รายงานสหกิจศึกษาฉบับสมบูรณ์

เว็บแอปพลิเคชันสำหรับการทดสอบซอฟต์แวร์แบบอัตโนมัติ
Web Application for Automated Software Testing

นางสาวณัฐกานต์ จึงไพศาลทรัพย์

ภาควิชาวิศวกรรมสารสนเทศ
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

รายงานสหกิจศึกษาฉบับสมบูรณ์

เว็บแอปพลิเคชันสำหรับการทดสอบซอฟต์แวร์แบบอัตโนมัติ

Web Application for Automated Software Testing

นางสาวณัฐกานต์ จึงไพศาลทรัพย์

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

ชื่อโครงงานสหกิจศึกษา เว็บแอปพลิเคชันสำหรับการทดสอบซอฟต์แวร์แบบอัตโนมัติ

ชื่อ-สกุล นักศึกษา นางสาวณัฐกานต์ จึงไพศาลทรัพย์

คณะ วิศวกรรมศาสตร์ **ภาควิชา** วิศวกรรมสารสนเทศ

ชื่อ-สกุล อาจารย์นิเทศ ผศ.มยุรี เลิศเวชกุล

ชื่อ-สกุล ผู้นิเทศงาน นายพุทธา กุศลกรรมบถ

สถานประกอบการ บริษัทรอยเตอร์ ซอฟต์แวร์ (ประเทศไทย) จำกัด

บทคัดย่อ

เว็บแอปพลิเคชันสำหรับการทดสอบซอฟต์แวร์แบบอัตโนมัติ ถูกจัดทำขึ้นเพื่ออำนวยความสะดวกในการทดสอบซอฟต์แวร์ โดยผู้ใช้สามารถสร้างการตั้งค่าเกี่ยวกับการทดสอบซอฟต์แวร์ เช่น เทสสคริปท์ (Test Script) สำหรับการทดสอบซอฟต์แวร์ เครื่องที่จะใช้ในการทดสอบ และเวลาที่ต้องการให้ทดสอบซอฟต์แวร์แบบอัตโนมัติ อีกทั้งยังสามารถเปิดหรือปิดโหมดการทดสอบซอฟต์แวร์แบบอัตโนมัติ และสามารถติดตามผลลัพธ์ของการทดสอบผ่านทางเว็บแอปพลิเคชัน

คำสำคัญ : เว็บแอปพลิเคชัน การทดสอบซอฟต์แวร์แบบอัตโนมัติ

Cooperative Title: Web Application for Automated Software Testing

Student intern name: Miss Nuttakarn Jungphisansup

Faculty: Engineering **Department:** Information Engineering

Advisor name: Asst.Prof. Mayuree Lertwetchakul

Mentor name: Mr. Phuttha Kusolkumbot

Company: Reuters Software (Thailand) Ltd.

ABSTRACT

Web application for automated software testing was prepared to facilitate software testing. Users can create Scenario (software testing scheme), such as a testing script, Instance (Machine to be used in the software testing) and Schedule (the time that users require to test the software at the point of time automatically). Moreover, users can also turn on/off automatic software testing mode and able to track results of the testing via the web application.

Keywords: Web Application, Automated Software Testing

กิตติกรรมประกาศ

โครงงานเว็บแอปพลิเคชันสำหรับการทดสอบซอฟต์แวร์แบบอัตโนมัติ ได้รับการสนันสนุนและการอนุเคราะห์จากผู้มีส่วนเกี่ยวข้องหลายฝ่าย ดิฉันในฐานะผู้จัดทำโครงงานขอแสดงความขอบคุณ

ผศ. มยุรี เลิศเวชกุล อาจารย์ที่ปรึกษาโครงงานสหกิจศึกษา ที่ให้กำลังใจและคำแนะนำตลอดการทำโครงงาน รวมถึงการตรวจข้อผิดพลาดของโครงงานจนโครงงานนี้สำเร็จไปได้ด้วยดี ดิฉันขอขอบพระคุณอาจารย์เป็นอย่างสูง

นายพุทธา กุศลกรรมบถ ผู้จัดการทีม CTA (Compass Test Automation) ที่ให้ทัศนคติในการทำงานให้สำเร็จภายใต้แรงกดดัน ความเอาใจใส่และการดูแลเป็นอย่างดีตลอดการเป็นนักศึกษาฝึกงาน

และสุดท้าย ขอขอบคุณครอบครัวและเพื่อนๆ ที่คอยให้กำลังใจเสมอ อยู่เคียงข้างตลอดในวันที่รู้สึกเหนื่อย ท้อแท้ หรือมีเรื่องแย่เข้ามาในชีวิต

ณัฐกานต์ จึงไพศาลทรัพย์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ หน้า

สารบัญภาพ ต่อ)

ภาพที่ หน้า

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1 บทนำ

1.1 ความเป็นมาและความสำคัญ

เนื่องจากทีม CTA (Compass Test Automation) ต้องการสร้างเว็บแอปพลิเคชันสำหรับการทดสอบซอฟต์แวร์แบบอัตโนมัติ ที่ใช้งานง่ายและสะดวกต่อการทดสอบซอฟต์แวร์ สามารถทดสอบซอฟต์แวร์แบบอัตโนมัติ และสามารถติดตามผลลัพธ์ของการทดสอบซอฟต์แวร์ผ่านทางเว็บแอปพลิเคชัน โดยทางทีมได้จัดเตรียมเซอร์วิส (Service) สำหรับระบบการทดสอบซอฟต์แวร์แบบอัตโนมัติไว้เป็นที่เรียบร้อยแล้ว แต่ยังขาดส่วนติดต่อผู้ใช้งานในรูปแบบของเว็บแอปพลิเคชันที่จะนำมาเรียกใช้งานเซอร์วิสในระบบ อีกทั้งตัวผู้จัดทำโครงงานเองมีความสนใจที่จะพัฒนาเว็บแอปพลิเคชัน จึงได้จัดทำโครงงานเว็บแอปพลิเคชันสำหรับการทดสอบซอฟต์แวร์แบบอัตโนมัติขึ้นมา

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อสร้างเว็บแอปพลิเคชันที่ง่ายและสะดวกต่อการทดสอบซอฟต์แวร์ เช่น การจัดการ Scenario (สร้าง/แก้ไข/รัน/ลบ การตั้งค่าเกี่ยวกับการทดสอบซอฟต์แวร์) การจัดการ Instance (แก้ไขคุณสมบัติของเครื่องที่จะใช้ในการทดสอบซอฟต์แวร์) และการแสดงผลของการทดสอบซอฟต์แวร์

1.2.2 เพื่ออำนวยความสะดวกให้ผู้ใช้สามารถทดสอบซอฟต์แวร์แบบอัตโนมัติผ่านเว็บแอปพลิเคชัน เช่น ผู้ใช้สามารถสั่งให้เว็บแอปพลิเคชันทดสอบเว็บไซต์ ทุกๆ 5 นาที

1.2.3 เพื่อให้ผู้ใช้สามารถติดตามผลลัพธ์ของการทดสอบซอฟต์แวร์ผ่านทางเว็บแอปพลิเคชัน โดยจะแสดงให้ทราบว่าเป็นการทดสอบอะไร เริ่มทดสอบเมื่อใด ใช้เวลาทดสอบเท่าไหร่ สถานะการทดสอบเป็นอย่างไร เป็นต้น

1.3 วิธีการดำเนินงาน

1.3.1 ศึกษา Angular4, Angular Material และ Jasmine สำหรับนำมาใช้ในการพัฒนาเว็บแอปพลิเคชัน

1.3.2 ทำความเข้าใจการไหลของเซอร์วิส (Service flow) และแบบจำลองข้อมูล (Data model) ของระบบ CTA

1.3.3 ทดสอบ API (Application Programming Interface) ของเซอร์วิสแต่ละตัวในระบบ CTA

1.3.4 ออกแบบส่วนติดต่อผู้ใช้งาน (User Interface)

1.3.5 พัฒนาโปรแกรม

1.3.6 ทดสอบเว็บแอปพลิเคชันและทำการแก้ไขหากพบข้อผิดพลาด

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ผู้ใช้สามารถจัดการการตั้งค่าเกี่ยวกับการทดสอบซอฟต์แวร์ (Scenario) และเครื่องที่จะใช้ในการทดสอบซอฟต์แวร์ (Instance) ได้อย่างง่ายและสะดวกสบาย

1.4.2 ผู้ใช้ไม่จำเป็นต้องสั่งให้เว็บแอปพลิเคชันทดสอบซอฟต์แวร์ด้วยตัวผู้ใช้เอง ในกรณีที่ผู้ใช้ต้องการทดสอบซอฟต์แวร์ทุกๆ ช่วงเวลาที่ซ้ำกัน เช่น ต้องการทดสอบซอฟต์แวร์ทุกๆ 5 นาที

1.4.3 ผู้ใช้สามารถติดตามผลของการทดสอบซอฟต์แวร์ผ่านทางเว็บแอปพลิเคชัน โดยไม่จำเป็นต้องเข้าไปดูผลลัพธ์ที่เกิดขึ้นจากเครื่องที่ใช้ทดสอบซอฟต์แวร์

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง

2.1 Angular [1]

2.1.1 Angular คืออะไร

Angular คือ แพลตฟอร์ม (Platform) ที่ทำให้สามารถสร้างแอปพลิเคชันด้วยเว็บได้อย่างง่ายดาย ซึ่งประกอบด้วย Declarative templates, Dependency injection, End to end tooling และแนวทางปฏิบัติที่ดีที่สุดสำหรับการพัฒนา

รูปที่ 2.1 สัญลักษณ์ของ Angular

2.1.2 Feature [2]

1) Cross platform

- Progressive Web Apps : ใช้ความสามารถของเว็บแพลตฟอร์มสมัยใหม่เพื่อมอบประสบการณ์แบบแอปพลิเคชั่น ซึ่งมีประสิทธิภาพสูง ออฟไลน์ และไม่มีขั้นตอนการติดตั้ง ทำให้เว็บแอปพลิเคชันมีความก้าวหน้า
- Native : สามารถสร้าง Native mobile app ด้วยกลวิธีจาก Ionic Framework, NativeScript และ React Native
- Desktop : สามารถสร้างแอปบนเดสก์ท็อปข้ามระบบปฏิบัติการ Mac, Windows และ Linux โดยใช้ความสามารถในการเข้าถึง Native OS API

2) Speed and performance

- Code Generation : สามารถเปลี่ยนเทมเพลต (Template) ให้กลายเป็นโค้ดที่มีความเหมาะสมสูงสำหรับ JavaScript virtual machines ทำให้ได้รับประโยชน์ทั้งหมดจากการเขียนโค้ดด้วยโปรดัก (Product) ของเฟรมเวิร์ก (Framework)

- Universal : ใช้มุมมองแรกของแอปพลิเคชันบน node.js, .NET, PHP และเซิร์ฟเวอร์ (Server) อื่นๆ สำหรับการแสดงผลเกือบทันทีในรูปแบบของ HTML และ CSS นอกจากนี้ยังปูทางเพื่อไซต์ที่เพิ่มประสิทธิภาพสำหรับการปรับแต่งเว็บไซต์ให้ติดอันดับการค้นหาในตำแหน่งที่ดีที่สุด (SEO : Search engine optimization)

- Code Splitting : สามารถโหลดแอปพลิเคชันได้อย่างรวดเร็วโดยใช้ Component router ทำให้ผู้ใช้โหลดเฉพาะโค้ดที่ต้องการจะแสดงผล

3) Productivity

- Templates : สามารถสร้างส่วนติดต่อผู้ใช้งานได้อย่างรวดเร็วด้วย Template syntax ที่เรียบง่ายและมีประสิทธิภาพ

- Angular CLI : เป็น Command line tool ที่ทำให้สามารถเริ่มต้นแอปพลิเคชัน เพิ่ม components และทำการทดสอบได้อย่างรวดเร็ว

- IDEs : ได้รับ Code completion, Instant errors และ feedback ใน editor และ IDE (Integrated development environment) ยอดนิยม

4) Full development story

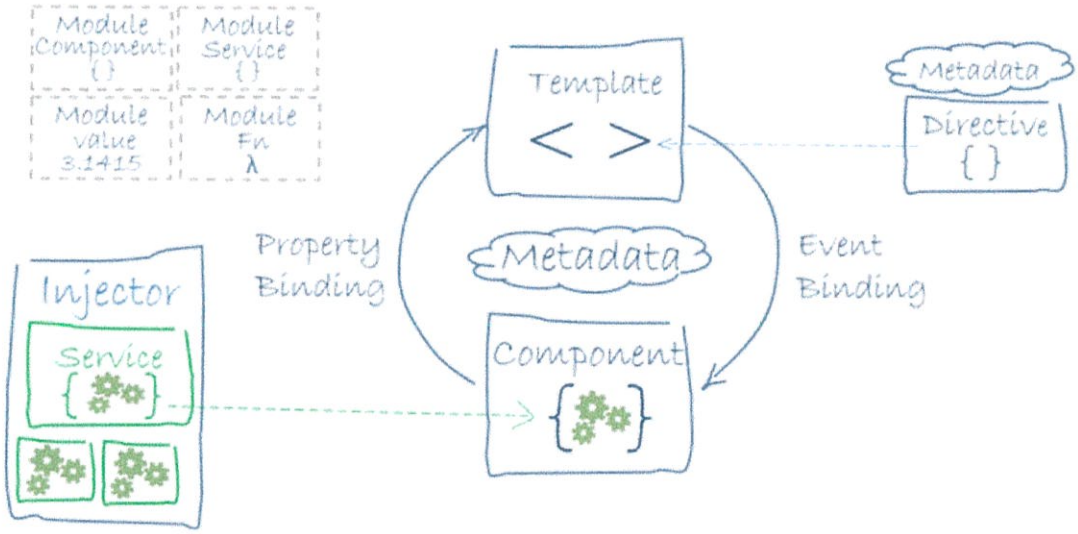
- Testing : ใช้ Karma สำหรับการทำ Unit test ซึ่งจะทำให้เราทราบผลการทดสอบทุกครั้งที่บันทึก และใช้ Protractor ทำให้รัน Scenario test ได้อย่างรวดเร็วและมีเสถียรภาพ

- Animation : สามารถสร้างผลงานที่ซับซ้อนและมีคุณภาพสูงด้วยโค้ดขนาดสั้นผ่าน Angular API ที่ใช้งานง่าย

- Accessibility : สามารถสร้างแอปพลิเคชันที่สามารถเข้าถึงด้วย ARIA-enabled component, Developer guide และ a11y Test infrastructure

2.1.3 Architecture [3]

Angular เป็นเฟรมเวิร์กสำหรับการสร้าง Client application ในรูปแบบ HTML และ JavaScript อย่างเช่น ภาษา TypeScript ที่คอมไพล์ไปเป็น JavaScript ซึ่งเฟรมเวิร์กนี้จะประกอบด้วยหลายไลบรารี (Library) บางไลบรารีเป็น Core และบางไลบรารีเป็น Optional เราสามารถเขียน Application โดยการเขียน HTML template เขียน Component class เพื่อจัดการ Template เพิ่มลอจิก (Logic) ในเซอร์วิส และเก็บ Component กับเซอร์วิสไว้ในโมดูล (Module) จากนั้นเราสามารถเปิดแอปพลิเคชันโดยการบูต Root module ซึ่งจะใช้เวลาแสดงเนื้อหาแอปพลิเคชันในเบราว์เซอร์ (Browser) และตอบสนองผู้ใช้ตามโครงสร้างที่เราได้จัดเตรียมไว้แล้ว



รูปที่ 2.2 ภาพรวมสถาปัตยกรรมของ Angular

1) Modules

- Angular มีระบบโมดูลที่เรียกว่า NgModule
- ทุกแอปพลิเคชันต้องมี NgModule อย่างน้อย 1 class นั่นคือ Root module ซึ่งโดยทั่วไปจะชื่อว่า AppModule
- ถึงแม้ว่า Root module อาจจะเป็นโมดูลในแอปพลิเคชันเล็กๆ แต่แอปพลิเคชันส่วนใหญ่จะมีหลาย Feature module เอาไว้สำหรับจำกัดโค้ดที่เป็น Application domain, workflow หรือชุดความสามารถที่เกี่ยวข้องกันอย่างใกล้ชิด
- เราจะประกาศ NgModule หรือ Feature module ไว้ใน @NgModule
- NgModule เป็น Decorator function ที่สร้าง Metadata object ไว้เป็นคุณสมบัติ (Property) ที่ใช้อธิบายโมดูล ซึ่งประกอบด้วย 5 คุณสมบัติดังนี้

    1. declarations : เป็น View class ของโมดูล ซึ่งมี 3 ประเภท ได้แก่ Component, Directive และ Pipe
    2. exports : เป็นสับเซตของ declarations ที่ควรถูกใช้ใน Component template ของโมดูลอื่น
    3. imports : โมดูลอื่นของ exported class ที่ Component template ในโมดูลนี้ต้องการ
    4. providers : เป็นตัวสร้างเซอร์วิสที่โมดูลนี้มีส่วนร่วมในการเก็บเซอร์วิส ทำให้สามารถเข้าถึงเซอร์วิสได้จากทุกส่วนของแอปพลิเคชัน
    5. bootstrap : เป็นส่วนที่ใช้ประกาศ Root component ซึ่งเป็นมุมมองหลักของแอปพลิเคชัน

src/app/app.module.ts

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

รูปที่ 2.3 ตัวอย่าง Root module

2) Angular libraries

- แต่ละ Angular library จะขึ้นต้นด้วย @angular
- เราสามารถติดตั้ง Angular library ด้วย npm (Package manager) และนำไลบรารีนั้นเข้ามาใช้งานด้วยประโยคคำสั่ง JavaScript เช่น import { Component } from '@angular/core';

3) Components

- Component จะเป็นส่วนที่ควบคุม View
- เราสามารถกำหนดลอจิกของ Component ว่าต้องการสนันสนุน View ไหนใน Class ซึ่ง Class จะติดต่อกับ View ผ่าน API และ Method

src/app/hero-list.component.ts (class)

```
export class HeroListComponent implements OnInit {
  heroes: Hero[];
  selectedHero: Hero;

  constructor(private service: HeroService) { }

  ngOnInit() {
    this.heroes = this.service.getHeroes();
  }

  selectHero(hero: Hero) { this.selectedHero = hero; }
}
```

รูปที่ 2.4 ตัวอย่าง Component

4) Templates

- เราสามารถกำหนด View ของ Component ด้วย Template ซึ่ง Template จะอยู่ในรูปแบบของ HTML ที่บอก Angular ว่าจะให้แสดง Component อย่างไร

src/app/hero-list.component.html

```
<h2>Hero List</h2>

<p><i>Pick a hero from the list</i></p>
<ul>
  <li *ngFor="let hero of heroes" (click)="selectHero(hero)">
    {{hero.name}}
  </li>
</ul>

<app-hero-detail *ngIf="selectedHero" [hero]="selectedHero"></app-hero-detail>
```

รูปที่ 2.5 ตัวอย่าง Template

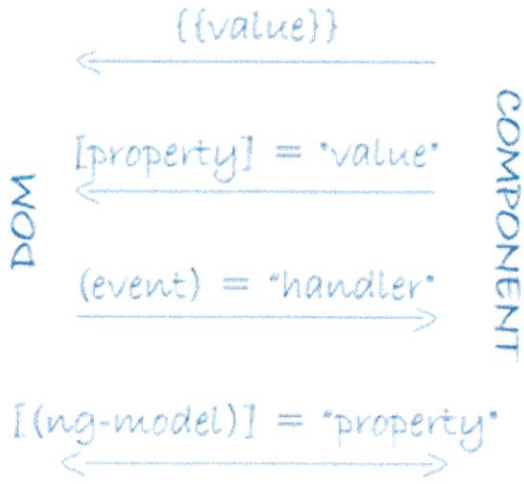
5) Metadata

- เป็นสิ่งที่บอก Angular ว่าจะโปรเซส Class อย่างไร
- Metadata ใน @Component จะบอก Angular ว่าจะรับ Building block หลัก ที่เราระบุสำหรับ Component ได้ที่ไหน ซึ่งประกอบด้วย 4 ทางเลือก

  1. selector : เป็น CSS selector ที่บอก Angular เพื่อสร้างหรือเพิ่ม Instance ของ Component
  2. templateUrl : ที่อยู่ของ Template ที่สัมพันธ์กับโมดูล
  3. providers : เป็น Array เก็บ Service ที่ Component ต้องการ

6) Data binding

- Angular สนับสนุนการผูกข้อมูลระหว่าง DOM (Document Object Model) กับ Component โดยการเพิ่ม Markup ที่ Template เพื่อบอก Angular ว่าทั้งสองส่วนเชื่อมต่อกันอย่างไร



รูปที่ 2.6 Data binding

src/app/hero-list.component.html (binding)

```
<li>{{hero.name}}</li>
<app-hero-detail [hero]="selectedHero"></app-hero-detail>
<li (click)="selectHero(hero)"></li>
```

รูปที่ 2.7 ตัวอย่างการทำ Data binding

7) Directives

- เป็น Class ที่ประกาศด้วย @Directive เช่น Component เป็น Directive อย่างหนึ่งที่มาพร้อมกับTemplate (@Component)
- Angular template มีลักษณะเป็นไดนามิค เมื่อ Angular ทำการเรนเดอร์ จะแปลง DOM ให้กลายเป็นโครงสร้างด้วย Directive

8) Services

- เป็นหมวดหมู่ที่ครอบคลุมต่างๆ ฟังก์ชั่น หรือฟีเจอร์ที่แอปพลิเคชันต้องการ
- เกือบทุกอย่างสามารถเป็นเซอร์วิสได้ โดยเซอร์วิสมักจะเป็น Class ที่มีจุดประสงค์ชัดเจน เช่น Logging service, Data service และ Message bus เป็นต้น

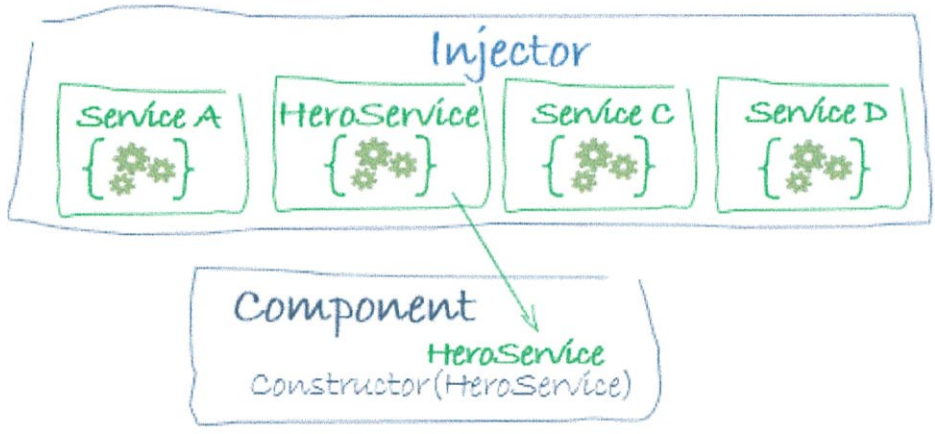
src/app/hero.service.ts (class)

```
export class HeroService {
  private heroes: Hero[] = [];

  constructor(
    private backend: BackendService,
    private logger: Logger) { }

  getHeroes() {
    this.backend.getAll(Hero).then( (heroes: Hero[]) => {
      this.logger.log(`Fetched ${heroes.length} heroes.`);
      this.heroes.push(...heroes); // fill cache
    });
    return this.heroes;
  }
}
```

รูปที่ 2.8 ตัวอย่าง Service

9) Dependency injection

- เป็นวิธีหนึ่งที่สนันสนุน Instance ใหม่ของ Class ด้วย Fully-formed dependency โดยส่วนใหญ่แล้ว Dependency จะเป็นเซอร์วิส ซึ่ง Angular ใช้ Dependency injection เพื่อจัดเตรียม Component ใหม่ด้วยเซอร์วิสที่ต้องการ



รูปที่ 2.9 Dependency injection

2.2 ภาษาที่ใช้ในการพัฒนาเว็บแอปพลิเคชัน

2.2.1 Typescript [4]

- คือ ซุปเปอร์เซตของ JavaScript แบบมีชนิดข้อมูล (Type) ที่คอมไพล์ (Compile) ไปเป็น Plain JavaScript
- ข้อดีของการใช้ TypeScript
  - สามารถใช้ความสามารถของ ES2015 (JavaScript ES6) ได้ เนื่องจาก TypeScript เป็น JavaScript สมัยใหม่ ที่รวมความสามารถของ ES2015 เอาไว้แล้ว
  - ตัวแปรที่ถูกประกาศขึ้นมาแล้ว จะไม่สามารถเปลี่ยนชนิดข้อมูลได้อีกต่อไป ทำให้ข้อผิดพลาดในโปรแกรมน้อยลง เพราะทำให้เราไม่มีโอกาสใส่ข้อมูลผิดชนิด
  - มีการตรวจสอบโค้ดในช่วง Compile time ทำให้ดักจับข้อผิดพลาดได้ตั้งแต่ต้น ไม่ปล่อยให้ข้อผิดพลาดไปเกิดในตอนทำงานจริง (Runtime)

- IDE และ Text Editor ที่ดีเยี่ยม ให้การสนับสนุน TypeScript

2.2.2 HTML (HyperText Markup Language) [5]

- เป็น Markup Language มาตรฐานที่ใช้ในการสร้างเว็บเพจและเว็บแอปพลิเคชัน ซึ่งภาษา HTML ถูกพัฒนามาจากภาษา SGML (Standard Generalized Markup Language) โดยทิม เบอร์เนอรส์ ลี (Tim Berners Lee) ปัจจุบัน World Wide Web Consortium (W3C) ประกาศให้ภาษา HTML เป็นมาตรฐานหนึ่งของ ISO และผลักดันภาษา XHTML ที่เป็นภาษา HTML รูปแบบใหม่ โดยมีลักษณะโครงสร้างของภาษา XML (Extensible Markup Language) มาใช้แทนภาษา HTML รุ่น 4.01

- เวอร์ชั่นของภาษา HTML

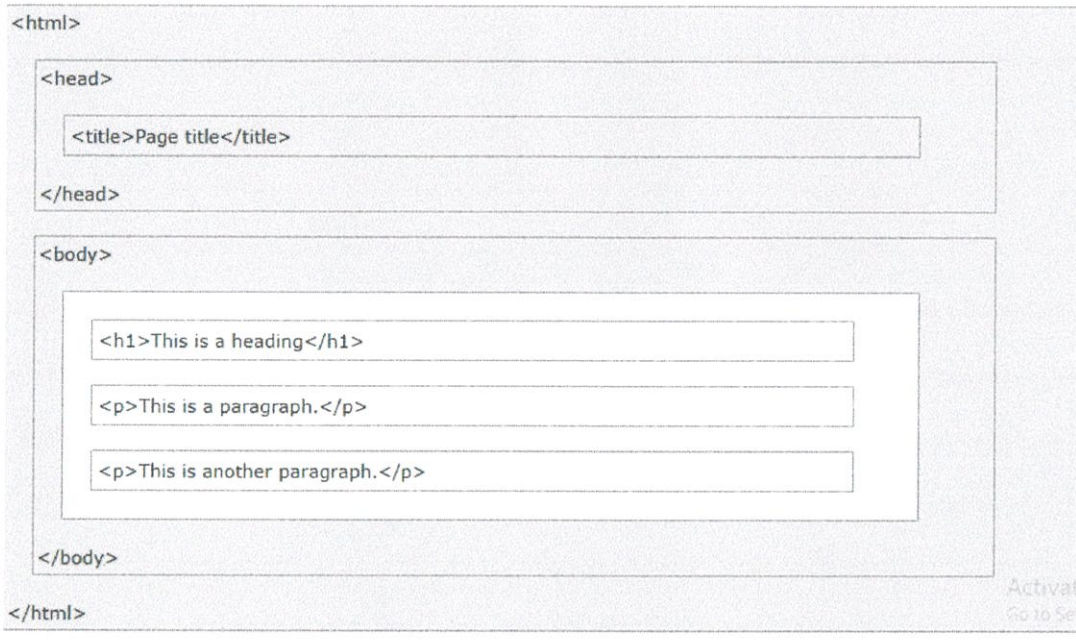
Version	Year
HTML	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2014

รูปที่ 2.10 เวอร์ชั่นของภาษา HTML

- โครงสร้างพื้นฐานของภาษา HTML

1. ส่วนเริ่มต้นของเอกสาร HTML : คือส่วนที่ทำให้เว็บเบราว์เซอร์ทราบว่าเป็นเอกสารข้อมูลชนิด HTML ซึ่งจะมีแท็ก (Tag) <html>...</html> กำกับอยู่ที่จุดเริ่มต้นและจุดสิ้นสุดของเอกสารข้อมูล
2. ส่วนหัวเรื่อง (Head) : คือส่วนที่กำหนดให้แสดงข้อความที่แถบหัวเรื่องของหน้าเว็บเพจ ซึ่งจะใช้แท็ก <head>....</head> กำกับไว้ในส่วนเริ่มต้นของเอกสาร HTML

3. ส่วนเนื้อหา (Body) : คือส่วนที่แสดงเนื้อหาของเว็บเพจทั้งหมด ประกอบด้วยข้อความและแท็กต่างๆ เช่น แท็กสำหรับจัดการกับรูปแบบของข้อความ ตาราง รูปภาพ กราฟิกต่างๆ สีของตัวอักษร สีพื้น เป็นต้น ซึ่งจะใช้แท็ก <body>...</body> กำกับไว้ในส่วนเริ่มต้นของเอกสาร HTML หลังจากประกาศส่วนหัวเรื่องแล้ว



รูปที่ 2.11 โครงสร้างพื้นฐานของภาษา HTML

2.2.3 CSS (Cascading Style Sheets) [6]

- คือ ภาษาที่ใช้เป็นส่วนของการจัดรูปแบบการแสดงผลเอกสาร HTML และ XHTML ซึ่งสามารถประยุกต์ใช้ร่วมกับภาษา XML, กราฟิกแบบเวคเตอร์ที่ใช้แสดงผลบนหน้าเว็บไซต์ (SVG : Scalable Vector Graphics) และ ภาษา XUL (XML User Interface Language) โดยที่ CSS จะกำหนดกฎเกณฑ์ในการระบุรูปแบบของเนื้อหาในเอกสาร ได้แก่ สีของข้อความ สีพื้นหลัง ประเภทตัวอักษร และการจัดวางข้อความ ซึ่งการกำหนดรูปแบบหรือสไตล์จะใช้หลักการของการแยกเนื้อหาเอกสาร HTML ออกจากคำสั่งที่ใช้ในการจัดรูปแบบการแสดงผล กำหนดให้รูปแบบของการแสดงผลเอกสารไม่ขึ้นอยู่กับเนื้อหาของเอกสาร เพื่อให้ง่ายต่อการจัดรูปแบบการ

แสดงผลลัพธ์ของเอกสาร HTML โดยเฉพาะในกรณีที่มีการเปลี่ยนแปลงเนื้อหาเอกสารบ่อยครั้งหรือต้องการควบคุมให้รูปแบบการแสดงผลเอกสาร HTML มีลักษณะของความสม่ำเสมอทั่วกันทุกหน้าเอกสารภายในเว็บไซต์เดียวกัน โดยกฎเกณฑ์ในการกำหนดรูปแบบ (Style) เอกสาร HTML ถูกเพิ่มเข้ามาครั้งแรกใน HTML 4.0 เมื่อปีพ.ศ. 2539 ในรูปแบบของ CSS level 1 Recommendations ที่กำหนดโดยองค์กร W3C

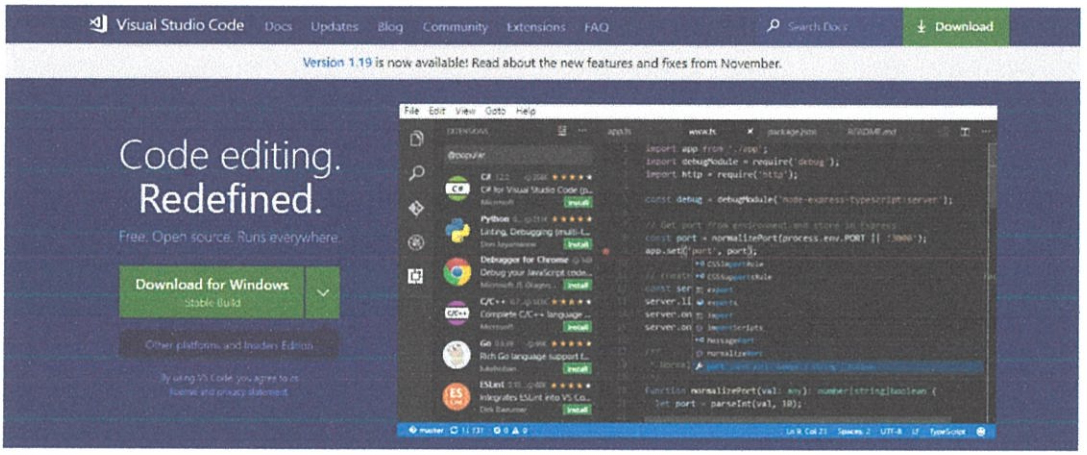
- ประโยชน์ของ CSS
  - ช่วยในการจัดรูปแบบแสดงผลให้กับภาษา HTML ซึ่งจะช่วยลดการใช้ภาษา HTML ให้น้อยลง โดยเหลือเพียงแต่ส่วนที่เป็นเอกสารที่เป็นภาษา HTML เท่านั้น ทำให้มีการแก้ไขและทำความเข้าใจได้ง่ายขึ้น
  - ทำให้ขนาดไฟล์ HTML น้อยลง เนื่องจาก ภาษา CSS จะช่วยลดการใช้ภาษา HTML ทำให้ขนาดไฟล์เล็กลงไปด้วย
  - Style Sheets ชุดเดียวสามารถใช้กำหนดรูปแบบการแสดงผลให้เอกสาร HTML ทั้งหน้า หรือทุกหน้ามีผลเหมือนกันได้ จึงทำให้เวลาที่มีการแก้ไขก็จะแก้ไขได้ง่ายขึ้นเพียงแก้ไข Style Sheets ที่ใช้งานเพียงชุดเดียวเท่านั้น
  - ทำให้เว็บไซต์มีมาตรฐาน เพราะจะทำให้การแสดงผลในสื่อต่างๆ ถูกปรับเปลี่ยนไปได้อย่างเหมาะสม เช่น การแสดงผลบนหน้าจอและการแสดงผลในมือถือ
  - สามารถที่จะใช้งานได้หลากหลายเว็บบราวเซอร์ ทำให้การใช้งานนั้นสะดวกมากยิ่งขึ้น
  - สามารถกำหนดแยกไว้ต่างหากจากไฟล์เอกสาร HTML และสามารถนำมาใช้ร่วมกับเอกสารหลายไฟล์ได้ การแก้ไขก็แก้เพียงจุดเดียวแล้วส่งผลกับเอกสารทั้งหมด

2.3 เครื่องมือที่ใช้ในการพัฒนาโปรแกรม

2.3.1 Visual studio code [7]

- เป็นโปรแกรม Code Editor ที่ใช้ในการแก้ไขและปรับแต่งโค้ดจากค่ายไมโครซอฟท์ มีการพัฒนาออกมาในรูปแบบของ OpenSource จึงสามารถนำมาใช้งานได้โดยไม่เสียค่าใช้จ่าย

- เหมาะสำหรับนักพัฒนาโปรแกรมที่ต้องการใช้งานข้ามแพลตฟอร์ม รองรับการใช้งานทั้งบน Windows, macOS และ Linux สนับสนุนทั้งภาษา JavaScript, TypeScript และ Node.js สามารถเชื่อมต่อกับ Git ได้ นำมาใช้งานได้ง่ายไม่ซับซ้อน มีเครื่องมือส่วนขยายต่างๆ ให้เลือกใช้อย่างมากมาก



รูปที่ 2.12 Visual studio code

2.3.2 Robomongo [8]

- เป็น Mongo GUI ที่มีน้ำหนักเบาและฟรีสำหรับผู้ที่ชื่นชอบ MongoDB

รูปที่ 2.13 Robomongo

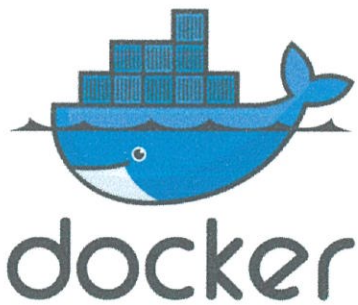
- MongoDB เป็น Open-source document database โดยเป็นฐานข้อมูลแบบ NoSQL (Not Only SQL) ไม่มี Relation ของตารางแบบ SQL ทั่วไป ซึ่งจะเก็บข้อมูลในแบบ JSON (JavaScript Object Notation) แทนการบันทึกข้อมูลทุกๆ Record ใน MongoDB เรียกว่า Document ซึ่งจะเก็บค่าเป็น key และ value

```
{
  "_id": ObjectId("554b8ee746e04bc5503aef47"),
  "name": "Chai"
}
```

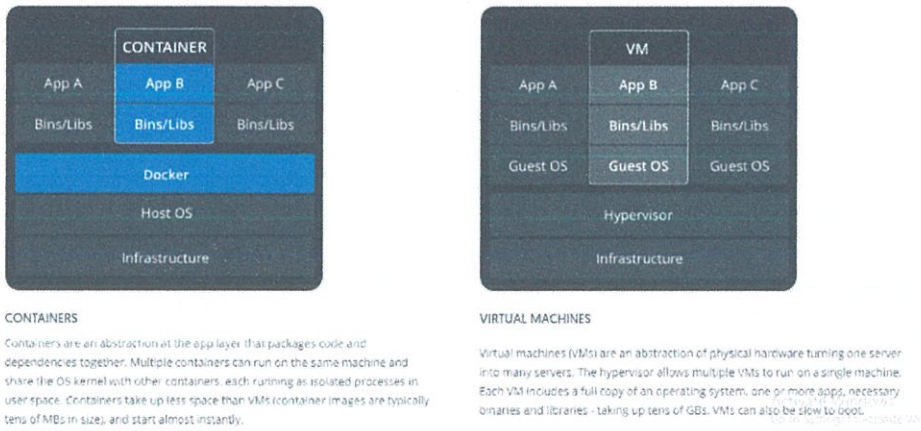
รูปที่ 2.14 ตัวอย่าง Document

2.3.3 Docker [9]

- Docker คือ Engine ตัวหนึ่งที่มีการทำงานในลักษณะจำลองสภาพแวดล้อมขึ้นมาบนเครื่อง server เพื่อใช้ในการ Run service ที่ต้องการ มีการทำงานคล้ายคลึงกับ Virtual Machine เช่น VMWare, VirtualBox, XEN และ KVM (Kernel-based Virtual Machine) แต่ข้อแตกต่างที่ชัดเจนคือ Virtual Machine เป็นการจำลองทั้งระบบปฏิบัติการ (OS : Operating system) เพื่อใช้งานและหากต้องการใช้งานเซอร์วิสใดๆ จะต้องทำการติดตั้งเพิ่มเติมบนระบบปฏิบัติการนั้นๆ ด้วย แต่สำหรับ Docker จะใช้ Container ในการจำลองสภาพแวดล้อมขึ้นมาเพื่อใช้งานสำหรับ 1 เซอร์วิสที่ต้องการใช้งานเท่านั้น โดยไม่ต้องมีส่วนของระบบปฏิบัติการเข้าไปเกี่ยวข้องเหมือน Virtual Machines



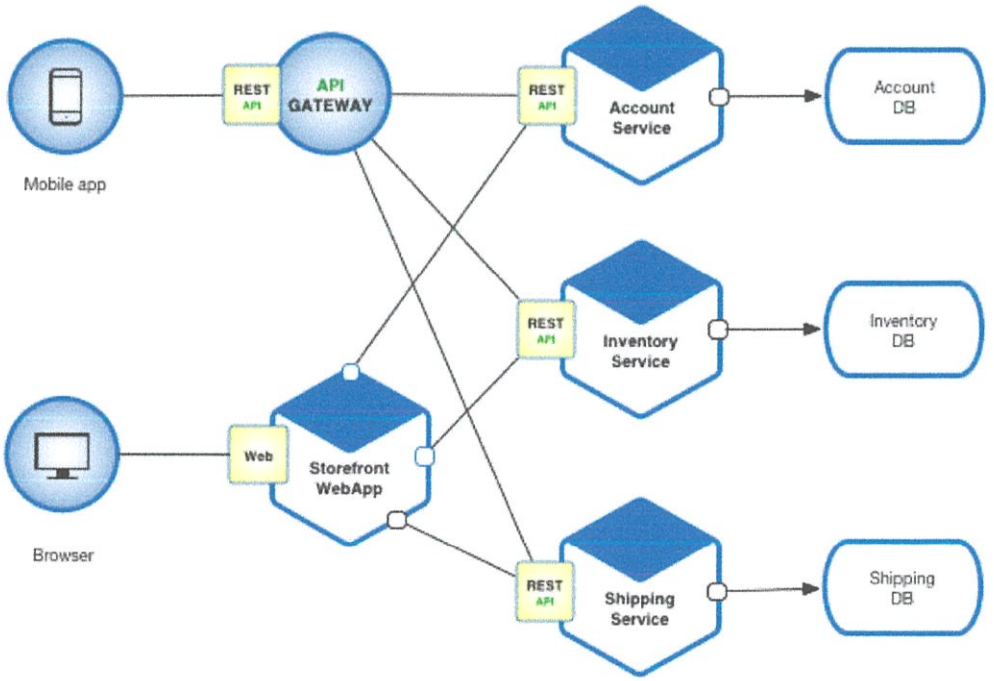
รูปที่ 2.15 Docker



รูปที่ 2.16 ข้อแตกต่างระหว่าง Container กับ VM

2.4 Microservice Architecture [10]

- สถาปัตยกรรมแบบ Microservice จะแยกพัฒนาแต่ละเซอร์วิสออกจากกันอย่างชัดเจน โดยกำหนด API ไว้ให้เรียกใช้แต่ละเซอร์วิส ซึ่งแต่ละเซอร์วิสสามารถทำงานได้อย่างเป็นอิสระ มีฐานข้อมูลเป็นของตัวเอง และหากจำเป็นต้องใช้ข้อมูลที่อยู่ในเซอร์วิสอื่นก็สามารถเรียกใช้ผ่าน API ได้

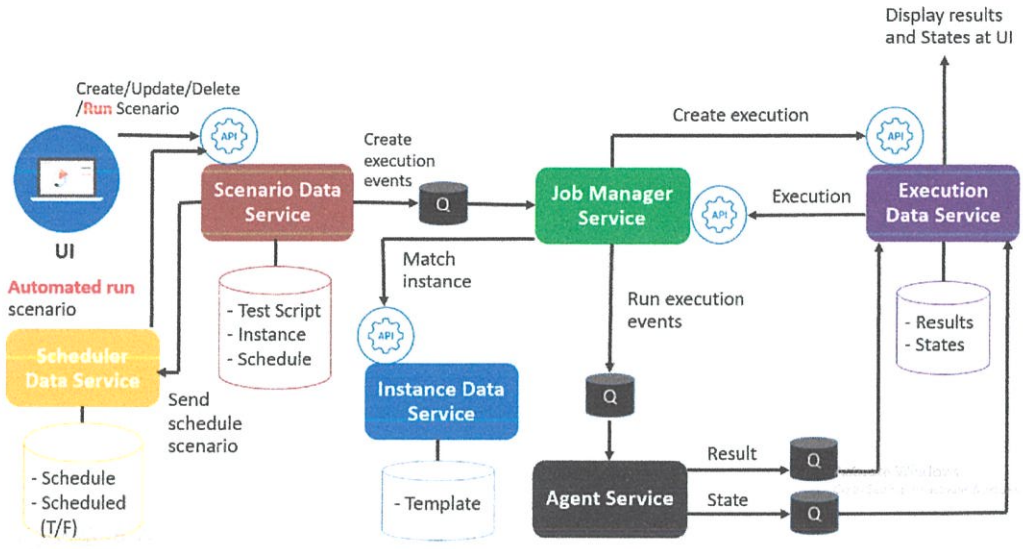


รูปที่ 2.17 Microservice

- ข้อดีของ Microservice
  - Technology independent : แต่ละเซอร์วิสมีอิสระในการเลือกใช้เทคโนโลยีที่ต่างกัน
  - Availability : หากเซอร์วิสตัวใดตัวหนึ่งไม่สามารถใช้งานได้ จะไม่ส่งผลกระทบต่อเซอร์วิสตัวอื่นๆ ทำให้ระบบยังสามารถทำงานได้อยู่
  - Release and deployment : สามารถบริหารจัดการเซอร์วิสแต่ละตัวได้ง่าย
  - Scalability : สามารถขยายเซิร์ฟเวอร์เพื่อรองรับผู้ใช้งานที่มากขึ้นโดยไม่จำเป็นต้องปรับเปลี่ยนแอปพลิเคชันทั้งหมด

- ข้อเสียของ Microservice
  - Boundary definition between services : หากทำการแบ่งเซอร์วิสตอนต้นได้ไม่ดี จะส่งผลให้ตอนหลังทำการแก้ไขได้ยากลำบาก เพราะเซอร์วิสอาจถูกเรียกใช้แล้ว การรักษา Backward compatibility จะทำให้การ Refactor โค้ดระหว่างเซอร์วิสยาก
  - Latency : การส่งข้อมูลระหว่างเซอร์วิสจะใช้เวลานานนานกว่าการส่งข้อมูลภายในโปรเซสเดียวกัน เช่น สถาปัตยกรรมแบบ Monolith
  - Consistency : ความสอดคล้องกันของข้อมูลจะรักษายากกว่าในกรณี Integrated Database
  - Infrastructure automation : การทำ Microservices ให้ได้ดี จำเป็นต้องมีการจัดการ Infrastructure ที่ดีมาก เพราะการพัฒนาเซอร์วิสจำนวนมากนั้นจะทำแบบ Manual ได้ยากมาก ส่วนใหญ่ต้องใช้การทำ Automation เกือบทั้งหมด

บทที่ 3 การทำงานของระบบ



รูปที่ 3.1 ภาพรวมการทำงานของระบบทดสอบซอฟต์แวร์แบบอัตโนมัติ

3.1 การสร้าง แก้ไข ลบ และรัน Scenario

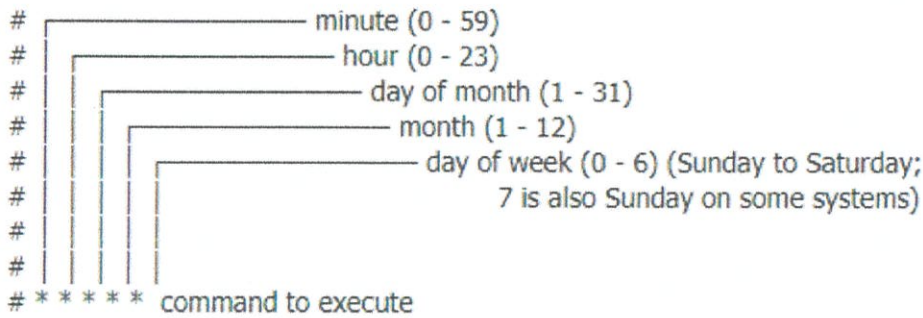
- ผู้ใช้ทำการสร้าง แก้ไข ลบ และ สั่งรัน Scenario ที่ส่วนติดต่อผู้ใช้งาน
- ส่วนติดต่อผู้ใช้ส่ง Scenario Object ไปที่ Scenario Data Service ผ่าน Scenario API
- Scenario Data Service เก็บ Scenario Object ลงใน Scenario Database


```
{
    "id" : "57e0e3ff7f256e3368cc4aaa",
    "name" : "sds-test",
    "description" : "test scenario data service",
    "testSuiteId" :"57e0e3ff7f256e3368cc4ecb",
    "testSuite" : {...},
    "configuration" : {...},
    "pendingTimeout" : 300000,
    "runningTimeout" : 300000,
    "schedule" : "*/10 * * * *",
    "scheduled" : false
}
```


รูปที่ 3.2 ตัวอย่าง Scenario Object

3.2 การทดสอบซอฟต์แวร์แบบอัตโนมัติ

- หาก Scenario Object ที่ Scenario Data Service ส่งมาให้ Scheduler Data Service ค่า scenario.scheduled = true จะทำให้ Scheduler Data Service สั่งรัน Scenario แบบอัตโนมัติผ่าน Scenario API ไปที่ Scenario Data Service ซึ่งจะสั่งรันแบบอัตโนมัติตามค่าเวลาใน scenario.schedule ที่เก็บค่าเวลาในรูปแบบ Crontab เช่น scenario.schedule = "0 0 * * 3 *" หมายถึง สั่งรัน Scenario ทุกๆ วันพุธตอนเที่ยงคืน



รูปที่ 3.3 Crontab Format

3.3 การจับคู่ Instance

- เมื่อ Job Manager Service ได้รับอีเว้นท์ให้สร้างการประมวลผล (Execution) จะทำการจับคู่ Instance ไปที่ Instance Data Service ผ่าน Instance API ซึ่ง Instance Data Service จะเก็บ Template ไว้ใน Instance Database


```
{
    "id" : "5a46094070f6f3000f11bae1",
    "hostname" : "agt-docker",
    "ip" : "172.18.0.5",
    "properties" : {
        "platform" : "linux",
        "hostname" : "agt-docker"
    },
    "executionId" : null,
    "state" : null
}
```


รูปที่ 3.4 ตัวอย่าง Template Object

3.4 การสร้าง Execution

- Job Manager Service สั่งให้ Execution Data Service สร้าง Execution ผ่าน Execution API
- Execution Data Service สร้าง Execution ที่ยังไม่มีส่งผลของการทดสอบซอฟต์แวร์ (Result) และสถานะของการประมวลผล (State) แล้วส่งไปให้ Job Manager Service ผ่าน Job Manager API

3.5 การรัน Execution

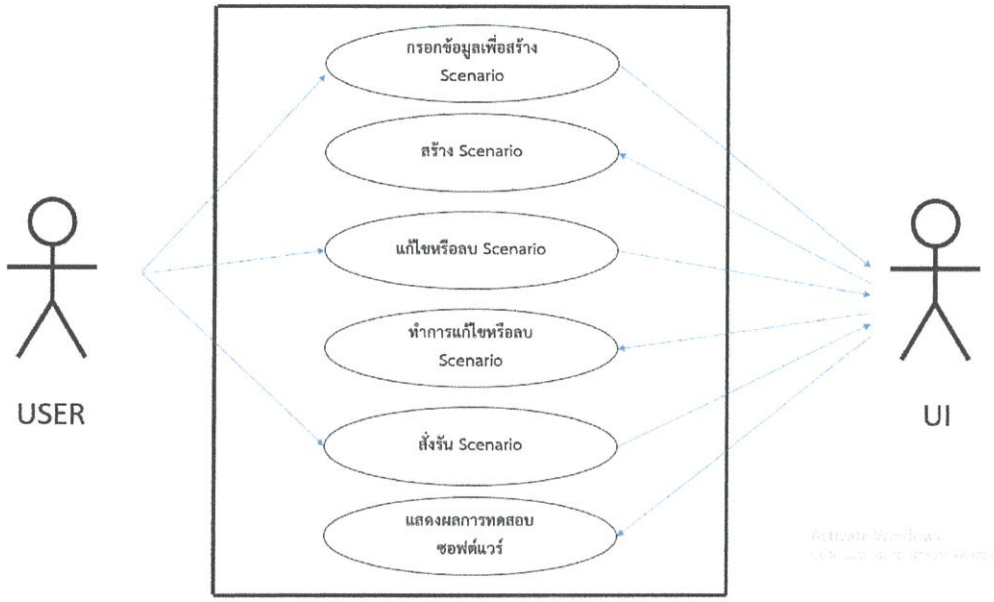
- เมื่อ Job Manager Service ทำการจับคู่ Instance และ ได้รับ Execution แล้ว จะส่งอีเว้นท์ไปให้ Agent Service รัน Execution
- เมื่อ Agent Service รัน Execution เสร็จแล้ว จะส่ง Result และ State ไปให้ Execution Data Service เก็บลงใน Execution Database


```
{
    "id" : "5a50be312081af000f09df2d",
    "scenarioId" : "5a50a47c1f2ba5000fd1e021",
    "scenarioData" : {...},
    "userId" : "5a50a47c1f2ba5000fd1e021",
    "requestTimestamp" : 1515241009725.0,
    "updateTimestamp" : 1515241010215.0,
    "completeTimestamp" : 1515241010253.0,
    "pendingTimeout" : 300000,
    "runningTimeout" : 300000,
    "pendingTimeoutScheduleId" : "5a50be31596462000fd88836",
    "result" : "ok",
    "ok" : 1,
    "partial" : 0,
    "inconclusive" : 0,
    "failed" : 0,
    "resultsCount" : 1,
    "instances" : [...],
    "commandsCount" : 1,
    "state" : "finished",
    "cancelDetails" : null }
```


รูปที่ 3.5 ตัวอย่าง Execution Object

บทที่ 4 การออกแบบเว็บแอปพลิเคชัน

4.1 Use case diagram



รูปที่ 4.1 Use case diagram แสดงภาพรวมการทำงานของระบบ

4.1.1 Actor (แอคเตอร์)

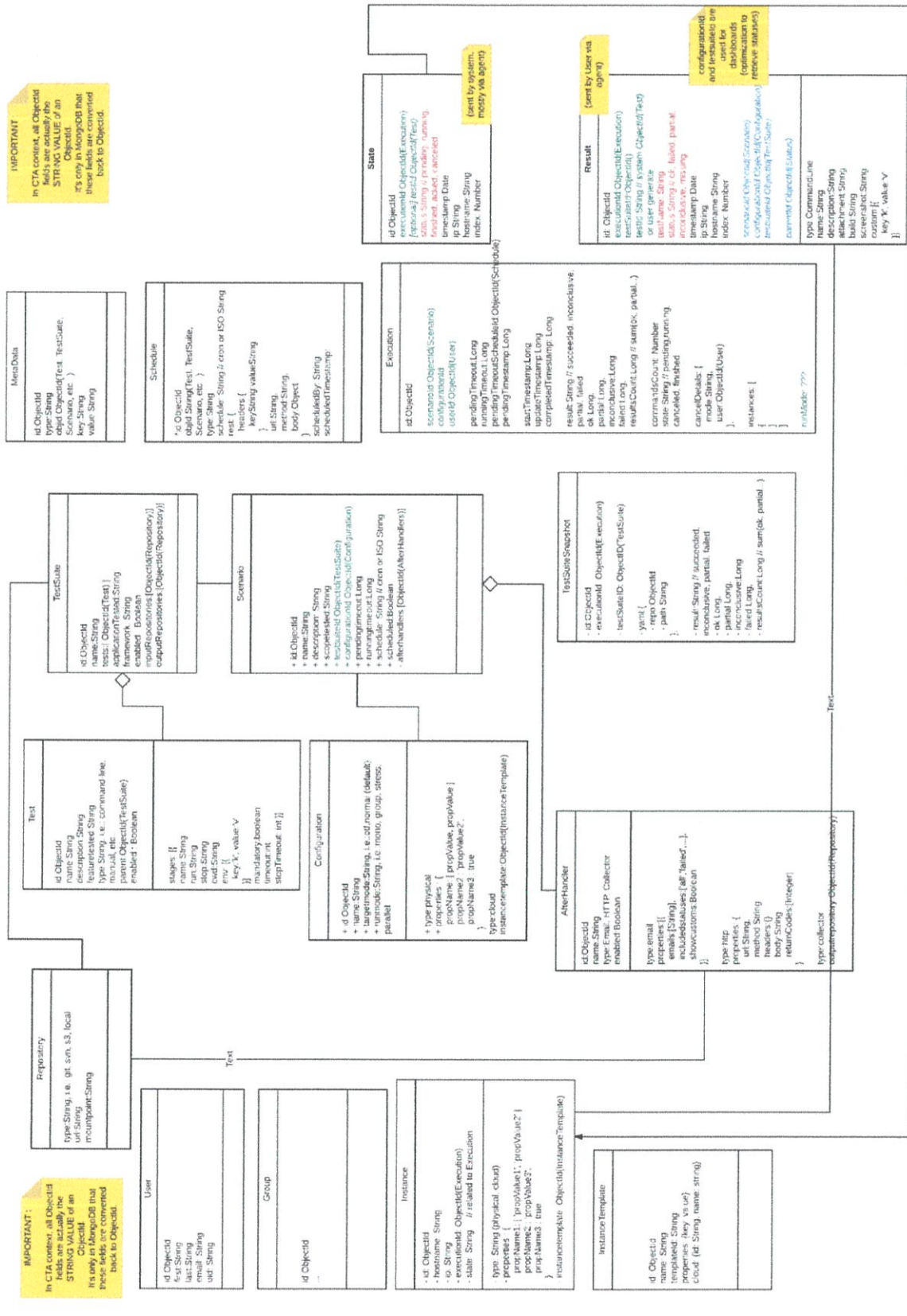
- User : ผู้ใช้งานของระบบ
- User interface : ส่วนติดต่อผู้ใช้งาน

4.1.2 Use case (ยูสเคส)

- กรอกข้อมูลเพื่อสร้าง Scenario : ผู้ใช้งานระบบทำการกรอกข้อมูลเกี่ยวกับ Scenario เพื่อร้องขอให้ส่วนติดต่อผู้ใช้งานสร้าง Scenario
- สร้าง Scenario : ส่วนติดต่อผู้ใช้งานทำการสร้าง Scenario ตามที่ผู้ใช้งานระบบได้ทำการร้องขอ

- แก้ไขหรือลบ Scenario : ผู้ใช้งานระบบร้องขอให้ส่วนติดต่อผู้ใช้งานแก้ไขหรือลบ Scenario
- ทำการแก้ไขหรือลบ Scenario : ส่วนติดต่อผู้ใช้งานทำการแก้ไขหรือลบ Scenario ตามที่ผู้ใช้งานระบบได้ทำการร้องขอ
- สั่งรัน Scenario : ผู้ใช้งานระบบทำการสั่งรัน Scenario ที่เคยสร้างขึ้นมาก่อนหน้านี้
- แสดงผลการทดสอบซอฟต์แวร์ : ส่วนติดต่อผู้ใช้งานทำการแสดงผลการทดสอบซอฟต์แวร์ตาม Scenario ที่ผู้ใช้งานระบบได้สร้างไว้

4.2 Class diagram



รูปที่ 4.2 Class diagram ของระบบ

จากรูปที่ 4.2 แสดง Class diagram ของระบบ ซึ่งประกอบด้วยเอนทิตี้ (Entity) ดังตารางต่อไปนี้

เอนทิตี้	หน้าที่
Scenario	จัดเก็บการตั้งค่าเกี่ยวกับการทดสอบซอฟต์แวร์
Configuration	จัดเก็บโหมดของการทดสอบซอฟต์แวร์และเครื่องที่ใช้ในการทดสอบซอฟต์แวร์
Instance	จัดเก็บรายละเอียดเกี่ยวกับเครื่องที่ใช้ในการทดสอบซอฟต์แวร์
TestSuite	จัดเก็บเทสสคริปท์ที่ต้องการใช้ในการทดสอบซอฟต์แวร์
Repository	จัดเก็บแหล่งที่มาของ Resource เช่น Gitlab
Afterhandler	จัดเก็บวิธีการแจ้งเตือนผลการทดสอบซอฟต์แวร์
Schedule	สั่งทดสอบซอฟต์แวร์แบบอัตโนมัติ
Execution	เก็บผลลัพธ์และสถานะการทดสอบซอฟต์แวร์
Result	เก็บผลลัพธ์การทดสอบซอฟต์แวร์
State	เก็บสถานะการทดสอบซอฟต์แวร์

ตารางที่ 4.1 ตารางแสดงหน้าที่ของแต่ละเอนทิตี้

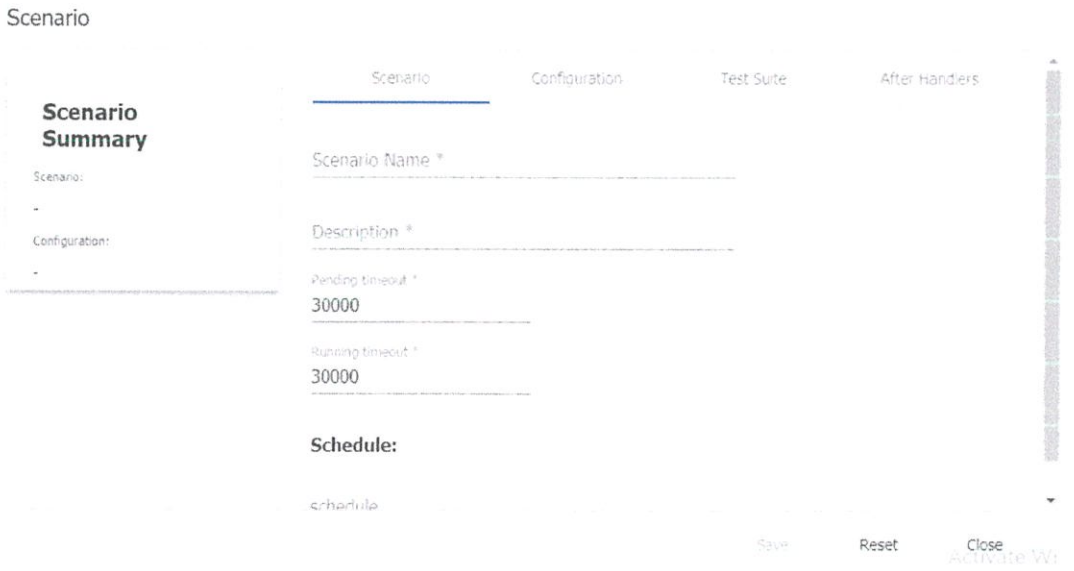
บทที่ 5 การทำงานของเว็บแอปพลิเคชัน

5.1 การสร้าง Scenario

- เมื่อเปิดเว็บแอปพลิเคชัน จะแสดงหน้าหลักคือ Scenarios

รูปที่ 5.1 หน้า Scenarios

- เมื่อผู้ใช้คลิกที่การ์ด Create a scenario จะมีไดอะล็อก (Dialog) ให้กรอกข้อมูลเพื่อสร้าง Scenario
  - เมื่อคลิกที่แท็บ Scenario จะมีอินพุท (Input) ให้กรอกข้อมูลเกี่ยวกับ Scenario

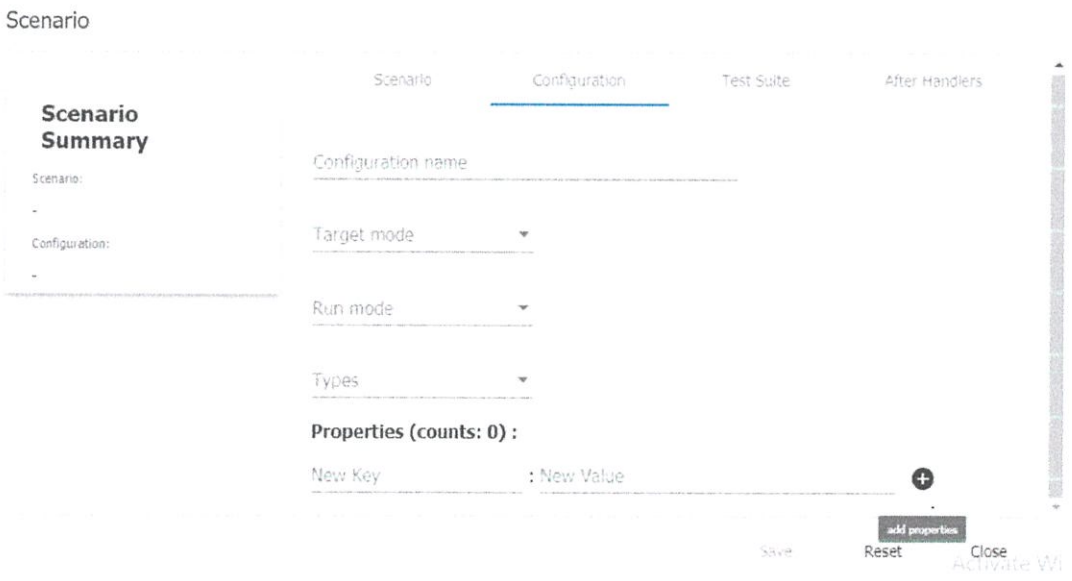


รูปที่ 5.2 Scenario Dialog – Scenario Tab

คีย์เวิร์ด	คำอธิบาย
Scenario Name	ชื่อของ Scenario
Description	รายละเอียดเกี่ยวกับ Scenario
Pending timeout	ระยะเวลาที่มากที่สุดที่สามารถรอการรัน Scenario
Running timeout	ระยะเวลาที่มากที่สุดที่สามารถทำการรัน Scenario
Schedule	หมายถึง เวลาที่จะทำการทดสอบซอฟต์แวร์แบบอัตโนมัติ

ตารางที่ 5.1 ตารางแสดงคำอธิบายสำหรับ Scenario Tab

- เมื่อคลิกที่แท็บ Configuration จะมีอินพุทให้กรอกข้อมูลเกี่ยวกับ Configuration

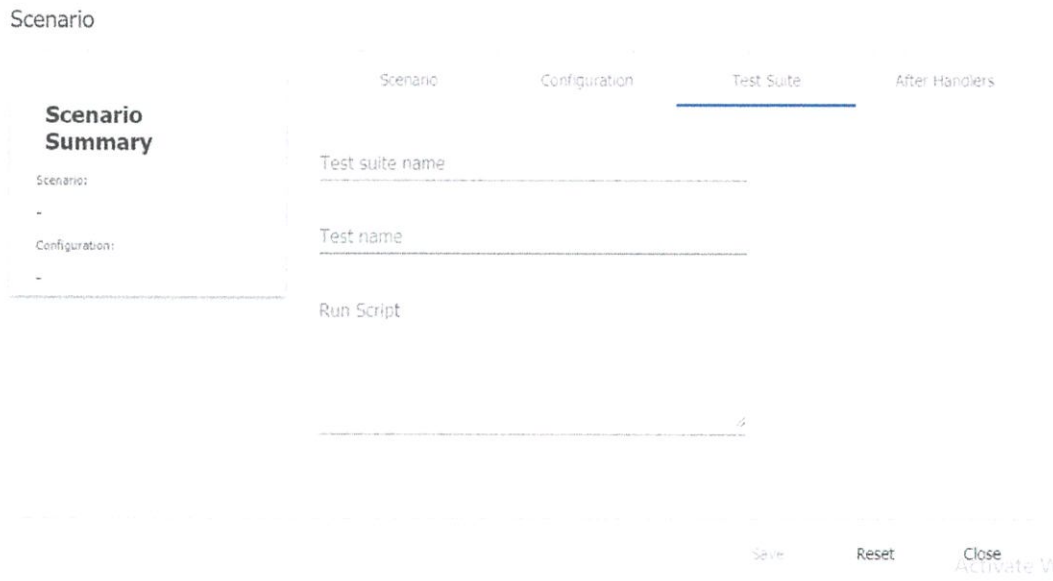


รูปที่ 5.3 Scenario Dialog – Configuration Tab

คีย์เวิร์ด	คำอธิบาย
Configuration name	ชื่อของ Configuration
Target mode	โหมดเป้าหมาย
Run mode	โหมดในการรัน
Types	ประเภทของ Configuration
Properties	คุณสมบัติของ Configuration เช่น เครื่องที่จะใช้ในการทดสอบซอฟต์แวร์ (Instance) ซึ่งผู้ใช้สามารถเพิ่ม Property โดยการคลิกที่ปุ่ม add properties

ตารางที่ 5.2 ตารางแสดงคำอธิบายสำหรับ Configuration Tab

- เมื่อคลิกที่แท็บ Test Suite จะมีอินพุทให้กรอกข้อมูลเกี่ยวกับ Test Suite

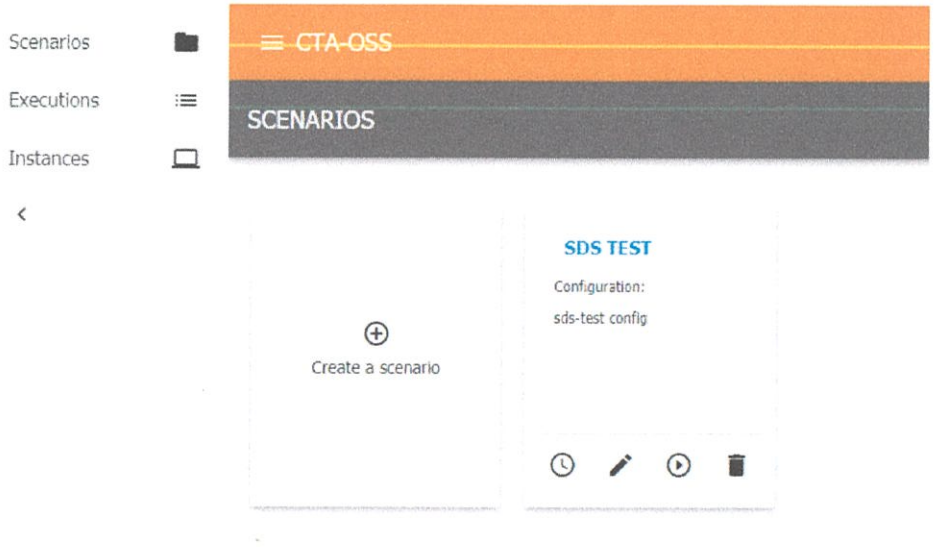


รูปที่ 5.4 Scenario Dialog – Test Suite Tab

คีย์เวิร์ด	คำอธิบาย
Test Suite name	หมายถึง ชื่อของ Test Suite
Test name	หมายถึง ชื่อของ Test
Run Script	หมายถึง สคริปท์คำสั่งสำหรับรันซอฟต์แวร์ที่ต้องการ

ตารางที่ 5.3 ตารางแสดงคำอธิบายสำหรับ Test Suite Tab

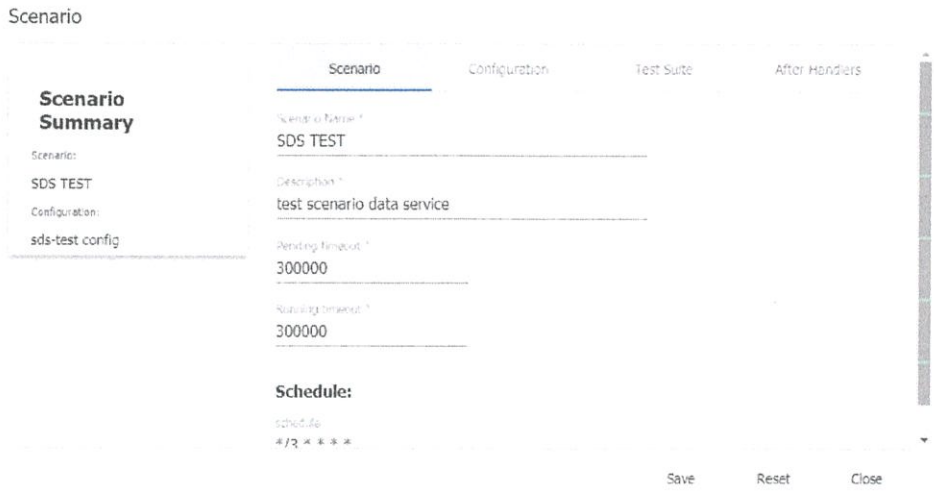
- เมื่อกรอกข้อมูลเพื่อสร้าง Scenario เสร็จเรียบร้อยแล้ว ผู้ใช้สามารถกดปุ่ม Save ด้านล่าง เพื่อทำการสร้าง Scenario จากนั้นจะเห็น Scenario card ปรากฏขึ้นมา



รูปที่ 5.5 Scenario card

5.2 การแก้ไข Scenario

- เมื่อเปิดเว็บแอปพลิเคชัน หรือกดที่เมนู Scenarios ด้านซ้ายมือ จะแสดงหน้า Scenarios
- เมื่อคลิกที่ปุ่ม Edit บน Scenario card จะมีไดอะล็อกที่เคยกรอกข้อมูลเพื่อสร้าง Scenario ไว้ ผู้ใช้สามารถแก้ไขข้อมูลได้ตามต้องการ



รูปที่ 5.6 Scenario Dialog – Scenario Tab

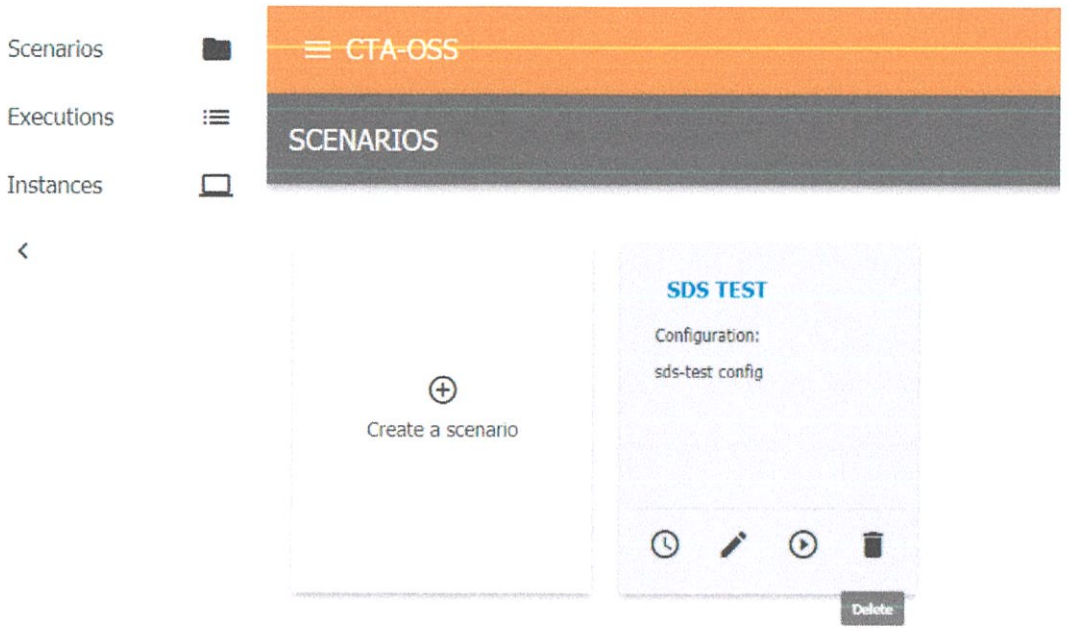
รูปที่ 5.7 Scenario Dialog – Configuration Tab

รูปที่ 5.8 Scenario Dialog – Test Suite Tab

- หากผู้ใช้แก้ไขข้อมูลไปแล้ว แต่อยากได้ข้อมูลดั่งเดิมที่เคยกรอกไปก่อนหน้านี้ สามารถกดปุ่ม Reset ด้านล่างของ Dialog เพื่อรีเซตค่า

5.3 การลบ Scenario

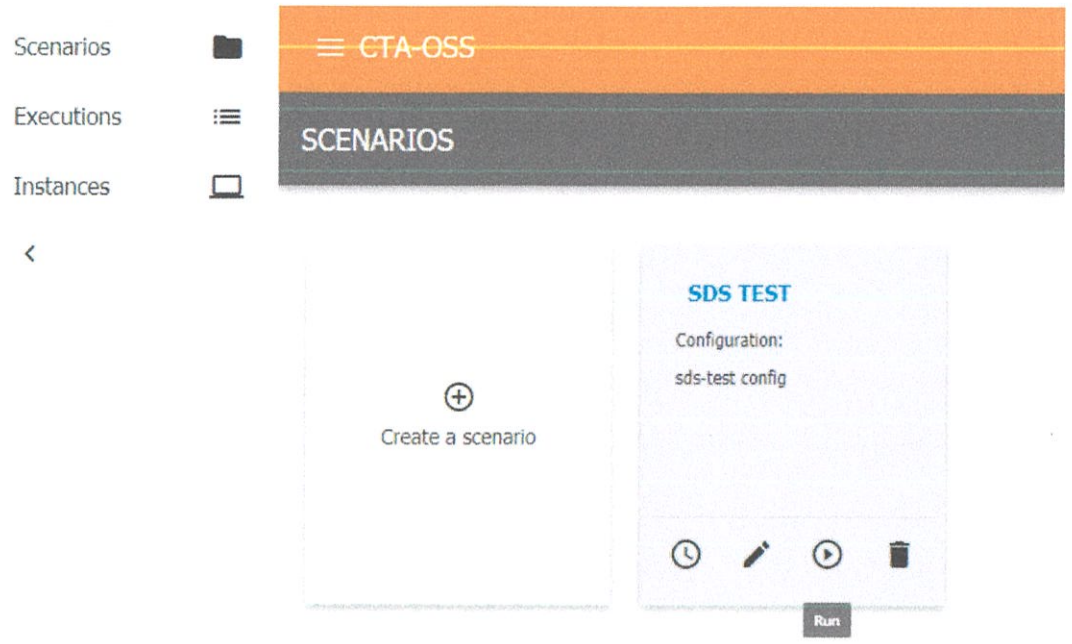
- เมื่อเปิดเว็บแอปพลิเคชัน หรือกดที่เมนู Scenarios ด้านซ้ายมือ จะแสดงหน้า Scenarios
- เมื่อผู้ใช้คลิกที่ปุ่ม Delete บน Scenario card จะสามารถลบ Scenario ได้



รูปที่ 5.9 การลบ Scenario

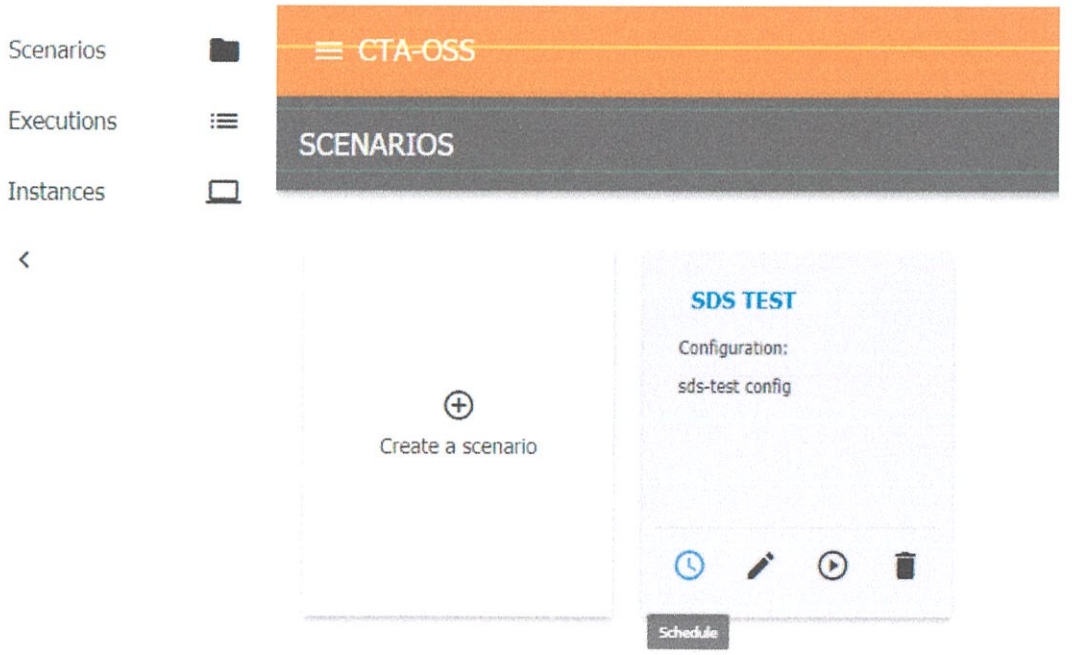
5.4 การรัน Scenario

- เมื่อเปิดเว็บแอปพลิเคชัน หรือกดที่เมนู Scenarios ด้านซ้ายมือ จะแสดงหน้า Scenarios
- เมื่อผู้ใช้คลิกที่ปุ่ม Run บน Scenario card จะสามารถรัน Scenario ได้



รูปที่ 5.10 การรัน Scenario

- หากผู้ใช้ต้องการรัน Scenario แบบอัตโนมัติ ให้ทำการกดปุ่ม Schedule บน Scenario card



รูปที่ 5.11 การรัน Scenario แบบอัตโนมัติ

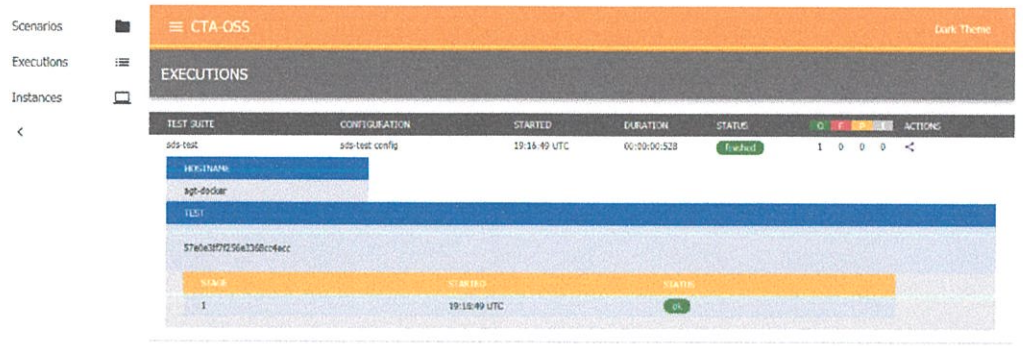
5.5 การแสดงผลลัพธ์ของการทดสอบซอฟต์แวร์

- เมื่อกดที่เมนู Executions ด้านซ้ายมือ จะแสดงหน้า Executions ซึ่งแสดงผลลัพธ์ของแต่ละ Test suite ว่าเริ่มรันเมื่อใด ใช้ระยะเวลาในการรันเท่าไหร่ สถานะของการรันเป็นอย่างไร และผลการทดสอบซอฟต์แวร์เป็นอย่างไร

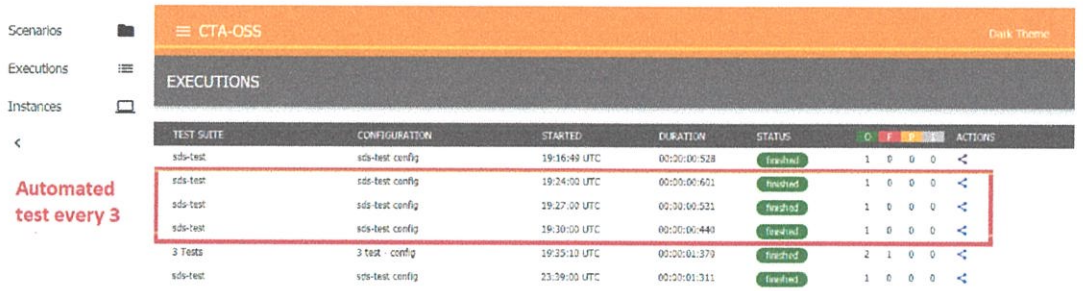
รูปที่ 5.12 หน้า Executions

- หากผู้ใช้ต้องการดูผลลัพธ์ของการทดสอบซอฟต์แวร์เพียง 1 Test Suite สามารถกดปุ่ม Share ที่ Test Suite นั้น แล้วเว็บแอปพลิเคชันจะเปิดหน้าต่างใหม่ขึ้นมาเพื่อแสดงผลลัพธ์เพียง 1 Test Suite

รูปที่ 5.13 การกดปุ่ม Share



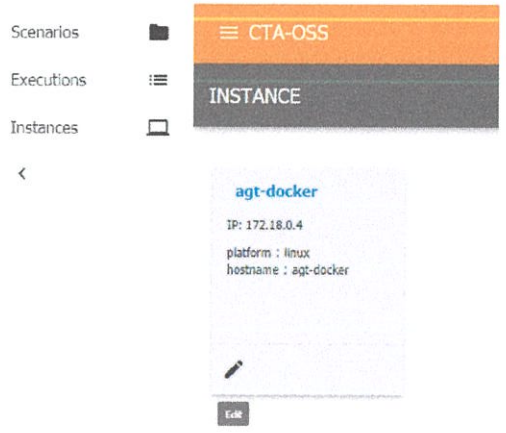
รูปที่ 5.14 การแสดงผลลัพธ์ของการทดสอบซอฟต์แวร์เพียง 1 Test Suite



รูปที่ 5.15 ตัวอย่างการแสดงผลลัพธ์ของการทดสอบซอฟต์แวร์แบบอัตโนมัติทุกๆ 3 นาที

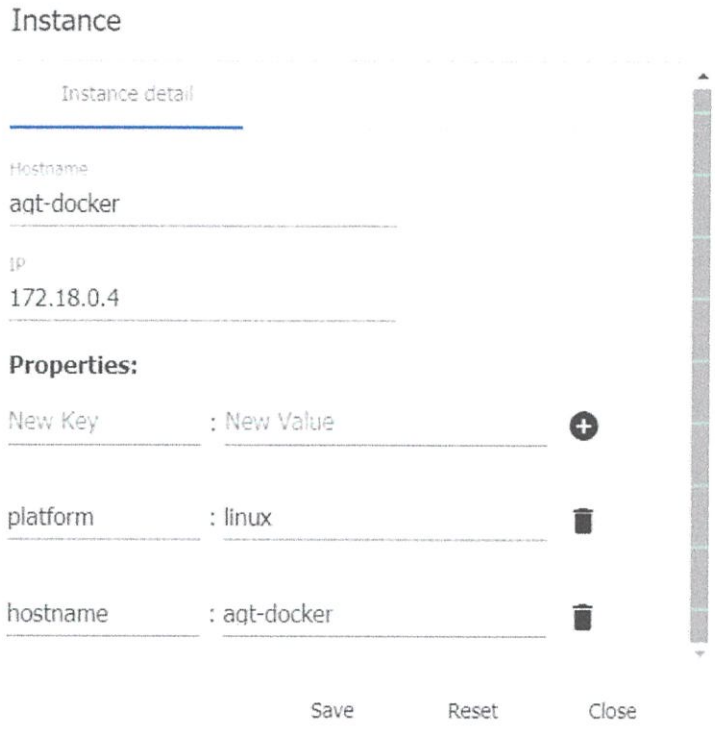
5.6 การแก้ไข เพิ่ม หรือลบคุณสมบัติของ Instance

- เมื่อกดที่เมนู Instances ด้านซ้ายมือ จะแสดงหน้า Instances ซึ่งแสดงเครื่องที่ใช้ในการทดสอบซอฟต์แวร์ตามที่ผู้ใช้ได้ระบุไว้ใน Scenario



รูปที่ 5.16 หน้า Instance

- เมื่อกดที่ปุ่ม Edit บน Instance card จะมีไดอะลอกแสดงข้อมูลเกี่ยวกับ Instance ซึ่งผู้ใช้สามารถแก้ไข เพิ่ม หรือลบคุณสมบัติ (Properties) ได้



รูปที่ 5.17 Instance dialog

- หากผู้ใช้แก้ไข เพิ่ม หรือลบ Properties ไปแล้ว แต่อยากได้ Properties ดั่งเดิม ผู้ใช้สามารถกดปุ่ม Reset ด้านล่างของ Dialog เพื่อรีเซตค่า

บทที่ 6 สรุปผลการดำเนินงาน ปัญหา และข้อเสนอแนะ

6.1 สรุปผลการดำเนินงาน

1) ผู้ใช้สามารถสร้าง Scenario ได้ โดยการกรอกข้อมูลที่ประกอบด้วย 3 ส่วน คือ Scenario, Configuration และ Test Suite

2) ผู้ใช้สามารถแก้ไขหรือรีเซ็ตค่าใน Scenario ที่เคยสร้างขึ้นมา

3) ผู้ใช้สามารถลบ Scenario ทิ้ง หากไม่ต้องการใช้แล้ว

4) ผู้ใช้สามารถสั่งรัน Scenario ด้วยตัวเองหรือสั่งรันแบบอัตโนมัติ

5) ผู้ใช้สามารถติดตามผลลัพธ์ของการทดสอบซอฟต์แวร์ ที่ประกอบด้วย ชื่อของ Test Suite ชื่อของ Configuration เวลาที่เริ่มทดสอบซอฟต์แวร์ (Started) ระยะเวลาที่ใช้ในการทดสอบซอฟต์แวร์ (Duration) สถานะทดสอบซอฟต์แวร์ (Status) ผลลัพธ์การทดสอบซอฟต์แวร์ และจำนวนผลลัพธ์ที่ทดสอบผ่านและไม่ผ่าน

6) ผู้ใช้สามารถแก้ไข เพิ่ม หรือลบคุณสมบัติของ Instance

6.2 ปัญหาและอุปสรรคที่พบ

1) ระบบ CTA มีความซับซ้อนค่อนข้างมาก และทางทีมพัฒนาระบบยังไม่มีเอกสารที่สมบูรณ์ให้ศึกษาระบบ อีกทั้งเอกสารบางส่วนมีข้อผิดพลาดที่ยังไม่ได้รับการแก้ไขปรับปรุง ทำให้ต้องใช้เวลาทำความเข้าใจเป็นเวลานาน จึงจะเข้าใจและสามารถอธิบายการทำงานของระบบทั้งหมดได้

2) เมื่อเว็บแอปพลิเคชันติดต่อกับเซอร์วิสมักจะเกิดปัญหาเกี่ยวกับ Access-Control-Allow-Origin ทำให้ต้องดาวน์โหลดปลั๊กอิน (Plugin) ที่มีชื่อว่า Allow-Control-Allow-Origin: * มาแก้ไขปัญหาดังกล่าว โดยปลั๊กอินตัวนี้จะสามารถใช้กับเบราว์เซอร์ของ Chrome เท่านั้น หากต้องการใช้เบราว์เซอร์อื่นในการเปิดเว็บแอปพลิเคชัน ต้องดาวน์โหลดปลั๊กอินตัวอื่นมาใช้งาน

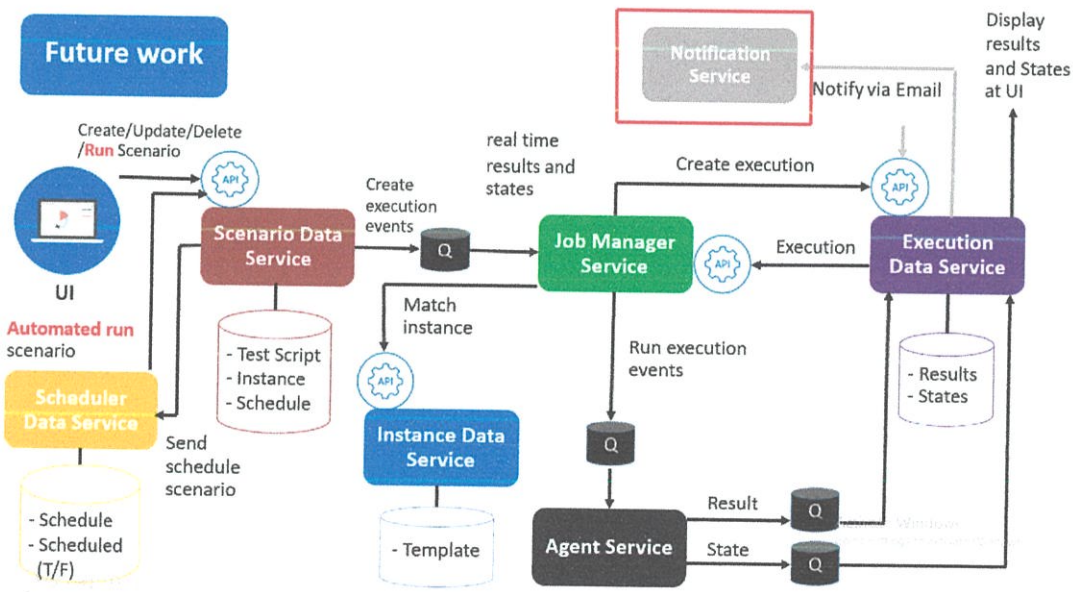
รูปที่ 6.1 ปลั๊กอิน Allow-Control-Allow-Origin: *

3) เซอร์วิสหลายตัวในระบบมีปัญหาไม่อนุญาตให้ใช้ HTTP Method บางตัว เช่น PATCH ทำให้ต้องรอการแก้ไขปรับปรุงเซอร์วิสจากคนในทีม ส่งผลให้โครงงานดำเนินได้ช้าขึ้น เพราะปัญหาดังกล่าวทำให้งานดำเนินต่อไปไม่ได้

6.3 ข้อเสนอแนะ

1) การแจ้งเตือนผลลัพธ์ของการทดสอบซอฟต์แวร์

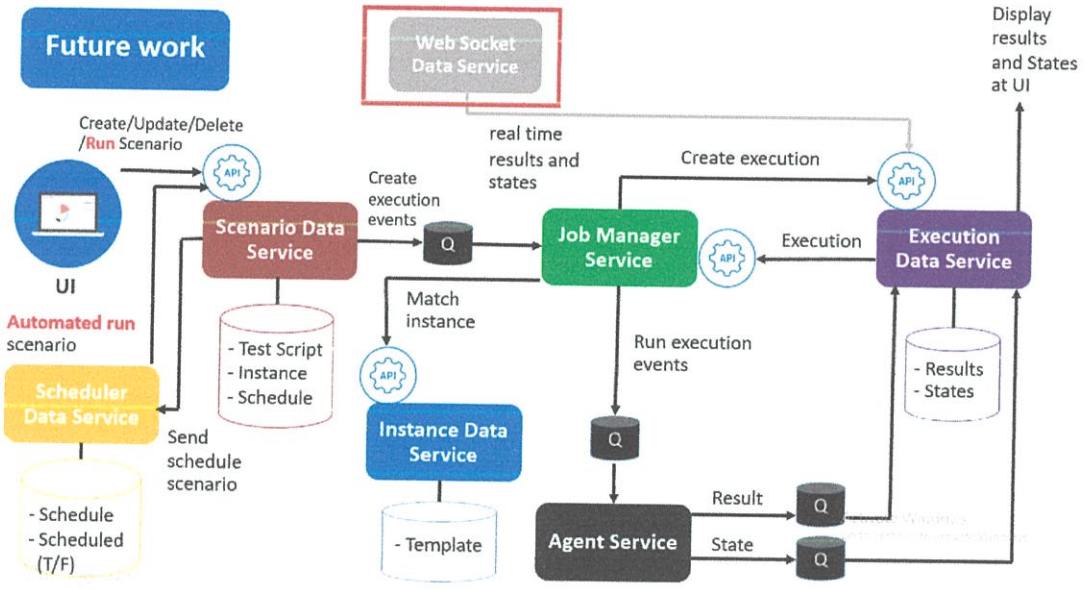
ควรเพิ่ม Notification Service ซึ่งทำหน้าที่แจ้งเตือนผลลัพธ์ไปให้ผู้ใช้แบบอัตโนมัติผ่านทางอีเมล เพื่อทำให้ผู้ใช้ไม่จำเป็นต้องคอยติดตามผลลัพธ์ที่ส่วนติดต่อผู้ใช้เพียงอย่างเดียว



รูปที่ 6.2 การแจ้งเตือนผลลัพธ์ของการทดสอบซอฟต์แวร์

2) การแสดงผลลัพธ์ของการทดสอบซอฟต์แวร์แบบเรียลไทม์

ควรเพิ่ม Web Socket Data Service ซึ่งทำหน้าที่แสดงผลการทดสอบซอฟต์แวร์และสถานะของการประมวลผลแบบเรียลไทม์ เพื่อทำให้ผู้ใช้ไม่จำเป็นต้องรีเฟรชส่วนติดต่อผู้ใช้ เมื่อต้องการดูผลของการทดสอบซอฟต์แวร์และสถานะของการประมวลผลล่าสุด



รูปที่ 6.3 การแสดงผลลัพธ์ของการทดสอบซอฟต์แวร์แบบเรียลไทม์

เอกสารอ้างอิง

[1] "Angular Doc"

https://angular.io/docs, [ออนไลน์]

[2] "Angular Feature"

https://angular.io/features, [ออนไลน์]

[3] "Angular Architecture "

https://angular.io/guide/architecture, [ออนไลน์]

[4] "TypeScript"

https://www.babelcoder.com/blog/posts/typescript-data-types, [ออนไลน์]

[5] "HTML"

http://www.kts.ac.th/kts/kanghan/html/unit202.htm, [ออนไลน์]

[6] "CSS"

http://www.seo-winner.com/CSS_What, [ออนไลน์]

[7] "Visual studio code"

http://www.mindphp.com/%E0%B8%9A%E0%B8%97%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B8%A1/microsoft/4829-visual-studio-code.html, [ออนไลน์]

[8] "Robomongo"

https://devahoy.com/posts/getting-started-with-mongodb/#step4, [ออนไลน์]

[9] "Docker"

https://www.hostpacific.com/using-docker-on-centos7/, [ออนไลน์]

[10] "Microservice"

https://www.techtalkthai.com/introduction-to-microservices-architecture/, [ออนไลน์]

ประวัติผู้เขียน

ชื่อ	นางสาวณัฐกานต์ จึงไพศาลทรัพย์
ชื่อเล่น	อิ๋งอิ๋ง
วัน เดือน ปีเกิด	9 กรกฎาคม 2538
ประวัติการศึกษา	1. ประถมศึกษาชั้นปีที่ 1-6 โรงเรียนอนุบาลสุรินทร์ จังหวัดสุรินทร์
	2. มัธยมศึกษาชั้นปีที่ 1-6 โรงเรียนสิรินธร จังหวัดสุรินทร์
	3. ปริญญาตรีวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ประวัติการทำงาน	ฝึกงานภาคฤดูร้อนและฝึกงานสหกิจศึกษา
	บริษัทรอยเตอร์ ซอฟต์แวร์ (ประเทศไทย) จำกัด
	(1 มิถุนายน - 31 ธันวาคม 2560)