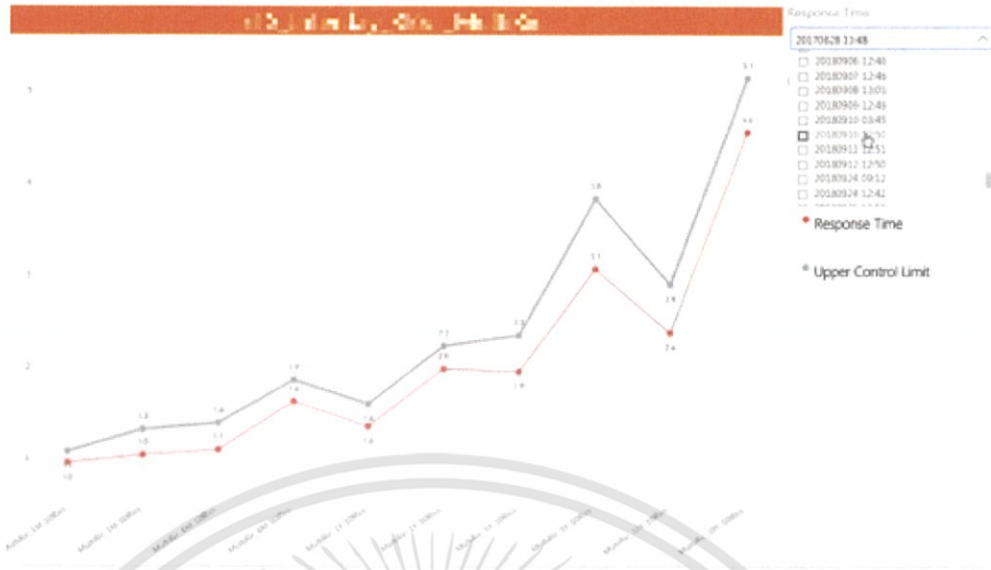
บทที่ 3
วิธีดำเนินการวิจัย

3.1. ขั้นตอนการศึกษาระบบเก่า

ขั้นแรก ผู้ที่มีหน้าที่ในการตรวจสอบประสิทธิภาพของระบบ จะนำผลที่ได้จากการทดสอบ ซึ่งเป็นการทดสอบที่จำลองการขอข้อมูลของ Client มาที่ระบบ โดยมีสิ่งที่สนใจคือ จำนวนข้อมูลที่ได้รับกลับมา เทียบกับจำนวนข้อมูลที่ขอไป และเวลาการตอบสนองของ Client ตั้งแต่เริ่มต้นส่งคำขอจนถึง Client ได้รับข้อมูล เข้าสู่ฐานข้อมูล SQL (SQL Database) ของ Microsoft Azure เพื่อที่จะนำไปแสดงผลบน Microsoft Power BI และทำการตรวจสอบโดยการสร้างกราฟเปรียบเทียบระหว่างค่าเฉลี่ยของผลการทดสอบในรอบปัจจุบัน กับค่าเฉลี่ยของผลการทดสอบของเดือนก่อนหน้า หากค่าเฉลี่ยของผลการทดสอบในรอบปัจจุบันมีค่ามากกว่าค่าเฉลี่ยของผลการทดสอบของเดือนก่อนหน้า จะถือว่าผลการทดสอบในรอบนั้นมีความผิดปกติ และเริ่มทำการตรวจสอบ โดยการนำไฟล์ผลการทดสอบ และ ไฟล์ Logs ที่เกิดจากการทำงานของระบบในขณะทำการทดสอบมาวิเคราะห์หาสาเหตุ ว่าองค์ประกอบใดในระบบ ที่ทำให้ ผลการทดสอบมีความผิดปกติ เพื่อที่จะทำการแก้ไขให้ระบบมีประสิทธิภาพสูงขึ้นต่อไป แต่เนื่องจากระบบที่ทำการทดสอบนั้นมีองค์ประกอบภายในจำนวนมากทำให้การเข้าไปเก็บรวบรวม Logs ที่เกิดขึ้นในขณะที่ทำการทดสอบ ของแต่ละองค์ประกอบนั้นเป็นกระบวนการที่สิ้นเปลืองเวลาเป็นอย่างมาก

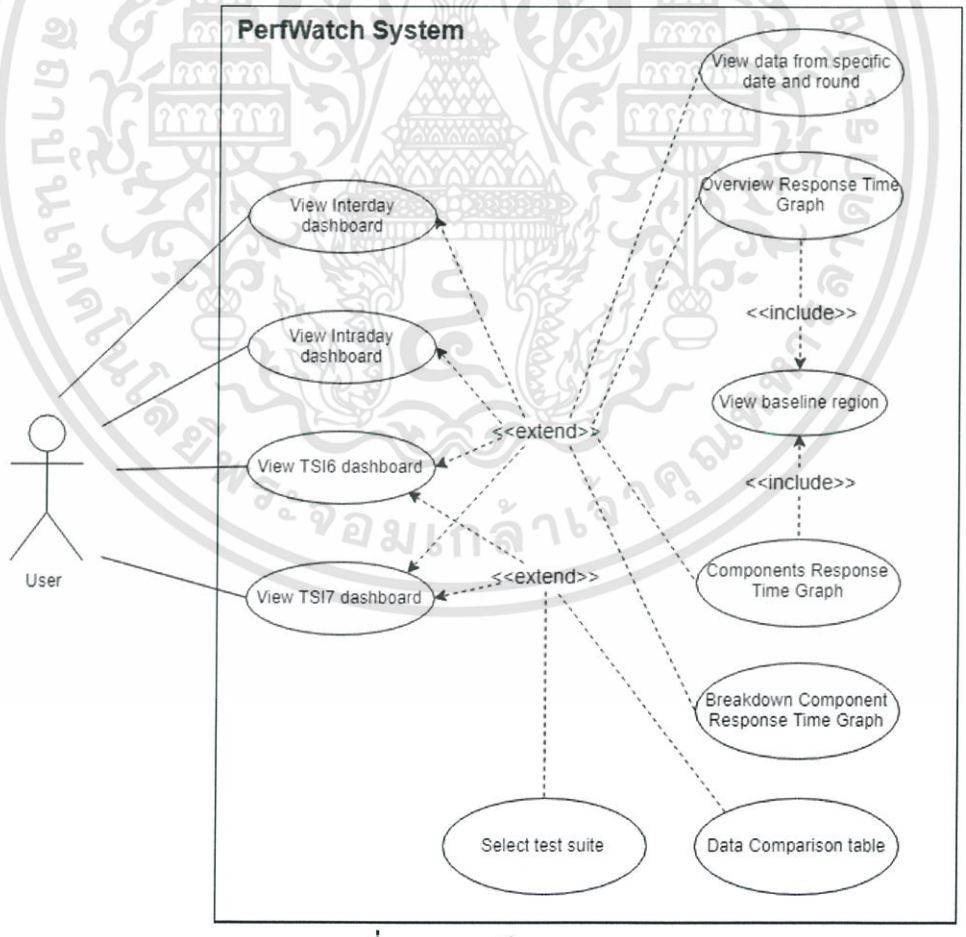


ภาพที่ 3.1 การเก็บและแสดงผลของการทดสอบรูปแบบเดิม



ภาพที่ 3.2 การแสดงผลของการทดสอบในรูปแบบเดิม

3.2. วิเคราะห์ Use case ของระบบ



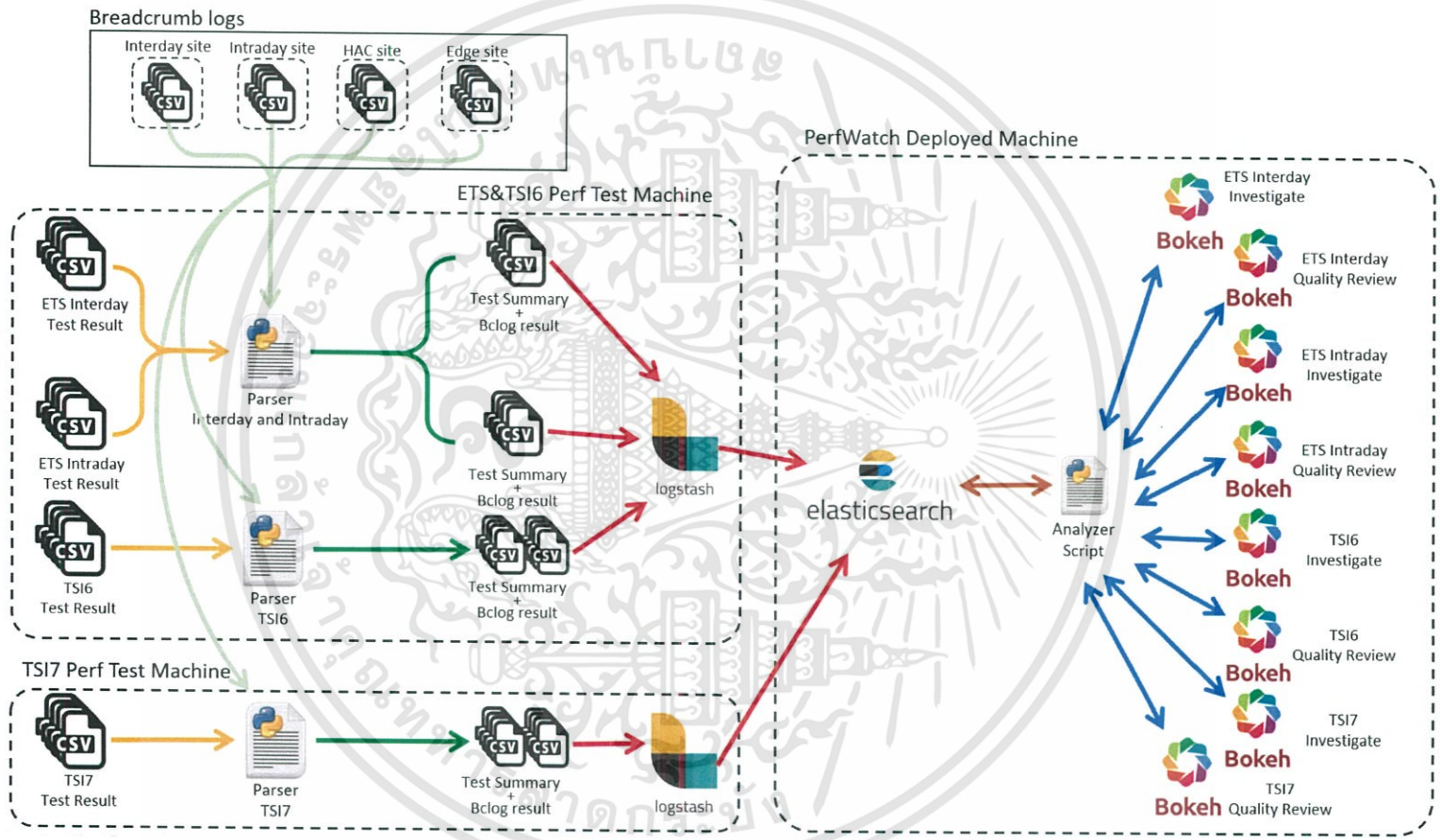
ภาพที่ 3.3 แผนผัง Use case

จากแผนผัง Use case ภาพที่ 3.2 มี Use case ทั้งหมด 4 Use cases โดยผลการทดสอบของแต่ละ Use case นั้นแตกต่างกันทำให้ผู้ตรวจสอบประสิทธิภาพของระบบต้องการการแสดงผล (Visualization) ที่แตกต่างกันทั้งด้านข้อมูลที่นำมาแสดง และรูปแบบการแสดงผล นอกจากนี้ ผู้ตรวจสอบประสิทธิภาพของระบบ ต้องการที่จะเลือกแสดงผลตาม วันของการทดสอบ, รอบของการทดสอบ, Test suite และ Test case ที่สนใจได้อีกด้วย

ข้อมูลที่นำมาใช้ในการแสดงผลได้มาจากผลของการทดสอบ ซึ่งขึ้นอยู่กับลักษณะการทดสอบของแต่ละ Use case โดยการทดสอบของแต่ละ Use case จะแบ่งเป็น Test suite ซึ่งก็คือกลุ่มของ Test case ที่มีลักษณะการทดสอบคล้ายคลึงกัน โดย Test case แต่ละ Test case มีพารามิเตอร์ในการทดสอบที่แตกต่างกันขึ้นอยู่กับ Business requirement ที่ได้จากลูกค้า และผลการทดสอบในแต่ละ Test case จะประกอบไปด้วย ระยะเวลาที่ใช้ทั้งหมดตั้งแต่ Client เริ่มส่งคำขอจน Client ได้รับข้อมูล, ระยะเวลาที่ใช้ในองค์ประกอบหลักของระบบ, ระยะเวลาที่ใช้ในองค์ประกอบย่อยซึ่งเป็นส่วนที่อยู่ภายในองค์ประกอบหลัก และข้อมูลพารามิเตอร์ต่างๆที่ใช้ทดสอบใน Test case นั้นๆ

เนื่องจากทีม Time Series มีการพัฒนาระบบอยู่อย่างต่อเนื่องทำให้ Test case ที่ใช้ในการทดสอบและผลของการทดสอบอาจมีการเปลี่ยนแปลงหรือเพิ่มเติมได้ตลอดเวลา ผู้ตรวจสอบประสิทธิภาพของระบบจึงมีความต้องการให้ระบบ PerfWatch สามารถรองรับกับความเปลี่ยนแปลงเหล่านั้นได้อีกด้วย เพื่อที่ในอนาคตจะสามารถนำ PerfWatch ไปประยุกต์ใช้ในการวิเคราะห์ประสิทธิภาพของระบบอื่นๆต่อไปได้

3.3. โครงสร้างของระบบ



ภาพที่ 3.4 โครงสร้างของระบบ PerfWatch

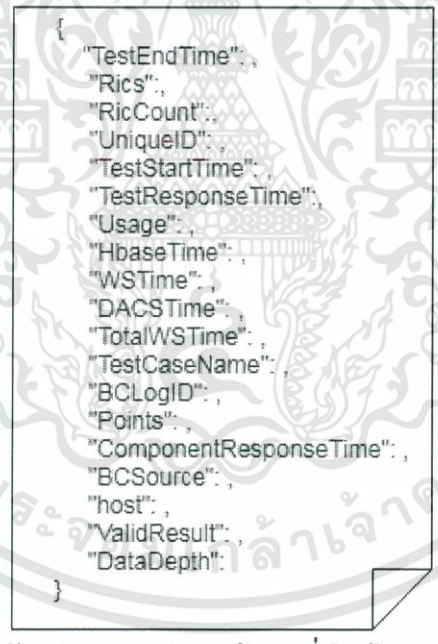
จากภาพที่ 3.3 สามารถอธิบายได้ดังนี้

3.3.1. รวบรวมและคำนวณผลการทดสอบ

รวบรวมผลของการทดสอบ และ Logs ที่เกิดจากการทำงานของระบบภายในเครื่องที่ทำการทดสอบ และองค์ประกอบทั้งหมดในระบบ โดยใช้ Script ไพธอน ซึ่งจะรวบรวมผลและคำนวนระยะเวลาที่ใช้ในการทำงานของแต่ละองค์ประกอบในระบบ ทั้งองค์ประกอบหลักและองค์ประกอบย่อย แต่เนื่องจากลักษณะของผลการทดสอบในแต่ละ Use case มีความแตกต่างกัน ทำให้มีความจำเป็นที่จะต้องสร้าง Script ไพธอนขึ้นมารองรับ ตามลักษณะของผลการทดสอบในแต่ละ Use case (ภาพที่ 3.5)

3.3.2. การนำผลลัพธ์เข้าสู่ Elasticsearch

นำผลลัพธ์ที่ได้จาก Script ไพธอนเข้าสู่ Elasticsearch โดยใช้ Logstash ซึ่งตั้งค่าให้นำข้อมูลที่อยู่ในโฟรเดอร์ผลลัพธ์เข้าสู่ Elasticsearch โดยการนำเข้าข้อมูล จะแยก Index ตาม Test suite ของผลลัพธ์นั้นๆ



ภาพที่ 3.5 ตัวอย่าง Field ของข้อมูลที่นำเข้าสู่ Elasticsearch

3.3.3. วิเคราะห์ข้อมูล

นำข้อมูลที่อยู่ใน Elasticsearch มาทำการวิเคราะห์ เพื่อหาว่ามีการทดสอบครั้งใดและ Test case ใดที่มีความผิดปกติบ้างโดยใช้วิธีทางสถิติ โดยผู้ใช้สามารถกำหนดวันที่จะทำการตรวจสอบและจำนวนของรอบการทำงาน (Sprint) ย้อนหลังที่จะนำมาวิเคราะห์หาขอบเขตของข้อมูลที่สามารถยอมรับได้ (Baseline) ผ่าน Dashboard จากนั้น Script ที่ใช้ในการวิเคราะห์ข้อมูล จะทำ

การเรียกข้อมูลของวันที่ผู้ใช้ต้องการตรวจสอบ และข้อมูลของวันในรอบการทำงานย้อนหลัง ที่ผู้ใช้เลือกไว้ที่หน้า Dashboard จาก Elasticsearch โดยการเลือกวันในรอบการทำงานย้อนหลังนั้น จะเลือกเฉพาะวันในช่วง Freeze period เท่านั้น

ช่วง Freeze period คือช่วงของวันที่ระบบจะมีความเสถียรมากที่สุด เนื่องจากจะไม่มีการเพิ่ม feature หรือการแก้ไขของระบบเลย ทำให้ผลการทดสอบในช่วงนี้จะมีความน่าเชื่อถือมากที่สุด ในหนึ่งรอบการทำงานจะมีระยะเวลาทั้งหมด 14 วัน ช่วง Freeze period จะมีสามวัน คือ จันทร์ อังคาร พุธ ของอาทิตย์แรกในรอบการทำงาน หรือบางครั้งอาจจะใช้ผลการทดสอบของวันศุกร์ในรอบการทำงานก่อนหน้าด้วย ตัวอย่างเช่น ผู้ใช้เลือกจำนวนของรอบการทำงานย้อนหลังเพื่อจะนำมาวิเคราะห์หาขอบเขตของข้อมูลที่สามารถยอมรับได้เป็นจำนวน 2 รอบการทำงาน แสดงว่าจะมีจำนวน Freeze period ที่จะนำมาใช้วิเคราะห์ เพียง 6 วันเท่านั้น

หลังจากได้ข้อมูลของวันที่จะนำมาวิเคราะห์หาขอบเขตของข้อมูลที่สามารถยอมรับได้แล้ว นำข้อมูลเหล่านั้นมาวิเคราะห์ โดยปัจจุบันการคำนวณหาขอบเขตของข้อมูลที่สามารถยอมรับได้จะใช้วิธีเดียวกับการหา Outlier ของ Boxplot นั้นก็คือใช้ Upper Whisker และ Lower Whisker เป็นขอบเขตที่สามารถยอมรับได้ เมื่อทำการวิเคราะห์เสร็จแล้ว จะได้ขอบเขตที่สามารถยอมรับได้ของผลการทดสอบในแต่ละ Test case จากนั้นนำข้อมูลของวันที่ต้องการตรวจสอบ ซึ่งเรียกมาจาก Elasticsearch มาเปรียบเทียบกับขอบเขตที่สามารถยอมรับได้ ที่ได้จากการวิเคราะห์ก่อนหน้านี้ โดยนำผลการทดสอบทุกรอบในวันที่ต้องการตรวจสอบ มาเทียบกับขอบเขตที่สามารถยอมรับได้ตาม Test case ของผลการทดสอบนั้น หากผลการทดสอบรอบใดมีค่ามากกว่า หรือน้อยกว่าขอบเขตที่สามารถยอมรับได้ จะถือว่าผลการทดสอบในรอบนั้นมีความผิดปกติ (Anomaly)

3.3.4. การแสดงผล (Visualization)

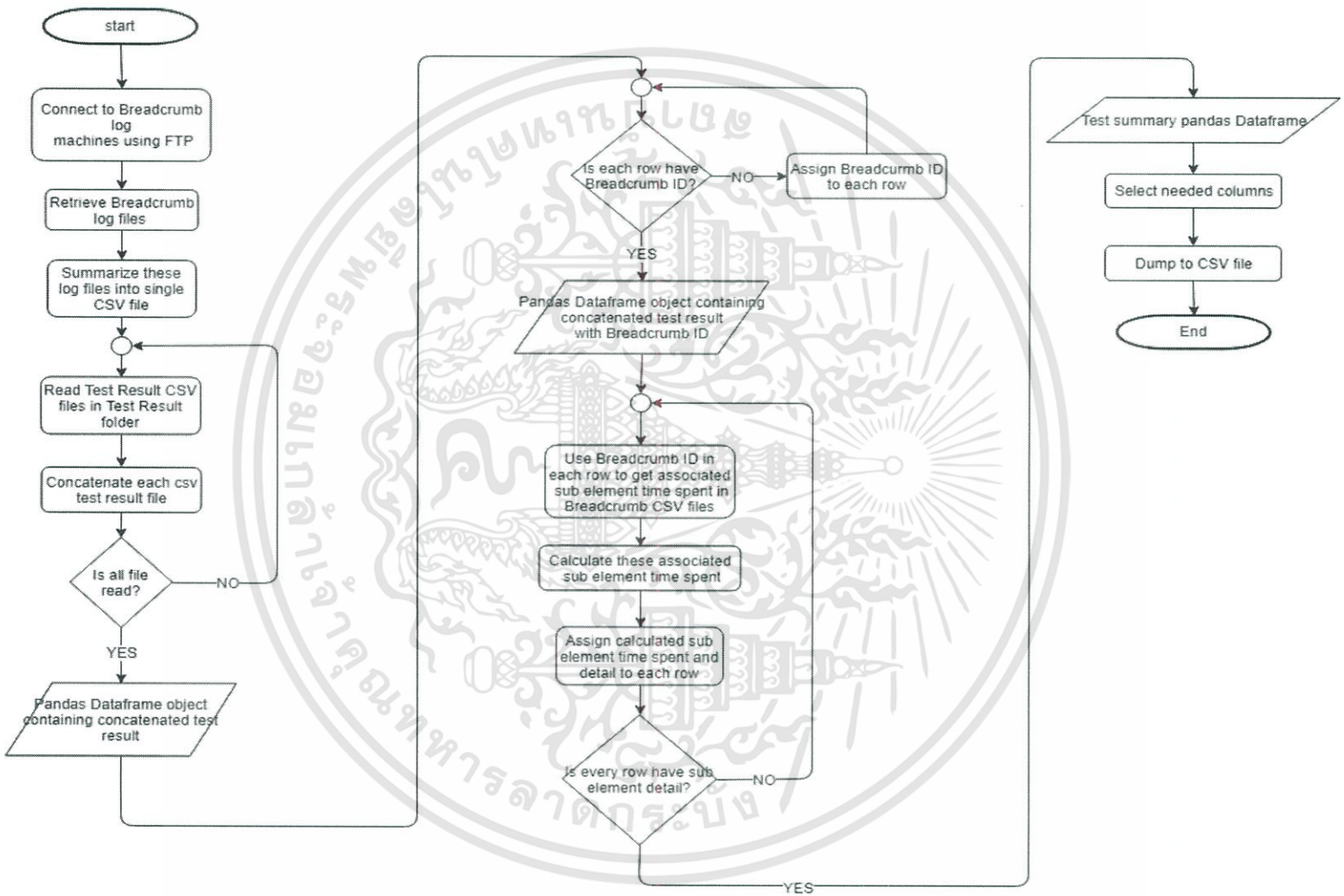
การแสดงผลจะทำในลักษณะ Dashboard เพื่อให้ผู้ใช้สามารถเลือกดูกราฟและข้อมูลที่ต้องการได้อย่างสะดวก โดยแต่ละ Use case จะมี Dashboard 2 ประเภทดังนี้

3.3.4.1. Investigated Dashboard

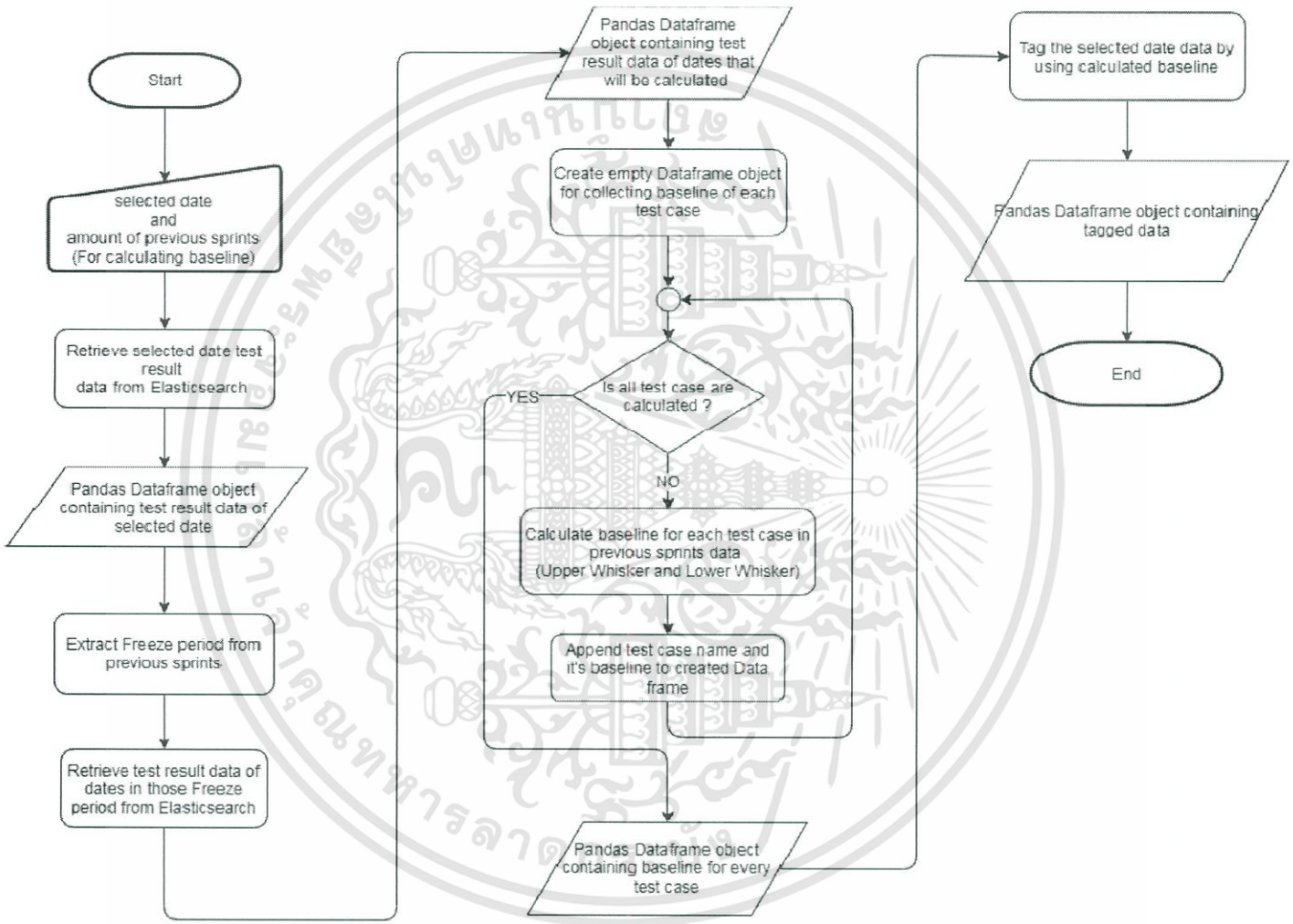
Investigated Dashboard คือ Dashboard สำหรับผู้ตรวจสอบประสิทธิภาพของระบบ ใช้ในการตรวจสอบผลการทดสอบ เพื่อหาความผิดปกติในระบบ โดยจะแสดงผลการทดสอบตามวันและรอบของการทดสอบที่เลือก ซึ่งสิ่งที่ทำการแสดงผลนั้น ประกอบด้วยเวลาในการตอบสนองทั้งหมดของระบบ เวลาในการตอบสนองขององค์ประกอบหลัก และเวลาในการตอบสนองขององค์ประกอบย่อย นอกจากนี้ในบาง Dashboard อาจมีการแสดงผลนอกเหนือจากที่กล่าวมา ขึ้นอยู่กับความต้องการของผู้ตรวจสอบประสิทธิภาพของระบบ เนื่องจากผลการทดสอบในแต่ละ Use case มีลักษณะที่ต่างกัน จึงอาจจะมีสิ่งที่ต้องพิจารณาต่างกัน

3.3.4.2. Quality Review Dashboard

Quality Review Dashboard คือ Dashborad ที่นำเสนอภาพรวมของผลการทดสอบของทั้งรอบการทำงาน โดยจะใช้ค่ากลางทางคณิตศาสตร์อย่างเช่น ค่าเฉลี่ยและค่ามัธยฐานของข้อมูลในช่วง Freeze period ในรอบการทำงานที่เลือก เพื่อใช้ในการตรวจสอบคุณภาพของระบบ (Quality Review) ในแต่ละรอบการทำงาน



ภาพที่ 3.6 Flowchart การทำงานของ script ที่ใช้ในการรวบรวมข้อมูล

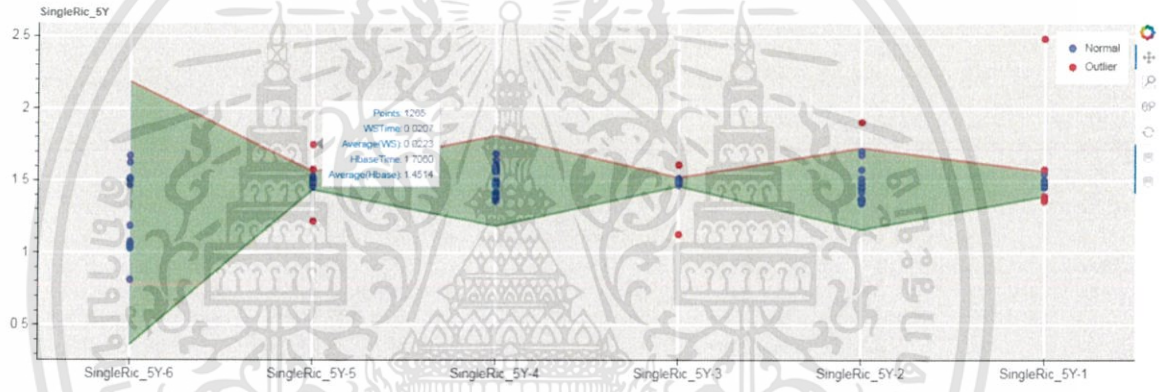


ภาพที่ 3.7 Flowchart การทำงานของ script ที่ใช้ในการวิเคราะห์ข้อมูล

3.4. รูปแบบการแสดงผล

3.4.1. Overview Response Time Graph

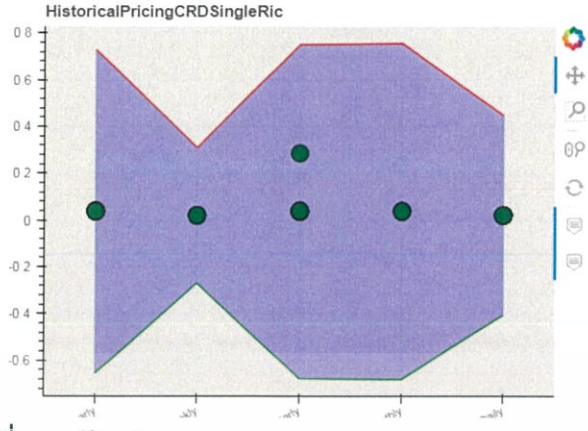
คือกราฟที่แสดงระยะเวลาที่ใช้ทั้งหมดตั้งแต่ Client เริ่มส่งคำขอ จน Client ได้รับข้อมูลของแต่ละ Test case กราฟนี้จะสามารถบอกได้ว่าการทดสอบครั้งใดบ้างที่ใช้เวลามากหรือน้อยผิดปรกติ โดยมีเส้นขอบเขตที่สามารถยอมรับได้ (Baseline region) ซึ่งได้จากการนำผลทดสอบของรอบการทดสอบก่อนๆ มาใช้การคำนวณทางสถิติมาเป็นสิ่งที่ใช้พิจารณา โดยหากการทดสอบครั้งใดที่มีเวลาในการตอบสนองอยู่นอกขอบเขตที่สามารถยอมรับได้ จะมีสีแดงและแสดงว่าการทดสอบครั้งนั้นมีความผิดปกติ แต่ในทางกลับกันหากการทดสอบครั้งใดมีเวลาในการตอบสนองอยู่ภายในขอบเขตที่สามารถยอมรับได้จะมีสีฟ้าและแสดงว่าการทดสอบครั้งนั้นมีไม่มีความผิดปกติ



ภาพที่ 3.8 ตัวอย่าง Overview Response Time Graph

3.4.2. Component Response Time Graph

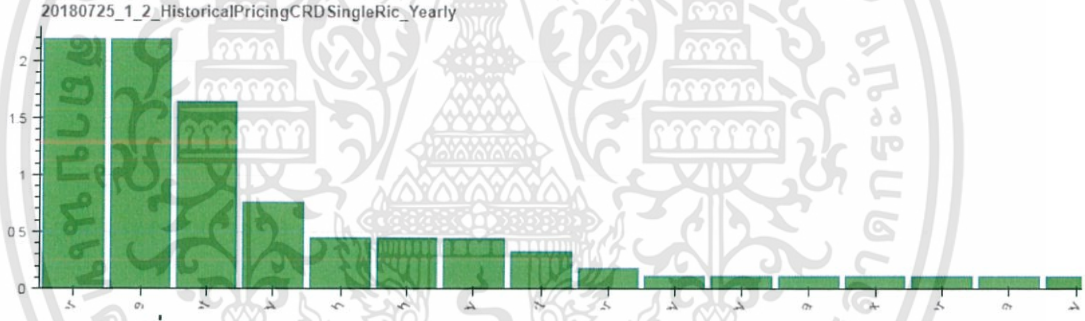
การแสดงผลจะเหมือนกับ Overview Response Time Graph เพียงแต่ข้อมูลที่นำมาแสดงจะเป็นระยะเวลาที่ใช้ในองค์ประกอบหลักของระบบแทนที่จะเป็นระยะเวลาการทำงานทั้งหมดของระบบ



ภาพที่ 3.9 ตัวอย่าง Component Response Time Graph

3.4.3. Breakdown Component Response Time Graph

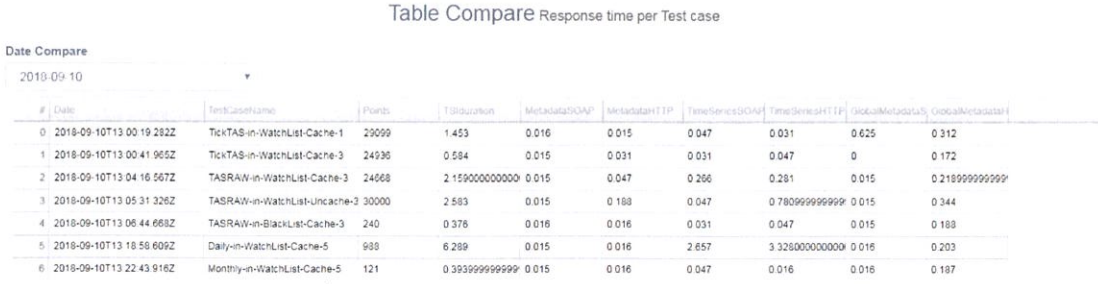
คือกราฟที่แสดงระยะเวลาที่ใช้ในองค์ประกอบย่อย ซึ่งอยู่ภายในองค์ประกอบหลัก โดยจะแสดงผลออกมาในรูปแบบแผนภูมิแท่ง (Histogram)



ภาพที่ 3.10 ตัวอย่าง Breakdown Component Response Time Graph

3.4.4. Data Comparison Table

คือตารางที่ใช้เปรียบเทียบผลการทดสอบของสองวัน โดยสามารถเลือกวันและ Test case ที่ต้องการเปรียบเทียบได้



#	Date	TestCaseName	Points	TSIduration	MetadataSOAP	MetadataHTTP	TimeSeriesISOAP	TimeSeriesHTTP	GlobalMetadataS	GlobalMetadataH
0	2018-09-10T13:00:19.282Z	TickTAS-in-WatchList-Cache-1	29099	1.453	0.016	0.015	0.047	0.031	0.625	0.312
1	2018-09-10T13:00:41.965Z	TickTAS-in-WatchList-Cache-3	24936	0.584	0.015	0.031	0.031	0.047	0	0.172
2	2018-09-10T13:04:16.567Z	TASRAW-in-WatchList-Cache-3	24668	2.159000000000(	0.015	0.047	0.266	0.281	0.015	0.218999999999
3	2018-09-10T13:05:31.326Z	TASRAW-in-WatchList-Uncache-3	30000	2.583	0.015	0.188	0.047	0.780999999999	0.015	0.344
4	2018-09-10T13:06:44.668Z	TASRAW-in-BlackList-Cache-3	240	0.376	0.016	0.016	0.031	0.047	0.015	0.188
5	2018-09-10T13:18:58.609Z	Daily-in-WatchList-Cache-5	988	6.289	0.015	0.016	2.657	3.326000000000(	0.016	0.203
6	2018-09-10T13:22:43.916Z	Monthly-in-WatchList-Cache-5	121	0.393999999999(	0.015	0.016	0.047	0.016	0.016	0.187

ภาพที่ 3.11 ตัวอย่าง Data comparison Table

3.5. การ Deploy

การ Deploy จะใช้ GitLab CI/CD เป็นเครื่องมือในการทำ CI/CD (Continuous integration and Continuous Deployment) เนื่องจากใช้บริการ GitLab ในการจัดการ Git Repository อยู่แล้ว การทำ CI/CD ด้วย GitLab CI/CD จึงทำได้อย่างมีประสิทธิภาพ

ขั้นตอนการทำ CI/CD ของ GitLab CI/CD จะทำตาม Pipeline ที่กำหนดไว้ในไฟล์ .gitlab-ci.yml ซึ่งจะระบุขั้นตอนการเตรียมการต่างๆก่อนการนำงานไป Deploy (Build) และ ขั้นตอนการ Deploy งาน (Deploy) โดย Pipeline จะทำงานก็ต่อเมื่อผู้พัฒนาทำการ Push งานขึ้นมาใน Branch ที่ระบุไว้ในไฟล์ .gitlab-ci.yml

```yaml
stages:
  - build
  - deploy

build:
  stage: build
  tags:
    - ets_ws_e2eperftest_119
  script:
    - pip install -r requirementPython.txt
    - pm2 delete PerfWatchDashboard Bokeh_App
  only:
    - master

bokeh_app:
  stage: deploy
  tags:
    - ets_ws_e2eperftest_119
  script:
    - cd ./Dashboard/PerfWatch-ETS_Interday-Visualize/Bokeh_apps
    - pm2 start bokeh_app.py --name Bokeh_App -- 8001
  only:
    - master
```

ภาพที่ 3.12 ตัวอย่างการเขียนไฟล์ .gitlab-ci.yml

จากไฟล์ .gitlab-ci.yml ข้างต้น การทำงานของ Pipeline จะแบ่งเป็น 2 ส่วนคือ Build และ Deploy มีรายละเอียดดังนี้

3.5.1. กำหนดขั้นตอนให้กับ Pipeline (stage)

3.5.1.1. ขั้นตอนการ Build

3.5.1.2. ขั้นตอนการ Deploy

3.5.2. ขั้นตอนการ Build

ทำการเตรียมสิ่งที่ต้องใช้ ก่อนการ Deploy

3.5.2.1. Tags: ระบุเครื่องที่ใช้ในการ Deploy

3.5.2.2. Script: คำสั่งที่จะรัน

1) Install แพ็จเกจของไพธอนที่จำเป็นต้องใช้ในโปรเจค

2) หยุดการทำงานของ Process เก่าที่เคย Deploy ไป

3.5.2.3. Only: กำหนดให้ดำเนินการเมื่อมีการ Push เข้า master branch เท่านั้น

3.5.3. ขั้นตอนการ Deploy

ทำการ Deploy

3.5.3.1. Tags: ระบุเครื่องที่ใช้ในการ Deploy

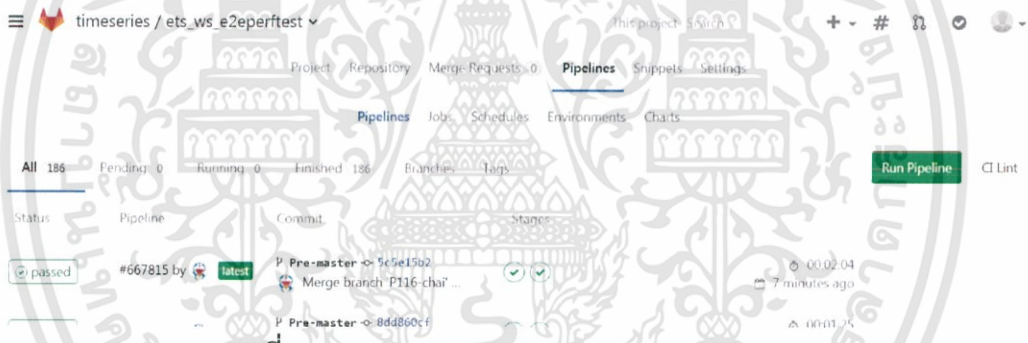
3.5.3.2. Script: คำสั่งที่จะรัน

1) เข้าไปยังโฟลเดอร์ของโปรเจคที่ต้องการ Deploy

2) ระบุคำสั่งที่ใช้ในการ Deploy

3.5.3.3. Only: กำหนดให้ดำเนินการเมื่อมีการ Push เข้า master branch เท่านั้น

เมื่อ Pipeline เริ่มทำงานแล้ว สามารถตรวจสอบการทำงานและรายละเอียดการทำงานของ Pipeline ได้ที่หน้าเว็ปไซต์ของ GitLab



ภาพที่ 3.13 การแสดงสถานะ Pipeline ของ GitLab

ภาพที่ 3.14 การแสดงสถานะในแต่ละขั้นตอนของ Pipeline ของ GitLab

บทที่ 4
ผลการดำเนินงาน

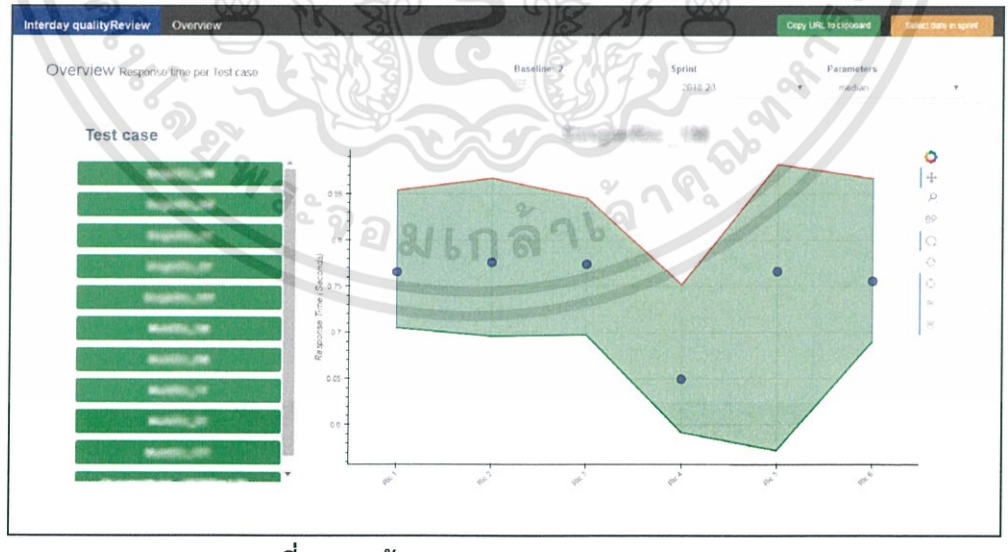
โครงงาน PerfWatch สามารถลดระยะเวลาและกระบวนการ ในการรวบรวมผล Logs จากองค์ประกอบต่างๆ ในระบบ และทำการแสดงผลออกมาในรูปแบบ Dashboard ซึ่งทำให้ผู้ตรวจสอบประสิทธิภาพของระบบ สามารถเปรียบเทียบและวิเคราะห์ผลการทดสอบได้สะดวกมากกว่าเดิม นอกจากนี้ผู้ตรวจสอบประสิทธิภาพของระบบยังสามารถกำหนดได้ว่า ผลการทดสอบใดเป็นผลการทดสอบที่ผ่านการพิจารณาแล้ว และจะถูกนำไปใช้ในการวิเคราะห์ผลการทดสอบอื่นๆ ต่อไป

4.1. ขั้นตอนการใช้งาน

ขั้นการการใช้งานจะแบ่งออกเป็น 2 ส่วนคือ Quality Review Dashboard และInvestigated Dashboard

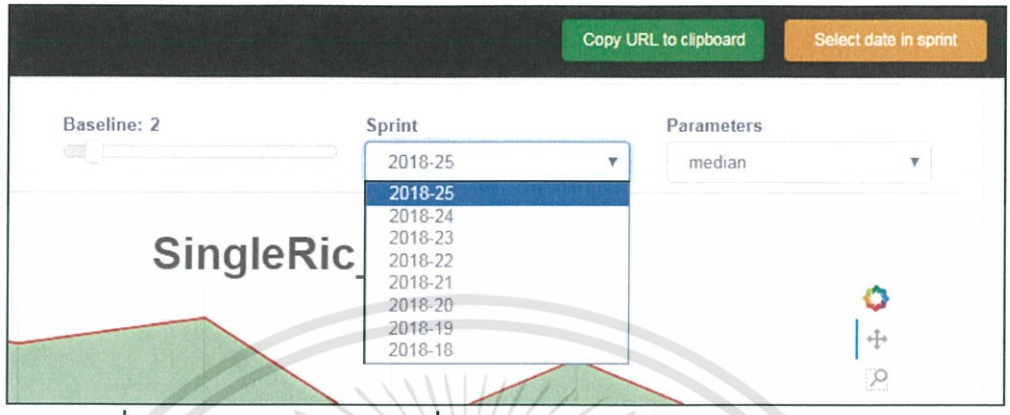
4.1.1. Quality Review Dashboard

ผู้ใช้สามารถพิจารณาภาพรวมของผลการทดสอบของทั้งรอบการทำงานได้ด้วย Dashboard นี้ โดย Dashboard จะแสดงข้อมูลของทั้งรอบการทำงานด้วยค่ากลางทางคณิตศาสตร์ อย่างเช่น ค่าเฉลี่ยหรือค่ามัธยฐาน



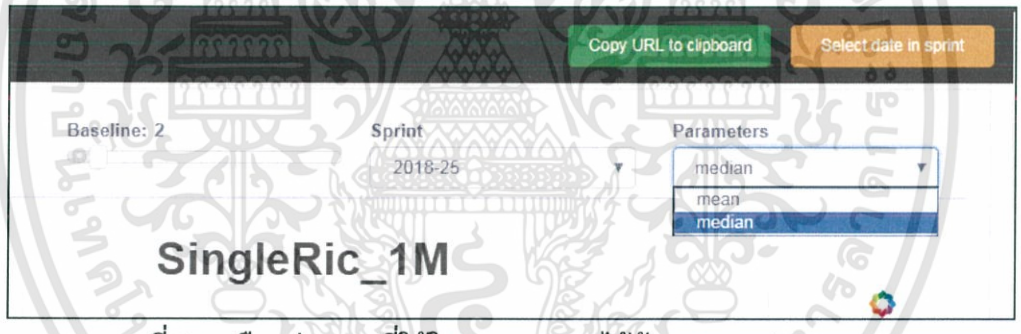
ภาพที่ 4.1 หน้า Quality Review Dashboard

4.1.1.1. ผู้ใช้สามารถเลือกรอบการทำงานที่ต้องการแสดงบนกราฟได้ด้วย Dropdown menu ดังนี้



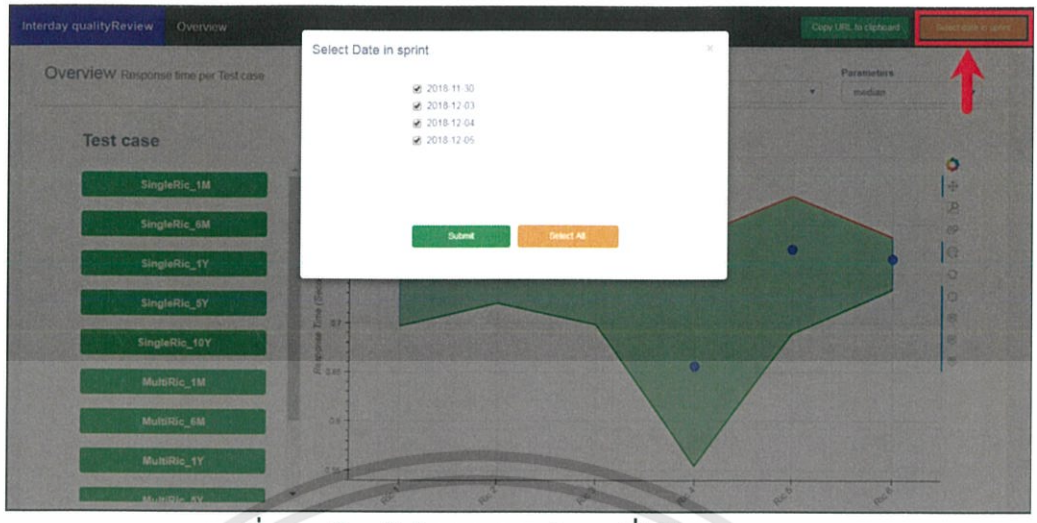
ภาพที่ 4.2 เลือกรอบการทำงานที่ต้องการแสดงผลได้ด้วย Dropdown menu

4.1.1.2. ผู้ใช้สามารถเลือกได้ว่าจะให้กราฟแสดงข้อมูลของรอบการทำงานด้วยค่าเฉลี่ยหรือค่ามัธยฐาน ได้ด้วย “Parameters” Dropdown menu ดังนี้



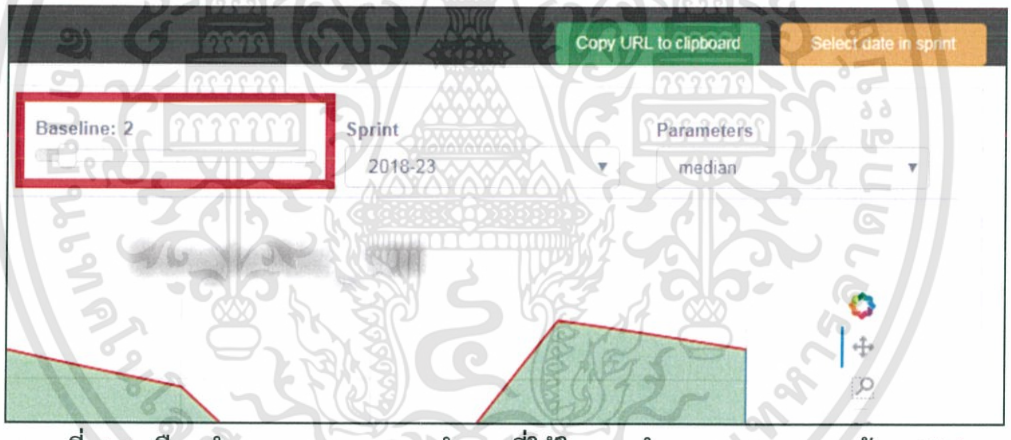
ภาพที่ 4.3 เลือกค่ากลางที่ใช้ในการแสดงผลได้ด้วย Dropdown menu

4.1.1.3. โดยเริ่มต้นเมื่อผู้ใช้เลือกรอบการทำงานที่ต้องการแสดงผลแล้ว Dashboard จะแสดงผลการทดสอบในช่วง Freeze period ทั้งหมดในรอบการทำงานนั้น แต่ผู้ใช้สามารถเลือกได้ว่าจะนำวันใดในรอบการทำงานมาแสดงผลได้ด้วยการกดที่ปุ่ม “Select date of sprint” จากนั้นจะมีป๊อปอัพปรากฏขึ้นเพื่อให้ผู้ใช้เลือกว่าจะใช้วันใดใน Freeze period ของรอบการทำงานนั้นในการแสดงผลบ้าง แล้วจึงกด “Submit”



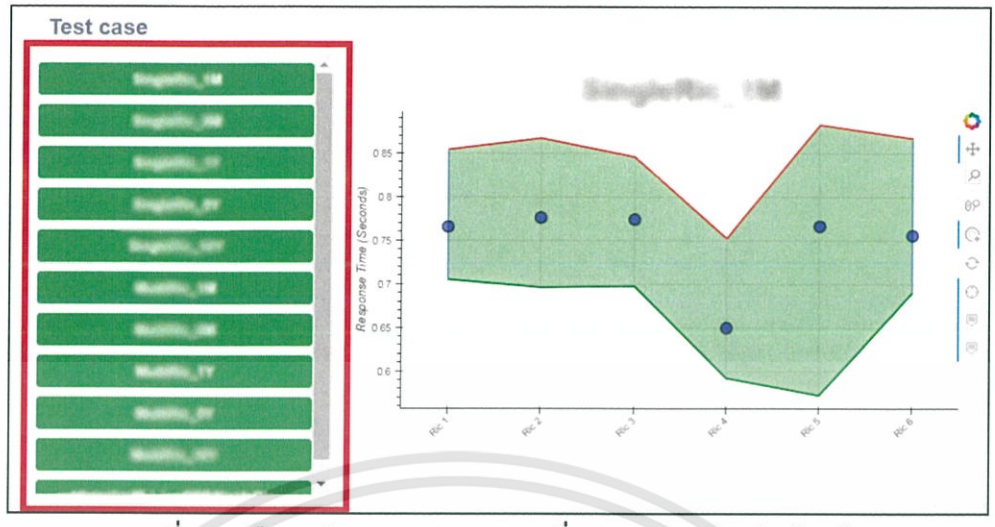
ภาพที่ 4.4 เลือกวันในรอบการทำงานที่ใช้ในการแสดงผล

4.1.1.4. ผู้ใช้สามารถเลือกจำนวนรอบการทำงานย้อนหลังเพื่อนำมาวิเคราะห์หาขอบเขตที่สามารถยอมรับได้ด้วย Slider ดังนี้



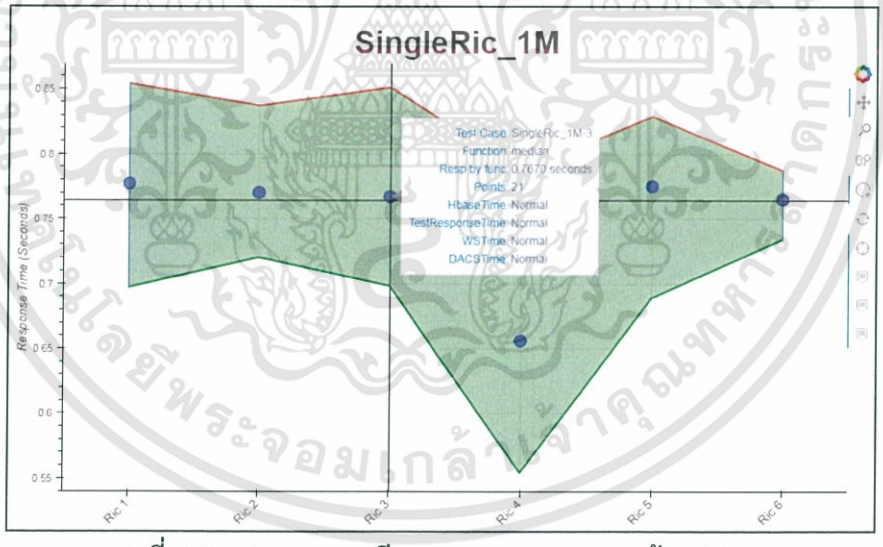
ภาพที่ 4.5 เลือกจำนวนของรอบการทำงานที่ใช้ในการคำนวณหาขอบเขตด้วย Slider

4.1.1.5. กราฟที่แสดงบน Quality Review Dashboard นี่จะแสดงผลการทดสอบของ Test case ที่อยู่ในกลุ่มเดียวกัน ผู้ใช้สามารถเปลี่ยนกลุ่มของ Test case ที่ต้องการแสดงผลได้ด้วยแถบปุ่มด้านซ้ายของ Dashboard ดังนี้



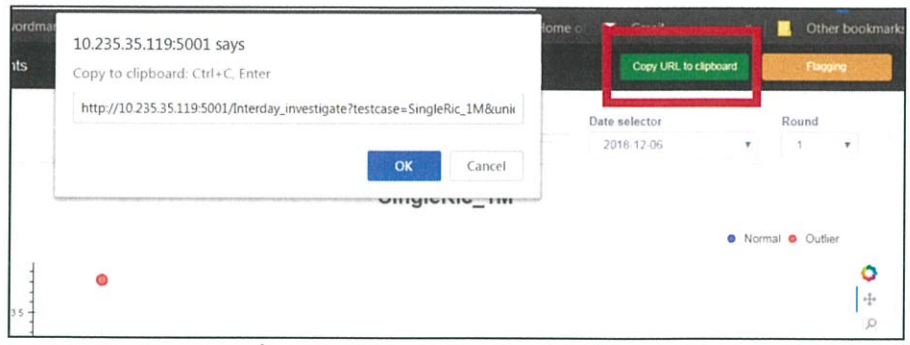
ภาพที่ 4.6 เลือกกลุ่มของ Test Case ที่จะแสดงผลด้วยปุ่มข้างซ้าย

4.1.1.6. เพื่อความสะดวกในการสืบหาสาเหตุของความผิดปกติ ผู้ใช้สามารถดูรายละเอียด และผลสรุปความของผิดปกติที่เกิดขึ้นของแต่ละองค์ ประกอบหลักในการทดสอบครั้งนั้นได้ด้วยการนำเมาส์ไปวางบนจุดดังนี้



ภาพที่ 4.7 แสดงรายละเอียดของผลการทดสอบด้วย Hover

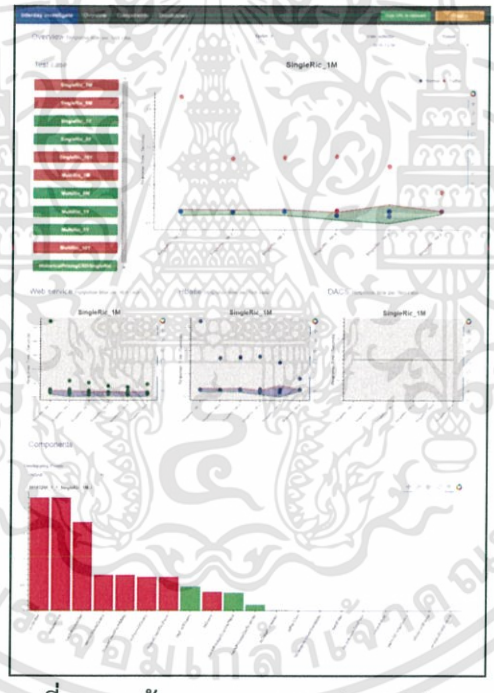
4.1.1.7. เพื่อไม่ให้ผู้ใช้ต้องทำการกดและเลือกค่าต่างๆใน Dashboard อีกครั้ง ในการดูข้อมูลเดิมที่เคยทำการเลือกไว้ก่อนหน้า ผู้ใช้สามารถเก็บสิ่งที่ผู้ใช้เลือกไว้ในหน้า Dashboard ได้ด้วยการกดปุ่ม “Copy URL to Clipboard” จากนั้นจะมีป๊อปอัพปรากฏขึ้นพร้อมกับ URL ซึ่งเมื่อผู้ใช้นำไปเปิดแล้วจะได้ Dashboard ซึ่งแสดงข้อมูลเหมือนกับ Dashboard ที่เคยเปิดเอาไว้



ภาพที่ 4.8 ปุ่ม “Copy URL to clipboard”

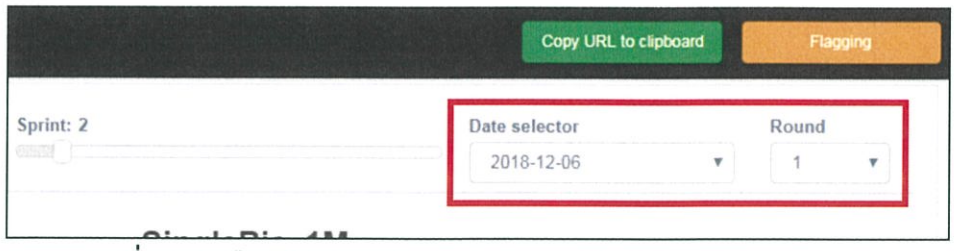
4.1.2. Investigated Dashboard

สำหรับผู้ที่ต้องการตรวจสอบผลการทดสอบอย่างละเอียดเพื่อหาสาเหตุของความผิดปกติของเวลาในการตอบสนองของระบบ



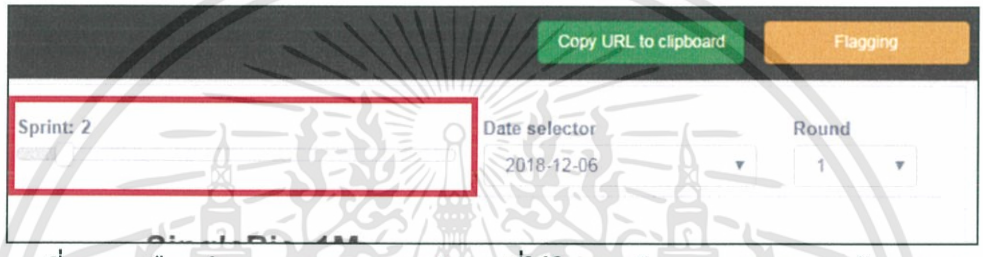
ภาพที่ 4.9 หน้า Investigated Dashboard

4.1.2.1. ผู้ใช้สามารถเลือกวันและรอบของผลการทดสอบที่ต้องการตรวจสอบได้ด้วย Dropdown menu ดังนี้



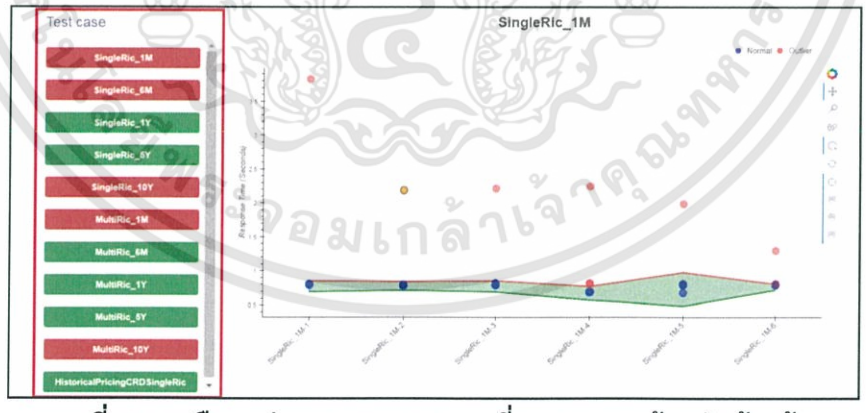
ภาพที่ 4.10 เลือกวันและรอบของการทดสอบด้วย Dropdown menu

4.1.2.2. ผู้ใช้สามารถเลือกจำนวนรอบการทำงานย้อนหลังเพื่อนำมาวิเคราะห์หาขอบเขตที่สามารถยอมรับได้ด้วย Slider ดังนี้



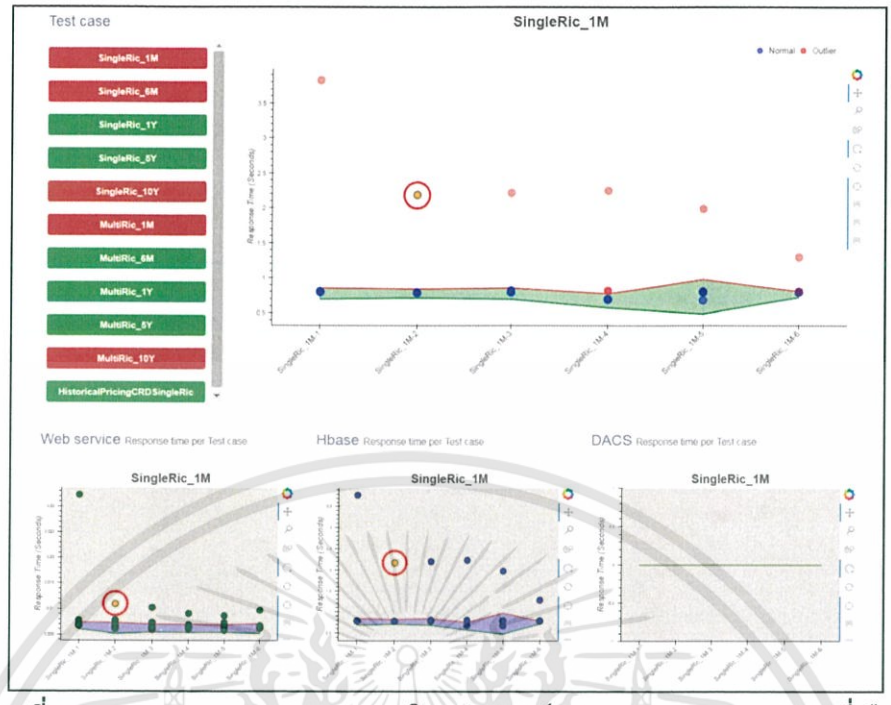
ภาพที่ 4.11 เลือกจำนวนของรอบการทำงานที่ใช้ในการคำนวณหาขอบเขตด้วย Slider

4.1.2.3. กราฟ Overview Response Time และกราฟ Component Response Time จะแสดงผลการทดสอบของ Test case ที่อยู่ในกลุ่มเดียวกัน ผู้ใช้สามารถเปลี่ยนกลุ่มของ Test case ที่ต้องการแสดงผลได้ด้วยแถบปุ่มด้านซ้ายของ Dashboard ดังนี้



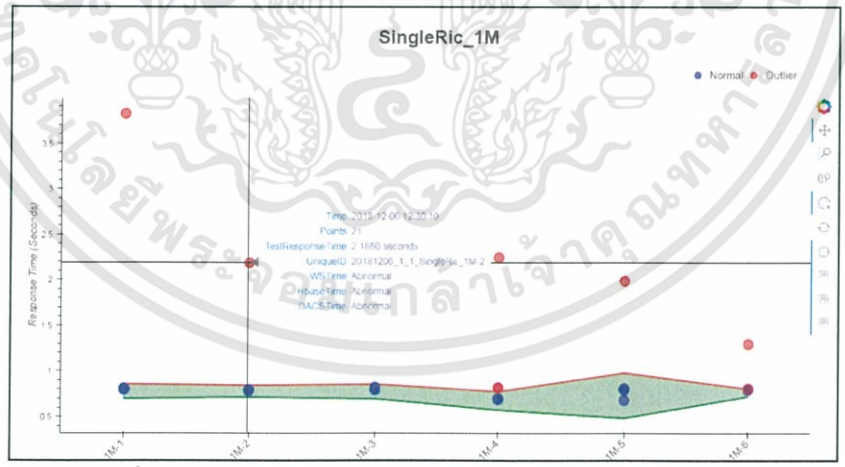
ภาพที่ 4.12 เลือกกลุ่มของ Test Case ที่จะแสดงผลด้วยปุ่มข้างซ้าย

4.1.2.4. ผู้ใช้สามารถพิจารณาเวลาในการตอบสนองของแต่ละองค์ประกอบในการทดสอบครั้งนั้นๆได้ด้วยการกดที่จุดที่อยู่บน กราฟ Overview Response Time และกราฟ Component Response Time ได้ ดังนี้



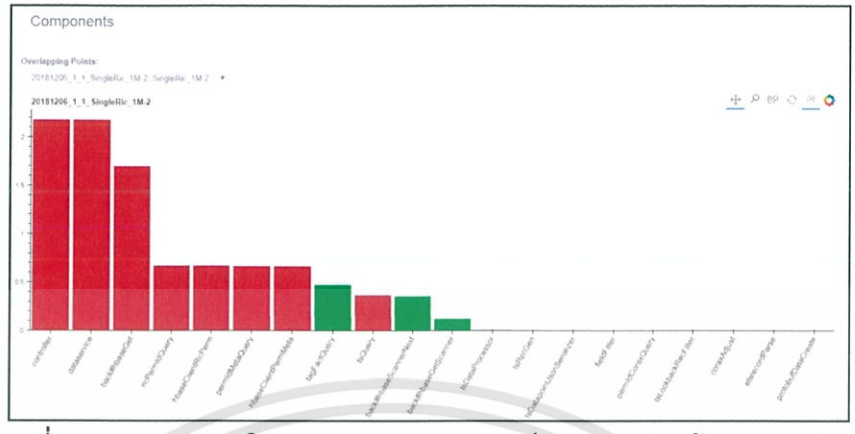
ภาพที่ 4.13 แสดงเวลาการตอบสนองในแต่ละองค์ประกอบของการทดสอบที่เลือก

4.1.2.5. เพื่อความสะดวกในการสืบหาสาเหตุของความผิดปกติ ผู้ใช้สามารถพิจารณารายละเอียด และผลสรุปความของผิดปกติที่เกิดขึ้นของแต่ละองค์ประกอบหลักในการทดสอบครั้งนั้นได้ด้วยการนำเมาส์ไปวางบนจุดดังนี้



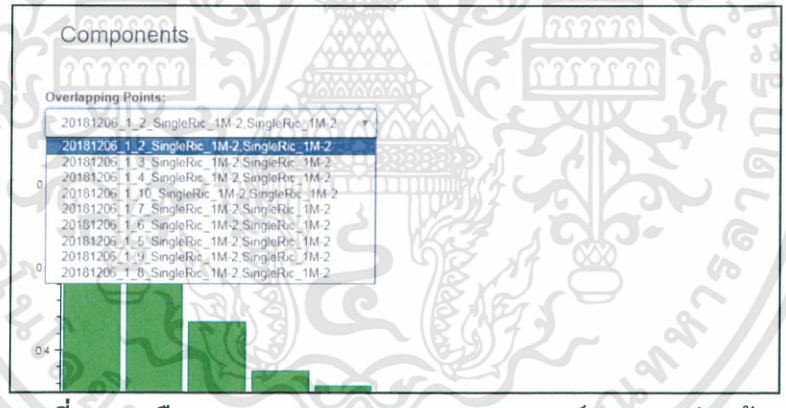
ภาพที่ 4.14 แสดงรายละเอียดของผลการทดสอบด้วย Hover

4.1.2.6. เมื่อผู้ใช้สังเกตเห็นจุดสีแดงบน กราฟ Overview Response Time และกราฟ Component Response Time ด้านบนซึ่งแสดงถึงความผิดปกติของข้อมูลผลการทดสอบแล้วผู้ใช้สามารถเลือกกดจุดดังกล่าว เพื่อให้กราฟ Breakdown Component Response Time ด้านล่างแสดงเวลาในการตอบสนองขององค์ประกอบย่อย เพื่อหาสาเหตุของความผิดปกตินั้น



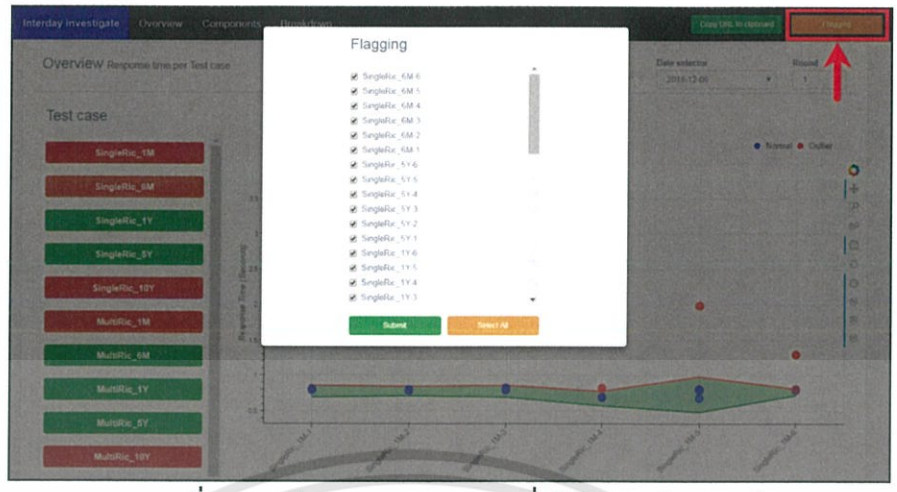
ภาพที่ 4.15 แสดงเวลาในการตอบสนองขององค์ประกอบย่อยด้วยแผนภูมิแท่ง

4.1.2.7. ในกรณีที่จุดที่ผู้ใช้เลือกซ้อนทับกันหลายจุด ผู้ใช้สามารถเลือกให้ Breakdown Component Response Time แสดงเวลาในการตอบสนองขององค์ประกอบย่อยของแต่ละจุดได้ด้วย “Overlapping Points” Dropdown menu ดังนี้



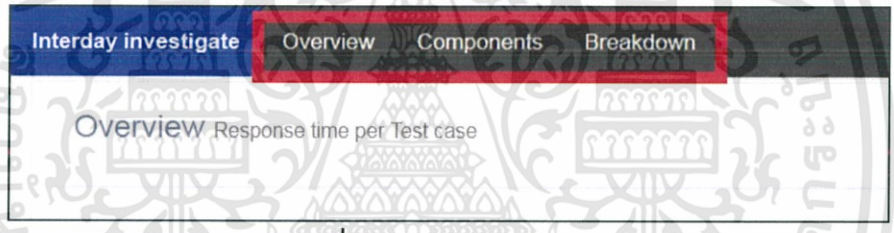
ภาพที่ 4.16 เลือกแสดงเวลาการตอบสนองขององค์ประกอบย่อยด้วย Dropdown menu

4.1.2.8. เนื่องจากผลการทดสอบในบางรอบ ผู้ใช้ได้พิจารณาแล้วว่ามีความปกติเกิดขึ้นและไม่ต้องการนำผลการทดสอบในรอบนั้นมาใช้ในการคิดหาขอบเขตที่สามารถยอมรับได้ ผู้ใช้สามารถกดที่ปุ่ม “Flagging” จากนั้นจะมีป๊อปอัพปรากฏขึ้นเพื่อให้ผู้ใช้เลือกว่าจะใช้ผลการทดสอบของ Test case ใดในรอบการทดสอบนั้นๆ บ้าง แล้วจึงกด “Submit”



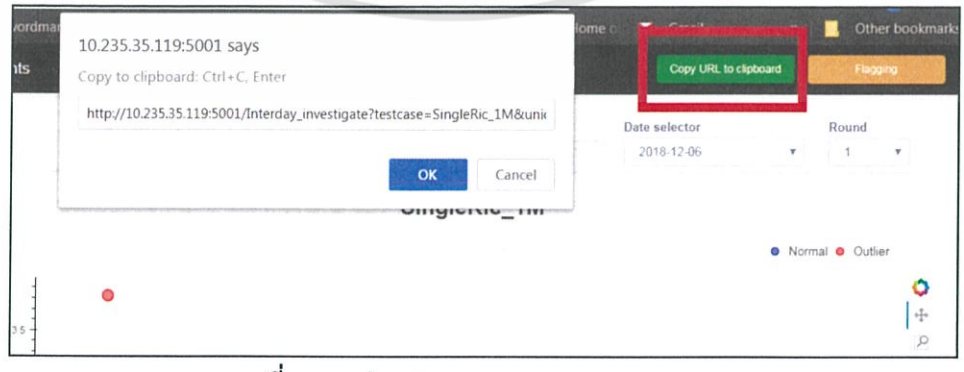
ภาพที่ 4.17 เลือกผลการทดสอบเพื่อใช้ในการวิเคราะห์

4.1.2.9. ผู้ใช้สามารถใช้ Navigation Bar ที่อยู่ด้านบนสุดของ Dashboard เพื่อไปยังกราฟที่ต้องการได้



ภาพที่ 4.18 Navigation bar

4.1.2.10. เพื่อไม่ให้ผู้ใช้ต้องทำการกดและเลือกค่าต่างๆใน Dashboard อีกครั้ง ในการดูข้อมูลเดิมที่เคยทำการเลือกไว้ก่อนหน้า ผู้ใช้สามารถเก็บสิ่งที่ผู้ใช้เลือกไว้ในหน้า Dashboard ได้ด้วยการกดปุ่ม “Copy URL to Clipboard” จากนั้นจะมีป๊อปอัพปรากฏขึ้นพร้อมกับ URL ซึ่งเมื่อผู้ใช้นำไปเปิดแล้วจะได้ Dashboard ซึ่งแสดงข้อมูลเหมือนกับ Dashboard ที่เคยเปิดเอาไว้



ภาพที่ 4.19 ปุ่ม “Copy URL to clipboard”

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

โครงงาน PerfWatch สามารถทำงานได้ตรงตามจุดประสงค์ที่คาดไว้นั้นคือ สามารถลดระยะเวลาที่ผู้ตรวจสอบประสิทธิภาพของระบบต้องใช้ในการตรวจสอบความผิดปกติในระบบและสามารถหาสาเหตุที่ทำให้เกิดความผิดปกตินั้นได้รวดเร็วขึ้น ซึ่งผู้ตรวจสอบประสิทธิภาพของระบบได้นำไปใช้งานจริงแล้ว สามารถสรุปผลการวิจัยได้ดังนี้

1) มี Script ใช้ในการรวบรวมผลการทดสอบและข้อมูลต่างๆที่อยู่ในองค์ประกอบในระบบจำนวน 3 Script ซึ่งใช้สำหรับ 4 Use Case

2) มี Elasticsearch ที่ใช้เป็นแหล่งเก็บข้อมูลของผลการทดสอบเพื่อใช้ในการแสดงผลและใช้ในการวิเคราะห์ในอนาคต

3) ติดตั้ง Logstash ให้กับเครื่อง Client ที่ใช้ในการทดสอบจำนวน 2 เครื่องเพื่อนำผลของการทดสอบเข้าสู่ Elasticsearch

4) มี Dashboard ที่ใช้ในการแสดงผล ซึ่งประกอบด้วย Investigated Dashboard และ Quality Review Dashboard ของแต่ละ Use case รวมเป็น 8 Dashboard

5.1. ปัญหาและอุปสรรคที่พบ

1) เนื่องจากช่วงแรกในการทำ Dashboard ทั้งผู้ใช้และผู้พัฒนายังไม่รู้ ความต้องการที่ชัดเจนทำให้ต้องลองผิดลองถูกไปเรื่อย ๆ จนกระทั่งเจอรูปแบบการแสดงผลที่มีประสิทธิภาพมากที่สุดในการตรวจสอบประสิทธิภาพของระบบ

2) เนื่องจากช่วงแรกไม่มีความรู้เกี่ยวกับโครงสร้างของระบบและขั้นตอนการตรวจสอบประสิทธิภาพของระบบทำให้งานที่ทำไม่ตรงกับความต้องการของผู้ใช้

3) ผลการทดสอบและสิ่งที่ผู้ใช้สนใจในแต่ละ Use case มีความแตกต่างกัน จึงเป็นการยากที่จะทำให้รองรับกับ Use case อื่นๆในอนาคต

4) ผู้ตรวจประสิทธิภาพของระบบยังไม่ได้ทำการยืนยันความถูกต้องของผลการทดสอบในแต่ละครั้งทำให้การวิเคราะห์เพื่อหาความผิดปกตินั้นยังไม่มีประสิทธิภาพเท่าที่ควร

5) การใช้ Git เนื่องจากในช่วงแรกไม่ได้มีข้อตกลงกันในทีมอย่างชัดเจน ทำให้เกิดความสับสนและทำให้งานที่ทำบางส่วนสูญหาย

6) เนื่องจากปัจจุบัน Dashboard ถูกพัฒนาขึ้นด้วย Bokeh ซึ่งเป็นเทคโนโลยีที่ยังใหม่อยู่ทำให้มีข้อมูลและตัวอย่างไม่มากนักให้ศึกษา ทำให้ต้องใช้เวลาในการพัฒนานาน

5.2. ข้อเสนอแนะ

1) ในอนาคตข้อมูลใน Elasticsearch จะมากขึ้นเรื่อยๆและอาจทำให้ความเร็วในการเอาข้อมูลจาก Elasticsearch นั้นช้าลง ดังนั้น ควรจะหาเครื่องเซิฟเวอร์เพิ่มเพื่อนำมาทำให้ระบบ Elasticsearch เป็น Cluster ที่มี Node มากกว่า 1 Node เพื่อให้สามารถรองรับข้อมูลจำนวนมหาศาลได้

2) หากในอนาคตมีข้อมูลของผลการทดลองที่ใช้งานได้มากพอ จะสามารถนำ Machine Learning มาประยุกต์ใช้ในการวิเคราะห์ข้อมูลเพื่อให้การตรวจสอบความผิดปกติที่เกิดขึ้นในระบบมีประสิทธิภาพมากขึ้น

เอกสารอ้างอิง

Python (2018) What's New in Python 2.7 (ออนไลน์) สืบค้นจาก: https://docs.python.org/2/whatsnew/2.7.html

Adi Bronshtein (2017) A Quick Introduction to the "Pandas" Python Library (ออนไลน์) สืบค้นจาก: https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673

Tutorialspoint (2018) Python Pandas - DataFrame (ออนไลน์) สืบค้นจาก: https://www.tutorialspoint.com/python_pandas/python_pandas_dataframe.htm

Elastic (2018) Elasticsearch (ออนไลน์) สืบค้นจาก: https://www.elastic.co/products/elasticsearch

Bo Andersen (2017) Understanding Sharding in Elasticsearch (ออนไลน์) สืบค้นจาก: https://codingexplained.com/coding/elasticsearch/understanding-sharding-in-elasticsearch

Bo Andersen (2017) Understanding Replication in Elasticsearch (ออนไลน์) สืบค้นจาก: https://codingexplained.com/coding/elasticsearch/understanding-replication-in-elasticsearch

Tanakorn Numrubporn (2018) Elasticsearch ภาคลุยสนาม (ออนไลน์) สืบค้นจาก: https://medium.com/machinereading/elasticsearch-%E0%B8%A0%E0%B8%B2%E0%B8%84%E0%B8%A5%E0%B8%B8%E0%B8%A2%E0%B8%AA%E0%B8%99%E0%B8%B2%E0%B8%A1-%E0%B8%95%E0%B8%AD%E0%B8%99%E0%B8%97%E0%B8%B5%E0%B9%88-1-19077ab210b3

Zkan (2014) Elasticsearch คืออะไร? (ออนไลน์) สืบค้นจาก: https://www.kanouivirach.com/2014/04/elasticsearch-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3/

Anaconda and Bokeh Contributors (2018) Bokeh (ออนไลน์) สืบค้นจาก:
https://bokeh.pydata.org/en/latest/#
Microsoft (2018) Visual Studio Code (ออนไลน์) สืบค้นจาก: https://code.visualstudio.com/
Michael Galarnyk (2018) Understanding Boxplots (ออนไลน์) สืบค้นจาก:
https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51
GitLab (2018) GitLab Continuous Integration (GitLab CI/CD) (ออนไลน์) สืบค้นจาก:
https://docs.gitlab.com/ee/ci/