โปรแกรมตรวจสอบทรัพยากรระบบ

SYSTEM RESOURCE MONITORING PROGRAM

พงศ์ศักดิ์ [illegible]

สหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2559

โปรแกรมตรวจสอบทรัพยากรระบบ

SYSTEM RESOURCE MONITORING PROGRAM

พงษ์สิทธิ์ กิตติสรนันท์

สหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2559

SYSTEM RESOURCE MONITORING PROGRAM

PONGSIT KITTISORANUN

A COOPEATIVE EDUCATION SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR
THE DEGREE OF BACHELOR OF SCIENCE (COMPUTER SCIENCE)
DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2016

หัวข้อสหกิจศึกษา	โปรแกรมตรวจสอบทรัพยากรระบบ System Resource Monitoring Program
ชื่อนักศึกษา	นายพงษ์สิทธิ์ กิตติสรนันท์ รหัสนักศึกษา 56050317
ปริญญา	วิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชา	วิทยาการคอมพิวเตอร์
ปีการศึกษา	2559
อาจารย์ที่ปรึกษา	ผศ.ดร.วรางคณา กิ้มปาน

คณะวิทยาศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้สหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์) ประจำปีการศึกษา 2559

คณะกรรมการสอบ	ลายมือชื่อ
ดร.ไพรัตน์ ธรเจริญศรี ประธานกรรมการ	
ผศ.ดร.วรางคณา กิ้มปาน กรรมการและอาจารย์ที่ปรึกษา	

หัวข้อสหกิจศึกษา โปรแกรมตรวจสอบทรัพยากรระบบ

System Resource Monitoring Program

ชื่อนักศึกษา นายพงษ์สิทธิ์ กิตติสรนันท์ รหัสนักศึกษา 56050317

ปริญญา วิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)

ภาควิชา วิทยาการคอมพิวเตอร์

คณะ วิทยาศาสตร์

มหาวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (สจล.)

ปีการศึกษา 2559

อาจารย์ที่ปรึกษา ผศ.ดร.วรางคณา กิ้มปาน

บทคัดย่อ

โปรแกรมตรวจสอบทรัพยากรระบบ เป็นเครื่องมือที่ใช้ตรวจสอบว่าระบบคอมพิวเตอร์นั้นผ่านความต้องการทางด้านฮาร์ดแวร์และซอฟต์แวร์ของโปรแกรมหรือไม่ เพื่อยืนยันว่าระบบจะสามารถใช้งานโปรแกรมได้โดยไม่มีข้อผิดพลาด และอำนวยความสะดวกให้กับผู้ใช้งานในการสร้างและแก้ไขกรณีทดสอบ ซึ่งสามารถทำได้ง่ายขึ้น เนื่องจากระบบเดิมนั้น การสร้างและแก้ไขกรณีทดสอบจะต้องเข้าไปแก้ไขที่โค้ดโปรแกรม ทำให้ต้องใช้เวลามาก และมีข้อจำกัดในการทำงานที่สามารถทำงานได้บนแพลตฟอร์มที่โปรแกรมหลักรองรับเท่านั้น ดังนั้น โปรแกรมตรวจสอบทรัพยากรระบบจึงพัฒนามาเพื่อให้สามารถทำงานได้บนหลายแพลตฟอร์ม โดยใช้เฟรมเวิร์ค Electron ในการจัดการ และเฟรมเวิร์ค Angular 2 ซึ่งเขียนด้วยภาษา TypeScript สำหรับการพัฒนาด้าน Front-end เพื่อให้โปรแกรมมีส่วนติดต่อผู้ใช้ที่สวยงามและเป็นมิตรกับผู้ใช้ ผู้ที่นำไปใช้งานจะสามารถทราบได้ว่าระบบคอมพิวเตอร์ส่วนใดที่ไม่ผ่านความต้องการของโปรแกรม เพื่อดำเนินการแก้ไขให้สามารถใช้งานได้

คำสำคัญ : ความต้องการของระบบ ข้ามแพลตฟอร์ม การทดสอบซอฟต์แวร์ ภาษา TypeScript เฟรมเวิร์ค Electron เฟรมเวิร์ค Angular 2

Title System Resource Monitoring Program

Student Mr. Pongsit Kittisoranun Student ID 56050317

Degree Bachelor of Science (Computer Science)

Department Computer Science

Faculty Science

University King Mongkut's Institute of Technology Ladkrabang (KMITL)

Academic Year 2016

Advisor Asst.Prof.Dr. Warangkhana Kimpan

Abstract

System Resource Monitoring Program is a tool for validating computer system whether it has passed either hardware or software requirement of a program in order to verify that the program will be run without any known errors. The user can easily create and modify test cases. Previously, creating and modifying the test cases need to edit the source code that take much time and can only be run on the same platform. Therefore, System Resource Monitoring Program was designed and developed as a cross-platform by using Electron and Angular 2 frameworks with TypeScript for nice-looking and user-friendly interface in front-end development. After executed, the user will be notified the components which does not pass the requirements in order to fix the defects easily.

Keywords : System Requirement, Cross-platform, Software Testing, TypeScript, Electron Framework, Angular 2 Framework

กิตติกรรมประกาศ

สหกิจศึกษาเล่มนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลือและความกรุณาจาก ดร.ไพรัตน์ ธรเจริญศรี ประธานกรรมการในการสอบสหกิจศึกษา และผู้ช่วยศาสตราจารย์ ดร.วรางคณา กิ้มปาน กรรมการและอาจารย์ที่ปรึกษาสหกิจศึกษา ที่ได้ให้คำแนะนำในการจัดทำรูปเล่มที่ถูกต้อง รวมถึงตรวจทานและปรับปรุงแก้ไขจนเสร็จสมบูรณ์ ผู้จัดทำสหกิจศึกษาจึงใคร่ขอขอบพระคุณท่านทั้งสองเป็นอย่างสูงไว้ ณ โอกาสนี้

ขอขอบพระคุณคณาจารย์ภาควิชาวิทยาการคอมพิวเตอร์ทุกท่านที่ได้มอบความรู้พื้นฐานทางด้านการเขียนโปรแกรม และได้ให้คำปรึกษาเป็นอย่างดีมาโดยตลอด จนกระทั่งสหกิจศึกษานี้สัมฤทธิ์ผลได้ด้วยดีทุกประการ

ขอขอบพระคุณทางบริษัท Thomson Reuters Software Thailand ที่สนับสนุนโครงการสหกิจศึกษาในครั้งนี้ เพื่อให้นักศึกษาได้เรียนรู้กระบวนการในการทำงานจริง ก่อนที่จะก้าวเข้าสู่โลกของการทำงาน และผลักดันให้สหกิจศึกษานี้สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบพระคุณโครงการ GE Foundation Scholar-Leaders Program ซึ่งได้ให้การสนับสนุนด้านทุนการศึกษาตลอดจนจบการศึกษา มีกิจกรรมเสริมสร้างทักษะความเป็นผู้นำ รวมถึงนักเรียนทุนท่านอื่น ๆ ที่ได้ให้คำแนะนำและกำลังใจที่ดีเสมอมา

สุดท้ายนี้ผู้จัดทำสหกิจศึกษาขอกราบขอบพระคุณบิดา มารดา ที่คอยให้คำปรึกษาและเป็นกำลังใจให้มาโดยตลอด ผู้จัดทำสหกิจศึกษาจึงใคร่ขอขอบพระคุณทุกท่านเป็นอย่างสูงไว้ ณ ที่นี้

พงษ์สิทธิ์ กิตติสรนันท์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

ในการติดตั้งโปรแกรมลงบนระบบคอมพิวเตอร์นั้นมีปัจจัยหลายอย่างที่ต้องคำนึงถึงเพื่อให้โปรแกรมสามารถทำงานได้อย่างถูกต้องตามที่ต้องการ เช่น ความเร็วของหน่วยประมวลผลกลาง ขนาดของหน่วยความจำ เป็นต้น จึงจำเป็นจะต้องมีการตรวจสอบทรัพยากรของระบบว่าเพียงพอต่อความต้องการของโปรแกรมหรือไม่ โดยการกำหนดเงื่อนไขที่จะทดสอบ พร้อมทั้งเก็บข้อมูลทรัพยากรของระบบที่จำเป็นต้องใช้ในการทดสอบ ซึ่งเงื่อนไขนั้นอาจมีการเปลี่ยนแปลงได้ตลอดเวลา โดยนักพัฒนาจะต้องคอยแก้ไขโค้ดโปรแกรมเพื่อให้รองรับเงื่อนไขใหม่ ๆ ทำให้ต้องใช้ต้นทุนในการพัฒนาที่เพิ่มมากขึ้น

ดังนั้น ในการจัดทำโครงการสหกิจศึกษาในครั้งนี้ ผู้จัดทำจึงได้มีการพัฒนาโปรแกรมที่สามารถตรวจสอบระบบคอมพิวเตอร์ได้ว่าทรัพยากรของระบบนั้นผ่านเงื่อนไขที่นำมาทดสอบหรือไม่ และสามารถที่จะสร้างเงื่อนไขใหม่หรือปรับเปลี่ยนเงื่อนไขเดิมได้โดยที่ไม่ต้องแก้ไขโค้ดโปรแกรม เพื่อให้สะดวกต่อการนำไปใช้งานในระยะยาว และจะต้องแยกออกจากโปรแกรมหลัก เพื่อให้สามารถนำไปประยุกต์ใช้กับโปรแกรมอื่น ๆ ได้

1.2 วัตถุประสงค์ของสหกิจศึกษา

1) เพื่อให้นักพัฒนาสามารถสร้างกรณีทดสอบและชุดทดสอบได้เอง และสามารถแปลงเป็นสคริปต์ทดสอบเพื่อนำไปใช้งานได้
2) เพื่อให้สามารถระบุได้ว่าระบบคอมพิวเตอร์ที่นำมาทดสอบนั้นผ่านความต้องการขั้นต่ำของโปรแกรมหรือไม่ จากการเรียกใช้งานสคริปต์ทดสอบที่สร้างขึ้น
3) เพื่อความสะดวกในการปรับเปลี่ยนเงื่อนไขของการทดสอบโดยไม่ต้องแก้ไขโค้ดโปรแกรม

1.3 ขอบเขตของสหกิจศึกษา

1) มีโปรแกรมสำหรับใช้ในการสร้างกรณีทดสอบและชุดทดสอบ
2) ผู้ใช้งานสามารถที่จะกำหนดเงื่อนไขที่จะใช้ทดสอบได้เองจากส่วนประกอบที่จัดเตรียมไว้ให้
3) สามารถแปลงชุดทดสอบให้เป็นสคริปต์ทดสอบเพื่อนำไปใช้งานได้
4) สามารถเก็บข้อมูลของระบบคอมพิวเตอร์ที่จำเป็นต่อการนำมาทดสอบได้
5) สามารถตรวจสอบข้อมูลของระบบคอมพิวเตอร์ได้ว่าผ่านเงื่อนไขที่นำมาทดสอบหรือไม่
6) สามารถจำลองข้อมูลของระบบคอมพิวเตอร์ที่จะนำมาทดสอบได้ เพื่อตรวจสอบความถูกต้องของเงื่อนไขที่กำลังสร้าง
7) ทำงานได้ในระบบปฏิบัติการคอมพิวเตอร์ Windows Linux และ OS X อย่างถูกต้อง

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ประโยชน์ต่อผู้ใช้
   - สามารถตรวจสอบทรัพยากรของระบบคอมพิวเตอร์ได้ว่าผ่านเงื่อนไขที่กำหนดหรือไม่
   - สามารถสร้างและแก้ไขกรณีทดสอบได้เองโดยที่ไม่ต้องแก้ไขโค้ดโปรแกรม
   - ลดระยะเวลาการพัฒนาในส่วนของการทดสอบความต้องการของโปรแกรม
   - สามารถนำโปรแกรมนี้ไปประยุกต์ใช้กับโปรแกรมอื่นได้อย่างสะดวก
   - สามารถจำลองข้อมูลของระบบคอมพิวเตอร์ได้ ทำให้ลดทรัพยากรในการทดสอบลง
2) ประโยชน์ต่อผู้พัฒนา
   - ได้ศึกษาและเรียนรู้การทำงานของการทดสอบระบบ และสามารถสร้างขึ้นมาได้เอง
   - ได้ศึกษาการใช้งานเฟรมเวิร์ค Electron ซึ่งใช้สำหรับสร้างแอปพลิเคชันที่ทำงานได้หลายระบบปฏิบัติการ
   - ได้ศึกษาการใช้งาน Angular2 ในการพัฒนาระบบ ซึ่งเขียนโดยใช้ภาษา TypeScript
   - ได้เรียนรู้การจัดการเว็บไซต์ทั้งเบื้องหน้า (Front-end) และเบื้องหลัง (Back-end) โดยใช้ภาษา HTML CSS และ JavaScript

1.5 ขั้นตอนการดำเนินงาน

1) เรียนรู้การเขียนโปรแกรมด้วยภาษาคอมพิวเตอร์ที่จำเป็นต้องใช้ในการพัฒนา คือ HTML CSS JavaScript JQuery และ TypeScript
2) ศึกษาการทำงานของเฟรมเวิร์ค Electron
3) สร้างส่วนประกอบที่สามารถเข้าไปดึงข้อมูลเบื้องต้นของระบบปฏิบัติการ ข้อมูล Microsoft Update Patch (KB) และข้อมูลของโปรแกรม Microsoft Office ที่ติดตั้งอยู่บนระบบคอมพิวเตอร์ได้
4) ออกแบบหน้าต่างผู้ใช้งานของโปรแกรมที่ใช้จัดการกรณีทดสอบ เพื่อให้ผู้ใช้งานสามารถสร้างกรณีทดสอบง่าย ๆ ได้เองจากส่วนประกอบที่มีให้
5) ทำให้โปรแกรมจัดการกรณีทดสอบสามารถใช้ตัวดำเนินการทางตรรกศาสตร์ได้ เพื่อให้สามารถสร้างเงื่อนไขที่มีความซับซ้อนมากขึ้น
6) เพิ่มฟังก์ชันการบันทึกและแก้ไขไฟล์กรณีทดสอบที่ได้สร้างขึ้น
7) สร้างโปรแกรมที่สามารถรวบรวมกรณีทดสอบต่าง ๆ ให้เป็นชุดทดสอบเดียว
8) สร้างโปรแกรมที่จะนำชุดทดสอบไปทำงานบนระบบคอมพิวเตอร์จริง ๆ
9) เพิ่มฟังก์ชันที่ให้ผู้ใช้สามารถจำลองข้อมูลในการทดสอบเงื่อนไขที่ได้สร้างขึ้น
10) ปรับแต่งโค้ดโปรแกรมให้มีความเป็นระเบียบมากขึ้น
11) เขียนรายงานอธิบายการใช้งานโปรแกรมและเครื่องมือต่าง ๆ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

การพัฒนาโปรแกรมตรวจสอบทรัพยากรระบบจะต้องมีความเข้าใจในเรื่องของการทดสอบซอฟต์แวร์ ความต้องการของระบบ รวมถึงความหมายของกรณีทดสอบและชุดทดสอบ เพื่อเป็นทฤษฎีที่ใช้อ้างอิงในการพัฒนาโปรแกรม จากนั้นศึกษาเครื่องมือและเทคโนโลยีที่จำเป็นต้องใช้ในการพัฒนาโปรแกรมบนเดสก์ท็อปสำหรับการทำงานหลายแพลตฟอร์ม ได้แก่ Node.js, Angular และเฟรมเวิร์ค Electron

2.1 การทดสอบซอฟต์แวร์

การทดสอบซอฟต์แวร์ [1] เป็นกระบวนการเพื่อช่วยให้ซอฟต์แวร์ที่พัฒนามีความถูกต้อง ความสมบูรณ์ ปลอดภัย และมีคุณภาพที่ดี โดยใช้ความรู้ทางเทคนิค เพื่อให้สามารถระบุหรือค้นหาความผิดพลาดของซอฟต์แวร์ที่อาจซ่อนอยู่ให้ปรากฏออกมา และสามารถระบุถึงแนวทางการเปิดปัญหาพร้อมสมมติฐานของความผิดพลาดที่เกิดขึ้นได้

การทดสอบซอฟต์แวร์นั้นสามารถช่วยลดค่าใช้จ่ายในการซ่อมบำรุงลงไปได้มาก โดยยิ่งพบข้อผิดพลาดได้เร็วเท่าไรก็จะยิ่งมีค่าใช้จ่ายในการซ่อมแซมน้อยลงเท่านั้น ตารางที่ 2.1 แสดงค่าใช้จ่ายในการแก้ไขข้อผิดพลาดโดยขึ้นอยู่กับเวลาที่พบข้อผิดพลาดนั้น ตัวอย่างเช่น ถ้าหากเกิดข้อผิดพลาดในส่วน Requirement ซึ่งได้พบข้อผิดพลาดนี้หลังจากที่ทำการปล่อยซอฟต์แวร์ออกไปแล้ว จะมีค่าใช้จ่ายในการซ่อมแซมมากถึง 10-100 เท่า เมื่อเทียบกับการพบข้อผิดพลาดในขั้นตอนของการตรวจสอบ Requirement เป็นต้น

ตารางที่ 2.1 ค่าใช้จ่ายในการแก้ไขข้อผิดพลาด

Cost to fix a defect		Time detected				
		Requirements	Architecture	Construction	System test	Post-release
Time introduced	Requirements	1x	3x	5-10x	10x	10-100x
	Architecture	-	1x	10x	15x	25-100x
	Construction	-	-	1x	10x	10-25x

2.1.1 คุณลักษณะที่นำมาทดสอบ

การทดสอบนั้นจะรวมถึงการทำงานในแต่ละส่วนของซอฟต์แวร์ เพื่อรวบรวมคุณลักษณะที่จะนำมาทดสอบให้ได้มากที่สุด โดยทั่วไปแล้วคุณลักษณะเหล่านี้จะเป็นตัวกำหนดขอบเขตของการทดสอบ เช่น

1) ทดสอบว่ากระบวนการออกแบบและพัฒนาซอฟต์แวร์เป็นไปตามความต้องการหรือไม่
2) สามารถตอบสนองต่อรูปแบบต่าง ๆ ของข้อมูลนำเข้าได้อย่างถูกต้อง
3) ประมวลผลฟังก์ชันต่าง ๆ ได้ภายในเวลาที่กำหนด
4) ซอฟต์แวร์นั้นเพียงพอต่อการนำไปใช้งาน
5) สามารถติดตั้งและทำงานได้ในสภาพแวดล้อมที่ต้องการ
6) ประสบผลสำเร็จตามความต้องการของผู้รับผลประโยชน์

เนื่องจากกรณีทดสอบสำหรับซอฟต์แวร์ทั่วไปนั้นสามารถมีได้ไม่จำกัด การทดสอบซอฟต์แวร์จึงต้องใช้การวิเคราะห์กรณีทดสอบเท่าที่จำเป็นต้องใช้เพื่อให้ครอบคลุมกรณีที่เป็นไปได้ให้ได้มากที่สุดสำหรับเวลาและทรัพยากรที่มีอยู่ เพื่อค้นหาข้อผิดพลาดหรือความบกพร่องของซอฟต์แวร์ขณะทำงาน โดยการทดสอบแต่ละกรณีจะเป็นไปตามลำดับ เมื่อข้อผิดพลาดหนึ่งได้รับการแก้ไขจึงจะทดสอบต่อไปเพื่อหาข้อผิดพลาดที่อยู่ลึกลงไป

2.1.2 ประเภทของการทดสอบ

การทดสอบซอฟต์แวร์แบ่งออกเป็นหลายประเภท ขึ้นอยู่กับขั้นตอนของการพัฒนาและวัตถุประสงค์ของการใช้งานซอฟต์แวร์ โดยประเภทของซอฟต์แวร์ที่สำคัญมีดังนี้

1) Installation Testing เป็นการทดสอบขณะทำการติดตั้งซอฟต์แวร์ เพื่อให้แน่ใจว่าซอฟต์แวร์นั้นสามารถทำงานในสภาพแวดล้อมของผู้ใช้งานได้โดยไม่มีข้อผิดพลาด
2) Compatibility Testing เป็นการทดสอบความเข้ากันได้กับซอฟต์แวร์อื่น ๆ หรือระบบปฏิบัติการที่ซอฟต์แวร์นั้นทำงานอยู่
3) Regression Testing เป็นการทดสอบการเปลี่ยนแปลงของซอฟต์แวร์ โดยเมื่อซอฟต์แวร์มีการเปลี่ยนแปลงจะต้องทำการทดสอบกรณีทดสอบที่ได้เคยทดสอบไปแล้วด้วย เพื่อให้แน่ใจว่าระบบใหม่ยังคงสามารถทำงานในฟังก์ชันเดิมได้อย่างถูกต้อง
4) Alpha Testing เป็นการทดสอบการทำงานของระบบที่กระทำโดยทีมทดสอบ ในสถานที่ของนักพัฒนา
5) Beta Testing จะทำหลังจาก Alpha Testing โดยการปล่อยซอฟต์แวร์ในเวอร์ชัน Beta โดยสามารถจะกัดกลุ่มของผู้ใช้ได้ หรือทำการปล่อยเป็นสาธารณะ โดยจะเก็บรวบรวมผลตอบรับ เพื่อนำไปพัฒนาและแก้ไขข้อผิดพลาดที่เกิดขึ้น
6) Usability Testing เป็นการทดสอบส่วนติดต่อผู้ใช้ของซอฟต์แวร์ว่าสามารถใช้งานได้ดีและเรียนรู้ง่ายหรือไม่

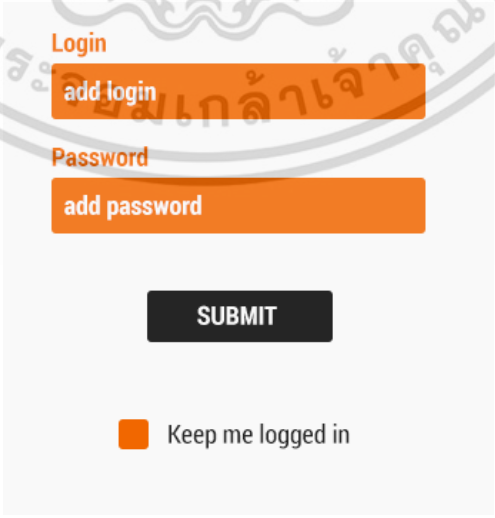
2.2 กรณีทดสอบและชุดทดสอบ

2.2.1 กรณีทดสอบ

กรณีทดสอบ (Test Case) [2] คือ กลุ่มของเงื่อนไขที่จะเป็นตัวทดสอบว่าแอปพลิเคชัน หรือซอฟต์แวร์นั้นสามารถทำงานได้ตรงตามที่ต้องการหรือไม่ โดยจะต้องกำหนดเงื่อนไขเพื่อทำการทดสอบ ซึ่งอาจจะมีได้หลายเงื่อนไขขึ้นอยู่กับการออกแบบกรณีทดสอบ ในส่วนใหญ่นั้นผลลัพธ์จะเป็นผ่านและไม่ผ่าน ขึ้นอยู่กับว่าจะเข้าเงื่อนไขใด โดยการสร้างกรณีทดสอบที่ดีนั้นจะต้องคำนึงถึงคุณสมบัติดังนี้

1) แต่ละกรณีทดสอบจะต้องแยกจากกันอย่างอิสระ ไม่มีกรณีทดสอบใดที่เกี่ยวข้องหรือมีผลกระทบต่อกรณีทดสอบอื่น
2) ในหนึ่งกรณีทดสอบจะมุ่งเน้นไปที่วัตถุเพียงชนิดเดียวในระบบ เพื่อให้กรณีทดสอบนั้นมีประสิทธิภาพมากที่สุด
3) ทุก ๆ กรณีทดสอบจะต้องสามารถเรียกใช้งานแยกจากกันได้ โดยที่ไม่จำเป็นต้องทำกรณีใดก่อน
4) แต่ละเงื่อนไขกรณีทดสอบนั้นสร้างขึ้นจากสถานการณ์ที่อาจเกิดขึ้นได้จริงในการทำงานของโปรแกรม
5) การทำงานของแต่ละกรณีทดสอบจะต้องใช้ระยะเวลาไม่นานเกินไป

การเขียนกรณีทดสอบจะเริ่มจากการเก็บรวบรวมทุกกรณีที่อาจเกิดขึ้นได้จากการทำงานของโปรแกรม หรือการป้อนข้อมูลจากผู้ใช้ ซึ่งผู้ใช้แต่ละคนนั้นมีความแตกต่างกัน จึงสามารถป้อนข้อมูลได้ทุกรูปแบบ หากไม่มีการป้องกันหรือเตรียมการไว้ก็อาจจะทำให้โปรแกรมทำงานผิดพลาด หรืออาจหยุดการทำงานได้ ซึ่งทำให้โปรแกรมนั้นดูไม่มีคุณภาพ การเขียนกรณีทดสอบให้ครอบคลุมจะช่วยปิดช่องโหว่เหล่านี้ได้ ตัวอย่างดังรูปที่ 2.1 ซึ่งเป็นหน้าต่างการลงชื่อเข้าใช้งานระบบ ที่มีช่องให้ผู้ใช้ป้อนข้อมูลชื่อผู้ใช้ และรหัสผ่าน ปุ่มคงสถานะของการลงชื่อเข้าใช้ และมีปุ่มยืนยันการลงชื่อเข้าใช้



รูปที่ 2.1 หน้าจอการลงชื่อเข้าใช้งานระบบ

ในการสร้างกรณีทดสอบสำหรับหน้าจอลงชื่อเข้าใช้งานระบบดังรูปที่ 2.1 นั้น เริ่มจากการตรวจสอบขั้นพื้นฐานก่อนว่าในหน้าจอนั้นมีส่วนประกอบครบ และจัดตำแหน่งตรงตามที่ระบุไว้หรือไม่ จากนั้นจึงตรวจสอบกรณีที่เป็นไปได้จากการป้อนข้อมูลของผู้ใช้ เริ่มจากการที่ผู้ใช้ไม่ป้อนข้อมูลใด ๆ เลย แล้วทำการกดปุ่มยืนยัน จะต้องไม่สามารถทำรายการได้ และมีข้อความแจ้งเตือนให้ผู้ใช้ทราบ จากนั้นก็เป็นการทดสอบความถูกต้องของการยืนยันตัวตนโดยจะต้องเขียนให้ครอบคลุมกรณีที่เป็นไปได้ทั้งหมด โดยตัวอย่างของกรณีทดสอบสำหรับหน้าจอลงชื่อเข้าใช้แสดงได้ดังรูปที่ 2.2

ID	Name / Idea	Precondition Steps	Steps	Expected Resault	Status	Pos/Neg	Comments
TC1	Check login form UI elements according to mockup from the spec (text field, button, chekbox, ect)	Specification with UI templetes	1. Open web page with login form 2. Check all UI elements	All elemetns are ok	Pass	1	
TC2	Authorization with empty fields (Login and Password field is empty)	Opened login form	1. Leave empty Login and password fields. 2. Press OK button	Displayed error message "Login and password fields can not be blank" user is not authorized.	Fail	0	Bug ID+ Bug summary
TC3	Correct Authorization	User already registered in the system	1. Enter correct Login in Login field. 2. Enter correct Password in Password field. 3. Press OK button	Authorization complete user entered the system.	Pass	1	
TC4	Incorrect Authorization (Wrong Login)	Make sure that user does not exist in the system	1. Enter incorrect Login in Login field. 2. Enter correct Password in Password field. 3. Press OK button	Displayed message "This user is not registered" Authorization is not completed.	Pass	0	
TC5	Incorrect Authorization (Correct Login and Wrong Password)	User have been registererd in the system	1. Enter correct Login in Login field. 2. Enter incorrect Password in Password field. 3. Press OK button	Authorization is not completed, error message displayed.	Pass	0	
TC6	Checking the maximum lenght of login and password	(According to spec) User have been registered in system with max characters in login and pass fields	1. Enter correct Login in Login field. 2. Enter correct Password in Password field. 3. Press OK button	Authorization complete user entered the system.	Pass	1	

รูปที่ 2.2 ตัวอย่างกรณีทดสอบสำหรับหน้าจอลงชื่อเข้าใช้ [3]

2.2.2 ชุดทดสอบ

ชุดทดสอบ (Test Suite) [4] คือ ชุดของกรณีทดสอบที่ใช้สำหรับทดสอบซอฟต์แวร์ เพื่อบอกว่าซอฟต์แวร์นั้นมีคุณสมบัติตามที่ต้องการ ซึ่งจะเป็นการทดสอบในภาพรวมที่ใหญ่กว่ากรณีทดสอบ โดยสามารถแบ่งกรณีทดสอบออกเป็นกลุ่มย่อย ๆ เพื่อทดสอบในแต่ละด้านได้ โดยผลลัพธ์จะขึ้นอยู่กับแต่ละกรณีทดสอบว่ามีกรณีใดบ้างที่ผ่านและไม่ผ่านการทดสอบ

ชุดทดสอบส่วนใหญ่มักจะระบุรายละเอียดหรือเป้าหมายของแต่ละกลุ่มของกรณีทดสอบ และข้อมูลการตั้งค่าของระบบที่ต้องการขณะทำการทดสอบ กลุ่มของกรณีทดสอบสามารถระบุสถานะหรือกระบวนการที่จำเป็นต้องมีก่อนทำการทดสอบได้

2.3 ความต้องการของระบบ

ในทุก ๆ ซอฟต์แวร์คอมพิวเตอร์นั้นต้องการฮาร์ดแวร์หรือซอฟต์แวร์บางตัวในการที่จะทำให้ซอฟต์แวร์นั้นทำงานได้ ข้อกำหนดเหล่านี้เรียกว่า ความต้องการของระบบ ซึ่งความต้องการเหล่านี้มักจะเพิ่มขึ้นตามกาลเวลา หรือเมื่อซอฟต์แวร์นั้นออกรุ่นใหม่ ซึ่งในแต่ละความต้องการของระบบนั้นจะมีการกำหนดประเภทเป็นความต้องการทางด้านฮาร์ดแวร์และซอฟต์แวร์ [5]

2.3.1 ความต้องการทางด้านฮาร์ดแวร์

ความต้องการทางฮาร์ดแวร์ถือเป็นปัจจัยหลักที่จะต้องตรวจสอบว่าซอฟต์แวร์จะสามารถทำงานได้หรือไม่ โดยจะเน้นไปที่ความเร็วของหน่วยประมวลผลและขนาดของหน่วยความจำเป็นหลัก ซึ่งเป็นสิ่งที่จำเป็นที่สุดในการทำงานของโปรแกรม หากระบบนั้นมีทรัพยากรไม่เพียงพอต่อความต้องการก็อาจจะทำให้การทำงานของโปรแกรมมีความผิดพลาดเกิดขึ้นได้ ความต้องการทางด้านฮาร์ดแวร์แสดงได้ดังตารางที่ 2.2

ตารางที่ 2.2 ความต้องการทางด้านฮาร์ดแวร์

Hardware Requirements	
CPU	Core 2 Quad Q6600 at 2.4 GHz or AMD Phenom 9850 at 2.5 GHz
Memory	4 GB of RAM
Hard Drive	65 GB of free space
Graphics Driver	GeForce 9800GT 1GB or ATI Radeon HD 4870 1GB

จากตารางที่ 2.2 แสดงให้เห็นถึงความต้องการทางด้านฮาร์ดแวร์ของโปรแกรมหนึ่ง ซึ่งต้องการหน่วยประมวลผลกลางตั้งแต่รุ่น Core 2 Quad Q6600 ที่มีความเร็ว 2.4 GHz เป็นต้นไป สำหรับชิปประมวลผลของบริษัท Intel หรือรุ่น Phenom 9850 ที่มีความเร็ว 2.5 GHz ขึ้นไป และต้องการหน่วยความจำหลักขนาด 4 GB ขึ้นไป เพื่อให้โปรแกรมสามารถทำงานได้อย่างถูกต้อง เป็นต้น

2.3.2 ความต้องการทางด้านซอฟต์แวร์

ความต้องการทางซอฟต์แวร์ คือ การที่โปรแกรมจะถูกติดตั้งได้นั้น ระบบจะต้องมีซอฟต์แวร์ที่จำเป็นติดตั้งไว้ก่อนแล้ว ซึ่งส่วนใหญ่จะเป็นซอฟต์แวร์ที่โปรแกรมนั้น ๆ จำเป็นต้องเรียกใช้งาน หรือเข้าถึงค่าที่เก็บไว้ใน Registry การระบุความต้องการทางด้านซอฟต์แวร์สามารถทำได้โดยระบุข้อกำหนดเบื้องต้นไว้ว่าโปรแกรมจะสามารถทำงานได้หากมีซอฟต์แวร์ที่ต้องการติดตั้งอยู่ในระบบแล้ว ซึ่งจะทำให้ซอฟต์แวร์นั้นสามารถทำงานได้อย่างถูกต้องและมีประสิทธิภาพ โดยประเภทของซอฟต์แวร์ดังกล่าวมีดังนี้

2.3.2.1 แพลตฟอร์ม

การที่ซอฟต์แวร์จะทำงานได้นั้นจำเป็นต้องกำหนดลักษณะแพลตฟอร์มที่ต้องการ ซึ่งแพลตฟอร์มก็รวมถึงสถาปัตยกรรมคอมพิวเตอร์ ระบบปฏิบัติการ ภาษาคอมพิวเตอร์ และไลบรารีต่าง ๆ ที่ต้องใช้ ซึ่งระบบปฏิบัติการนั้นเป็นสิ่งสำคัญที่จะต้องถูกระบุไว้ในความต้องการซอฟต์แวร์ และซอฟต์แวร์อาจจะไม่สามารถทำงานได้กับระบบปฏิบัติการคนละรุ่นกัน ถึงแม้ว่าจะเป็นประเภท

เดียวกันก็ตาม เช่น ซอฟต์แวร์ที่ออกแบบมาเพื่อทำงานบนระบบปฏิบัติการ Windows XP ก็อาจจะไม่สามารถทำงานบนระบบปฏิบัติการ Windows 7 ซึ่งใหม่กว่าได้

2.3.2.2 เอพีไอและไดรเวอร์

เอพีไอ หรือส่วนต่อประสานโปรแกรมประยุกต์ หมายถึงวิธีการที่ระบบปฏิบัติการ ไลบรารี หรือบริการอื่น ๆ เปิดให้โปรแกรมคอมพิวเตอร์สามารถติดต่อเรียกใช้งานได้ และไดรเวอร์ก็คือซอฟต์แวร์ที่ติดต่อกับอุปกรณ์ต่อพ่วงต่าง ๆ ให้สามารถใช้งานได้ โดยซอฟต์แวร์ที่ต้องการใช้งานอุปกรณ์พิเศษบางชนิดอย่างการ์ดแสดงผลระดับสูง จำเป็นจะต้องใช้เอพีไอพิเศษ หรือไดรเวอร์ที่ทันสมัย ตัวอย่างเช่น DirectX ซึ่งเป็นซอฟต์แวร์ที่รวบรวมเอพีไอที่ใช้สำหรับจัดการสื่อประสมต่าง ๆ โดยเฉพาะเกม ซึ่งอยู่บนแพลตฟอร์มของ Microsoft และซอฟต์แวร์ที่ต้องการใช้อุปกรณ์ต่อพ่วงก็ต้องเรียกใช้งานผ่านทางไดรเวอร์นั่นเอง

2.3.2.3 แอปพลิเคชัน

ซอฟต์แวร์บางประเภทต้องการเข้าถึงการใช้งานแอปพลิเคชัน เช่น Microsoft Outlook เพื่อใช้ในการรับส่งจดหมายอิเล็กทรอนิกส์ Microsoft Excel ในการสร้างตารางเพื่อแสดงข้อมูลที่มีปริมาณมากให้อยู่ในรูปแบบที่เข้าใจง่าย เป็นต้น จึงจำเป็นต้องติดตั้งแอปพลิเคชันที่ระบุก่อนที่จะติดตั้งซอฟต์แวร์นั้น ๆ เพื่อให้สามารถทำงานได้อย่างมีประสิทธิภาพ

2.4 เว็บแอปพลิเคชันแบบหน้าเดียว

เว็บแอปพลิเคชันแบบหน้าเดียว [6] คือ เว็บแอปพลิเคชันหรือเว็บไซต์ที่การทำงานทั้งหมดอยู่ในหน้าเว็บเพียงหน้าเดียวเท่านั้น โดยหน้าเว็บนั้นจะไม่มีการโหลดใหม่หรือเปลี่ยนไปยังหน้าอื่น ซึ่งจะทำให้ผู้ใช้รู้สึกเหมือนใช้งานแอปพลิเคชันบนเดสก์ท็อป โดยการทำงานนั้นจะโหลดข้อมูลทั้งหมดในครั้งเดียว คือ HTML JavaScript และ CSS หรือทยอยโหลดและเพิ่มลงไปในหน้าเว็บเมื่อจำเป็นต้องใช้เท่านั้น โดยมีเฟรมเวิร์คมากมายที่เข้ามาช่วยในการจัดการเว็บแอปพลิเคชันแบบหน้าเดียว เช่น Angular.js, React.js, Ember.js, และ Meteor.js เป็นต้น รูปแบบการส่งข้อมูลที่นิยมใช้ในการติดต่อกับ Back-end คือ XML และ JSON ซึ่งเป็นรูปแบบที่มีโครงสร้างที่ไม่ซับซ้อน สามารถเข้าใจได้ง่าย

ในการสร้างเว็บแอปพลิเคชันแบบหน้าเดียวนอกจากจะเน้นการทำงานที่ฝั่งไคลเอนต์แล้ว ยังต้องมีการออกแบบโครงสร้างของเซิร์ฟเวอร์ เพื่อให้สามารถติดต่อและรับส่งข้อมูลได้ โดยอิงตามสถาปัตยกรรมของเซิร์ฟเวอร์ ดังนี้

2.4.1 Thin Server Architecture

สถาปัตยกรรมนี้จะทำให้เซิร์ฟเวอร์นั้นทำงานน้อยที่สุด โดยจะย้ายข้อมูลจากเซิร์ฟเวอร์ไปเก็บไว้ที่ไคลเอนต์แทน ซึ่งจะทำให้บทบาทของเซิร์ฟเวอร์นั้นเปลี่ยนไปคล้ายกับเว็บเซอร์วิส โดยมีจุดมุ่งหมายเพื่อที่จะลดความซับซ้อนของทั้งระบบ

2.4.2 Thick Stateful Server Architecture

ในสถาปัตยกรรมนี้เซิร์ฟเวอร์จะเก็บสถานะที่จำเป็นของไคลเอนต์ไว้ในหน่วยความจำ เมื่อมีการร้องขอข้อมูลส่งมายังเซิร์ฟเวอร์ เซิร์ฟเวอร์ก็จะส่งกลับไปเฉพาะส่วนที่ร้องขอเท่านั้น ซึ่งมักจะเป็นการเพิ่ม ลบ หรือแก้ไข ส่วนแสดงข้อมูลของไคลเอนต์ เมื่อเสร็จสิ้นแล้วไคลเอนต์ก็จะปรับเป็นสถานะใหม่และเซิร์ฟเวอร์ก็ปรับเช่นเดียวกัน ซึ่งการประมวลผลต่าง ๆ จะอยู่บนเซิร์ฟเวอร์ และ HTML ก็จะถูกสร้างขึ้นที่เซิร์ฟเวอร์ก่อนจะส่งไปให้ไคลเอนต์ ซึ่งทำให้เซิร์ฟเวอร์นั้นต้องมีหน่วยความจำและหน่วยประมวลผลที่มากขึ้น แต่มีข้อดีคือทำให้พัฒนาได้ง่ายเนื่องจากการเขียนโปรแกรมหลักจะอยู่ทางด้านเซิร์ฟเวอร์

2.4.3 Thick Stateless Server Architecture

สถาปัตยกรรมนี้จะคล้ายกับ Thick Stateful Server Architecture แต่ต่างกันตรงที่ไคลเอนต์จะส่งข้อมูลสถานะปัจจุบันไปยังเซิร์ฟเวอร์ (ปกติมักใช้ Ajax ในการส่ง) ซึ่งเซิร์ฟเวอร์จะต้องสร้างโครงสร้างของหน้าเว็บจากสถานะที่ไคลเอนต์ส่งมาแล้วจึงส่งข้อมูลที่จำเป็นกลับไปให้ไคลเอนต์ตามที่ร้องขอ ทำให้ไคลเอนต์ต้องส่งข้อมูลจำนวนมากไปยังเซิร์ฟเวอร์ และเซิร์ฟเวอร์ก็จะต้องใช้ทรัพยากรในการจัดการโครงสร้างของแต่ละไคลเอนต์ที่ส่งมา แต่ก็ทำให้จัดการได้ง่ายขึ้นเนื่องจากเซิร์ฟเวอร์นั้นไม่จำเป็นต้องเก็บข้อมูลสถานะของไคลเอนต์เอาไว้

2.5 Node.js

Node.js [7] เป็นซอฟต์แวร์โอเพนซอร์สข้ามแพลตฟอร์ม ใช้สำหรับสร้างเว็บเซิร์ฟเวอร์หรือเครื่องมือทางเครือข่ายคอมพิวเตอร์โดยใช้ภาษา JavaScript และโมดูลต่าง ๆ ที่ช่วยจัดการฟังก์ชันหลัก ซึ่งโมดูลพื้นฐานส่วนใหญ่ก็จะถูกเขียนด้วยภาษา JavaScript เช่นกัน โดยโมดูลพื้นฐานนั้นจะมีการเตรียมฟังก์ชันที่จำเป็นไว้ให้เรียกใช้งานได้ เช่น การเข้าถึงแฟ้มข้อมูลของระบบ การจัดการระบบเครือข่าย การเข้ารหัสข้อมูล และอื่น ๆ

การเขียนโปรแกรมส่วนใหญ่นั้นจะมีการใช้งานไลบรารีที่มีผู้เขียนไว้แล้ว โดยแต่ละภาษาโปรแกรมก็จะมีเครื่องมือจัดการไลบรารีที่แตกต่างกันไป ซึ่ง Node.js นั้นก็มีเครื่องมือที่มาช่วยจัดการไลบรารีด้วยเช่นกัน นั่นคือ npm [8] โดย npm จะถูกติดตั้งมาพร้อมกับ Node.js เพื่อทำหน้าที่ติดตั้ง อัพเดท หรือลบไลบรารีเสริมต่าง ๆ โดยการระบุชื่อให้ตรงกันแล้ว npm จะนำชื่อไปตรวจสอบใน registry เมื่อพบก็จะทำการดาวน์โหลดและติดตั้งให้โดยอัตโนมัติ นอกจากนั้นการนำแอปพลิเคชันที่สร้างขึ้นเองไปเพิ่มไว้ใน registry ของ npm ก็สามารถทำได้เช่นกัน

ดังที่กล่าวไปข้างต้น แอปพลิเคชัน Node.js สามารถทำงานข้ามแพลตฟอร์มได้ โดยไม่ต้องเขียนโปรแกรมใหม่ ซึ่งสามารถทำงานบนเครื่องเซิร์ฟเวอร์ Mac OS X, Microsoft Windows และ Unix ได้ อีกทั้งยังสามารถเขียนด้วยภาษา CoffeeScript, Dart, TypeScript, และภาษาอื่น ๆ ที่สามารถคอมไพล์ไปเป็น JavaScript ได้

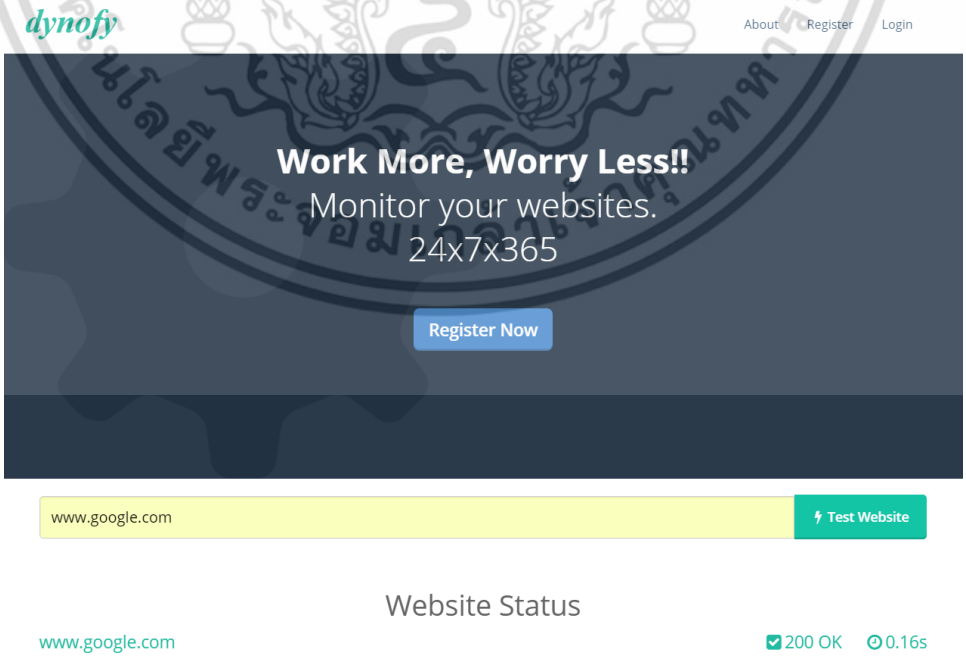
2.6 Angular

Angular [9] เป็นเว็บแอปพลิเคชันเฟรมเวิร์คแบบโอเพนซอร์ซ สำหรับการทำงานด้าน Front-end พัฒนาโดยบริษัท Google เพื่อที่จะตอบสนองความต้องการของการทำแอปพลิเคชันแบบหน้าเดียว ในการทำงานนั้น Angular จะอ่านข้อมูลของ HTML ที่อยู่ในแท็กที่สร้างขึ้นเอง แล้วจึงนำไปเชื่อมต่อกับตัวแปร JavaScript โดยค่าของตัวแปรสามารถกำหนดได้เองในโค้ดโปรแกรม หรือจะดึงข้อมูลมาในรูปแบบ JSON ก็ได้

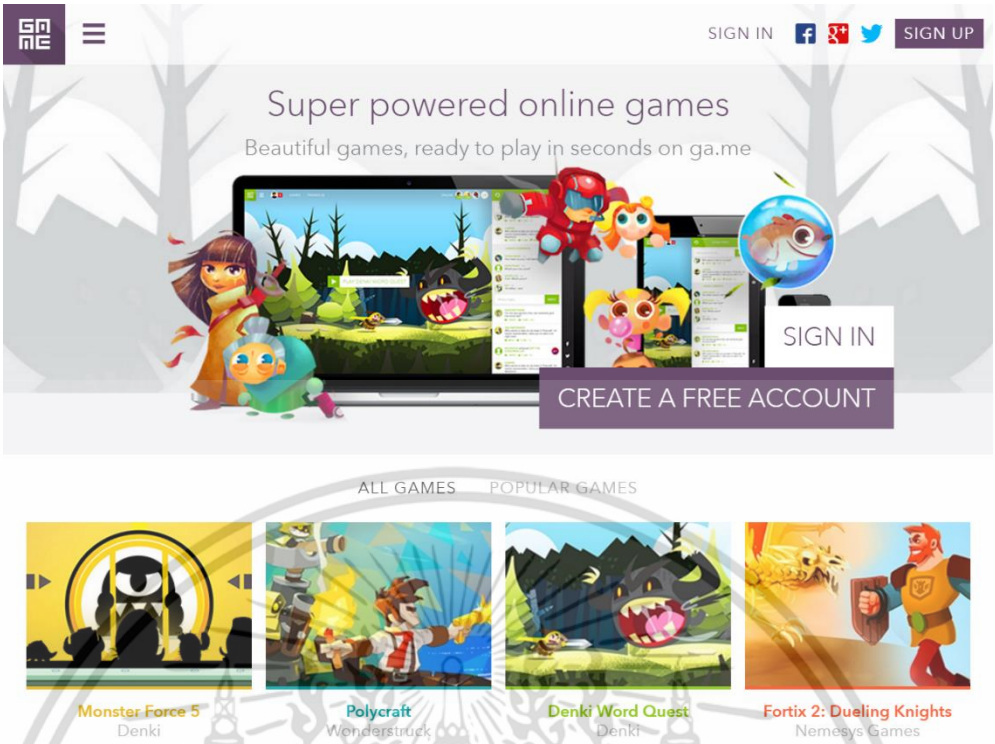
วัตถุประสงค์ของ Angular คือ เพื่อแยกฝั่งไคลเอนต์ออกจากฝั่งเซิร์ฟเวอร์ ทำให้สามารถแยกกันพัฒนาได้ และความสามารถในการนำกลับมาใช้ใหม่ของทั้งสองฝั่ง และเพื่อจัดวางโครงสร้างที่ดีสำหรับแอปพลิเคชัน เพื่อให้ง่ายต่อการออกแบบส่วนต่อประสานกับผู้ใช้ ไปจนถึงการทดสอบระบบและในแง่ของตรรกะทางธุรกิจ

Angular ตั้งแต่เวอร์ชัน 2 เป็นต้นมาจะใช้ภาษา TypeScript เป็นในการพัฒนา ซึ่งเป็นภาษาที่พัฒนาโดยบริษัท Microsoft โดยเป็นการต่อยอดจากภาษา JavaScript สิ่งหลัก ๆ ที่เพิ่มมาคือมีการสนับสนุน Type System ซึ่งโดยปกติแล้ว JavaScript เป็นภาษาที่ไม่มี Type นั่นคือตัวแปรสามารถถูกเปลี่ยนประเภทได้หลังจากประกาศไปแล้ว ทำให้ง่ายต่อการเขียน แต่อาจทำให้เกิดความผิดพลาดจากการใช้งานตัวแปรผิดประเภทได้

ปัจจุบันมีเว็บแอปพลิเคชันมากมายที่ได้นำเฟรมเวิร์ค Angular มาใช้ เช่น dynofy.com ซึ่งเป็นเว็บแอปพลิเคชันสำหรับตรวจสอบสถานะของเว็บไซต์ว่าออนไลน์อยู่หรือไม่ ดังรูปที่ 2.3 และ ga.me เป็นเว็บแอปพลิเคชันที่มีเกมหลากหลายรูปแบบให้ผู้ใช้สามารถเล่นผ่านทางหน้าเว็บได้ ดังรูปที่ 2.4



รูปที่ 2.3 ตัวอย่างหน้าเว็บไซต์ dynofy.com



รูปที่ 2.4 ตัวอย่างหน้าเว็บไซต์ ga.me

2.7 Electron

Electron [10] เป็นแอปพลิเคชันเฟรมเวิร์คแบบโอเพนซอร์ส พัฒนาโดย GitHub ที่ให้นักพัฒนาสามารถสร้างเดสก์ท็อปแอปพลิเคชันโดยใช้ Node.js และภาษา HTML, JavaScript, CSS ในการพัฒนา และใช้ Chromium ซึ่งเป็นเว็บเบราว์เซอร์แบบโอเพนซอร์สของบริษัท Google ในการแสดงผล โดยทั่วไปแล้วจะใช้สำหรับพัฒนาในส่วน Back-end ของเว็บแอปพลิเคชัน เนื่องจาก Electron นั้นมีโครงสร้างเป็นเว็บแอปพลิเคชัน จึงมีความสามารถในการทำงานได้หลายแพลตฟอร์ม ซึ่งในการพัฒนานั้นสามารถเขียนโปรแกรมเพียงครั้งเดียวแล้วนำไปสร้างเป็น executable file สำหรับแพลตฟอร์มนั้น ๆ

โครงสร้างของ Electron จะแบ่งออกเป็น Main Process และ Renderer Process โดยที่ Main Process จะจัดการสร้างหน้าต่างเบราว์เซอร์ขึ้นมา และในแต่ละหน้าต่างจะถูกแสดงผลโดย Renderer Process ของมันเอง หมายความว่าถ้าหน้าต่างใดถูกปิด Renderer Process นั้น ๆ ก็จะถูกทำลายไปด้วย แต่ Main Process จะยังคงอยู่เพื่อจัดการกับหน้าต่างอื่น ๆ

การมาของเฟรมเวิร์ค Electron ทำให้ภาษา JavaScript ได้ก้าวขึ้นไปอีกขั้น เกิดเป็นช่องทางใหม่ในการสร้างเดสก์ท็อปแอปพลิเคชันโดยใช้ภาษาเดียวกับที่ใช้พัฒนาเว็บไซต์ ซึ่งในปัจจุบันมีแอปพลิเคชันจำนวนมากที่ได้ใช้เฟรมเวิร์ค Electron ในการพัฒนา ดังแสดงได้ในรูปที่ 2.5

รูปที่ 2.5 แอปพลิเคชันที่ใช้เฟรมเวิร์ค Electron ในการพัฒนา

บทที่ 3

การวิเคราะห์และออกแบบระบบ

ในขั้นตอนการพัฒนาโปรแกรมตรวจสอบความต้องการของระบบ ได้มีการวิเคราะห์ปัญหาของระบบเดิม ซึ่งเป็นที่มาของการพัฒนาระบบใหม่ รวมถึงการออกแบบแผนภาพที่จำเป็น ได้แก่ แผนภาพแสดงการทำงานของผู้ใช้ระบบ แผนภาพลำดับการทำงานของระบบ แผนภาพบริบท และแผนภาพกระแสข้อมูล จากนั้นจึงออกแบบส่วนติดต่อผู้ใช้

3.1 การวิเคราะห์ระบบเดิม

3.1.1 ปัญหาของระบบเดิม

เนื่องจากในการพัฒนาโปรแกรมโดยทั่วไปนั้น หากต้องการเพิ่มฟังก์ชันที่จะตรวจสอบทรัพยากรของระบบว่าเพียงพอต่อความต้องการของโปรแกรมหรือไม่ จะต้องเขียนโปรแกรมเพิ่มในขั้นตอนของการพัฒนาโปรแกรม ซึ่งความต้องการของโปรแกรมนั้นอาจมีการเปลี่ยนแปลงได้ตลอดเวลา ตามความเหมาะสมของระบบและเทคโนโลยี จึงต้องมีการแก้ไขปรับเปลี่ยนอยู่เสมอ โดยหากต้องการเพิ่มกรณีทดสอบใหม่ ๆ หรือปรับแก้เงื่อนไขเดิม จะต้องทำการแก้ไขที่ซอร์สโค้ดของโปรแกรม ซึ่งจะต้องให้นักพัฒนาเป็นผู้แก้ไข ทำให้เสียค่าใช้จ่ายในการบำรุงรักษา และฟังก์ชันการตรวจสอบทรัพยากรระบบแบบที่ฝังอยู่ในโปรแกรมหลักนั้นจะมีข้อจำกัดในการทำงานที่สามารถทำงานได้บนแพลตฟอร์มที่โปรแกรมหลักรองรับเท่านั้น

3.1.2 วิธีการแก้ไขปัญหา

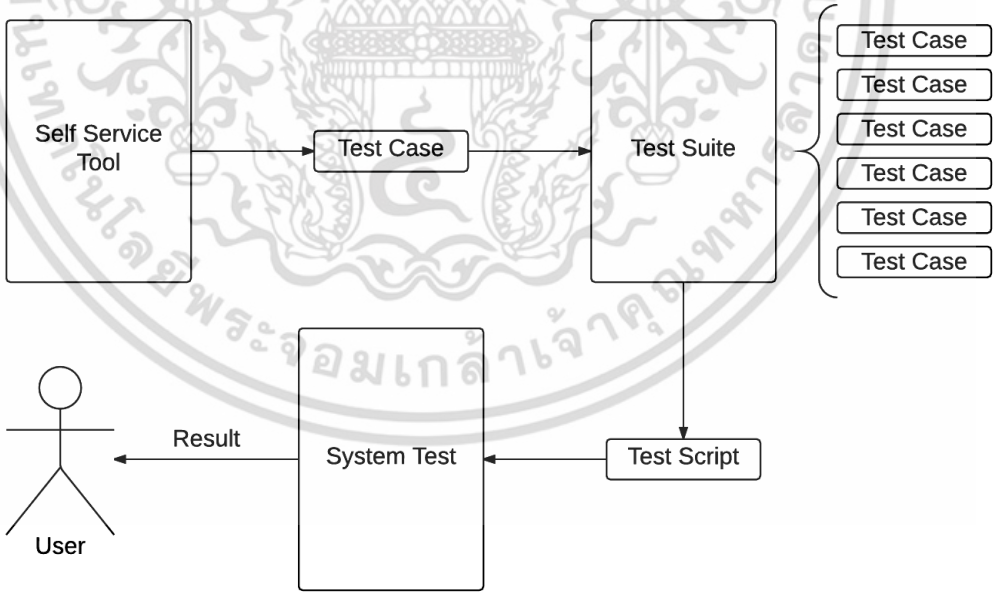
จากปัญหาข้างต้น จึงได้มีการคิดหาวิธีที่จะทำให้การเพิ่มหรือแก้ไขเงื่อนไขของการตรวจสอบสามารถทำได้โดยใครก็ได้ ซึ่งไม่จำเป็นต้องมีความรู้ทางด้านการเขียนโปรแกรม และทำการแยกฟังก์ชันการทำงานของการตรวจสอบทรัพยากรระบบออกจากตัวโปรแกรมหลัก เพื่อให้ทั้งสองโปรแกรมไม่มีความเกี่ยวข้องกัน ทำให้สามารถแยกกันพัฒนาได้ อีกทั้งยังสามารถทำงานได้หลายแพลตฟอร์ม โดยไม่ขึ้นกับโปรแกรมหลัก

3.2 การวิเคราะห์ระบบใหม่

ในการออกแบบระบบใหม่จะแบ่งเงื่อนไขในการตรวจสอบออกเป็นกรณีทดสอบ ซึ่งจะตรวจสอบเฉพาะด้านใดด้านหนึ่งเท่านั้น โดยกรณีทดสอบที่จะนำไปใช้งานจริง จะถูกเก็บอยู่ในชุดทดสอบ การออกแบบโครงสร้างในลักษณะนี้จะมีข้อดีคือสามารถแก้ไขกรณีทดสอบได้ง่าย และไม่กระทบกับกรณีทดสอบอื่น ๆ และเพื่อให้สนับสนุนการทำงานข้ามแพลตฟอร์มจะต้องใช้เครื่องมือหรือโปรแกรมที่ออกแบบมาสำหรับงานลักษณะนี้ จึงได้เลือกใช้เฟรมเวิร์ค Electron ซึ่งสามารถสร้างโปรแกรมบนเดสก์ท็อปที่สามารถทำงานได้หลายแพลตฟอร์ม โดยในระบบใหม่นี้จะมุ่งเน้นให้ผู้ใช้งาน

สามารถที่จะสร้างหรือกำหนดเงื่อนไขได้ด้วยตนเอง โดยที่ไม่จำเป็นจะต้องมีความรู้ทางด้านการเขียนโปรแกรมมาก่อนก็สามารถใช้งานได้ ซึ่งจะต้องออกแบบส่วนติดต่อผู้ใช้ให้สามารถเข้าใจได้ง่าย เรียนรู้ได้เร็ว และมีประสิทธิภาพ โดยได้มีการปรับโครงสร้างของระบบใหม่ ทำให้เมื่อมีการปรับเปลี่ยนเงื่อนไขหรือกรณีทดสอบจะไม่ต้องเข้าไปเปลี่ยนแปลงซอร์สโค้ด และในระบบใหม่นี้จะมีฟังก์ชันในการสร้างหน่วยทดสอบสำหรับกรณีทดสอบที่มีขนาดใหญ่มาก ๆ ซึ่งกรณีทดสอบที่มีเงื่อนไขจำนวนมากจะทำให้การทดสอบให้ครอบคลุมทุกเงื่อนไขนั้นทำได้ยากขึ้น จึงมีหน่วยทดสอบเพื่อให้สามารถจำลองค่าต่าง ๆ ของระบบได้ และทำการทดสอบได้ว่าแต่ละเงื่อนไขนั้นได้ผลลัพธ์ออกมาตรงตามที่ต้องการหรือไม่

ในระบบใหม่นี้จะแบ่งออกเป็น 3 โปรแกรมย่อย ได้แก่ Self Service Tool, Test Suite และ System Test ซึ่งจะทำงานเป็นขั้นตอน โดยเริ่มจากการสร้างกรณีทดสอบโดยใช้ Self Service Tool ซึ่งเป็นโปรแกรมที่ให้ผู้ใช้สามารถสร้างกรณีทดสอบได้ด้วยตนเอง โดยมีส่วนต่อประสานกราฟิกกับผู้ใช้ที่เข้าใจง่าย ผู้ใช้ไม่จำเป็นต้องมีความรู้ในด้านการเขียนโปรแกรมก็สามารถใช้งานได้ หลังจากสร้างกรณีทดสอบจนเพียงพอแล้วก็จะนำกรณีทดสอบเหล่านั้นมารวมเป็นชุดทดสอบด้วยโปรแกรมที่ชื่อว่า Test Suite ซึ่งสามารถเพิ่มกรณีทดสอบและจัดลำดับของกรณีทดสอบได้ว่าควรจะทำกรณีใดก่อนและหลัง และสามารถสร้างสคริปต์จากชุดทดสอบเพื่อนำไปใช้งานจริงได้ และโปรแกรมสุดท้ายคือ System Test ซึ่งทำหน้าที่ทดสอบระบบจากสคริปต์ที่สร้างมา พร้อมทั้งแจ้งผลลัพธ์ของแต่ละกรณีทดสอบ โดยลำดับการทำงานของโปรแกรมทั้งสามนั้นสามารถแสดงได้ดังรูปที่ 3.1

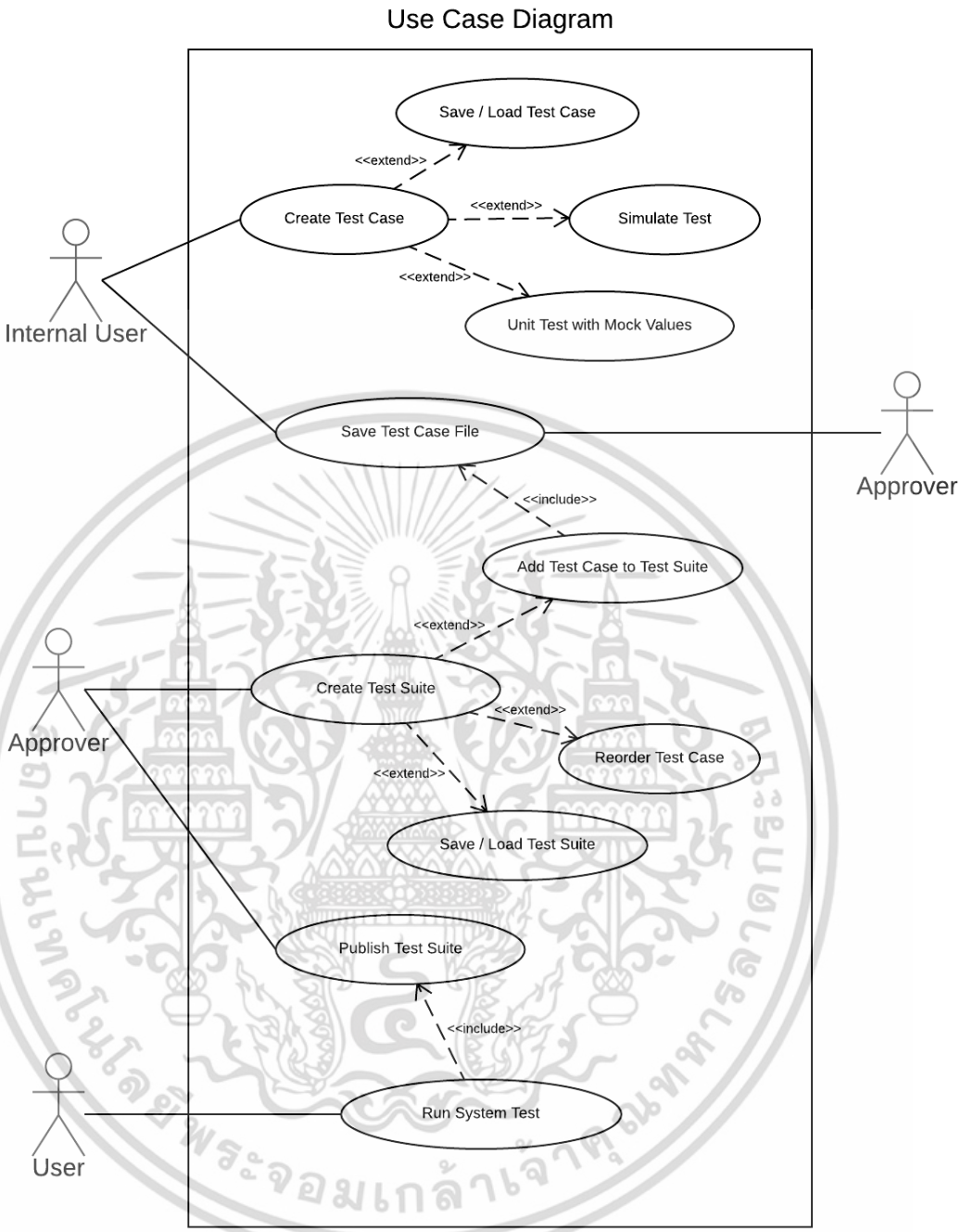


รูปที่ 3.1 ลำดับการทำงานของโปรแกรม

3.3 แผนภาพแสดงการทำงานของผู้ใช้ระบบ (Use Case Diagram)

การเข้าใช้งานของผู้ใช้จะเริ่มจากการสร้างกรณีทดสอบ โดยเมื่อมีความต้องการที่จะเพิ่มเงื่อนไขเพื่อตรวจสอบก็จะเปิดโปรแกรม Self Service Tool ขึ้นมาเพื่อทำการสร้างกรณีทดสอบด้วยตนเอง เมื่อสร้างแล้วสามารถบันทึกไว้ในที่เก็บข้อมูลของผู้ใช้ได้ เพื่อที่สามารถเปิดมาทำต่อได้หากต้องการแก้ไข หรือสร้างเงื่อนไขเพิ่มเติม หลังจากสร้างกรณีทดสอบแล้วจะต้องทำการทดสอบว่ากรณีทดสอบที่สร้างมานั้นจะให้ผลลัพธ์เป็นไปตามที่ต้องการหรือไม่ ซึ่งในขั้นตอนนี้ก็จะมีเครื่องมือที่ช่วยให้ผู้ใช้สามารถสั่งให้กรณีทดสอบนี้ทำงานบนระบบคอมพิวเตอร์ของผู้ใช้เอง แต่ในบางครั้งก็อาจจะไม่สามารถทดสอบให้ครบทุกเงื่อนไขได้เนื่องจากทรัพยากรนั้นมีจำกัด จึงได้มีเครื่องมือที่สามารถจำลองค่าต่าง ๆ ของระบบได้เองโดยที่ไม่ต้องนำไปใช้งานจริง ซึ่งจะสามารถกำหนดค่าทางฮาร์ดแวร์ได้ตั้งแต่ความเร็วของหน่วยประมวลผลกลาง ขนาดของหน่วยความจำหลัก ความเร็วของหน่วยความจำหลัก ขนาดของที่เก็บข้อมูลสำรอง หรือทางด้านซอฟต์แวร์ เช่น แพลตฟอร์ม รุ่นของระบบปฏิบัติการ หรือสถาปัตยกรรมของระบบปฏิบัติการ โปรแกรม Microsoft Office ที่ติดตั้งอยู่ เป็นต้น โดยหลังจากที่ทดสอบจนมั่นใจว่ากรณีทดสอบนี้มีความถูกต้องครบถ้วนแล้ว ก็ดำเนินการส่งกรณีทดสอบไปยังผู้ตรวจสอบเพื่อทำการยืนยันว่ากรณีทดสอบนี้มีความปลอดภัยต่อระบบ

จากนั้นเมื่อตรวจสอบผ่านแล้วก็จะทำการเพิ่มกรณีทดสอบนี้เข้าไปในชุดทดสอบ โดยใช้โปรแกรม Test Suite เพื่อทำการสร้างชุดทดสอบใหม่ หรือเพิ่มกรณีทดสอบใหม่เข้าไปในชุดทดสอบที่มีอยู่แล้ว พร้อมทั้งดำเนินการจัดลำดับของกรณีทดสอบว่าควรทดสอบกรณีใดก่อนและหลัง และสามารถทำการบันทึกชุดทดสอบที่กำลังแก้ไขอยู่ได้เช่นเดียวกับกรณีทดสอบ เพื่อให้สามารถกลับมาแก้ไขได้ในภายหลัง จากนั้นเมื่อชุดทดสอบพร้อมที่จะนำไปใช้งานแล้วก็จะทำการสร้างเป็นสคริปต์เพื่อให้สามารถนำไปใช้งานได้ที่โปรแกรม System Test ซึ่งโปรแกรมนี้จะนำสคริปต์ที่ได้มาทำการประมวลผล โดยจะเก็บข้อมูลทรัพยากรของระบบที่ของผู้ใช้งานเท่าที่จำเป็น แล้วนำมาเปรียบเทียบตามเงื่อนไขของแต่ละกรณีทดสอบ แล้วจึงแสดงผลลัพธ์ว่ามีกรณีทดสอบใดบ้างที่ผ่านและไม่ผ่านการทดสอบ โดยแผนภาพแสดงการทำงานของผู้ใช้ระบบ แสดงได้ดังรูปที่ 3.2



รูปที่ 3.2 แผนภาพแสดงการทำงานของผู้ใช้ระบบ

จากแผนภาพแสดงการทำงานของผู้ใช้ระบบในรูปที่ 3.2 สามารถเขียนคำอธิบายได้ดังตารางที่ 3.1 ถึงตารางที่ 3.11

ตารางที่ 3.1 คำอธิบายสำหรับการสร้างกรณีทดสอบ

หัวข้อ	**คำอธิบาย**
Use Case Name	สร้างกรณีทดสอบ (Create Test Case)
Scenario	ผู้ใช้งานต้องการสร้างกรณีทดสอบ
Description	เมื่อผู้ใช้งานเปิดโปรแกรม Self Service Tool ขึ้นมาจะปรากฏหน้าต่างสำหรับสร้างกรณีทดสอบพร้อมทั้งเครื่องมือต่าง ๆ ที่อำนวยความสะดวกในการทดสอบกรณีทดสอบที่กำลังสร้าง
Trigger	ผู้ใช้งานต้องการสร้างกรณีทดสอบ
Actor	Internal User
Pre-Conditions	ผู้ใช้งานจะต้องเป็นผู้ใช้ภายในเท่านั้น ซึ่งมีสิทธิ์ในการสร้างกรณีทดสอบ
Post-Conditions	โปรแกรมจะแสดงหน้าต่างของกรณีทดสอบที่สร้างขึ้น
Flow	1. ผู้ใช้งานเปิดโปรแกรม Self Service Tool 2. เลือกแพลตฟอร์มสำหรับกรณีทดสอบ 3. เพิ่มเงื่อนไขในการทดสอบ 4. แสดงข้อมูลทั้งหมดของกรณีทดสอบ

ตารางที่ 3.2 คำอธิบายสำหรับการแก้ไขกรณีทดสอบ

หัวข้อ	**คำอธิบาย**
Use Case Name	แก้ไขกรณีทดสอบ (Save / Load Test Case)
Scenario	ผู้ใช้งานต้องการแก้ไขกรณีทดสอบ
Description	เมื่อผู้ใช้งานต้องการเปิดกรณีทดสอบที่ได้สร้างไว้ขึ้นมาเพื่อแก้ไข โดยกดปุ่ม Load จะปรากฏหน้าต่างสำหรับเลือกกรณีทดสอบที่ต้องการแก้ไข หลังจากที่แก้ไขเสร็จแล้วกดปุ่ม Save จะปรากฏหน้าต่างสำหรับบันทึกกรณีทดสอบ และยืนยันการบันทึก
Trigger	ผู้ใช้งานต้องการแก้ไขกรณีทดสอบ
Actor	Internal User
Pre-Conditions	ผู้ใช้งานจะต้องเป็นผู้ใช้ภายในเท่านั้น ซึ่งมีสิทธิ์ในการแก้ไขกรณีทดสอบ และจะต้องมีไฟล์กรณีทดสอบที่ได้สร้างไว้แล้ว
Post-Conditions	โปรแกรมจะบันทึกไฟล์กรณีทดสอบไว้ที่คอมพิวเตอร์ของผู้ใช้
Flow	1. ผู้ใช้งานเปิดโปรแกรม Self Service Tool 2. กดปุ่ม Load 3. เลือกกรณีทดสอบที่ต้องการแก้ไข 4. แก้ไขกรณีทดสอบ 5. กดปุ่ม Save 6. ยืนยันการบันทึก

ตารางที่ 3.3 คำอธิบายสำหรับการจำลองการทดสอบ

หัวข้อ	**คำอธิบาย**
Use Case Name	จำลองการทดสอบ (Simulate Test)
Scenario	ผู้ใช้งานต้องการจำลองการทำงานของกรณีทดสอบ
Description	เมื่อผู้ใช้งานต้องการจำลองการทำงานของกรณีทดสอบด้วยคอมพิวเตอร์ของผู้ใช้ โดยกดปุ่ม Run ระบบจะทำการสร้างสคริปต์ทดสอบแล้วเรียกใช้งานทันที จากนั้นจึงแสดงผลลัพธ์ของการทดสอบ
Trigger	ผู้ใช้งานต้องการจำลองการทำงานของกรณีทดสอบ
Actor	Internal User
Pre-Conditions	ผู้ใช้งานจะต้องเป็นผู้ใช้ภายในเท่านั้น
Post-Conditions	โปรแกรมจะแสดงผลลัพธ์ของการทดสอบ
Flow	1. ผู้ใช้งานกดปุ่ม Run 2. สร้างสคริปต์ทดสอบจากกรณีทดสอบ 3. สั่งให้สคริปต์ทดสอบทำงาน 4. แสดงผลลัพธ์การทดสอบ

ตารางที่ 3.4 คำอธิบายสำหรับการสร้างหน่วยทดสอบด้วยค่าจำลอง

หัวข้อ	**คำอธิบาย**
Use Case Name	สร้างหน่วยทดสอบด้วยค่าจำลอง (Unit Test with Mock Values)
Scenario	ผู้ใช้งานต้องการสร้างหน่วยทดสอบด้วยค่าจำลอง
Description	เมื่อผู้ใช้งานต้องการกำหนดค่าของระบบคอมพิวเตอร์ที่จะนำมาใช้ทดสอบ โดยเลือกไปที่หน้าต่าง Test Panel จากนั้นกดปุ่ม Add New Mock Case เพื่อเพิ่มระบบใหม่ และกำหนดค่าของระบบตามต้องการ แล้วกดปุ่ม Run ทางด้านขวา หรือปุ่ม All เพื่อทดสอบระบบทั้งหมดในครั้งเดียว
Trigger	ผู้ใช้งานต้องการสร้างหน่วยทดสอบด้วยค่าจำลอง
Actor	Internal User
Pre-Conditions	ผู้ใช้งานจะต้องเป็นผู้ใช้ภายในเท่านั้น และกรณีทดสอบจะต้องมีอย่างน้อย 1 เงื่อนไขเพื่อให้สามารถจำลองค่าได้
Post-Conditions	โปรแกรมจะแสดงผลลัพธ์ของการทดสอบด้วยค่าจำลองของแต่ละระบบ
Flow	1. ผู้ใช้งานสลับไปยังหน้าต่าง Test Panel 2. กดปุ่ม Add New Mock Case 3. กำหนดค่าต่าง ๆ ของระบบ 4. กดปุ่ม Run หรือปุ่ม All 5. แสดงผลลัพธ์การทดสอบของแต่ละระบบ

ตารางที่ 3.5 คำอธิบายสำหรับการบันทึกกรณีทดสอบ

หัวข้อ	**คำอธิบาย**
Use Case Name	บันทึกกรณีทดสอบ (Save Test Case File)
Scenario	ผู้ใช้งานต้องการบันทึกกรณีทดสอบ
Description	เมื่อผู้ใช้งานต้องการส่งกรณีทดสอบที่สมบูรณ์แล้วไปรวมเป็นชุดทดสอบ โดยส่งไฟล์กรณีทดสอบไปยังผู้ตรวจสอบ
Trigger	ผู้ใช้งานต้องการบันทึกกรณีทดสอบ
Actor	Internal User, Approver
Pre-Conditions	ผู้ใช้งานจะต้องเป็นผู้ใช้ภายในเท่านั้น และต้องมีไฟล์กรณีทดสอบ
Post-Conditions	ได้ผลลัพธ์การตรวจสอบว่ากรณีทดสอบนั้นผ่านหรือไม่
Flow	1. ผู้ใช้งานเลือกกรณีทดสอบที่ต้องการ 2. ส่งไปยังผู้ตรวจสอบ 3. รอผลลัพธ์การตรวจสอบ

ตารางที่ 3.6 คำอธิบายสำหรับการสร้างชุดทดสอบ

หัวข้อ	**คำอธิบาย**
Use Case Name	สร้างชุดทดสอบ (Create Test Suite)
Scenario	ผู้ใช้งานต้องการสร้างชุดทดสอบ
Description	เมื่อผู้ใช้งานเปิดโปรแกรม Test Suite ขึ้นมาจะปรากฏหน้าต่างสำหรับสร้างชุดทดสอบ
Trigger	ผู้ใช้งานต้องการสร้างชุดทดสอบ
Actor	Approver
Pre-Conditions	ผู้ใช้งานจะต้องเป็นผู้ที่มีสิทธิ์ในการจัดการชุดทดสอบ
Post-Conditions	โปรแกรมจะแสดงหน้าต่างของชุดทดสอบที่สร้างขึ้น
Flow	1. ผู้ใช้งานเปิดโปรแกรม Test Suite 2. แสดงหน้าต่างสำหรับสร้างชุดทดสอบ

ตารางที่ 3.7 คำอธิบายสำหรับการเพิ่มกรณีทดสอบไปยังชุดทดสอบ

หัวข้อ	**คำอธิบาย**
Use Case Name	เพิ่มกรณีทดสอบไปยังชุดทดสอบ (Add Test Case to Test Suite)
Scenario	ผู้ใช้งานต้องการเพิ่มกรณีทดสอบไปยังชุดทดสอบ
Description	เมื่อผู้ใช้งานต้องการเพิ่มกรณีทดสอบที่สมบูรณ์แล้วไปยังชุดทดสอบ โดยการกดปุ่ม Add Existing Test Case จะปรากฏหน้าต่างให้เลือกกรณีทดสอบ จากนั้นกดปุ่มยืนยัน
Trigger	ผู้ใช้งานต้องการเพิ่มกรณีทดสอบไปยังชุดทดสอบ
Actor	Approver
Pre-Conditions	ผู้ใช้งานจะต้องเป็นผู้ที่มีสิทธิ์ในการจัดการชุดทดสอบ
Post-Conditions	โปรแกรมจะเพิ่มกรณีทดสอบที่ได้เลือกไว้ไปยังชุดทดสอบ
Flow	1. ผู้ใช้งานกดปุ่ม Add Existing Test Case 2. เลือกกรณีทดสอบที่ต้องการเพิ่ม 3. กดปุ่มยืนยันการเลือก 4. แสดงรายการกรณีทดสอบของชุดทดสอบ

ตารางที่ 3.8 คำอธิบายสำหรับการจัดลำดับกรณีทดสอบ

หัวข้อ	**คำอธิบาย**
Use Case Name	จัดลำดับกรณีทดสอบ (Reorder Test Case)
Scenario	ผู้ใช้งานต้องการจัดลำดับกรณีทดสอบ
Description	เมื่อผู้ใช้งานต้องการจัดลำดับกรณีทดสอบ โดยเลือกกรณีทดสอบที่ต้องการเปลี่ยนลำดับ จากนั้นกดปุ่ม Ctrl และลูกศรขึ้นหรือลงที่แป้นพิมพ์ กรณีทดสอบจะถูกเลื่อนลำดับขึ้นหรือลง 1 ลำดับ และแสดงรายการของกรณีทดสอบ
Trigger	ผู้ใช้งานต้องการจัดลำดับกรณีทดสอบ
Actor	Approver
Pre-Conditions	ผู้ใช้งานจะต้องเป็นผู้ที่มีสิทธิ์ในการจัดการชุดทดสอบ และในชุดทดสอบต้องมีอย่างน้อย 2 กรณีทดสอบเพื่อให้สามารถจัดลำดับได้
Post-Conditions	โปรแกรมจะแสดงรายการของกรณีทดสอบหลังจัดลำดับแล้ว
Flow	1. ผู้ใช้งานเลือกกรณีทดสอบที่ต้องการเปลี่ยนลำดับ 2. กดปุ่ม Ctrl ตามด้วยลูกศรขึ้นหรือลง 3. จัดลำดับกรณีทดสอบ 4. แสดงรายการกรณีทดสอบหลังจัดลำดับ

ตารางที่ 3.9 คำอธิบายสำหรับการแก้ไขชุดทดสอบ

หัวข้อ	**คำอธิบาย**
Use Case Name	แก้ไขชุดทดสอบ (Save / Load Test Suite)
Scenario	ผู้ใช้งานต้องการแก้ไขชุดทดสอบ
Description	เมื่อผู้ใช้งานต้องการเปิดชุดทดสอบขึ้นมาเพื่อแก้ไข โดยกดปุ่ม Load จะปรากฏหน้าต่างสำหรับเลือกชุดทดสอบ หลังจากที่แก้ไขเสร็จแล้วกดปุ่ม Save จะปรากฏหน้าต่างสำหรับบันทึกชุดทดสอบ และยืนยันการบันทึก
Trigger	ผู้ใช้งานต้องการแก้ไขชุดทดสอบ
Actor	Approver
Pre-Conditions	ผู้ใช้งานจะต้องเป็นผู้ที่มีสิทธิ์ในการจัดการชุดทดสอบ
Post-Conditions	โปรแกรมจะบันทึกไฟล์ชุดทดสอบไว้ที่คอมพิวเตอร์ของผู้ใช้ และต้องมีไฟล์ชุดทดสอบ
Flow	1. ผู้ใช้งานเปิดโปรแกรม Test Suite 2. กดปุ่ม Load 3. เลือกชุดทดสอบที่ต้องการ 4. แก้ไขชุดทดสอบ 5. กดปุ่ม Save 6. ยืนยันการบันทึก

ตารางที่ 3.10 คำอธิบายสำหรับการสร้างสคริปต์ทดสอบ

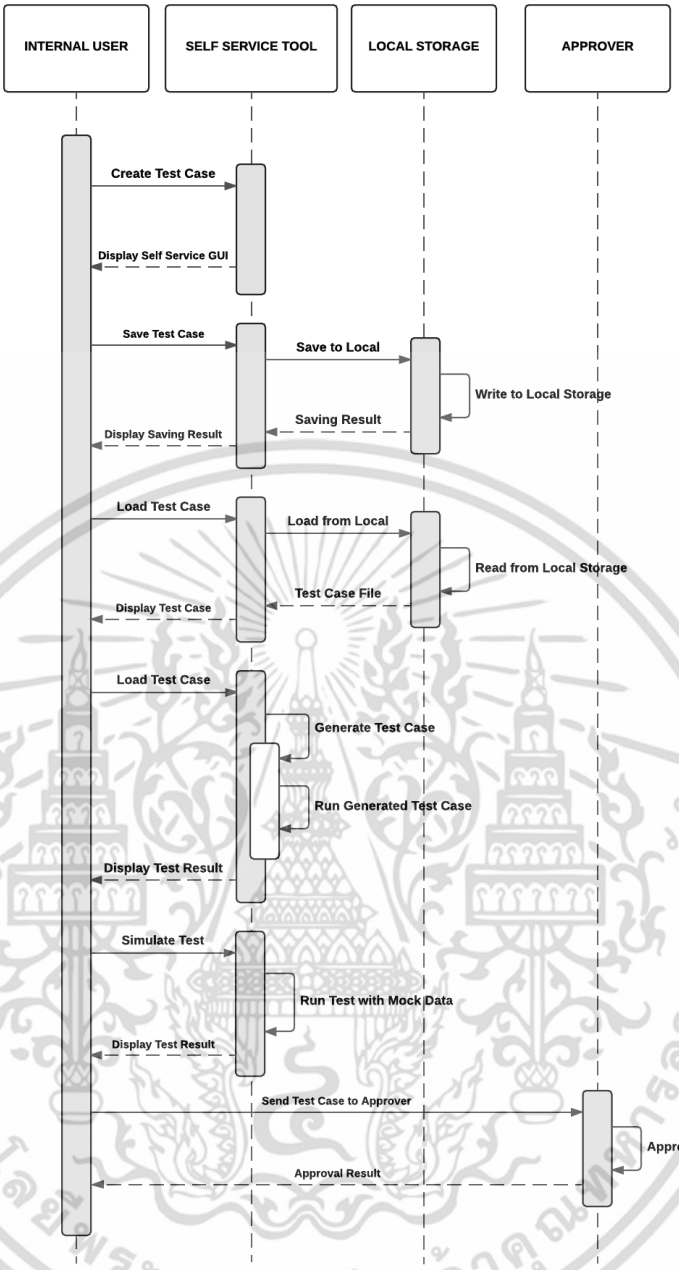
หัวข้อ	**คำอธิบาย**
Use Case Name	สร้างสคริปต์ทดสอบ (Publish Test Suite)
Scenario	ผู้ใช้งานต้องการสร้างสคริปต์ทดสอบ
Description	เมื่อผู้ใช้งานต้องการนำชุดทดสอบที่สมบูรณ์แล้วไปใช้งานจริง โดยกดปุ่ม Publish โปรแกรมจะสร้างสคริปต์ทดสอบจากชุดทดสอบ และนำไปบันทึกไว้ที่เซิร์ฟเวอร์
Trigger	ผู้ใช้งานต้องการสร้างสคริปต์ทดสอบ
Actor	Approver
Pre-Conditions	ผู้ใช้งานจะต้องเป็นผู้ที่มีสิทธิ์ในการจัดการชุดทดสอบ และในชุดทดสอบต้องมีอย่างน้อย 2 กรณีทดสอบเพื่อให้สคริปต์สามารถทำงานได้
Post-Conditions	โปรแกรมจะบันทึกสคริปต์ทดสอบไว้ที่เซิร์ฟเวอร์
Flow	1. ผู้ใช้งานเลือกกรณีทดสอบที่ต้องการ 2. ส่งไปยังผู้ตรวจสอบ 3. รอผลลัพธ์การตรวจสอบ

ตารางที่ 3.11 คำอธิบายสำหรับการทดสอบระบบ

หัวข้อ	**คำอธิบาย**
Use Case Name	ทดสอบระบบ (Run System Test)
Scenario	ผู้ใช้งานต้องการทดสอบระบบ
Description	เมื่อผู้ใช้งานต้องการทดสอบระบบ โดยเปิดโปรแกรม System Test โปรแกรมจะไปโหลดไฟล์สคริปต์ทดสอบที่เก็บไว้ที่เซิร์ฟเวอร์แล้วสั่งให้ทำงาน จากนั้นจึงแสดงผลลัพธ์ของการทดสอบ
Trigger	ผู้ใช้งานต้องการทดสอบระบบ
Actor	User
Pre-Conditions	ต้องมีไฟล์สคริปต์ทดสอบที่สามารถทำงานได้
Post-Conditions	โปรแกรมจะแสดงผลลัพธ์ของการทดสอบ
Flow	1. ผู้ใช้งานเปิดโปรแกรม System Test 2. โหลดสคริปต์ทดสอบจากเซิร์ฟเวอร์ 3. สั่งให้สคริปต์ทดสอบทำงาน 4. แสดงผลลัพธ์การทดสอบ

3.4 แผนภาพลำดับการทำงานของระบบ (Sequence Diagram)

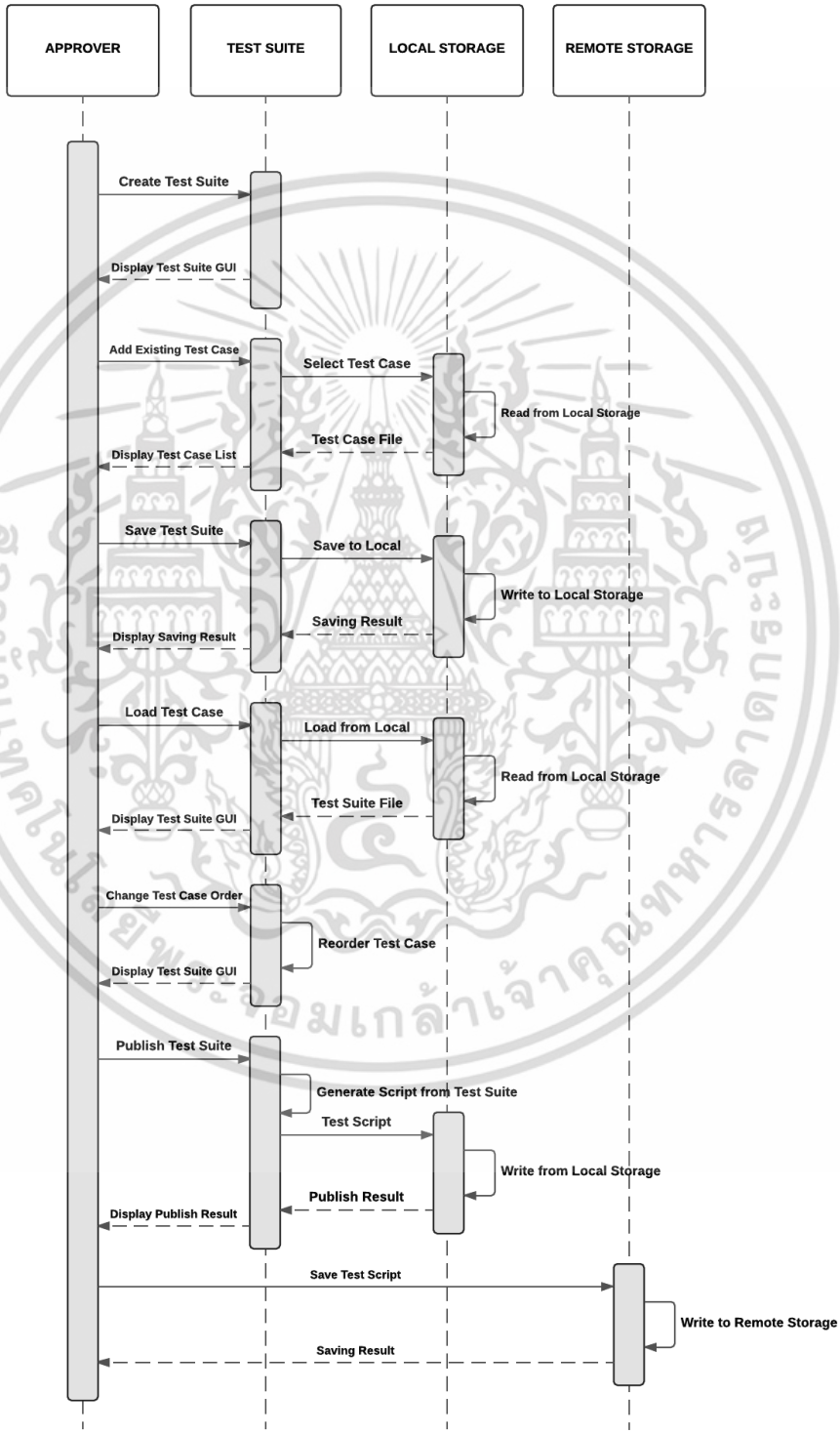
จากแผนภาพแสดงการทำงานของผู้ใช้ระบบในรูปที่ 3.2 สามารถนำมาสร้างเป็นแผนภาพลำดับการทำงานของระบบ ได้ดังรูปที่ 3.3 ถึงรูปที่ 3.5



รูปที่ 3.3 แผนภาพลำดับการทำงานของโปรแกรม Self Service Tool

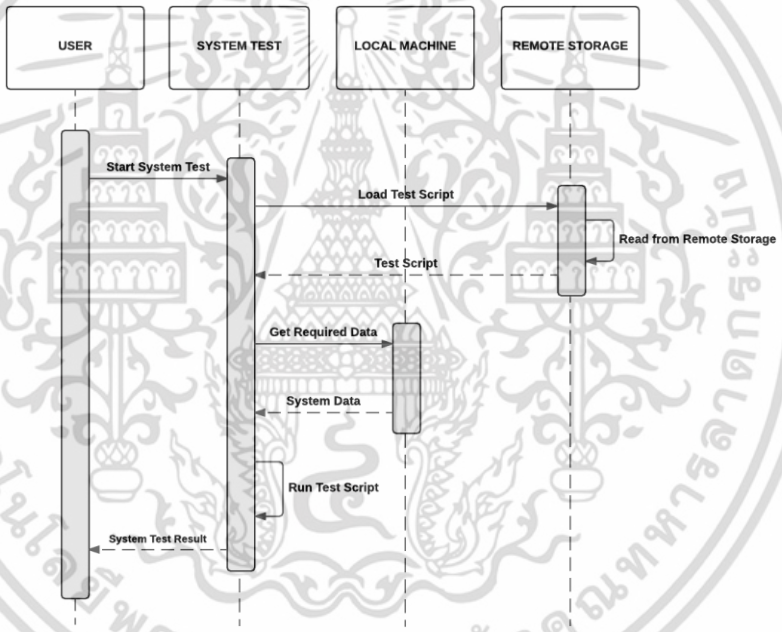
จากรูปที่ 3.3 เป็นการลำดับทำงานของโปรแกรม Self Service Tool โดยเริ่มจากการสร้างกรณีทดสอบ โปรแกรมจะแสดงหน้าต่างเครื่องมือสำหรับการสร้างกรณีทดสอบ ซึ่งผู้ใช้สามารถสร้างกรณีทดสอบตามเงื่อนไขที่ต้องการได้ จากนั้นจะเป็นการบันทึกกรณีทดสอบที่ได้สร้างไว้ ซึ่งการบันทึกนั้นเป็นการบันทึกไปยังที่เก็บข้อมูลของผู้ใช้ แล้วแสดงผลลัพธ์ของการบันทึก ซึ่งเมื่อมีการบันทึกแล้วจึงต้องมีการเปิดกรณีทดสอบที่ได้บันทึกไว้ เพื่อทำการแก้ไข โดยจะเป็นการนำไฟล์กรณีทดสอบมาจากที่เก็บข้อมูลของผู้ใช้เช่นกัน พร้อมทั้งแสดงหน้าต่างแก้ไขกรณีทดสอบ เมื่อต้องการทดสอบกรณีทดสอบที่ได้สร้างขึ้น ผู้ใช้สามารถเรียกใช้งานฟังก์ชันทดสอบการทำงานของกรณีทดสอบได้ โดยจะสร้างสคริปต์ทดสอบที่สามารถทำงานได้จริงจากกรณีทดสอบนั้น หลังจากนั้นจึงสั่งให้สคริปต์ทดสอบ

ทำงาน แล้วจึงแจ้งผลลัพธ์ไปยังผู้ใช้ ฟังก์ชันถัดมาคือการจำลองการทดสอบ โดยการระบุค่าจำลองของระบบที่ต้องการทดสอบ จากนั้นจึงทำการทดสอบแล้วแสดงผลลัพธ์ให้ผู้ใช้ทราบ และลำดับสุดท้ายคือการส่งกรณีทดสอบไปยังผู้ตรวจสอบ ซึ่งหลังจากได้รับการตรวจสอบและยืนยันความถูกต้องแล้วก็จะแจ้งผลลัพธ์กลับมาให้ผู้ใช้ทราบ



รูปที่ 3.4 แผนภาพลำดับการทำงานของโปรแกรม Test Suite

จากรูปที่ 3.4 เป็นลำดับการทำงานของโปรแกรม Test Suite ซึ่งผู้ใช้งานโปรแกรมนี้จะต้องเป็นผู้ที่มีหน้าที่ในการรวบรวมกรณีทดสอบเข้าไว้ด้วยกันเพื่อสร้างเป็นชุดทดสอบ โดยเริ่มจากการเปิดโปรแกรมเพื่อสร้างชุดทดสอบ โปรแกรมก็จะแสดงหน้าต่างเครื่องมือสำหรับการสร้างชุดทดสอบขึ้น จากนั้นจึงทำการเพิ่มกรณีทดสอบเข้ามาในโปรแกรมจากที่เก็บข้อมูลของผู้ใช้ โดยแสดงเป็นรายการของกรณีทดสอบ โปรแกรม Test Suite นั้นมีฟังก์ชันการบันทึกและแก้ไขเช่นเดียวกับโปรแกรม Self Service Tool โดยสามารถบันทึกชุดทดสอบและเปิดเพื่อแก้ไขได้เช่นกัน ถัดมาเป็นการจัดลำดับกรณีทดสอบ โดยเมื่อผู้ใช้ทำการจัดลำดับกรณีทดสอบ โปรแกรม Test Suite จะปรับลำดับของรายการกรณีทดสอบ แล้วแสดงผลลัพธ์ไปยังผู้ใช้ ฟังก์ชันถัดมาคือการแปลงชุดทดสอบให้เป็นสคริปต์ทดสอบ โดยเมื่อมีการสั่งให้แปลงชุดทดสอบ โปรแกรม Test Suite ก็จะทำการสร้างสคริปต์ทดสอบขึ้นมาแล้วทำการบันทึกไปยังที่เก็บข้อมูลของผู้ใช้ และสุดท้ายคือการส่งสคริปต์ทดสอบไปเก็บไว้ที่เซิร์ฟเวอร์ จากนั้นจึงแจ้งผลลัพธ์ให้ผู้ใช้ทราบ

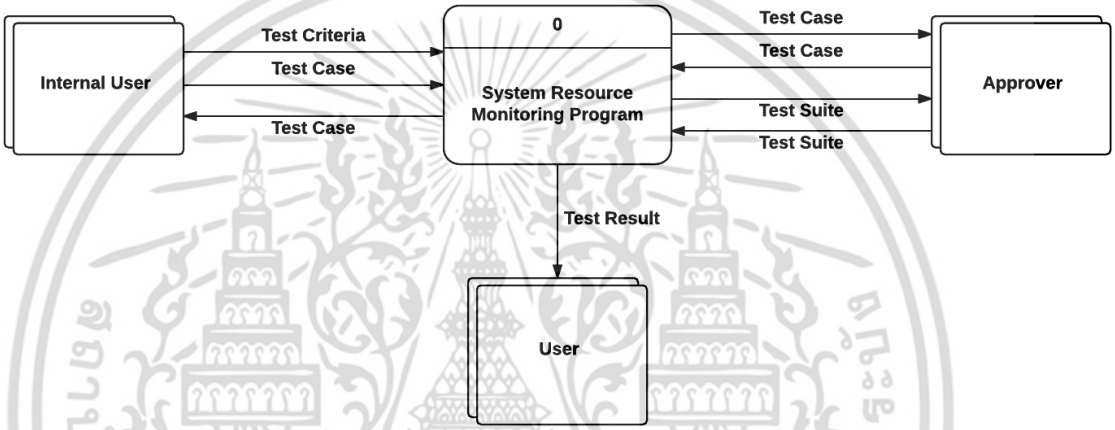


รูปที่ 3.5 แผนภาพลำดับการทำงานของโปรแกรม System Test

จากรูปที่ 3.5 เป็นลำดับการทำงานของโปรแกรม System Test ซึ่งจะทำงานโดยอัตโนมัติหลังจากเรียกใช้งาน โดยจะไปดาวน์โหลดไฟล์สคริปต์ทดสอบจากเซิร์ฟเวอร์ แล้วสั่งให้สคริปต์ทำงาน ซึ่งในสคริปต์ทดสอบจะบอกว่าข้อมูลใดบ้างที่จำเป็นต้องใช้ในการทดสอบ จากนั้นจึงไปเก็บข้อมูลเหล่านั้นจากคอมพิวเตอร์ของผู้ใช้ โดยเมื่อได้ข้อมูลที่จำเป็นครบแล้วก็จะทำการทดสอบตามรายการของกรณีทดสอบที่อยู่ในสคริปต์ จากนั้นจึงแสดงผลลัพธ์ของการทดสอบและคำอธิบายในแต่ละกรณีทดสอบที่หน้าจอของโปรแกรม

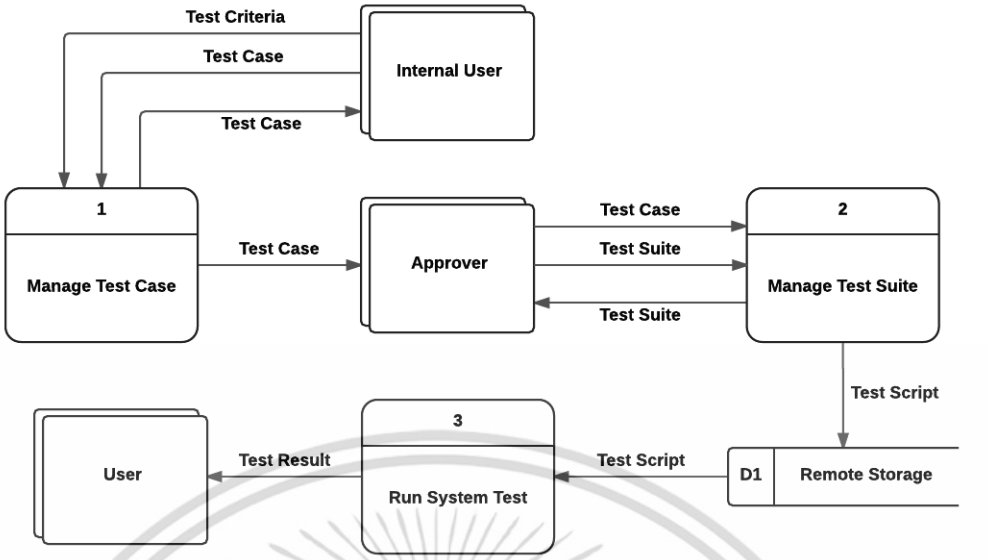
3.5 แผนภาพกระแสข้อมูล (Data Flow Diagram)

แผนภาพกระแสข้อมูลของโปรแกรมนั้นเริ่มจากแผนภาพบริบทซึ่งแสดงทิศทางการไหลของข้อมูลทั้งระบบ โดยผู้ที่มีส่วนเกี่ยวข้องกับระบบ คือ ผู้ใช้ทั่วไป ผู้ใช้ภายใน และผู้ตรวจสอบ โดยผู้ใช้ภายในนั้นจะมีการส่งเงื่อนไขของการทดสอบไปที่โปรแกรม เพื่อนำเงื่อนไขเหล่านั้นไปสร้างเป็นกรณีทดสอบ และมีการส่งและรับกรณีทดสอบในกระบวนการบันทึกและโหลดกรณีทดสอบ ถัดมาเป็นผู้ตรวจสอบ ซึ่งต้องมีการส่งและรับกรณีทดสอบในการเพิ่มกรณีทดสอบเข้าไปยังชุดทดสอบ และมีการส่งและรับชุดทดสอบในกระบวนการบันทึกและโหลดชุดทดสอบ และสุดท้ายคือผู้ใช้ทั่วไป ซึ่งมีเพียงข้อมูลผลลัพธ์ของการทดสอบเท่านั้นที่ถูกส่งไปยังผู้ใช้ทั่วไป โดยแผนภาพบริบทแสดงได้ดังรูปที่ 3.6



รูปที่ 3.6 แผนภาพบริบท

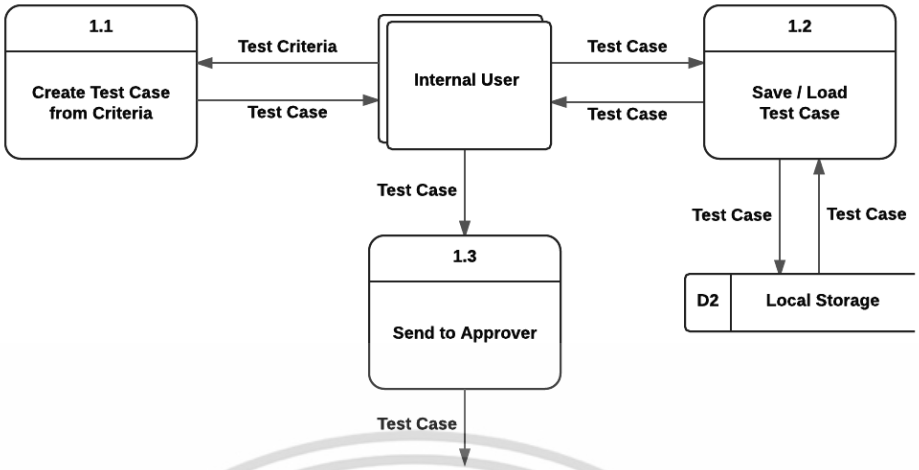
จากรูปที่ 3.6 เป็นแผนภาพบริบทซึ่งแสดงทิศทางการไหลของข้อมูลทั้งระบบ รวมถึงวัตถุภายนอกที่สนใจ และมีบทบาทต่อข้อมูลในระบบ เพื่อให้สามารถอธิบายการทำงานของระบบได้ละเอียดมากยิ่งขึ้น จึงแบ่งเป็นระบบย่อยที่มีขนาดเล็กลง คือ แผนภาพกระแสข้อมูลระดับที่ 1 ซึ่งแบ่งออกเป็นทั้งหมด 3 กระบวนการ ได้แก่ การจัดการกรณีทดสอบ การจัดการชุดทดสอบ และการทดสอบระบบ โดยแผนภาพกระแสข้อมูลระดับที่ 0 แสดงได้ดังรูปที่ 3.7



รูปที่ 3.7 แผนภาพกระแสข้อมูลระดับที่ 0

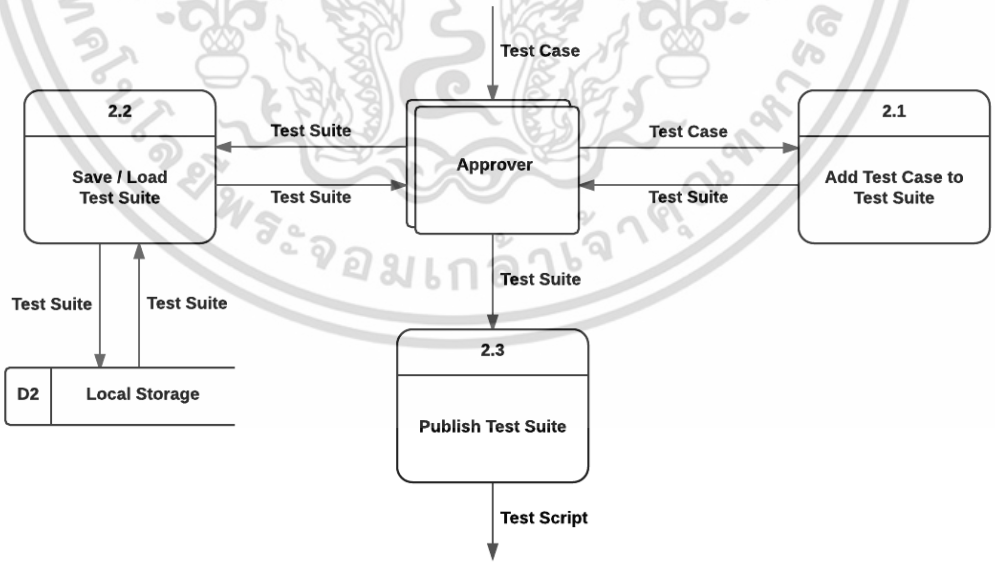
จากรูปที่ 3.7 แสดงให้เห็นกระบวนการภายในระบบที่ซับซ้อนมากขึ้น โดยกระแสข้อมูลเริ่มจากผู้ใช้ภายในทำการส่งเงื่อนไขหรือเกณฑ์การทดสอบไปยังกระบวนการจัดการกรณีทดสอบ (กระบวนการที่ 1) จึงได้เป็นกรณีทดสอบ แล้วส่งกรณีทดสอบนั้นไปยังผู้ตรวจสอบ จากนั้นผู้ตรวจสอบจะส่งกรณีทดสอบไปยังกระบวนการจัดการชุดทดสอบ (กระบวนการที่ 2) เพื่อเพิ่มกรณีทดสอบใหม่เข้าไปยังชุดทดสอบ เมื่อเสร็จสิ้นแล้วกระบวนการจัดการชุดทดสอบจะทำการสร้างสคริปต์จากชุดทดสอบแล้วจึงส่งไปยังที่จัดเก็บข้อมูลระยะไกล เพื่อให้กระบวนการทดสอบระบบ (กระบวนการที่ 3) สามารถนำสคริปต์นี้มาใช้งานได้ แล้วจึงแจ้งผลลัพธ์ของการทดสอบไปยังผู้ใช้งาน

จากลำดับการทำงานของแผนภาพกระแสข้อมูลระดับที่ 0 ทำให้ทราบถึงการกระบวนการที่ซับซ้อนมากขึ้น แต่สำหรับกระบวนการจัดการกรณีทดสอบและกระบวนการจัดการชุดทดสอบนั้นสามารถทำการแบ่งเป็นแผนภาพกระแสข้อมูลระดับที่ 1 ได้ ดังรูปที่ 3.8 และรูปที่ 3.9



รูปที่ 3.8 แผนภาพกระแสข้อมูลระดับที่ 1 (กระบวนการจัดการกรณีทดสอบ)

จากรูปที่ 3.8 จะเริ่มจากการที่ผู้ใช้ภายในทำการส่งเงื่อนไขหรือเกณฑ์การทดสอบไปยังกระบวนการที่ 1.1 เพื่อสร้างเป็นกรณีทดสอบ หากต้องการบันทึกกรณีทดสอบก็ส่งกรณีทดสอบไปยังกระบวนการที่ 1.2 จากนั้นกรณีทดสอบจะถูกส่งไปเก็บไว้ที่แหล่งเก็บข้อมูลภายใน และหากต้องการนำกรณีทดสอบมาแก้ไข กระบวนการที่ 1.2 ก็จะไปดึงกรณีทดสอบมาจากแหล่งเก็บข้อมูลภายในเพื่อส่งกลับมาให้ผู้ใช้ เมื่อกรณีทดสอบนั้นสมบูรณ์แล้วจะถูกส่งไปยังกระบวนการที่ 1.3 เพื่อทำการส่งกรณีทดสอบไปยังผู้ตรวจสอบต่อไป



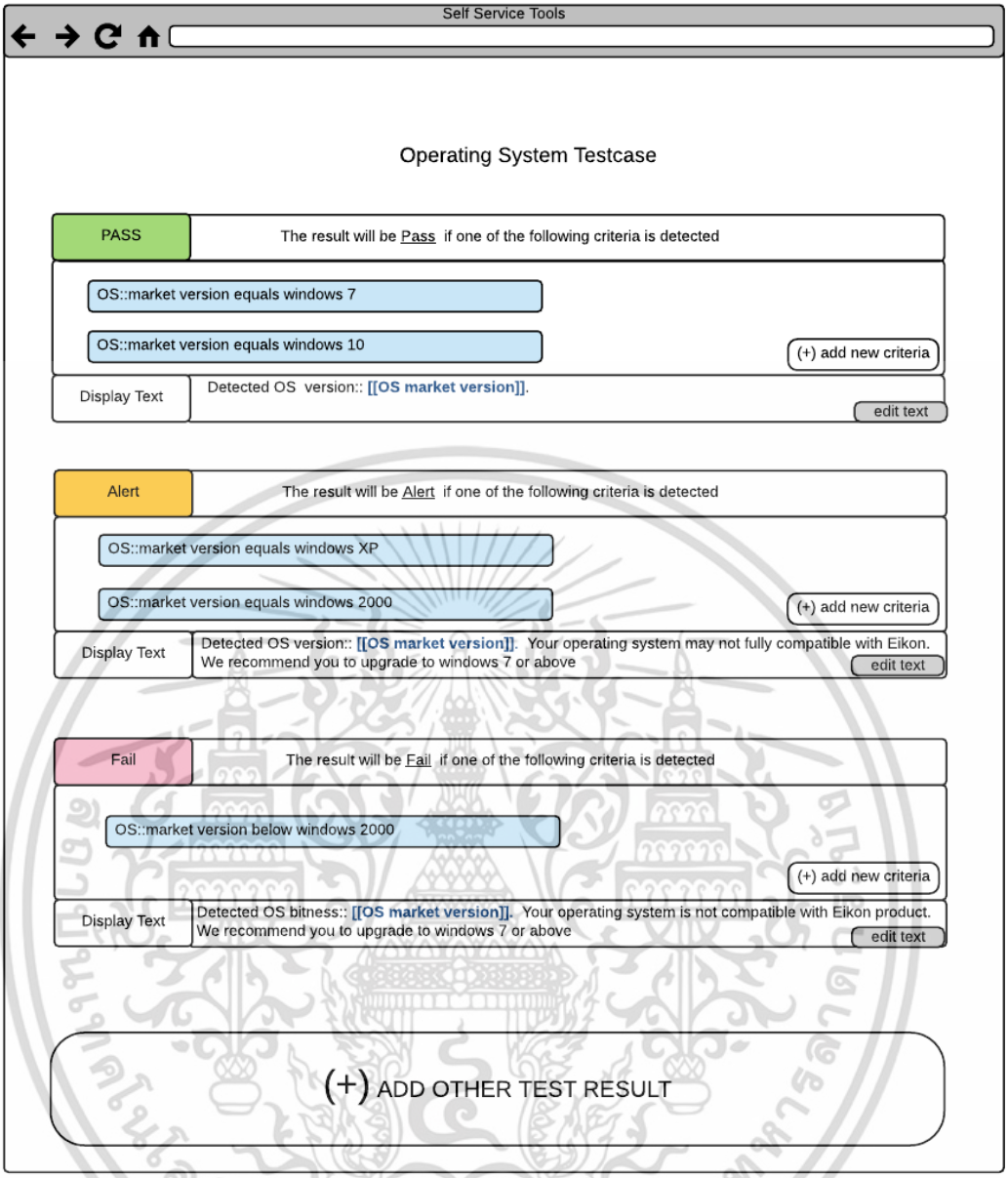
รูปที่ 3.9 แผนภาพกระแสข้อมูลระดับที่ 1 (กระบวนการจัดการชุดทดสอบ)

จากรูปที่ 3.9 เริ่มจากการที่ผู้ตรวจสอบได้รับกรณีทดสอบมาจากกระบวนการจัดการกรณีทดสอบ เมื่อตรวจสอบแล้วว่ากรณีทดสอบนี้มีความถูกต้องและไม่เป็นอันตรายต่อผู้ใช้ก็จะส่งกรณีทดสอบไปยังกระบวนการที่ 2.1 เพื่อทำการเพิ่มกรณีทดสอบใหม่เข้าไปที่ชุดทดสอบ กระบวนการนี้จึงส่งชุดทดสอบกลับมายังผู้ตรวจสอบ กระบวนการจัดการชุดทดสอบนี้ก็มีการบันทึกและแก้ไขชุดทดสอบเช่นเดียวกับกระบวนการจัดการกรณีทดสอบ โดยจัดเก็บข้อมูลไว้ที่แหล่งเก็บข้อมูลภายในเช่นเดียวกัน ซึ่งเป็นกระบวนการที่ 2.2 จากนั้นเมื่อต้องการสร้างเป็นสคริปต์สำหรับการนำไปใช้งานก็ส่งชุดทดสอบไปยังกระบวนการที่ 2.3 ซึ่งกระบวนการนี้จะทำการแปลงชุดทดสอบให้กลายเป็นสคริปต์ แล้วจึงส่งสคริปต์นั้นไปยังแหล่งเก็บข้อมูลระยะไกลซึ่งสามารถเข้าถึงได้ผ่านระบบเครือข่าย

จากนั้นเมื่อผู้ใช้ทั่วไปเรียกใช้กระบวนการทดสอบระบบ กระบวนการทดสอบระบบก็จะนำสคริปต์จากแหล่งเก็บข้อมูลระยะไกลมาประมวลผลที่เครื่องคอมพิวเตอร์ของผู้ใช้ แล้วจึงส่งผลลัพธ์ของการทดสอบกลับไปยังผู้ใช้

3.6 ส่วนติดต่อผู้ใช้ (User Interface)

การออกแบบส่วนติดต่อกับผู้ใช้เป็นขั้นตอนสุดท้ายของการวิเคราะห์และออกแบบระบบ โดยจะต้องออกแบบให้สามารถใช้งานได้ดีและเรียนรู้ได้เร็ว และเนื่องจากการแบ่งออกเป็นโปรแกรมย่อยจึงต้องออกแบบหน้าต่างการใช้งานของทั้งสามโปรแกรมให้สอดคล้องกัน การออกแบบส่วนติดต่อผู้ใช้ของทั้งสามโปรแกรมและหน้าต่างย่อย แสดงได้ดังรูปที่ 3.10 ถึงรูปที่ 3.13



รูปที่ 3.10 การออกแบบส่วนติดต่อผู้ใช้ของโปรแกรม Self Service Tool

จากรูปที่ 3.10 เป็นตัวอย่างส่วนติดต่อผู้ใช้สำหรับโปรแกรม Self Service Tool ซึ่งมีวัตถุประสงค์หลักคือการจัดการกรณีทดสอบ โดยจะแบ่งกลุ่มตามสถานะของการทดสอบซึ่งมีทั้งหมด 4 สถานะ ได้แก่

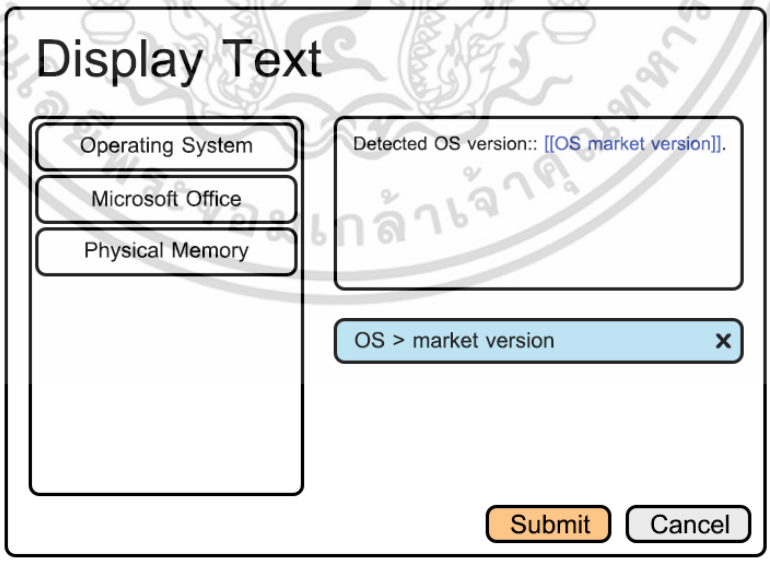
1) Pass หมายถึงกรณีทดสอบนี้ผ่าน สามารถใช้งานโปรแกรมได้
2) Alert เป็นการเตือนว่าโปรแกรมอาจทำงานผิดพลาดได้ เนื่องจากระบบยังไม่รองรับคุณสมบัติบางอย่างที่โปรแกรมต้องการ แต่ยังสามารถทำงานได้
3) Fail หมายถึงกรณีทดสอบนี้ไม่ผ่าน เนื่องจากทรัพยากรของระบบไม่เพียงพอต่อความต้องการของโปรแกรม ทำให้ไม่สามารถใช้งานโปรแกรมได้
4) Info เป็นการแสดงข้อมูลของระบบ จะไม่มีผลลัพธ์ว่าผ่านหรือไม่

โดยการเพิ่มสถานะทดสอบใหม่สามารถทำได้โดยกดปุ่มที่อยู่ด้านล่าง พร้อมทั้งเลือกว่าจะให้ผลลัพธ์ของการทดสอบเป็นสถานะใด ในแต่ละสถานะของการทดสอบก็จะมีเงื่อนไขเพื่อตรวจสอบว่าข้อมูลทรัพยากรของระบบที่นำไปทดสอบนั้นตรงกับเงื่อนไขที่ระบุหรือไม่ หากตรงเงื่อนไขใดเงื่อนไขหนึ่งก็จะแสดงผลลัพธ์เป็นข้อความและสถานะของการทดสอบในกรณีทดสอบนั้น ๆ หากไม่ตรงก็จะตรวจสอบเงื่อนไขถัดไปตามลำดับ ซึ่งเมื่อไม่มีเงื่อนไขให้ตรวจสอบแล้วก็แสดงผลลัพธ์ตามค่าเริ่มต้นที่ตั้งไว้ โดยในกล่องของสถานะทดสอบจะมีปุ่มให้เพิ่มเงื่อนไขของการทดสอบได้

จากนั้นจะมีแถบเครื่องมือที่ใช้จัดการกับกรณีทดสอบ เช่น สร้างกรณีทดสอบใหม่ บันทึกกรณีทดสอบ หรือจำลองการทดสอบ เป็นต้น ซึ่งแถบเครื่องมือเหล่านี้จะอยู่ทางด้านมุมบนขวาเพื่อใช้พื้นที่ให้น้อยที่สุดและยังสามารถใช้งานได้อย่างสะดวก

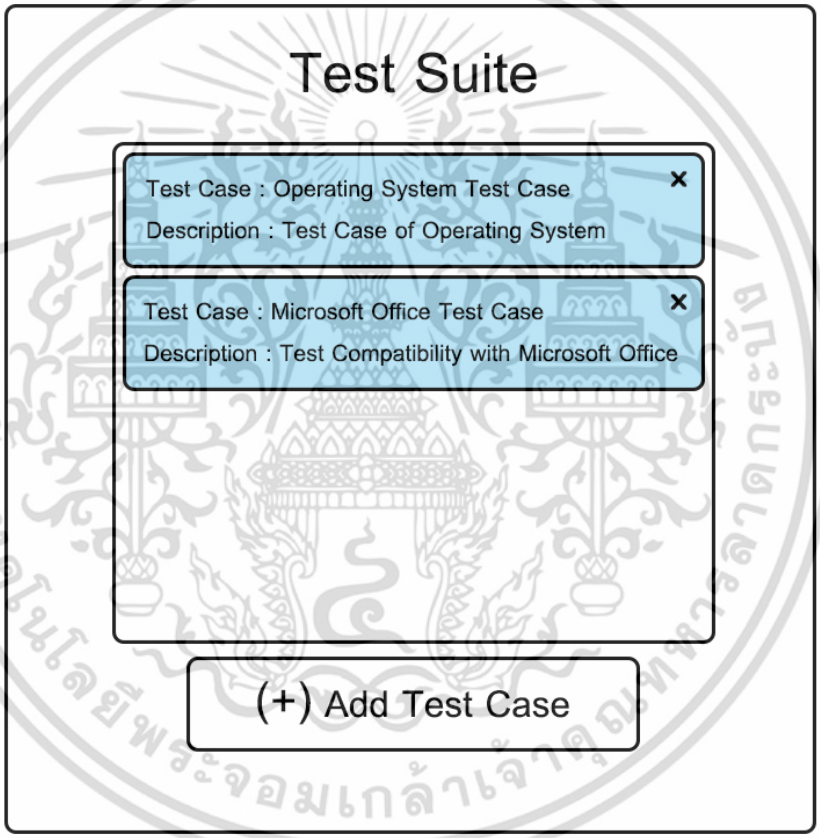
ถัดมาจะเป็นการออกแบบหน้าต่างที่ใช้สร้างหรือแก้ไขข้อความที่ใช้แสดงผลลัพธ์ของการทดสอบ โดยในแต่ละกล่องสถานะการทดสอบก็จะมีข้อความที่ใช้แสดงผลด้วย ซึ่งสามารถกดปุ่มหรือคลิกที่ข้อความเพื่อเปิดหน้าต่างแก้ไขขึ้นมาได้ แต่หน้าต่างนี้มีความพิเศษตรงที่นอกจากจะใส่ข้อความเพื่อแสดงผลตามปกติแล้ว ยังสามารถแสดงค่าต่าง ๆ ของระบบที่นำมาตรวจสอบได้อีกด้วย เพื่อแจ้งผู้ใช้ให้ทราบถึงความแตกต่างระหว่างค่าที่หามาได้กับค่าที่โปรแกรมต้องการ โดยเฉพาะในกรณีที่ผลการทดสอบนั้นไม่ผ่าน

การออกแบบหน้าต่างจึงต้องมีส่วนที่ให้เลือกส่วนประกอบเพื่อให้สามารถเลือกได้ว่าต้องการจะแสดงค่าของส่วนประกอบใด และสามารถลบส่วนประกอบที่เลือกไว้ได้หากไม่ต้องการให้แสดงค่าแล้ว ซึ่งหน้าต่างแก้ไขข้อความแสดงผลนั้นจะเป็นหน้าต่างแบบ Pop-up ซึ่งเป็นหน้าต่างที่เปิดขึ้นมาทับหน้าจอหลัก เมื่อใช้งานเสร็จแล้วก็จะปิดลงไป เนื่องจากการเพิ่มหรือแก้ไขข้อความแสดงผลนั้นไม่ได้ถูกเรียกใช้งานบ่อยนัก จึงต้องออกแบบเพื่อไม่ให้ไปบดบังพื้นที่ใช้งานของหน้าจอหลัก ดังรูปที่ 3.11



รูปที่ 3.11 การออกแบบหน้าต่างแสดงผลลัพธ์

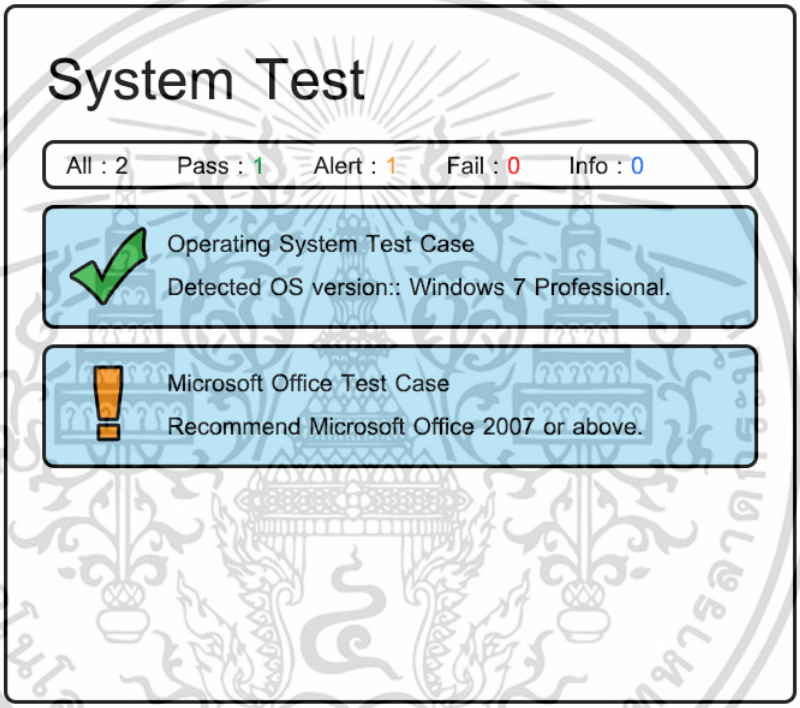
การออกแบบถัดมาคือโปรแกรม Test Suite ซึ่งเป็นหน้าต่างจัดการชุดทดสอบ ที่จะต้องสามารถเพิ่มกรณีทดสอบใหม่เข้ามาได้ และสามารถจัดลำดับของกรณีทดสอบได้ ซึ่งเมื่อทำการทดสอบระบบก็จะทดสอบเรียงตามกรณีทดสอบที่ได้จัดลำดับไว้ โดยจะต้องมีปุ่มเพิ่มกรณีทดสอบ ที่ถือว่าเป็นฟังก์ชันหลักสำหรับโปรแกรมนี้ ซึ่งได้จัดไว้ด้านล่างและปรับให้มีขนาดใหญ่เพื่อให้สามารถใช้งานได้สะดวก ในแต่ละกรณีทดสอบที่ได้เพิ่มมาแล้วก็จะต้องสามารถลบออกได้ จึงได้จัดวางปุ่มสำหรับลบกรณีทดสอบไว้ที่มุมบนขวา เพื่อไม่ให้บดบังข้อมูลของกรณีทดสอบ และโปรแกรม Test Suite นั้นก็จะมีแถบเครื่องมือที่ไว้จัดการชุดทดสอบเช่นเดียวกันกับโปรแกรม Self Service Tool คือ เครื่องมือสร้างชุดทดสอบใหม่ หรือบันทึกชุดทดสอบ และมีเครื่องมือที่ใช้สร้างสคริปต์จากชุดทดสอบเพิ่มมาด้วย การออกแบบส่วนติดต่อผู้ใช้ของโปรแกรม Test Suite แสดงได้ดังรูปที่ 3.12



รูปที่ 3.12 การออกแบบส่วนติดต่อผู้ใช้ของโปรแกรม Test Suite

โปรแกรมสุดท้ายคือโปรแกรม System Test ซึ่งจะเป็นการทดสอบระบบจากสคริปต์ที่ได้สร้างไว้ โดยการออกแบบหน้าต่างจะไม่ซับซ้อนมากนัก เนื่องจากโปรแกรมนี้จะไม่มีการโต้ตอบกับผู้ใช้ มีเพียงการแสดงผลลัพธ์ของการทดสอบเท่านั้น

ผลลัพธ์จากการทดสอบจะแสดงตามลำดับของกรณีทดสอบที่ได้จัดไว้ และมีสัญลักษณ์ของสถานะการทดสอบ เช่น ถ้าหากกรณีทดสอบนี้ให้ผลลัพธ์เป็นผ่านก็จะแสดงสัญลักษณ์เป็นเครื่องหมายถูกสีเขียว หรือถ้าไม่ผ่านก็จะแสดงเป็นเครื่องหมายกากบาทสีแดง เป็นต้น และจะมีการนับจำนวนของแต่ละสถานะพร้อมทั้งแสดงผลเป็นแท็บด้านบน โดยสีของแต่ละสถานะก็จะแตกต่างกันตามระดับของความสำคัญ ดังรูปที่ 3.13



รูปที่ 3.13 การออกแบบส่วนติดต่อผู้ใช้ของโปรแกรม System Test

บทที่ 4

การพัฒนาแอปพลิเคชัน

โปรแกรมนี้พัฒนาโดยใช้แอปพลิเคชันเฟรมเวิร์คที่ชื่อว่า Electron ซึ่งใช้สำหรับสร้างเดสก์ท็อปแอปพลิเคชัน ร่วมกับ Angular2 ซึ่งเป็นเฟรมเวิร์คทางด้าน Front-end ที่ใช้สำหรับจัดโครงสร้างของส่วนต่อประสานกราฟิกกับผู้ใช้ให้ดูสวยงามและตอบสนองการใช้งานมากยิ่งขึ้น โดยในการพัฒนาจะใช้ภาษา TypeScript เป็นหลัก

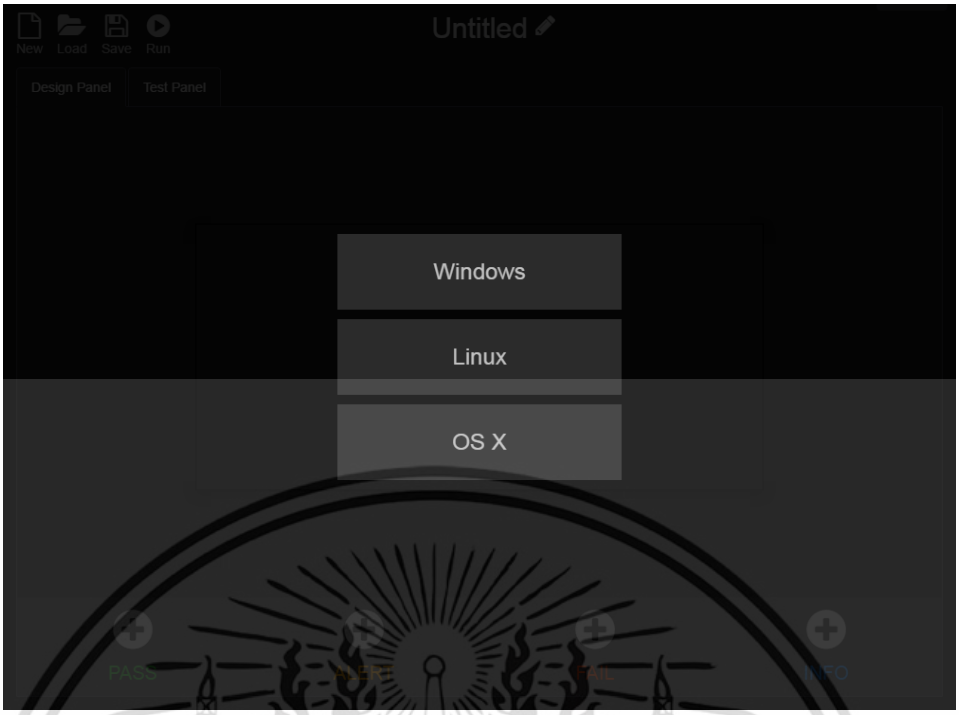
โปรแกรมตรวจสอบทรัพยากรระบบแบ่งเป็น 3 โปรแกรมย่อย ได้แก่ Self Service Tool, Test Suite, และ System Test เนื่องจากแต่ละโปรแกรมมีการทำงานที่เป็นอิสระต่อกัน จึงได้แยกกันพัฒนาตามลำดับการทำงาน ดังนี้

4.1 โปรแกรม Self Service Tool

ลำดับการทำงานของโปรแกรมตรวจสอบทรัพยากรระบบนั้น จะเริ่มจากการสร้างกรณีทดสอบโดยใช้โปรแกรม Self Service Tool ซึ่งสามารถกำหนดเงื่อนไขของการทดสอบ และผลลัพธ์จากการทดสอบได้ อีกทั้งยังมีเครื่องมือต่าง ๆ ที่ช่วยอำนวยความสะดวกในการสร้างกรณีทดสอบอีกมากมาย เช่น การทดสอบในเครื่องของผู้ใช้ หรือจำลองค่าของระบบเพื่อใช้ในการทดสอบ เป็นต้น

4.1.1 การเลือกแพลตฟอร์ม

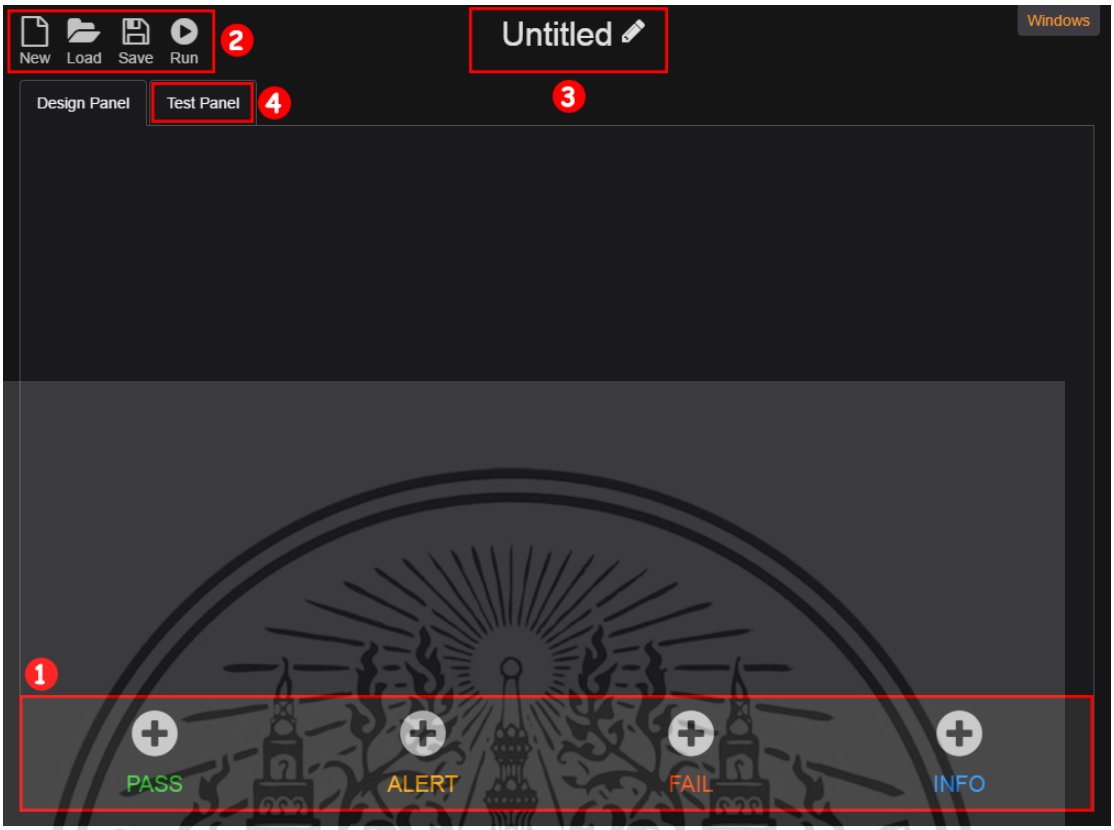
หลังจากที่เปิดโปรแกรมขึ้นมาแล้ว จะปรากฏหน้าต่างให้เลือกแพลตฟอร์มสำหรับการสร้างกรณีทดสอบ ซึ่งในแต่ละกรณีทดสอบจะรองรับเพียงแพลตฟอร์มเดียวเท่านั้น โดยเมื่อเลือกแพลตฟอร์มแล้วจะไม่สามารถเปลี่ยนภายหลังได้ ผู้ใช้งานจึงต้องทราบก่อนว่ากรณีทดสอบที่กำลังสร้างนั้นจะทำงานบนแพลตฟอร์มใด และการเลือกแพลตฟอร์มนั้นมีผลต่อส่วนประกอบที่จะมีให้เลือก เช่น หากเลือกกรณีทดสอบสำหรับแพลตฟอร์ม Linux ก็จะไม่สามารถใช้งานส่วนประกอบของ Microsoft Update Patch (KB) ได้ เนื่องจากส่วนประกอบนี้มีในแพลตฟอร์ม Windows เท่านั้น เป็นต้น หน้าจอการเลือกแพลตฟอร์มแสดงดังรูปที่ 4.1



รูปที่ 4.1 การเลือกแพลตฟอร์มสำหรับกรณีทดสอบ

4.1.2 หน้าต่างการทำงานหลัก

หลังจากที่เลือกแพลตฟอร์มแล้ว ก็จะปรากฏหน้าต่างการทำงานหลักของโปรแกรม ซึ่งมีแถบเครื่องมืออยู่มุมบนซ้าย และแพลตฟอร์มที่เลือกไว้จะระบุอยู่มุมบนขวาของหน้าจอ การใช้งานจะแบ่งออกเป็น 2 ส่วน คือ การออกแบบ และการทดสอบ โดยแบ่งอยู่ในรูปของแท็บ ซึ่งสามารถสลับไปมาได้ตลอดเวลา ฟังก์ชันหลักที่ใช้ คือ การเพิ่มเงื่อนไขของการทดสอบ โดยแต่ละเงื่อนไขจะแบ่งตามสถานะของการทดสอบ ซึ่งมี 4 สถานะ คือ สถานะผ่าน สถานะแจ้งเตือน สถานะไม่ผ่าน และสถานะของการแสดงข้อมูล และมีการแบ่งสีตามสถานะนั้น ๆ ดังรูปที่ 4.2

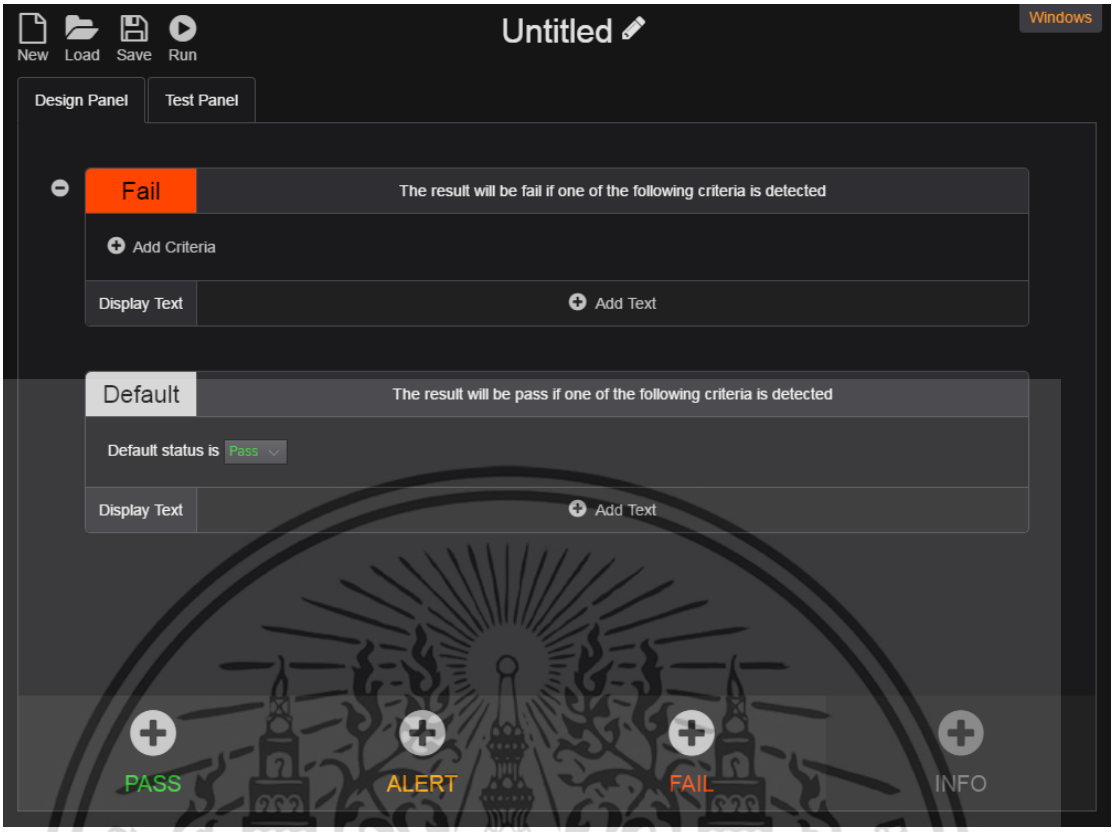


รูปที่ 4.2 หน้าต่างหลักของโปรแกรม Self Service Tool

จากรูปที่ 4.2 ประกอบไปด้วยส่วนที่ 1 เป็นส่วนของปุ่มเพิ่มผลลัพธ์ของการทดสอบซึ่งแบ่งตามสถานะของการทดสอบทั้ง 4 แบบ ส่วนที่ 2 เป็นส่วนของแถบเครื่องมือต่าง ๆ ได้แก่ การสร้างกรณีทดสอบใหม่ การเปิดกรณีทดสอบที่บันทึกไว้ การบันทึกกรณีทดสอบที่กำลังทำอยู่ และการทดสอบที่เครื่องของผู้ใช้ ถัดมาในส่วนที่ 3 คือส่วนรายละเอียดของกรณีทดสอบ คือ ชื่อกรณีทดสอบ และคำอธิบายของกรณีทดสอบ โดยสามารถคลิกที่รูปดินสอเพื่อแก้ไขได้ และส่วนสุดท้ายคือส่วนที่ 4 ซึ่งเป็นเครื่องมือที่ใช้จำลองการทดสอบโดยระบุค่าต่าง ๆ ของระบบที่ต้องการทดสอบ

4.1.3 การเพิ่มผลลัพธ์ของการทดสอบ

การเพิ่มผลลัพธ์ของการทดสอบสามารถทำได้โดยการกดปุ่มเพิ่มเงื่อนไข โดยจะอยู่ในส่วนที่ 1 ของรูปที่ 4.2 ซึ่งแบ่งออกตามสถานะของการทดสอบ เมื่อผู้ใช้กดปุ่ม PASS โปรแกรมก็จะสร้างกล่องผลลัพธ์ของกรณีทดสอบที่ให้ผลลัพธ์เป็นผ่านเพิ่มมา ซึ่งกล่องผลลัพธ์นั้นก็สามารถเพิ่มเงื่อนไขที่ต้องการได้ พร้อมทั้งเพิ่มข้อความที่จะแสดงผลเมื่อเงื่อนไขเป็นจริง โดยหน้าจอการเพิ่มกล่องผลลัพธ์ของกรณีทดสอบแสดงดังรูปที่ 4.3



รูปที่ 4.3 กล่องผลลัพธ์ของกรณีทดสอบ

จากรูปที่ 4.3 จะเห็นว่าเมื่อกดปุ่ม FAIL แล้วจะมีกล่องผลลัพธ์เพิ่มมา 2 กล่อง นั่นคือ กล่อง Fail ซึ่งจะมีปุ่มให้เพิ่มเงื่อนไขของการทดสอบ โดยถ้าเงื่อนไขเป็นจริงก็จะแสดงผลลัพธ์ของการทดสอบตาม Display Text ของกล่องนั้น ๆ ถัดมาคือกล่อง Default เป็นกล่องที่ถูกสร้างขึ้นมาโดยอัตโนมัติ เพื่อไว้รองรับในกรณีที่การทดสอบนั้นไม่ตรงกับเงื่อนไขใดเลย ก็จะแสดงผลลัพธ์สถานะและข้อความตามที่ระบุไว้ในกล่อง Default

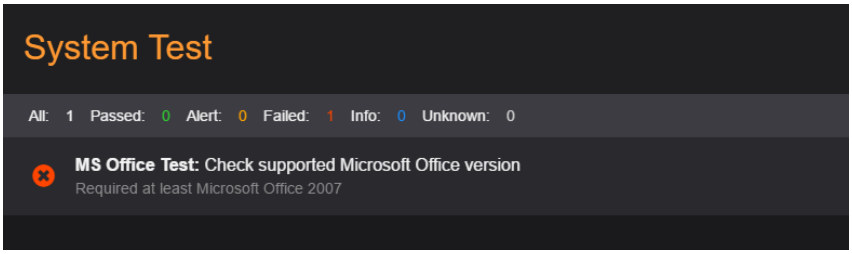
ประเภทของกรณีทดสอบนั้นแบ่งออกเป็น 2 ประเภท คือ การทดสอบตามปกติซึ่งจะมีผลลัพธ์เป็น Pass, Alert, หรือ Fail และการแสดงข้อมูลของระบบเท่านั้น (Info) โดยจะแยกกันอย่างชัดเจน จากรูปที่ 4.3 จะเห็นว่าเมื่อเพิ่มกล่องผลลัพธ์ Fail ซึ่งหมายถึงเป็นประเภทการทดสอบแบบปกติ ก็จะไม่สามารถเพิ่มผลลัพธ์ที่เป็น Info ได้ และหากเลือกเป็นการแสดงข้อมูลระบบ ก็จะไม่สามารถเพิ่มกล่องผลลัพธ์อื่นได้เช่นกัน

4.1.4 การเพิ่มเงื่อนไขของการทดสอบ

ในแต่ละกล่องผลลัพธ์ของการทดสอบนั้นจะมีปุ่มให้กดเพื่อเพิ่มเงื่อนไขของการทดสอบ เมื่อกดแล้วจะปรากฏหน้าต่างขึ้นดังรูปที่ 4.4 โดยการเพิ่มเงื่อนไขจะเริ่มจากการเลือกส่วนประกอบที่มีมาให้ก่อน โดยจะเป็นรายการอยู่ทางด้านซ้าย ซึ่งแบ่งตามกลุ่มของส่วนประกอบ ถัดมาเป็นการกำหนดค่าและเงื่อนไขของส่วนประกอบนั้น โดยตัวอย่างในรูปที่ 4.4 นั้นเลือกส่วนประกอบที่จะนำมาทดสอบเป็น Microsoft Office Word โดยมีเงื่อนไขคือ หากเวอร์ชันของ Microsoft Office Word นั้นต่ำกว่า 12.0 (Microsoft Word 2007) ก็จะให้ผลลัพธ์เป็น Fail ดังรูปที่ 4.5

เงื่อนไขของการทดสอบในแต่ละกล่องผลลัพธ์นั้นสามารถเพิ่มได้ไม่จำกัด และเงื่อนไขเหล่านั้นนำมาเชื่อมกันด้วยตัวดำเนินการทางตรรกศาสตร์ คือ “หรือ” (OR) หมายความว่าเมื่อเงื่อนไขใดเงื่อนไขหนึ่งเป็นจริงก็จะแสดงผลลัพธ์ของการทดสอบตามกล่องนั้น ๆ

รูปที่ 4.4 การเพิ่มเงื่อนไขที่จะใช้ทดสอบ



รูปที่ 4.5 ผลลัพธ์เมื่อไม่ผ่านการทดสอบ

4.1.5 การเพิ่มข้อความแสดงผล

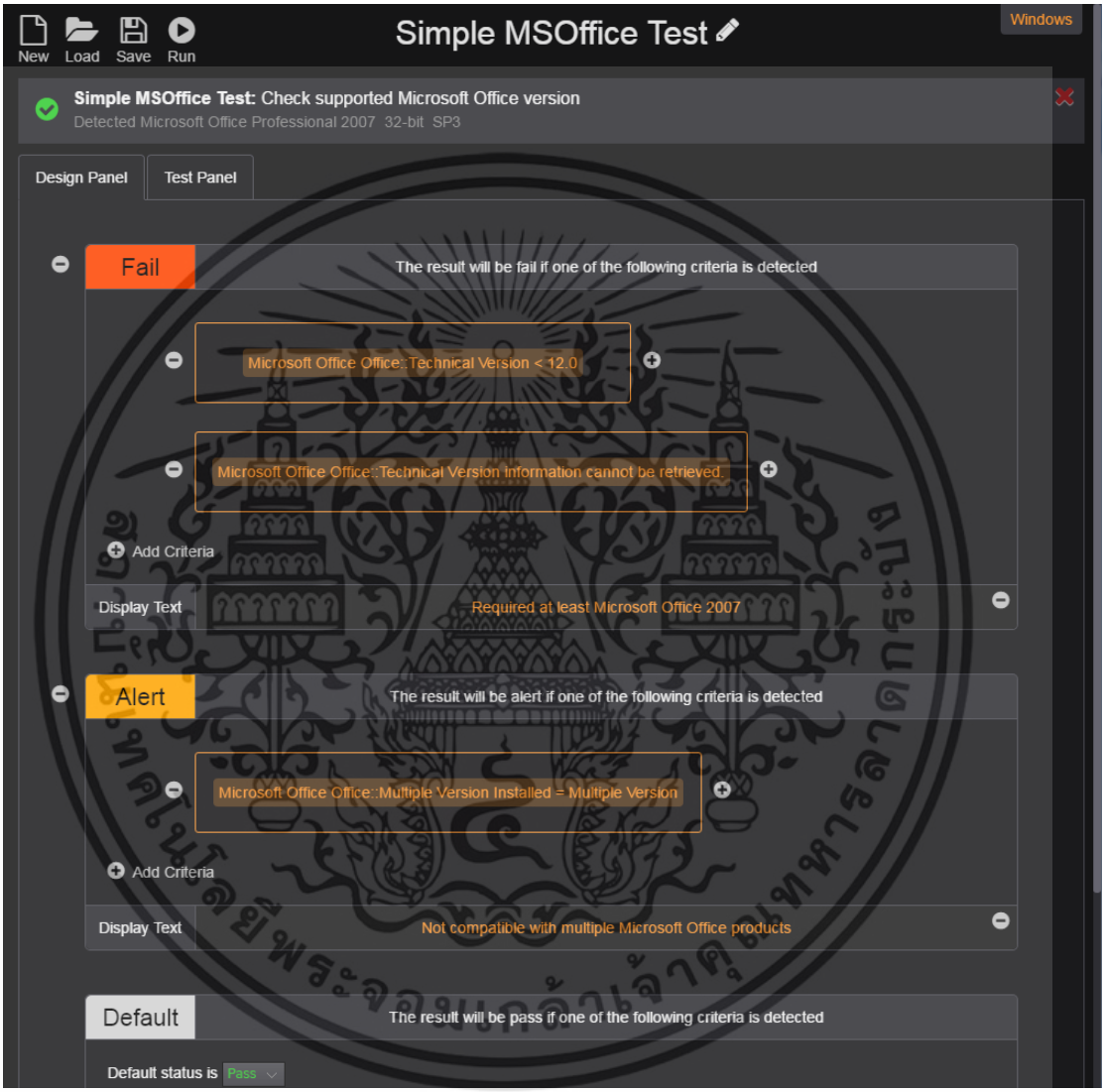
การเพิ่มข้อความแสดงผลของแต่ละกล่องผลลัพธ์การทดสอบ โดยขณะทำการทดสอบนั้น เมื่อเงื่อนไขใดเงื่อนไขหนึ่งเป็นจริงก็จะแสดงผลลัพธ์ที่หน้าจอ และเนื่องจากในหนึ่งกรณีทดสอบนั้นสามารถแสดงได้เพียงผลลัพธ์เดียว จึงต้องใส่ผลลัพธ์ให้ครบถ้วนและสอดคล้องกับกรณีทดสอบ เช่น การทดสอบ Microsoft Office Word หากไม่รองรับเวอร์ชันที่ระบุในเงื่อนไข ในการแสดงผลก็ควรจะบอกด้วยว่ารองรับเวอร์ชันใดบ้าง เป็นต้น

การแสดงผลนั้นนอกจากจะแสดงเป็นข้อความตามปกติแล้วยังสามารถแสดงข้อมูลอื่น ๆ ที่เก็บมาได้ด้วย โดยเลือกจากส่วนประกอบเช่นเดียวกับการเพิ่มเงื่อนไขของการทดสอบ เมื่อกดเลือกส่วนประกอบที่ต้องการแล้วจะปรากฏข้อความพิเศษเพิ่มเข้ามาในกล่องข้อความและกล่องแสดงข้อมูลของส่วนประกอบที่เลือก ซึ่งหากไม่ต้องการแล้วก็สามารถกดลบได้ เมื่อลบแล้วข้อความพิเศษก็จะหายไปเช่นกัน หน้าจอการเพิ่มข้อความแสดงผลของกล่องผลลัพธ์แสดงดังรูปที่ 4.6

รูปที่ 4.6 การเพิ่มข้อความแสดงผลของกล่องผลลัพธ์

4.1.6 การทดสอบที่เครื่องผู้ใช้

ในขณะที่กำลังสร้างกรณีทดสอบนั้น ผู้ใช้อาจจะต้องการทดสอบการทำงานของกรณีทดสอบที่สร้างว่าจะได้ผลลัพธ์ตรงตามที่ต้องการหรือไม่ ซึ่งสามารถทำได้โดยใช้เครื่องที่ช่วยทดสอบในระบบคอมพิวเตอร์ของผู้ใช้เอง โดยค่าทรัพยากรต่าง ๆ ที่จะนำมาทดสอบได้มาจากการไปเก็บข้อมูลจริงภายในระบบคอมพิวเตอร์ ดังรูปที่ 4.7



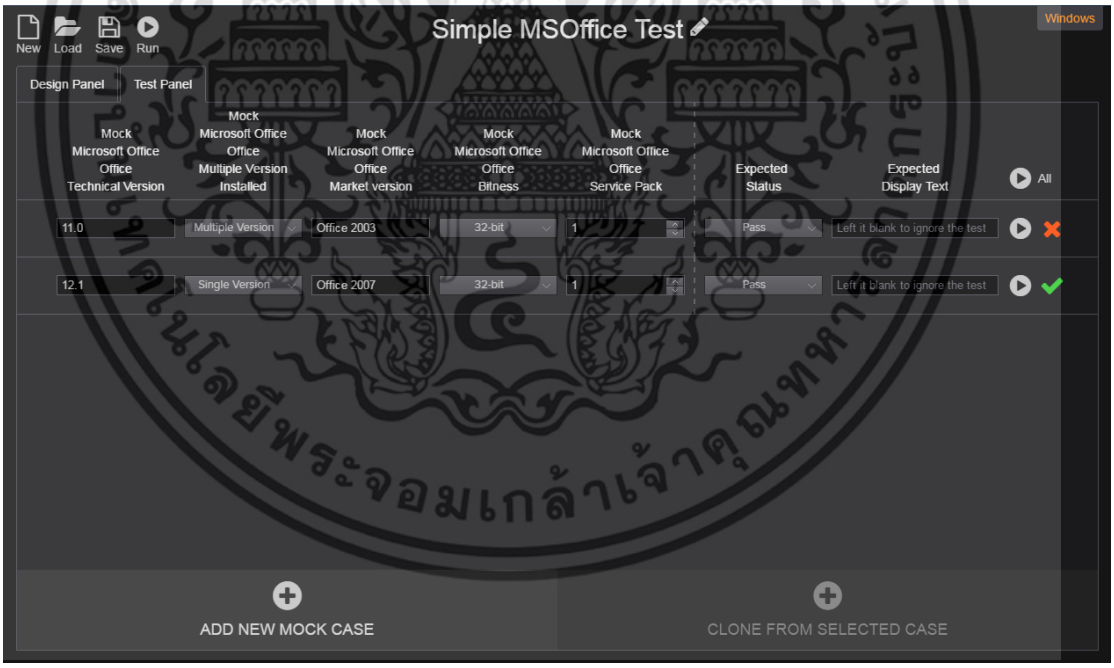
รูปที่ 4.7 การทดสอบในระบบของผู้ใช้

จากรูปที่ 4.7 จะเห็นว่ามีการกำหนดเงื่อนไขของการทดสอบโดยใช้ข้อมูลเวอร์ชันของโปรแกรม Microsoft Office ที่ถูกติดตั้งอยู่ในระบบคอมพิวเตอร์ เมื่อกดปุ่มทดสอบการทำงานจึงมีการเข้าไปเก็บข้อมูลจริงจากระบบคอมพิวเตอร์นั้น ๆ โดยข้อมูลที่ได้ คือ Microsoft Office เวอร์ชัน 2007 ที่มีเลขเวอร์ชันทางเทคนิค คือ 12.0 ซึ่งไม่ตรงกับเงื่อนไขใดเลย จึงให้ผลลัพธ์เป็นผ่านการทดสอบ ตามที่กำหนดไว้ในกล่อง Default

4.1.7 การจำลองค่าของระบบ

จากหัวข้อที่ 4.1.6 จะเห็นว่าผู้ใช้สามารถที่จะทดสอบการทำงานของกรณีทดสอบที่กำลังสร้างอยู่ได้ในระบบของตนเอง เป็นการพิสูจน์ว่ากรณีทดสอบนี้สามารถใช้งานได้จริง แต่ก็ยังมีข้อจำกัดทางด้านทรัพยากร ซึ่งหากต้องการทดสอบให้ครบทุกกรณีจะต้องใช้ทรัพยากรจำนวนมากมาทดสอบ จึงได้มีเครื่องมือที่ช่วยให้การทดสอบนั้นง่ายขึ้น โดยผู้ใช้สามารถกำหนดค่าต่าง ๆ ของระบบที่จะทดสอบได้เองผ่านเครื่องมือจำลองการทดสอบ ดังรูปที่ 4.8 โดยจะแบ่งออกเป็นสองฝั่ง ฝั่งแรกเป็นการจำลองค่า ซึ่งจะมีช่องให้ใส่ค่าเข้าไปได้ ตามส่วนประกอบทั้งหมดที่ได้เลือกไว้ ถัดมาจะเป็นการใส่ค่าผลลัพธ์ของการทดสอบที่คาดหวัง ซึ่งหากกรณีทดสอบนั้นใช้ส่วนประกอบจำนวนมาก ก็จะมีค่าให้ใส่จำนวนมากเช่นกัน จึงได้มีเครื่องมือที่ใช้คัดลอกการจำลองค่าได้ โดยการคลิกเลือกกรณีที่สนใจ จากนั้นกดปุ่ม CLONE FROM SELECTED CASE ด้านล่าง เพื่อทำการคัดลอกค่าที่จำลองไว้มาใส่ในกรณีใหม่

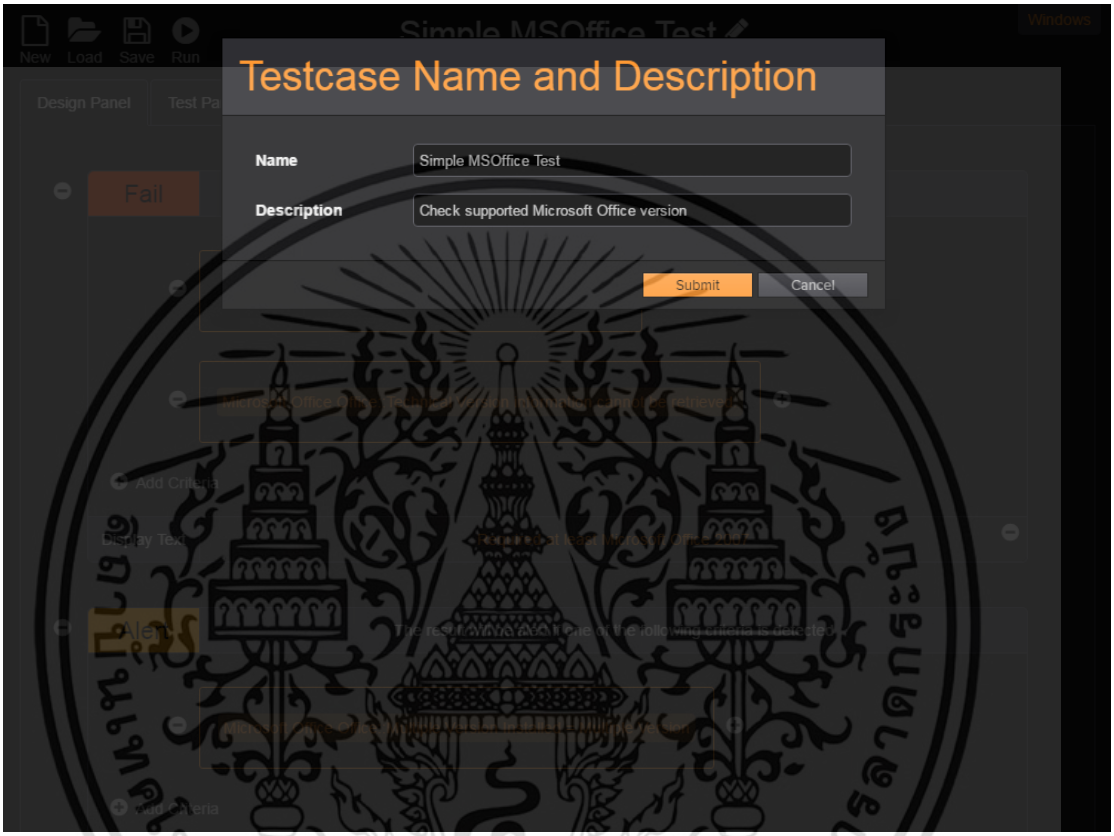
การทดสอบค่าที่จำลองพร้อมผลลัพธ์ที่คาดหวังสามารถทำได้สองวิธี คือ ทดสอบแยกแต่ละกรณี หรือทดสอบทั้งหมดในครั้งเดียว ซึ่งในกรณีปกตินั้นผลลัพธ์จากการจำลองการทดสอบจะต้องผ่านทั้งหมด เพื่อเป็นการบอกว่ากรณีทดสอบนี้ให้ผลลัพธ์ถูกต้องสำหรับทุก ๆ ระบบคอมพิวเตอร์ที่นำไปใช้งานนั่นเอง



รูปที่ 4.8 การจำลองค่าของทรัพยากรระบบในการทดสอบ

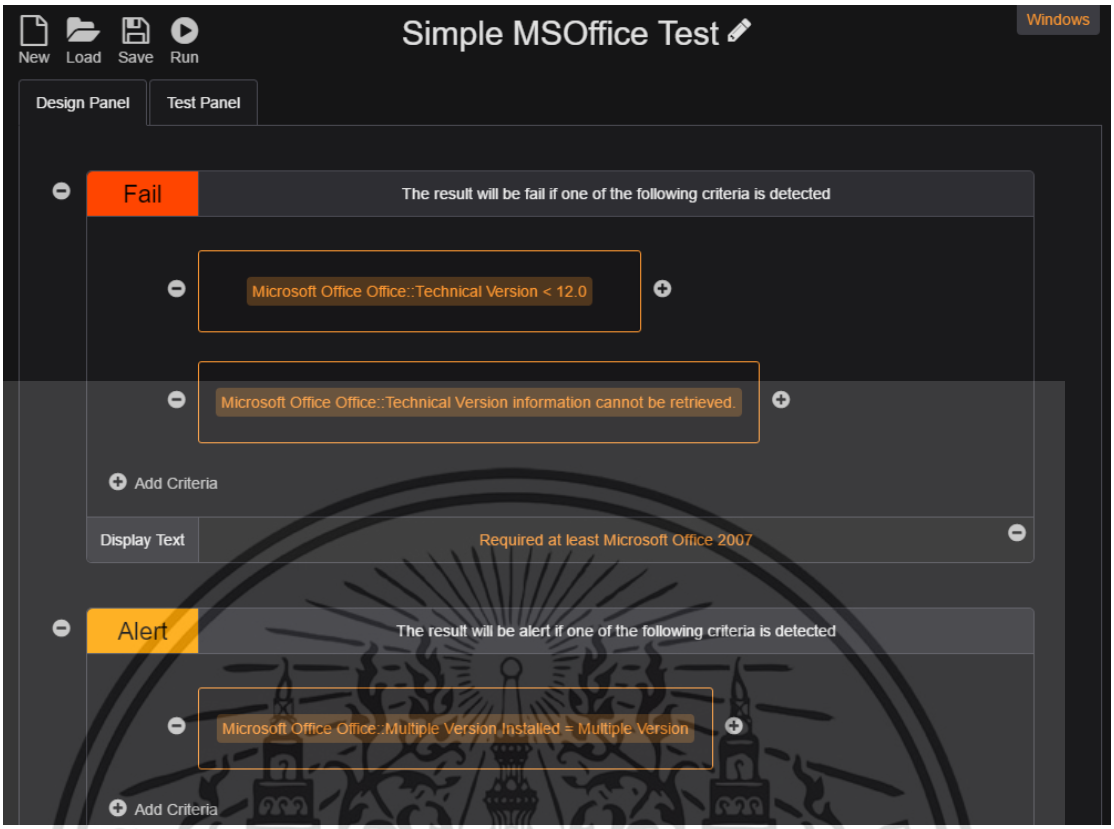
4.1.8 การเพิ่มชื่อและคำอธิบายกรณีทดสอบ

การอธิบายว่ากรณีทดสอบนั้นเป็นการทดสอบในรูปแบบใด สามารถทำได้โดยการกำหนดชื่อของกรณีทดสอบ และคำอธิบายกรณีทดสอบ เพื่อให้ผู้ตรวจสอบสามารถทำความเข้าใจกับกรณีทดสอบใหม่ ๆ ได้เร็วยิ่งขึ้น และที่สำคัญคือชื่อและคำอธิบายกรณีทดสอบนั้นจะแสดงต่อผู้ใช้ขณะนำไปทดสอบจริงด้วยสคริปต์ทดสอบ การเพิ่มชื่อและคำอธิบายกรณีทดสอบแสดงดังรูปที่ 4.9



รูปที่ 4.9 การเพิ่มชื่อและคำอธิบายกรณีทดสอบ

จากรูปที่ 4.9 ได้กำหนดชื่อและคำอธิบายของกรณีทดสอบเรียบร้อยแล้ว จึงกดปุ่มยืนยัน จากนั้นในหน้าต่างโปรแกรมก็จะปรากฏชื่อของกรณีทดสอบตามที่ได้กำหนดไว้ และเพื่อเป็นการประหยัดพื้นที่ คำอธิบายของกรณีทดสอบที่ได้กำหนดไว้จะไม่ถูกนำมาแสดงผล และผู้ใช้สามารถแก้ไขชื่อและคำอธิบายกรณีทดสอบได้ตลอดเวลา โดยการกดปุ่มรูปดินสอที่อยู่ถัดจากชื่อกรณีทดสอบ ดังรูปที่ 4.10



รูปที่ 4.10 กรณีทดสอบหลังแก้ไขชื่อและคำอธิบาย

โปรแกรม Self Service Tool ใช้เพื่อสร้างกรณีทดสอบ โดยมีฟังก์ชันสำหรับอำนวยความสะดวกในการใช้งานมากมาย เช่น ฟังก์ชันทดสอบการทำงาน การจำลองค่าของระบบที่ต้องการจะทดสอบ การบันทึกกรณีทดสอบที่ยังสร้างไม่เสร็จ เป็นต้น โดยตัวอย่างหน้าจอของกรณีทดสอบแสดงดังรูปที่ 4.11

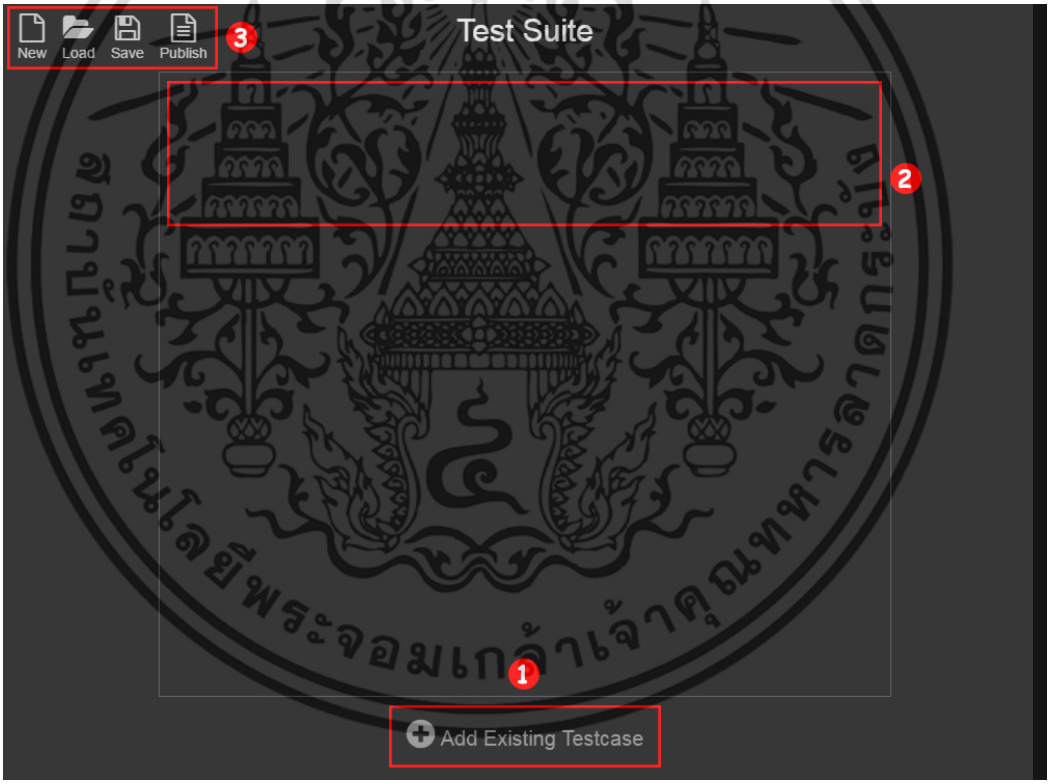
รูปที่ 4.11 ตัวอย่างกรณีทดสอบ

4.2 โปรแกรม Test Suite

โปรแกรม Test Suite มีไว้สำหรับจัดการกับชุดทดสอบ โดยชุดทดสอบนั้นจะประกอบไปด้วยกรณีทดสอบจำนวนมาก สำหรับโปรแกรมตรวจสอบทรัพยากรระบบนั้นจะมีชุดทดสอบเพียงชุดเดียว ดูแลและจัดการโดยผู้ตรวจสอบ ที่มีหน้าที่ตรวจสอบกรณีทดสอบที่ส่งมาจากผู้ใช้ภายใน หลังจากที่สร้างเสร็จแล้ว เมื่อทดสอบแล้วว่ามีความปลอดภัยและใช้งานได้อย่างถูกต้องก็จะทำการเพิ่มกรณีทดสอบใหม่เข้าไปในชุดทดสอบ

4.2.1 หน้าต่างการทำงานหลัก

เมื่อเปิดโปรแกรมขึ้นมาแล้ว จะปรากฏหน้าต่างสำหรับจัดการชุดทดสอบ โดยมีแถบเครื่องมืออยู่ทางด้านซ้ายบนเช่นเดียวกันกับโปรแกรม Self Service Tool เพื่อให้เป็นไปในแนวทางเดียวกัน และมีปุ่มสำหรับเพิ่มกรณีทดสอบ โดยจะเพิ่มกรณีทดสอบใหม่เข้าไปในรายการของกรณีทดสอบซึ่งอยู่ตรงกลางของหน้าจอ ดังรูปที่ 4.12

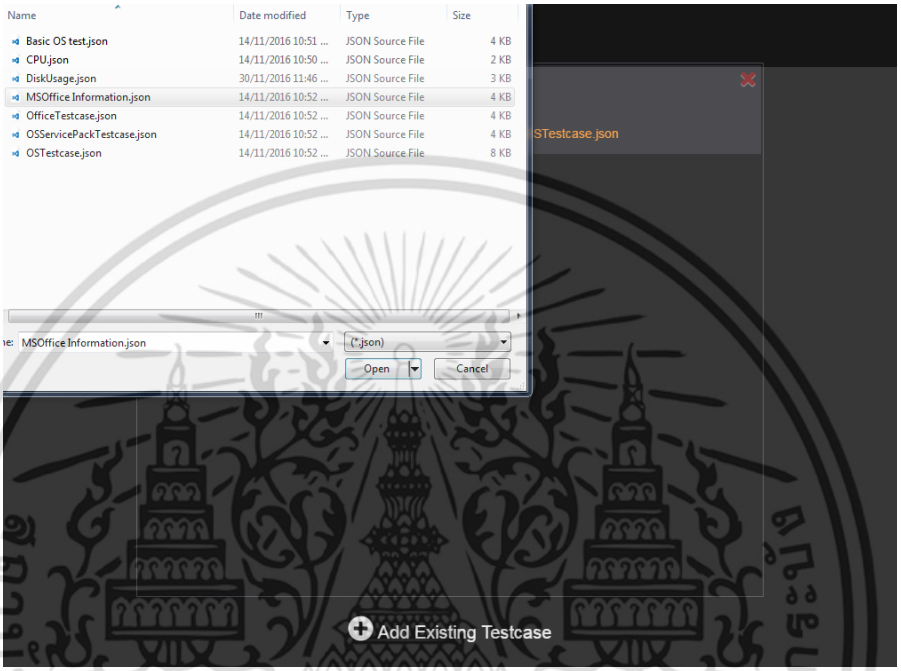


รูปที่ 4.12 หน้าต่างหลักของโปรแกรม Test Suite

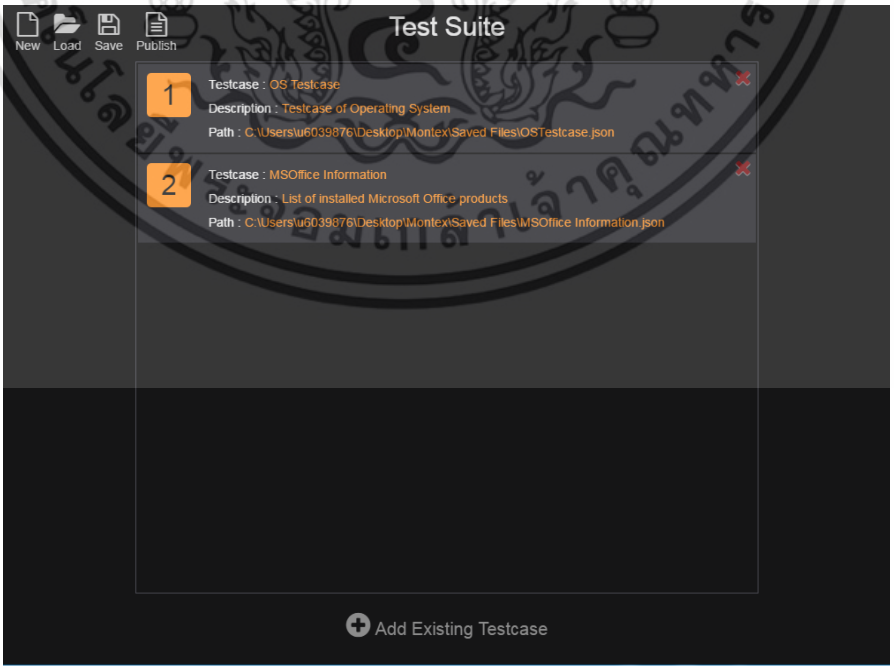
จากรูปที่ 4.12 ส่วนที่ 1 คือปุ่มเพิ่มกรณีทดสอบ สำหรับกรณีทดสอบที่ตรวจสอบแล้ว ส่วนที่ 2 คือรายการของกรณีทดสอบที่อยู่ในชุดทดสอบ โดยกรณีทดสอบใหม่ที่เพิ่มเข้ามาก็จะมาต่อท้ายรายการ ซึ่งรายการนี้สามารถจัดลำดับได้ว่าต้องการให้ทดสอบกรณีใดก่อน และส่วนที่ 3 เป็นแถบเครื่องมือที่ใช้จัดการชุดทดสอบ ได้แก่ การสร้างชุดทดสอบใหม่ การเปิดชุดทดสอบที่บันทึกไว้ การบันทึกชุดทดสอบ และการแปลงชุดทดสอบให้เป็นสคริปต์ที่สามารถนำไปใช้งานได้

4.2.2 การเพิ่มกรณีทดสอบใหม่

การเพิ่มกรณีทดสอบใหม่ ทำได้โดยกดปุ่มเพิ่มกรณีทดสอบที่อยู่ด้านล่างของหน้าต่าง จากนั้นจะมีหน้าต่างให้เลือกไฟล์กรณีทดสอบจากที่เก็บข้อมูลของผู้ใช้ โดยกรณีทดสอบนั้นถูกบันทึกเป็นนามสกุล JSON ซึ่งเป็นรูปแบบการเก็บข้อมูลที่เป็นที่นิยม หลังจากเลือกกรณีทดสอบแล้ว จะเห็นว่ากรณีทดสอบที่เพิ่มเข้ามาใหม่นั้นจะอยู่ลำดับท้ายสุดของรายการ ดังรูปที่ 4.13 และรูปที่ 4.14



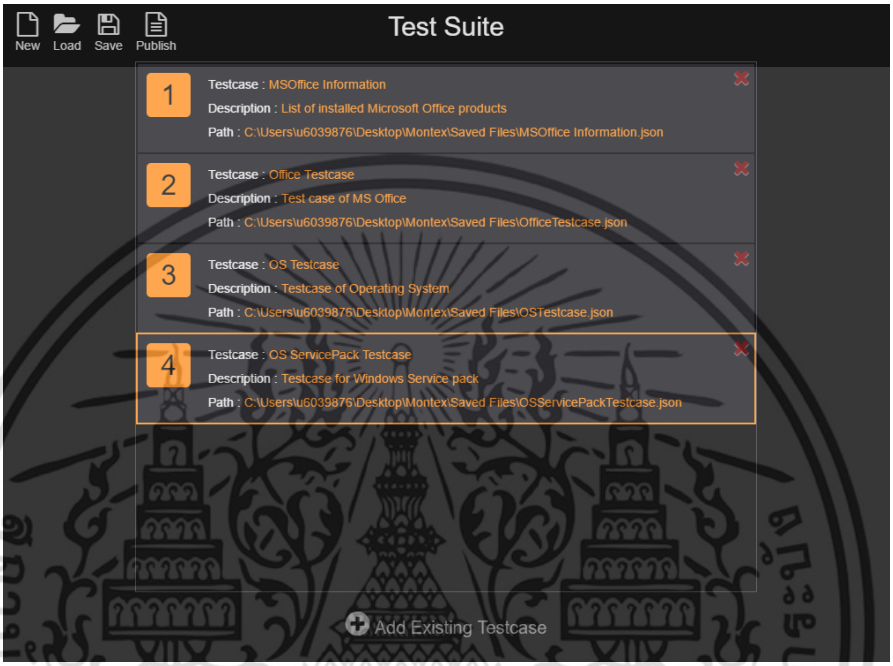
รูปที่ 4.13 การเพิ่มกรณีทดสอบเข้าไปในชุดทดสอบ



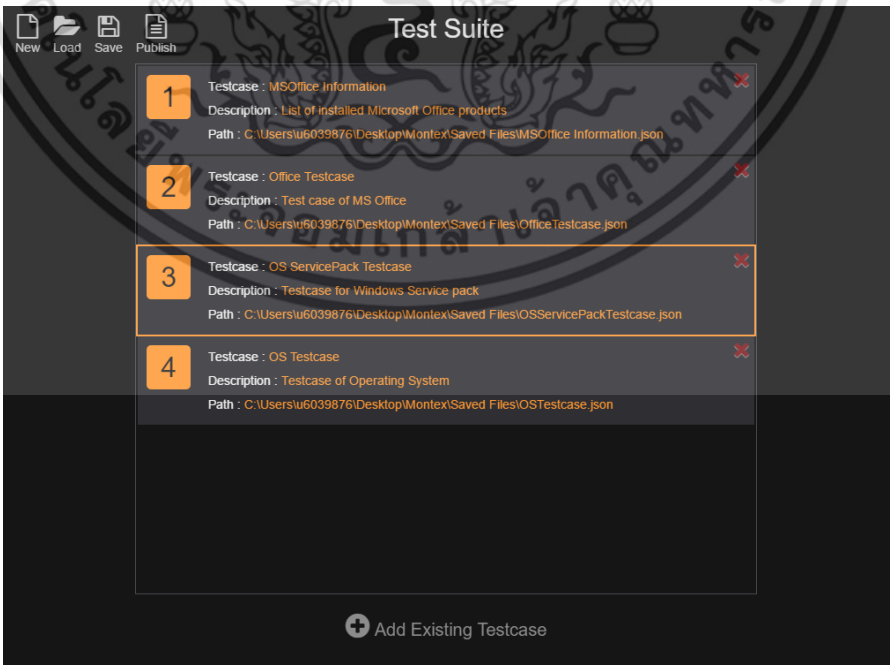
รูปที่ 4.14 รายการของกรณีทดสอบที่อยู่ในชุดทดสอบ

4.2.3 การจัดลำดับกรณีทดสอบ

การทำงานของโปรแกรมนั้นจะเรียงลำดับตามกรณีทดสอบที่อยู่ในรายการ ดังนั้น จึงต้องสามารถปรับเปลี่ยนลำดับของกรณีทดสอบได้เพื่อความสะดวกในการใช้งาน ทำให้ไม่ต้องคำนึงถึงการจัดลำดับขณะเพิ่มกรณีทดสอบใหม่เข้ามา เนื่องจากสามารถปรับเปลี่ยนลำดับของกรณีทดสอบได้ตลอดเวลา การจัดลำดับกรณีทดสอบแสดงดังรูปที่ 4.15 และรูปที่ 4.16



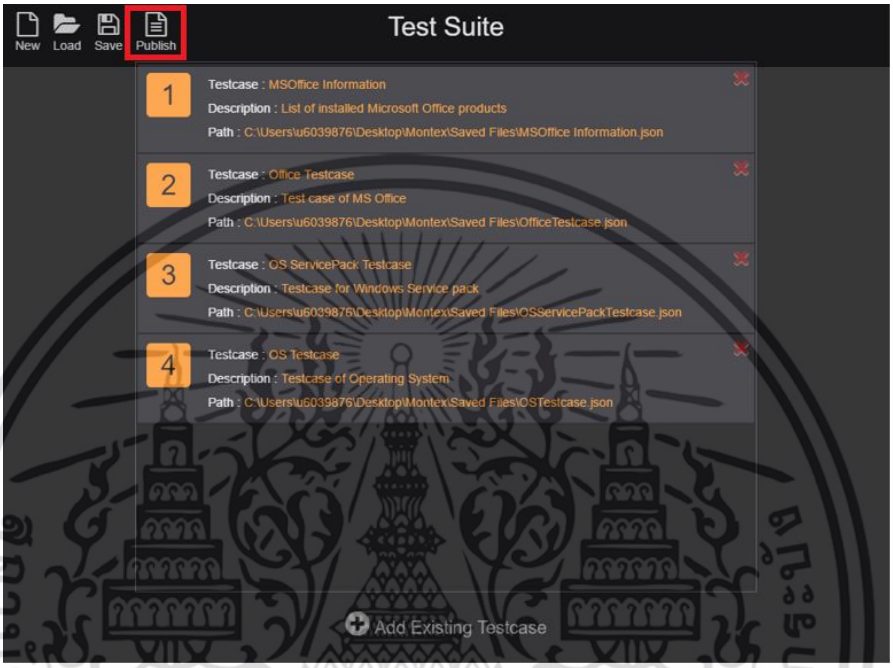
รูปที่ 4.15 เลือกกรณีทดสอบที่ต้องการปรับลำดับ



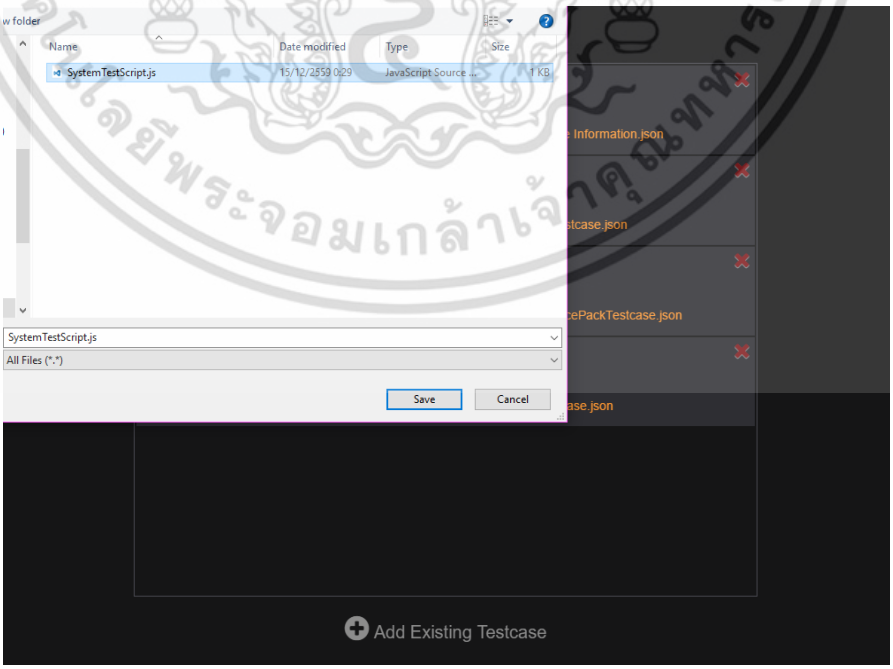
รูปที่ 4.16 รายการของกรณีทดสอบหลังการปรับลำดับ

4.2.4 การสร้างสคริปต์ทดสอบ

ในการทำให้ชุดทดสอบสามารถนำไปใช้งานได้จริงนั้น จะต้องถูกแปลงเป็นสคริปต์ทดสอบเสียก่อน โดยการกดปุ่ม Publish ซึ่งกรณีทดสอบทั้งหมดที่อยู่ในชุดทดสอบจะถูกแปลงเป็นสคริปต์ทดสอบเพียงหนึ่งไฟล์ ที่สามารถนำไปใช้งานได้ทันที หลังจากทำการแปลงเสร็จเรียบร้อยแล้วจะมีหน้าต่างขึ้นมาให้ใส่ชื่อและบันทึกไฟล์สคริปต์ทดสอบ ดังรูปที่ 4.17 และรูปที่ 4.18

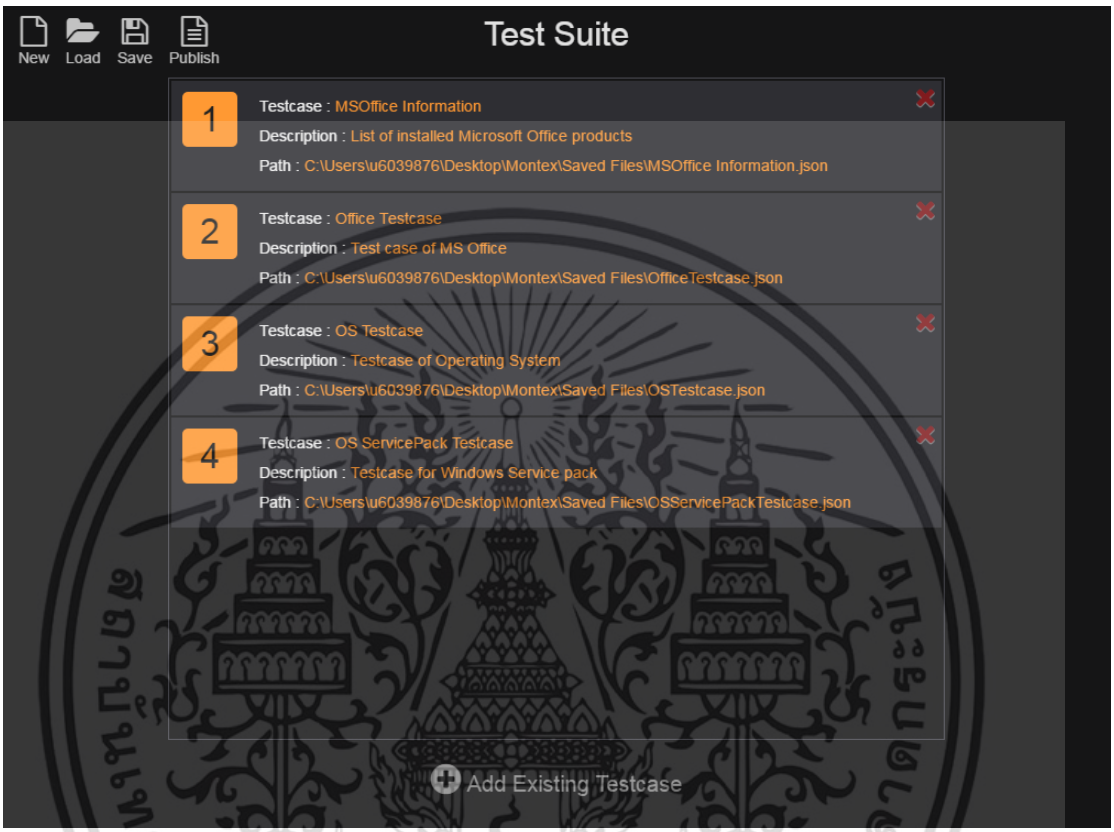


รูปที่ 4.17 การสร้างสคริปต์ทดสอบ



รูปที่ 4.18 การบันทึกสคริปต์ทดสอบ

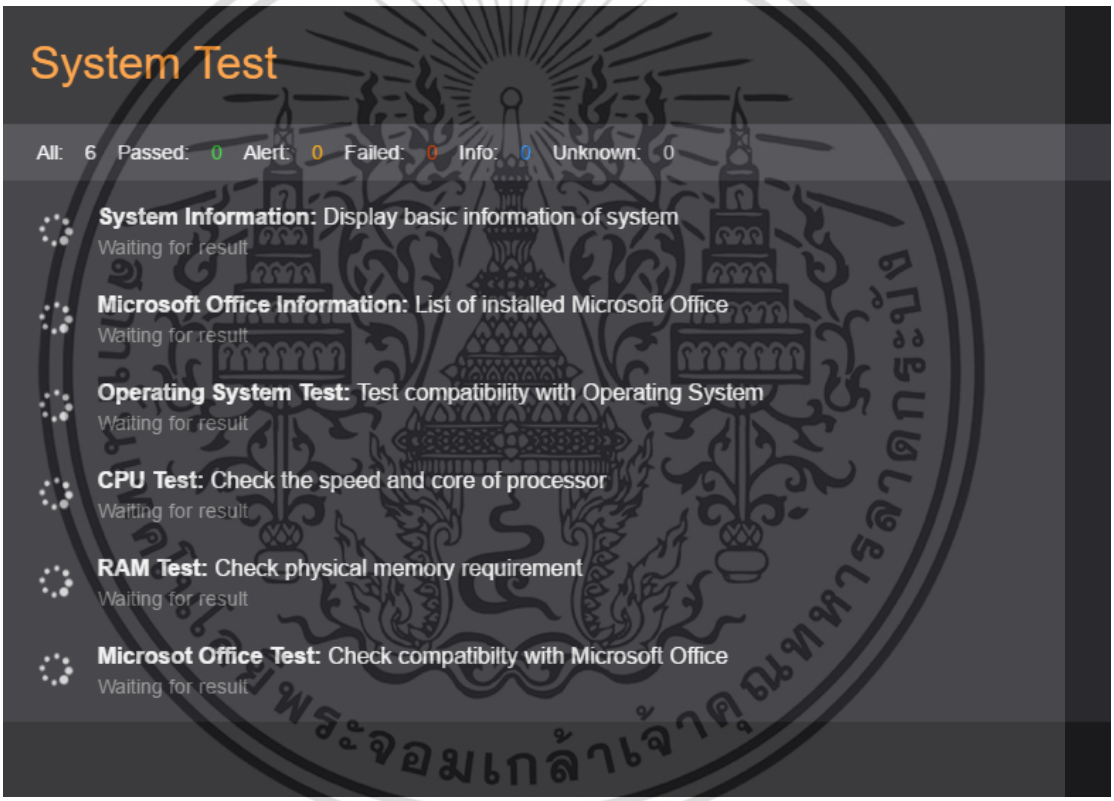
โปรแกรม Test Suite ใช้เพื่อรวบรวมกรณีทดสอบที่สมบูรณ์แล้วเข้าด้วยกันเป็นชุดทดสอบ สามารถจัดลำดับกรณีทดสอบได้หากต้องการทดสอบตามลำดับความสำคัญ โดยทดสอบกรณีที่มีความสำคัญมากที่สุดก่อน และสามารถแปลงชุดทดสอบที่สมบูรณ์แล้วให้กลายเป็นสคริปต์ทดสอบเพื่อนำไปใช้งานจริง ตัวอย่างของชุดทดสอบแสดงดังรูปที่ 4.19



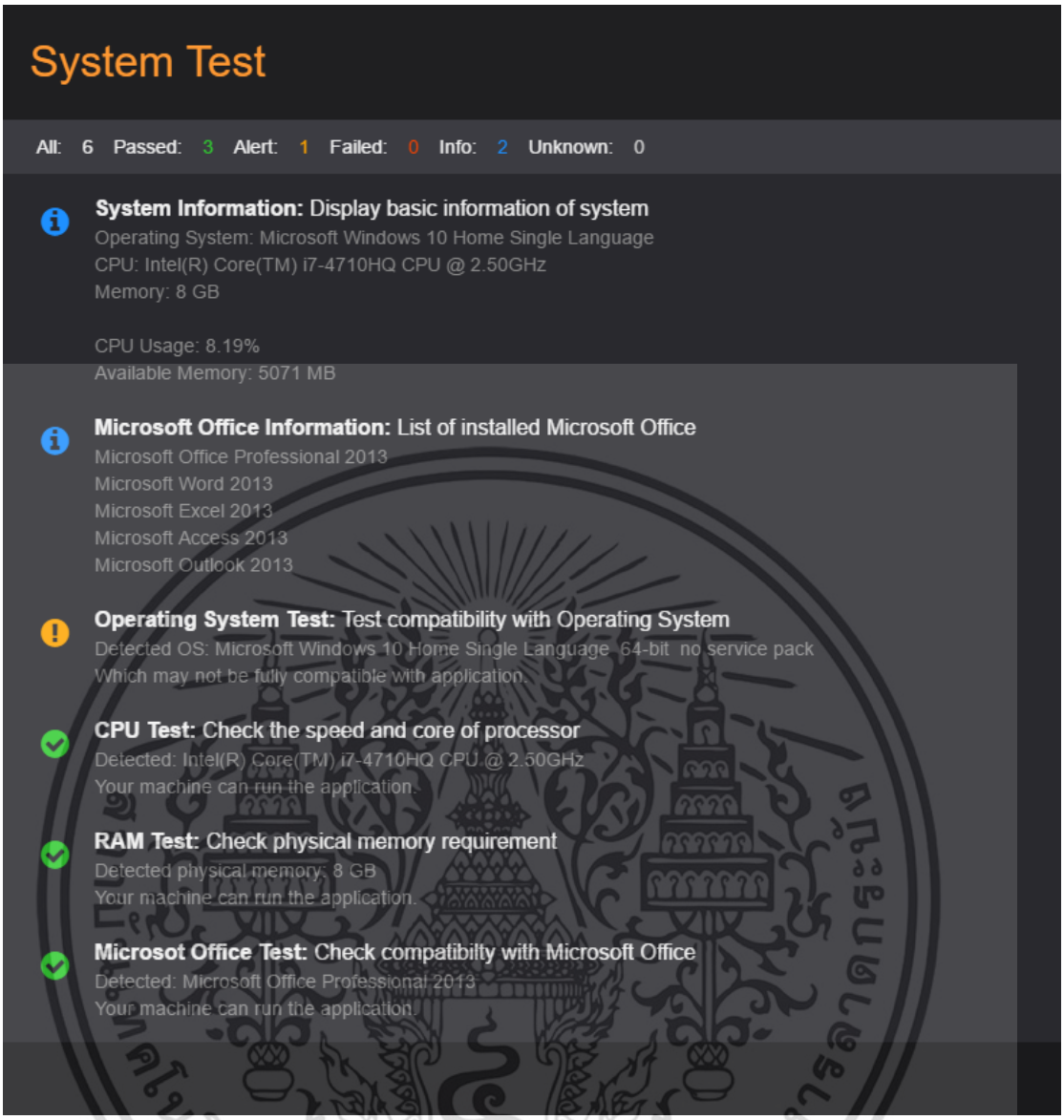
รูปที่ 4.19 ตัวอย่างชุดทดสอบ

4.3 โปรแกรม System Test

โปรแกรม System Test เป็นโปรแกรมที่ใช้สำหรับการทดสอบที่เครื่องของผู้ใช้จริง ๆ โดยใช้สคริปต์ทดสอบที่ได้จากโปรแกรม Test Suite ซึ่งจะสั่งสคริปต์นั้นให้ทำงาน โดยการทำงานของ System Test คือจะไปเก็บข้อมูลของระบบคอมพิวเตอร์ที่เรียกใช้โปรแกรมตามที่กำหนดไว้ในสคริปต์ทดสอบ เช่น ข้อมูลของระบบปฏิบัติการ ความเร็วของหน่วยประมวลผล หรือข้อมูลของโปรแกรม Microsoft Office ที่ติดตั้งอยู่ เป็นต้น ขึ้นอยู่กับว่าในแต่ละกรณีทดสอบนั้นต้องการข้อมูลใดเพื่อนำมาทดสอบบ้าง หลังจากได้ข้อมูลทรัพยากรของระบบมาครบถ้วนแล้วก็จะนำมาตรวจสอบกับแต่ละกรณีทดสอบ ว่าตรงกับเงื่อนไขใดบ้าง แล้วจึงแสดงผลลัพธ์พร้อมทั้งสถานะของการทดสอบของแต่ละกรณีทดสอบ โดยตัวอย่างของการทำงานแสดงดังรูปที่ 4.20 และรูปที่ 4.21



รูปที่ 4.20 โปรแกรม System Test ขณะทำการทดสอบ



รูปที่ 4.21 ผลลัพธ์การทดสอบของโปรแกรม System Test

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

โปรแกรมตรวจสอบทรัพยากรระบบสามารถช่วยอำนวยความสะดวกให้กับผู้ใช้งานที่กำลังสร้างแอปพลิเคชันใหม่ แล้วต้องการเครื่องมือที่ช่วยในการตรวจสอบว่าทรัพยากรของระบบที่จะทำงานนั้นเพียงพอต่อความต้องการหรือไม่ โดยมีความสะดวกในการใช้งาน เนื่องจากโปรแกรมนี้ออกแบบมาสำหรับผู้ใช้งานทั่วไป ใช้งานผ่านส่วนต่อประสานกราฟิกที่เข้าใจง่าย มีเครื่องมือที่ช่วยให้การสร้างกรณีทดสอบมีความสะดวกมากขึ้น ผู้ใช้ไม่จำเป็นต้องมีความสามารถในการเขียนโปรแกรมด้วยภาษาคอมพิวเตอร์ ก็สามารถสร้างกรณีทดสอบและชุดทดสอบได้ อีกทั้งยังรองรับการทำงานแบบหลายแพลตฟอร์ม ทำให้สามารถนำไปใช้งานได้ในทุกแอปพลิเคชัน

5.2 ข้อจำกัดของสหกิจศึกษา

1) ส่วนประกอบพื้นฐานที่มีมาให้นั้นยังมีจำนวนไม่มากนัก ซึ่งสามารถใช้สร้างกรณีทดสอบโดยทั่วไปได้ แต่หากต้องการทดสอบค่าบางอย่างของระบบที่โปรแกรมยังไม่รองรับ ผู้ใช้จะต้องเขียนโปรแกรมเพื่อสร้างส่วนประกอบใหม่ขึ้นมาด้วยตนเอง

2) จำเป็นต้องใช้เซิร์ฟเวอร์ในการเก็บสคริปต์ทดสอบที่แปลงมาจากชุดทดสอบ เพื่อให้ผู้ใช้โปรแกรมทั้งหมดได้ใช้สคริปต์ที่ปรับปรุงล่าสุดเสมอ

5.3 ข้อเสนอแนะในการพัฒนาสหกิจศึกษา

1) เพื่อความสะดวกมากยิ่งขึ้นในการทดสอบโดยการจำลองค่าของทรัพยากรระบบ หากมีเครื่องมือที่ช่วยติดตามการทดสอบในกรณีที่ผลการทดสอบนั้นไม่ผ่าน ว่ามีการตรวจสอบเงื่อนไขใดบ้าง พร้อมทั้งแสดงค่าและผลลัพธ์ของการตรวจสอบ โดยเรียงตามลำดับของเงื่อนไขที่ตรวจสอบ ก็จะทำให้ผู้ใช้ทราบว่าผลการทดสอบนั้นไม่ผ่านเพราะเหตุใด

2) การแยกกลุ่มของกรณีทดสอบในหน้าต่างการจัดการชุดทดสอบ โดยขณะเพิ่มกรณีทดสอบใหม่เข้ามาจะสามารถเลือกได้ว่าจะจัดกรณีทดสอบนั้นอยู่ในกลุ่มใด เช่น กลุ่มการทดสอบฮาร์ดแวร์ กลุ่มการทดสอบซอฟต์แวร์ในระบบ หรือกลุ่มการทดสอบระบบเครือข่าย เป็นต้น

เอกสารอ้างอิง

[1] **Software Testing.** [ออนไลน์]. สืบค้นจาก: https://en.wikipedia.org/wiki/Software_testing. เข้าถึงเมื่อวันที่ 1 พฤษภาคม 2560

[2] **Test Case.** [ออนไลน์]. สืบค้นจาก: https://en.wikipedia.org/wiki/Test_case. เข้าถึงเมื่อวันที่ 1 พฤษภาคม 2560

[3] **Test Case Execution.** [ออนไลน์]. สืบค้นจาก: https://testfort.com/test-case-execution. เข้าถึงเมื่อวันที่ 9 พฤษภาคม 2560

[4] **Test Suite.** [ออนไลน์]. สืบค้นจาก: https://en.wikipedia.org/wiki/Test_suite. เข้าถึงเมื่อวันที่ 1 พฤษภาคม 2560

[5] **System Requirements.** [ออนไลน์]. สืบค้นจาก: https://en.wikipedia.org/wiki/System_requirements. เข้าถึงเมื่อวันที่ 1 พฤษภาคม 2560

[6] **Single-page Application.** [ออนไลน์]. สืบค้นจาก: https://en.wikipedia.org/wiki/Single-page_application. เข้าถึงเมื่อวันที่ 1 พฤษภาคม 2560

[7] **Node.js.** [ออนไลน์]. สืบค้นจาก: https://nodejs.org. เข้าถึงเมื่อวันที่ 1 พฤษภาคม 2560

[8] **npm.** [ออนไลน์]. สืบค้นจาก: https://npmjs.com. เข้าถึงเมื่อวันที่ 9 พฤษภาคม 2560

[9] **One framework – Angular.** [ออนไลน์]. สืบค้นจาก: https://angular.io. เข้าถึงเมื่อวันที่ 1 พฤษภาคม 2560

[10] **Electron | Build cross platform desktop apps with JavaScript, HTML, and CSS.** [ออนไลน์]. สืบค้นจาก: https://electron.atom.io. เข้าถึงเมื่อวันที่ 1 พฤษภาคม 2560