Test Plan

1.Introduction

parking fee เป็นโปรแกรมที่ทำหน้าที่ในการคำนวณเวลาและค่าใช้บริการของการเข้าใช้งานที่จอด ได้แก่ การนำเวลาที่เข้าใช้บริการไปเช็คกับเวลาที่ใช้บริการเสร็จสิ้น ฟังก์ชั่นเหล่านี้จะถูกทดสอบด้วยชุดกรณีทดสอบในรายงานฉบับนี้ ส่วนรายละเอียดที่ของข้อมูลที่จะใช้ทดสอบ

2.Test Item

- วัน (Day)
- เดือน (Month)
- ปี (Year)
- เวลา (Time)

3.Feature ที่ต้องทดสอบ

Functionality

4. Feature ที่ไม่ต้องทดสอบ

Non - Functionality ได้แแก่ performance , Security , Usability เพราะโปรแกรมยังไม่เป็นระบบที่สมบูรณ์และยังไม่มีส่วนเก็บรักษาข้อมูลถาวร

5.วิธีการและแนวทางในการทดสอบ

สร้างชุดกรณีทดสอบที่ครอบคลุมตามกฎเกณฑ์ที่ได้กำหนดไว้ใช้ JUnit ช่วยในการทดสอบ โดยทดสอบด้วยการแยกฟังก์ชั่นของการทำงาน

6.กฎเกณฑ์ที่ถือว่าผ่าน และไม่ผ่านการทดสอบ

ถ้าผลลัพธ์ที่ได้จากกรณีทดสอบทุกกรณีตรงตามผลลัพธ์ที่คาดหวังถือว่าผ่านการทดสอบ กรณีที่ไม่ผ่านการทดสอบ เป็นเพราะมีบางกรณีที่ไม่ได้ผลลัพธ์ตามที่ต้องการ

7.สิ่งที่ต้องส่งมอบ

- โปรแกรมในรูปแบบ java
- Code ทั้งหมดของโปรแกรมนี้
- รายงานฉบับนี้ ซึ่งรวบรวมการวางแผน การออกแบบ การทดสอบ และสรุปผลการทดสอบ

8.งานที่ต้องทำ

- ออกแบบชุดกรณีทดสอบ
- สร้างโค้ดที่ใช้ในการทดสอบ
- ทำการทดสอบ
- บันทึกและสรุปผลการทดสอบ

9.สิ่งแวดล้อมที่ต้องการ

- Software
    - Eclipse for Java
    - page
- Hardware
    - Laptop

10.ผู้ที่รับผิดชอบ

- ในการทำรายงาน
    - สหวัชร รอดกลาง
- ในการออกแบบชุดกรณีทดสอบ
    - กิตติศักดิ์ ศรีเดช
- ในการทดสอบและบันทึกผลการทดสอบ
    - ธนพล โสมนะพันธุ์
    - ธนานนท์ คำวัน

11.ความเสี่ยงและผลที่ตามมา

- ความเสี่ยง : User กรอกข้อมูลไม่ตรงตามเกณฑ์ที่ระบุไว้
- ผลที่ตามมา : เกิด Error ขึ้น
- แนวทางในการป้องกัน : แสดงข้อความชี้แจ้งเกี่ยวกับลักษณะของข้อมูล

12.การอนุมัติ

รองศาสตราจารย์ สุชาดา รัตนคงเนตร ผู้สอนวิชา Software Testing รหัสวิชา 040613348 ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

Test Case Specification

1.Feature และ Test Item ที่ต้องการทดสอด

การทำงานของฟังก์ชั่นทำการตรวจสอบเวลาเข้าและเวลาออก เพื่อทำการทดสอบว่าเมื่อกำหนดอินพุตลักษณะนี้ จะได้ผลลัพธ์ตามที่ต้องการหรือไม่

2.ลักษณะอินพุต

ตัวเลข(เวลาเข้า และ เวลาออก)

3.ลักษณะเอาท์พุต

ข้อความแสดงผลลัพท์ต่างๆที่เป็นได้ทั้ง Valid และ Invalid

4.สิ่งแวดล้อมที่ต้องการ

โปรแกรม Eclipse for Java

5.กระบวนการพิเศษที่ต้องการ

ไม่มี

6.Test Case ที่ต้องการทดสอบ

ฟังก์ชั่น Kidtang

Tc1:เมื่อรับค่า

เวลาเข้า"13/03/14 09:00:00";

เวลาออก "13/03/14 10:00:00"

และจำนวนที่ต้องจ่ายคือ 20 บาท

Tc2:เมื่อรับค่า

เวลาเข้า"13/03/14 09:00:00";

เวลาออก "13/03/14 10:20:00"

และจำนวนที่ต้องจ่ายคือ 20 บาท

Tc3:เมื่อรับค่า

เวลาเข้า"13/03/14 09:00:00";

เวลาออก "13/03/14 10:31:00"

และจำนวนที่ต้องจ่ายคือ 40 บาท

Tc4:เมื่อรับค่า

เวลาเข้า"13/03/14 09:00:00";

เวลาออก "13/03/14 10:29:59"

และจำนวนที่ต้องจ่ายคือ 20 บาท

Tc5:เมื่อรับค่า

เวลาเข้า"13/03/14 09:00:00";

เวลาออก "13/03/14 09:29:59"

และจำนวนที่ต้องจ่ายคือ 0 บาท

Tc6:เมื่อรับค่า

เวลาเข้า"13/03/14 09:00:00";

เวลาออก "13/03/15 08:29:59"

และจำนวนที่ต้องจ่ายคือ 260 บาท

Tc7:เมื่อรับค่า

เวลาเข้า"วันนี้ตอนเที่ยง";

เวลาออก "13/03/15 08:29:59"

Tc8:เมื่อรับค่า

เวลาเข้า"08:29:59";

เวลาออก "08:29:59"

Test Log

1.คำอธิบาย

ระบบ : ระบบทำการตรวจสอบค่าลานจอดรถ (Kidtang)

สิ่งแวดล้อม : (Eclipse For Java,JUnit,Laptop)

2.รายการของกิจกรรม

ในการทดสอบ จะทดสอบการ Validate Input ที่มาผู้ใช้เป็นหลัก โดยการทดสอบหากว่าผู้ใช้นำอินพุตลักษณะต่างๆ เข้ามาระบบจะ Validate ตามเกณฑ์ที่กำหนดให้ได้อย่างถูกต้องหรือไม่ ในการทดสอบจะเขียนฟังก์ชันเสมือนในการรับ อินพุตเข้ามานำอินพุตผ่านฟังก์ชัน Validate ที่ใช้งานได้จริงและแสดงเอาท์พุตเป็น String เพื่อให้ทดสอบได้โดยง่าย

Output ที่เกิดจากการทดสอบ จะแสดง 3 ค่านี้

- “Valid” ที่เกิดขึ้นในกรณีที่ input ทุกตัวผ่านเกณฑ์หมด
- “Invalid” ที่เกิดขึ้นในกรณีที่ input ใส่เป็นตัวอักษร ,ใส่เวลาก่อนวันที่

เมื่อทดสอบตามกรณีขั้นตอนข้างต้น ปรากฏว่า ระบบสามารถ validate input ได้ทุกกรณีตามที่กำหนดไว้โดยมีการบันทึกดังนี้

ทดสอบฟังก์ชั่น Kidtang

- TC1 ผลการทดสอบ ผ่าน
- TC2 ผลการทดสอบ ผ่าน
- TC3 ผลการทดสอบ ผ่าน
- TC4 ผลการทดสอบ ผ่าน
- TC5 ผลการทดสอบ ผ่าน
- TC6 ผลการทดสอบ ผ่าน
- TC7 ผลการทดสอบ ไม่ผ่าน
- TC8 ผลการทดสอบ ไม่ผ่าน

Test Summary Report

1.ข้อสรุป

ระบบสามารถประมวลผลกรณีทดสอบได้ตามผลลัพธ์ที่คาดหวังไว้

2.ความหลากหลาย

การทดสอบไม่มีความแตกต่างระหว่างผลลัพธ์ที่คาดหวังกับผลลัพธ์ที่ได้ เพราะเมื่อมีการรันผลการทดสอบ ผลที่ได้คือผ่านทุกกรณี เพราะฉะนั้นจึงไม่มีความหลากหลาย

3.ความครอบคลุม

การทดสอบมีความครอบคลุม เนื่องจาก input ที่เข้ามานั้นไม่ได้มีความหลากหลายมากนั้น จึงทำให้การทดสอบนี้มีความครอบคลุมทุกกรณี

4.สรุปผลการทดสอบ

- สามารถทดสอบการ Validate ในฟังก์ชั่น Kidtung ได้ทุกกรณีและผ่านทุกกรณี
- สามารถทดสอบการ Validate ในฟังก์ชันได้ทุกกรณีและผ่านทุกกรณี

5. ประเมินแต่ละ Item

ทุก Item สามารถใช้งานได้จริงตามฟังก์ชันที่กำหนด

6.สรุปกิจกรรมการทดสอบ

การทดสอบนี้เป็นการทดสอบ validate แบบฟอร์มในระบบ ซึ่งทดสอบโดยฟังก์ชั่นเสมือนเพื่อให้ทดสอบได้โดยง่าย ปรากฏว่าระบบสามารถประมวลผลทุกกรณีทดสอบได้ผลลัพธ์ตามที่คาดหวังไว้

Use Case Testing

ฟังก์ชั่น Kidtang

Use Case component	Description
Use Case ID	SR01
Use Case Name	Kidtang
Goal in Context	คำนวณราคาที่จอดรถ
Level	Primary task
Primary Actor	ผู้ใช้
Preconditions	-
Success End Condition	สามารถบอกราคาที่ต้องชำระ
Trigger	-
Main Success Scenario	Step1 A: ใส่ปีเดือนวัน ชั่วโมงนาทีวินาทีที่รถเข้า Step2 A: ใส่ปีเดือนวัน ชั่วโมงนาทีวินาทีที่รถออก Step3 S:คำนวณจำนวนชั่วโมงที่รถอยู่ในระบบ Step4 S:คำนวณราคาที่ต้องชำระ Step5 S:แสดงผลลัพธ์
Extensions	1a ผู้ใช้ใส่ข้อมูลไม่ครบ
	2a ผู้ใช้กรอกข้อมูลเป็นตัวอักษร
	3a ผู้ใช้กรอกข้อมูลผิดรูปแบบ

Sub-Variations	-
Priority	Criticality
Response Time	30 วินาที
Frequency	ไม่จำกัด
Channel to Primary Actor	N/A
Secondary Actor	ไม่มี
Channel to Secconfdary	N/A
Data Due	29 April
Completeness level	0.5
Open Issue	ไม่มี

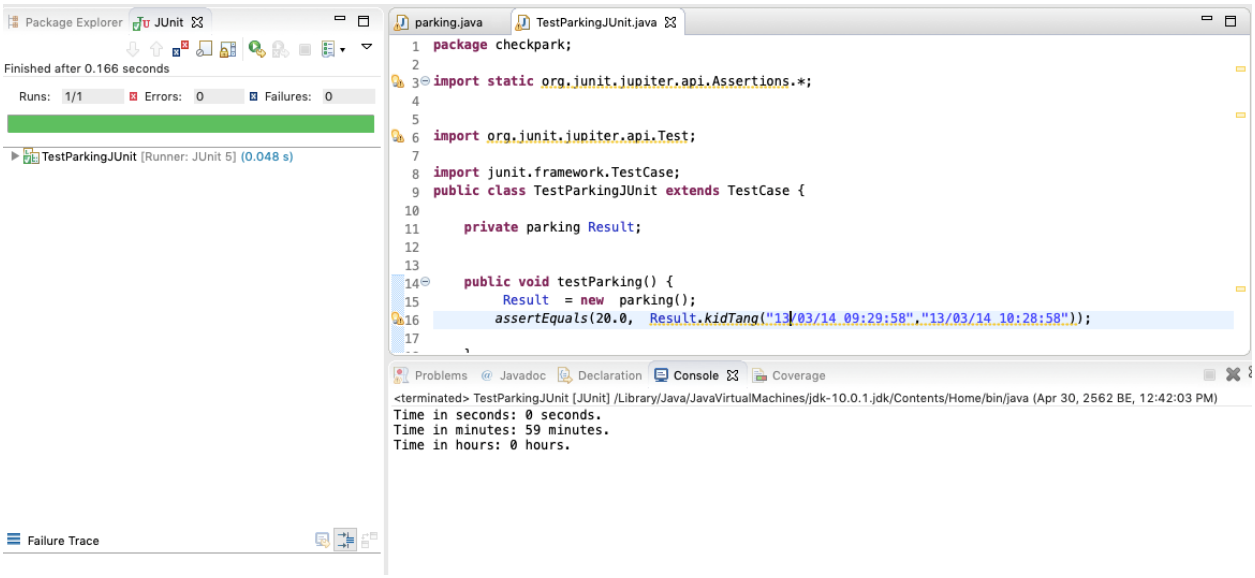
Test Case

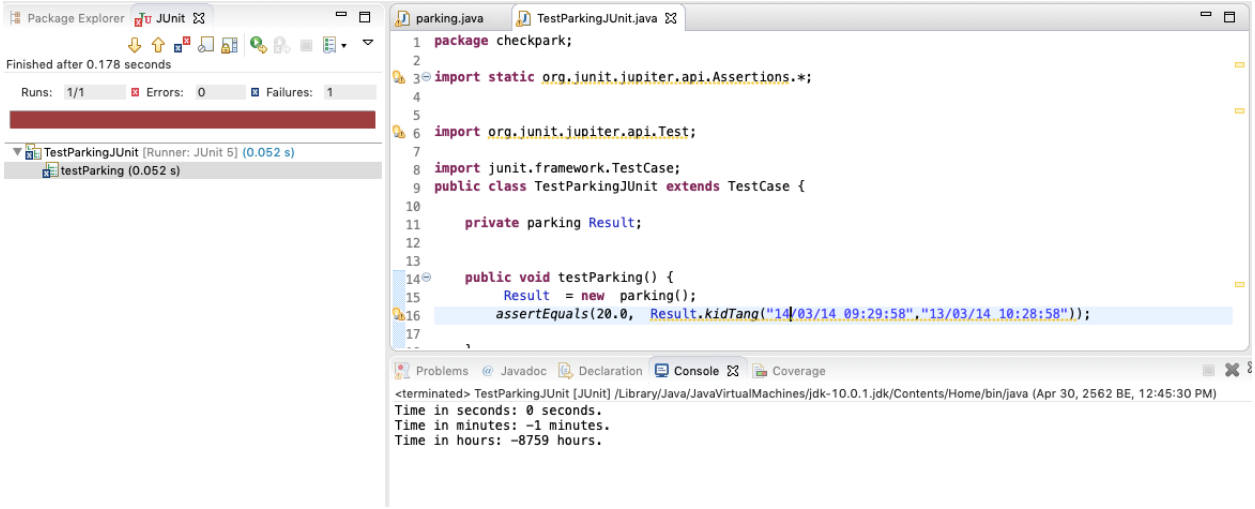
ฟังก์ชั่น Kidtang

TEST CASE	INPUT		OUTPUT
	Time In	**Time Out**	**Price**
1	13/03/14 09:00:00	13/03/14 09:29:00	0
2	13/03/14 09:00:00	13/03/14 09:30:00	20
3	13/03/14 09:00:00	13/03/14 09:31:00	20
4	13/03/14 09:00:00	13/03/14 10:30:00	40
5	13/03/14 09:00:00	13/03/14 12:29:59	60
6	13/03/14 09:00:00	13/03/14 23:00:00	170
7	13/03/14 09:00:00	13/03/14 09:00:00	0
8	13/03/14 09:00:00	13/03/13 09:00:00	Error
9	ตอนเที่ยง	ตอนเย็น	Error
10	09:00:00	10:00:00	Error

JUnit Preview

ฟังก์ชั่น TestParkingJUnit





Source Code

class parking

```
package checkpark;

import java.math.RoundingMode;
import java.text.DateFormat;
import java.text.NumberFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class parking {
	public static void main(String[] args)   {
		String dateStart = "11/03/14 09:29:58";
		String dateStop = "11/03/14 13:33:43";
		System.out.println("You will have to pay : " + kidTang(dateStart, dateStop) + " Baht");
	}
	public static double kidTang(String dateStart, String dateStop){
		SimpleDateFormat format = new SimpleDateFormat("yy/MM/dd HH:mm:ss");

		Date d1 = null;
		Date d2 = null;
		try {
			d1 = format.parse(dateStart);
			d2 = format.parse(dateStop);
		}catch (ParseException e) {
			e.printStackTrace();
		}
		long diff = d2.getTime() - d1.getTime();
		long diffSeconds = diff / 1000 % 60;
		long diffMinutes = diff / (60 * 1000) % 60;
		long diffHours = diff / (60 * 60 * 1000);
		System.out.println("Time in seconds: " + diffSeconds + " seconds.");
		System.out.println("Time in minutes: " + diffMinutes + " minutes.");
		System.out.println("Time in hours: " + diffHours + " hours.");
		if (diffMinutes>30){
			diffHours += 1;
		}
		long hour = diffHours;
		double sum = 0;
		sum = sum + (hour/24)*500;
		hour = hour%24;
		if(hour <= 3) sum = sum + hour*20;
		else sum = sum + 3*20 + (hour-3)*10;
		return sum;
```

```
        }
}
```

class TestParkingJunit

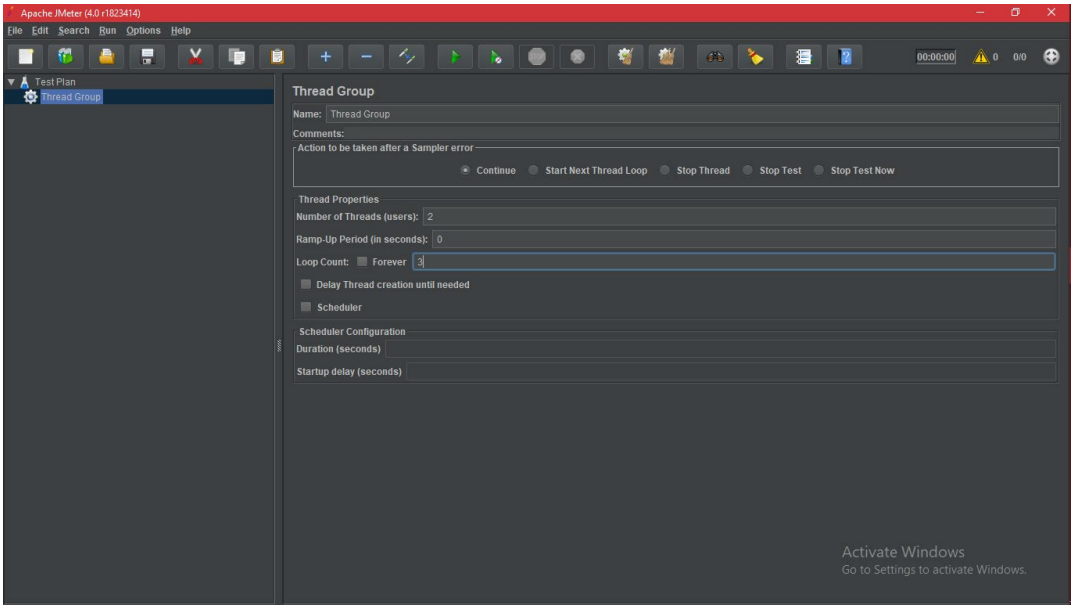
```
package checkpark;

import static org.junit.jupiter.api.Assertions.*;


import org.junit.jupiter.api.Test;

import junit.framework.TestCase;
public class TestParkingJUnit extends TestCase {

   private parking Result;


   public void testParking() {
      Result  = new  parking();
      assertEquals(20.0,  Result.kidTang("14/03/14 09:29:58","13/03/14 10:28:58"));

   }
}
```

Non-Function

โปรแกรม Apache JMeter

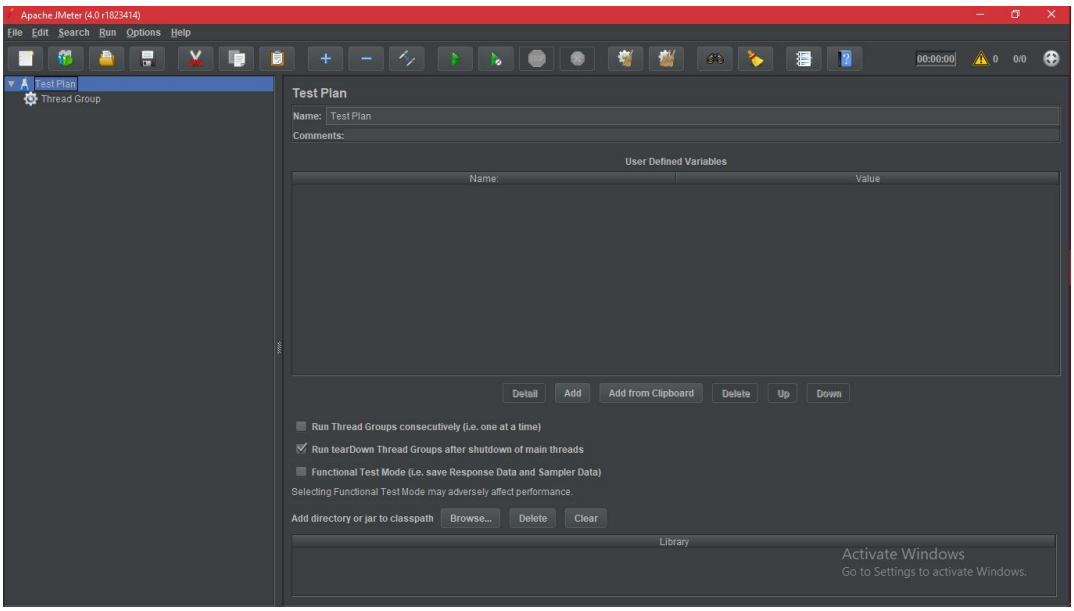
ความสามารถ

Apache JMeter เป็น Java Application Desktop เป็น Open Source ถูกพัฒนามาเพื่อใช้สำหรับการLoad Test เพื่อวัด Performance ตัว JMeter สามารถจำลอง การ load ในแต่ละ scenarios ต่างๆ ได้ และสามารถ output ค่าต่างๆ ออกมาอยู่ใน รูปแบบ ของ Graph, CSV หรือ XML ได้



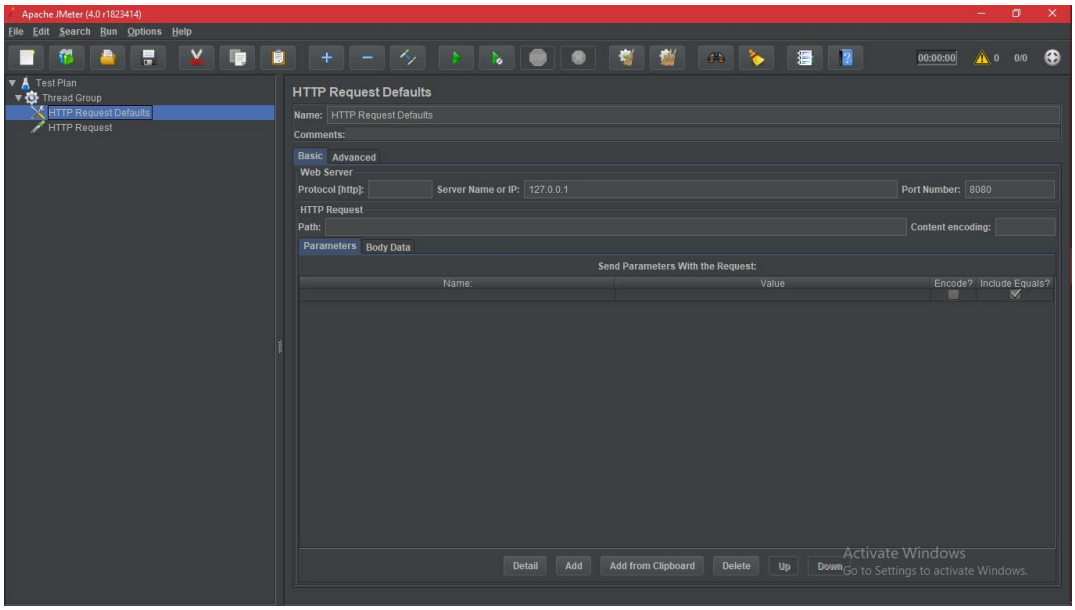
ขั้นตอนการทำงาน

เตรียม web application ที่ต้องการทดสอบจากนั้น download JMeter แต่ก่อนจะใช้งานเราต้องมีการติดตั้ง JDK ตั้งแต่เวอร์ชั่น 8 ขึ้นไป จากนั้นแตกไฟล์ออกมาแล้วเข้าไปที่ directory /bin แล้วรันตัว JMeter สำหรับ windows ให้รันไฟล์ jmeter.bat สำหรับ linux ให้รันไฟล์ jmeter.sh



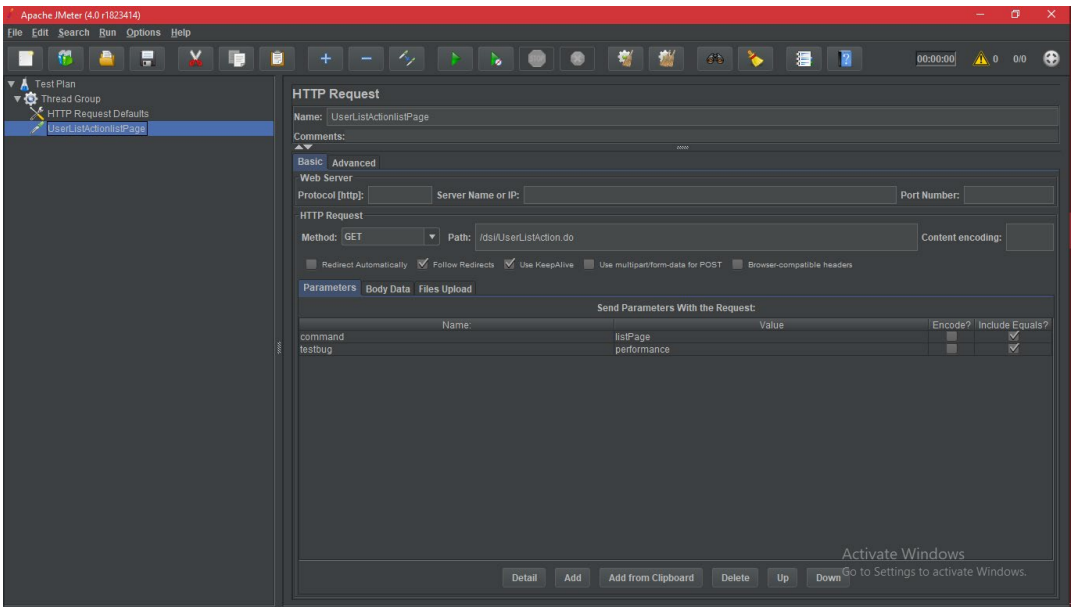
ขั้นตอนการทำงาน

สร้าง Thread Group โดยการคลิกขวาที่ TestPlan -> Add -> Thread Group ซึ่ง Thread Group จะเปรียบเสหมือนผู้ใช้ที่เราจะจำลองพฤติกรรมการทำงาน



ขั้นตอนการทำงาน

เพิ่ม HTTP Request Defulth โดยคลิกขวาที่ Thread Group -> Add -> Config Element -> HTTP Request Defualt สร้าง HTTP Request ซึ่งตรงนี้เป็นการสร้างต้นแบบให้แก่ทุกๆ request โดยที่เราจะสร้างขึ้นต่อจากนี้ใช้ค่าตามต้นแบบนี้



ขั้นตอนการทำงาน

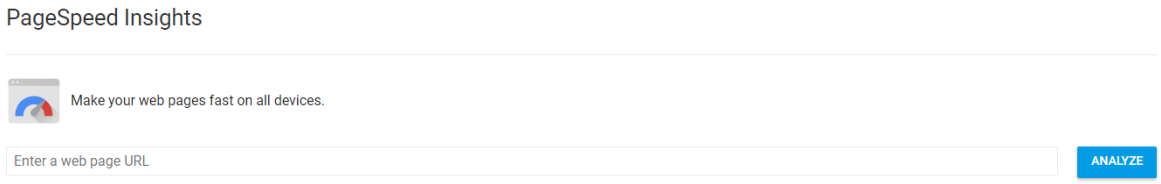
เพิ่ม HTTP Request โดยคลิกขวาที่ Thread Group -> Add -> Sampler -> HTTP Request ตรงนี้เป็นการเพิ่ม request ที่เราต้องการทดสอบ

Google PageSpeed

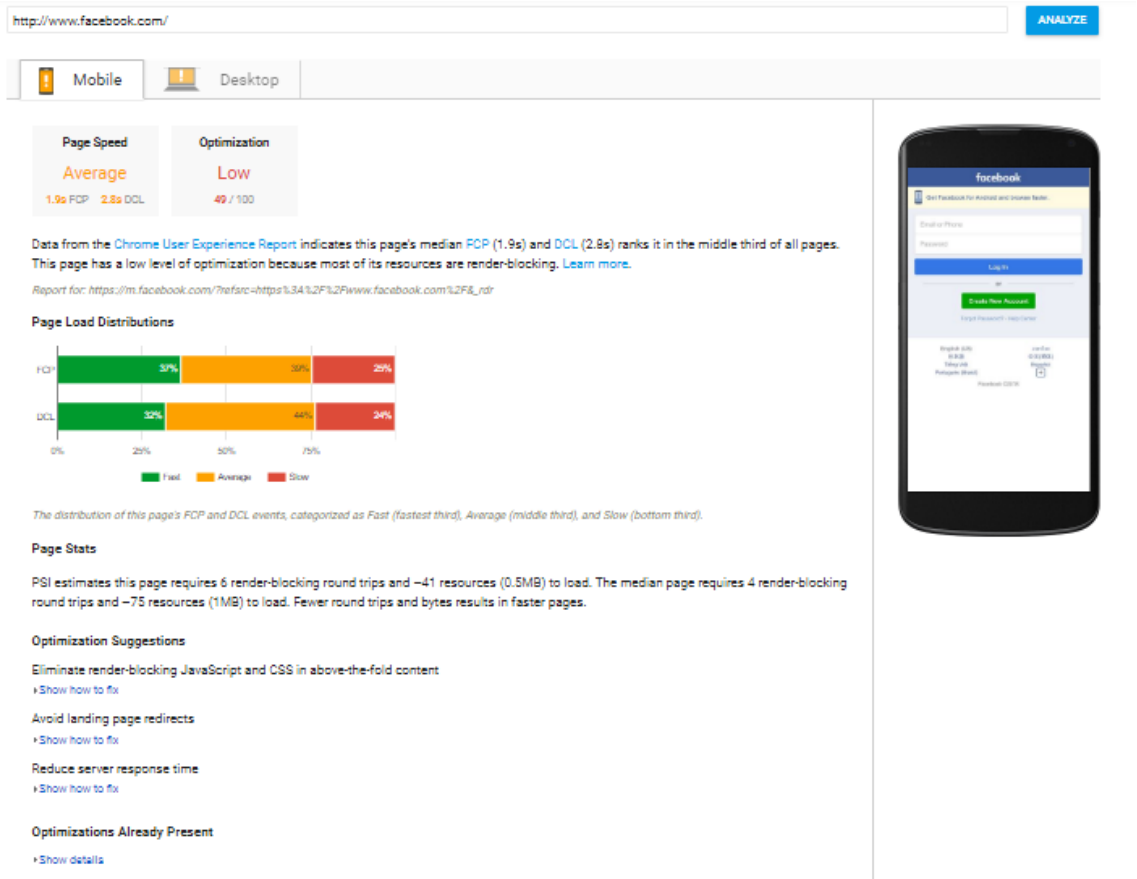
ค่า **Page Speed** คือ ค่าของความเร็วในการโหลดหน้าเว็บไซต์ โดยต้องเป็นเว็บไซต์ที่น่าเชื่อถือ Google ให้คะแนนการโหลดเว็บไซต์อยู่ที่ 100 คะแนน และตรวจสอบสิ่งที่มีผลกระทบต่อเว็บไซต์

การวัดความเร็วเว็บ มีขั้นตอน ดังนี้

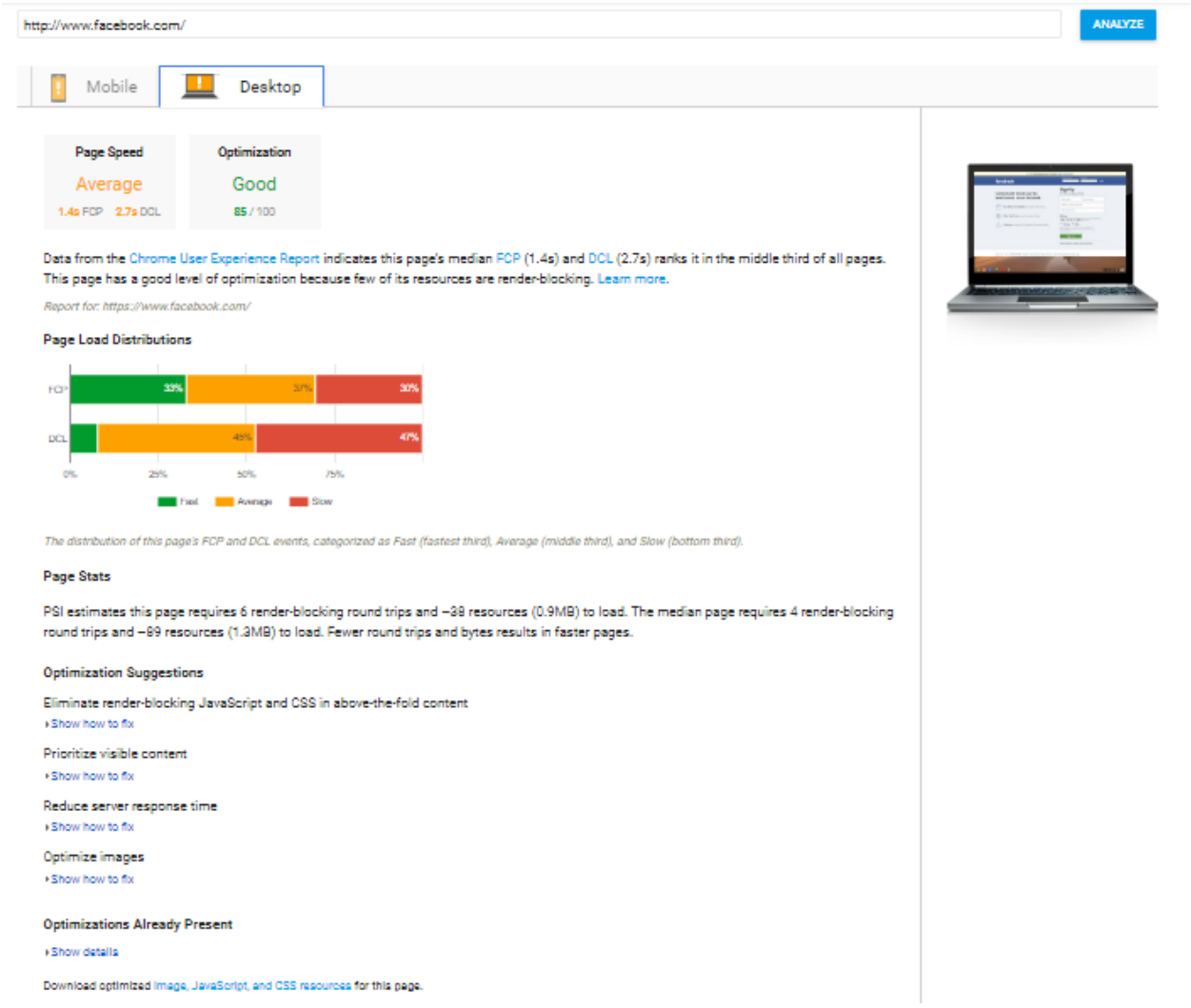
1. ไปที่ https://developers.google.com/speed/pagespeed/insights/ กรอก url ของเว็บที่ต้องการทดสอบ และรอผลการวิเคราะห์



2. แสดงผลการวิเคราะห์ Mobile



3. แสดงผลการวิเคราห์ Desktop



จากรูปข้างต้นที่ทำการทดสอบจะบอกเกณฑ์การให้คะแนนของทั้ง mobile และ Desktop ซึ่ง mobile อยู่ในเกณฑ์ที่แย่ต้องทำการแก้ไข มีส่วนแนะนำวิธีการแก้ไขว่าควรทำอย่างไรบ้างพร้อมลิ้งค์

ไปที่หน้าอธิบายข้อมูลต่างๆที่ต้องทำการแก้ไข

ปัจจัยที่ใช้วัดความเร็ว มีอยู่ 3 ส่วน คือ

- Request คือ การเรียกขอใช้งานต่างๆ ในการแสดงเว็บไซต์ เช่น ภาพ , Css , scriptต่างๆ หรือ JQuery
- Load Time คือ ระยะเวลาที่ใช้ในการโหลดทั้งหมดต่อหน้าโฮมเพจ
- Page Size คือ ขนาดของหน้าไฟล์ทั้งหมดของเพจนั้นๆ

ปัจจัยที่ทำให้เว็บไซต์โหลดได้ช้า คือ

1.Server ไม่มีประสิทธิภาพ สาเหตุอาจมาจากจำนวนการแชรีเว็บไซต์บน server นั้นๆมากเกินไป ทำให้ server ทำงานหนัก

2.Script Code ต่างๆ ที่รันอยู่บนเว็บไซต์มีมากเกินไป ปัญหานี้ค่อนข้างซับซ้อนเพราะต้องมีความรู้เรื่อง Coding ที่ใช้แต่งสคริปต่างๆ

3.Network โดยเฉพาะอย่างยิ่ง Bandwidth (ปริมารการรับและส่งข้อมูลทางอินเตอร์เน็ต Hosting) ของ Server ที่มีค่า Download และ Upload ต่ำ หรือ Server ราคาถูกทั้งหลาย เพราะยิ่งค่า Bandwidth ต่ำ ราคายิ่งถูก นั่นเอง

เบื้องต้นจะเห็นได้ว่า การทำเว็บไซต์ให้มีค่า Page speed ที่สูง ก็ต้องปรับทั้งปัจจัยทางด้าน Server และ การทำเว็บไซต์ ซึ่งการเพิ่มความเร็วให้เว็บไซต์ยังเป็นอีกเกณฑ์หนึ่งที่ถูกนำมาให้คะแนน SEO ของเว็บไซต์อีกด้วย

นอกจากนี้จะให้คะแนน **Google's Page Speed Insights ยังให้คำแนะนะเพื่อปรับปรุงเว็บไซต์อีกด้วย เช่น**

- ใช้รูปที่ใหญ่จนเกินไป
- ไม่มีการบีบอัดหรือย่อขนาดไฟล์ต่างๆ
- ไม่กำหนดข้อมุลของ Cache
- User Interface ยังไม่เหมาะสมกับผู้ใช้
- อื่นๆ

เอกสารอ้างอิง

https://tumit.wordpress.com/2012/06/02/assert-methods-3/

http://www.somkiat.cc/junit-test-isolation/

https://developers.google.com/speed/pagespeed/insights/

http://www.seomodify.com/seo/seo-3117/