ท่องเที่ยวอีสาน
TravelEsan

นายธนภูมิ สุปันนุช

โครงงานนี้เป็นส่วนหนึ่งของการศึกษา
หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ คณะวิทยาศาสตร์
มหาวิทยาลัยอุบลราชธานี
ปีการศึกษา 2562

โครงงาน	:	ท่องเที่ยวอีสาน TravelEsan
โดย	:	นายธนภูมิ สุปันนุช
	:	
อาจารย์ที่ปรึกษา	:	อาจารย์ วาสนา เหง้าเกษ
ระดับการศึกษา	:	วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ปีการศึกษา	:	2562

ได้รับการพิจารณาให้เป็นส่วนหนึ่งของการศึกษา

ตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์

คณะกรรมการสอบประเมินความรู้โครงงานคอมพิวเตอร์

. . . . . อาจารย์ที่ปรึกษา
(อาจารย์ วาสนา เหง้าเกษ)

. . . . . กรรมการ
(ดร.สุภาวดี หิรัญพงศ์สิน)

. . . . . กรรมการ
(ดร.วาโย ปุยะติ)

. . . . . หัวหน้าภาควิชา
(ผศ.ดร. สุพจน์ สีบุตร)

วันที่ . . . / . . . / . . .

กิตติกรรมประกาศ

การพัฒนาโครงงานระบบท่องเที่ยวอีสาน สำเร็จลุล่วงได้ด้วยความกรุณาและความช่วยเหลือจากหลายๆท่าน ข้าพเจ้าขอขอบพระคุณทุกท่าน ที่มีส่วนร่วมในการพัฒนาโครงงานนี้

ขอขอบพระคุณอาจารย์วาสนา เหง้าเกษ อาจารย์ที่ปรึกษาโครงงานที่ได้แนะนำทฤษฎีและแนวทางในแก้ปัญหาต่าง ๆ ที่เกิดขึ้นระหว่างการพัฒนาระบบ อีกทั้งยังคอยตรวจสอบความก้าว-หน้าของการทำงานเป็นระยะ รวมทั้งสร้างกำลังใจให้ผู้พัฒนาอยู่เสมอ

ขอขอบพระคุณทุกแหล่งข้อมูลที่ไม่สามารถเอ่ยนามได้หมด ที่ได้ให้ความร่วมมือเป็นอย่างดี

ขอกราบขอบพระคุณบิดา มารดา ที่คอยให้กำลังใจ คอยให้ความรักและความห่วงใยเสมอมา ตลอดจนคอยช่วยเหลือทุนทรัพย์ทางด้านการศึกษาและอุปกรณ์ในการพัฒนาโครงงาน

นายธนภูมิ สุปันนุช

พฤษภาคม 63

โครงงาน	:	ท่องเที่ยวอีสาน
โดย	:	นายธนภูมิ สุปันนุช
	:	
อาจารย์ที่ปรึกษา	:	อาจารย์ วาสนา เหง้าเกษ
ระดับการศึกษา	:	วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ปีการศึกษา	:	2562

บทคัดย่อ

ระบบท่องเที่ยวอีสานมีจุดประสงค์จัดทำขึ้นเพื่อเป็นการบอกลำดับการเดินทางไปยังสถานที่ท่องเที่ยวในภาคตะวันออกเฉียงเหนือที่ผู้ใช้งานสนใจ พร้อมแสดงเส้นทางการเดินทางไปยังสถานที่เหล่านั้น อีกทั้งยังแสดงข้อมูลสถานที่ท่องเที่ยว เช่น รายละเอียดของสถานที่ท่องเที่ยว และตำแหน่งของสถานที่ท่องเที่ยว การพัฒนาระบบท่องเที่ยวอีสานใช้ Laravel Framework และBootstrap ในการพัฒนาเว็บไซต์ และใช้ Distance Matrix API ในการหาระยะทาง

Topic	:	TravelEsan
Author	:	THANAPHUM SUPANNUT
	:	
Advisor	:	Wasana Ngaogate, Lecturer/Ph.D.
Degree	:	Bachelor of Science (Computer Science)
Academic Year	:	2019

Abstract

TravelEsan aims to provide a sequence of travel routes of attractions of the Northeastern region including detail of each route. Attraction information and location is also given. TravelEsan is developed by using the Laravel framework and bootstrap. The Distance matrix API is used for distance calculation.

สารบัญ

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

บทที่ 1

บทนำ

1.1 ที่มาและเหตุผล

คนไทยในปัจจุบันนิยมท่องเที่ยวด้วยตนเองมากขึ้น เห็นได้จากจำนวนบทความที่เขียนเผยแพร่ทางสื่อออนไลน์ เช่น เว็บไซต์พันทิป บล็อกหรือเฟซบุ๊ก เป็นต้น นักท่องเที่ยวเหล่านี้ส่วนใหญ่นิยมไปเที่ยวต่างจังหวัด แต่การเดินทางไปเที่ยวต่างจังหวัด นักท่องเที่ยวบางส่วนอาจยังไม่รู้จักสถานที่ท่องเที่ยวในจังหวัดนั้นมากนัก และยังไม่ค่อยรู้เส้นทาง ทำให้นักท่องเที่ยวประสบปัญหาในการวางแผนว่าควรไปสถานที่ไหนก่อน ซึ่งในปัจจุบันก็มีแอปพลิเคชันที่ใช้ในการดูเส้นทางอยู่มากมาย เช่น google map แต่แอปพลิเคชั่นเหล่านั้นยังไม่ได้อำนวยความสะดวกให้กับนักท่องเที่ยวเท่าที่ควร เนื่องจากการเรียงลำดับสถานที่ของ google map จะเรียงลำดับตามสถานที่ที่ผู้ใช้งานเพิ่มเข้ามา ซึ่งอาจทำให้มีระยะทางที่มาก ดังนั้น การพัฒนาซอฟต์แวร์ที่ช่วยอำนวยความสะดวกในการแนะนำสถานที่ท่องเที่ยวและวางแผนการเดินทางให้นักท่องเที่ยวจึงเป็นประโยชน์อย่างมาก

1.2 วัตถุประสงค์

1. เพื่อแสดงสถานที่ท่องเที่ยวต่างๆ ให้กับนักท่องเที่ยว
2. เพื่อบอกลำดับการเดินทางให้กับนักท่องเที่ยว

1.3 ขอบเขตของโครงงาน

1. สามารถแสดงตำแหน่งแหล่งท่องเที่ยวได้
2. แสดงสถานที่ท่องเที่ยวในจังหวัดอุบลราชธานี,ยโสธร
3. ผู้ใช้สามารถค้นหาสถานที่ท่องเที่ยวจากจังหวัดที่สนใจได้
4. มีการแสดงเส้นทางจากจุดที่ผู้ใช้อยู่ไปยังจุดหมาย
5. แสดงลำดับการเดินทางให้นักท่องเที่ยวได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ทำให้นักท่องเที่ยวรู้เส้นทางการเดินทาง
2. พื่อลดระยะเวลาในการเดินทาง

1.5 เครื่องมือที่ใช้ในการพัฒนา (Development tools)

1.5.1 ฮาร์ดแวร์

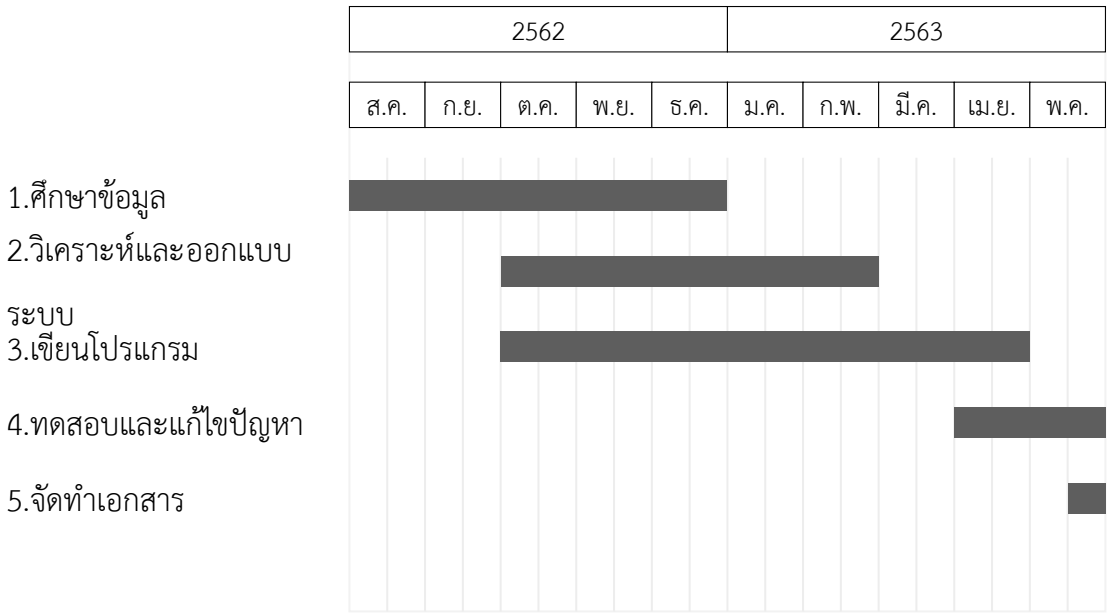
1. เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal computer)
    - ทำงานบนระบบปฏิบัติการ Windows 10
    - หน่วยประมวลผลกลาง (CPU) ความเร็ว 1.8 GHz
    - หน่วยความจำหลัก 4 กิกะไบต์ (Gigabyte, GB)

1.5.2 ซอฟต์แวร์ (Software)

1. visual studio code เป็น IDE Tools
2. google map api ใช้สำหรับเรียกใช้แผนที่
3. distance matrix api ใช้สำหรับหาระยะทาง
4. Laravel ซึ่งเป็น Frontend Framework สำหรับพัฒนาเว็บไซต์

1.5.3 แผนการดำเนินการ

ตารางที่ 1.1: ขั้นตอนการดำเนินงาน



บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

ในการจัดทำโครงงานคอมพิวเตอร์ในครั้งนี้ ผู้จัดทำโครงงานได้ศึกษาเอกสารและจากเว็บไซต์ต่างๆ ที่เกี่ยวข้องดังต่อไปนี้

2.1 ความรู้พื้นฐาน PHP

2.1.1 คุณสมบัติของภาษา PHP

ภาษา PHP [1] ที่ลักษณะเป็นแบบ Server Side Script คือจะอ่านโค้ด และทำงานที่เซิร์ฟ-เวอร์ จากนั้นจะส่งผลลัพธ์ที่ได้จากการประมวลผลกลับมาที่เครื่องของผู้ใช้ในรูปแบบของ HTML ซึ่งโค้ด PHP จะไม่สามารถมองเห็นได้ PHP สามารถทำงานร่วมกับระบบจัดการฐานข้อมูลที่หลากหลาย เช่น Oracle, MySQL เป็นต้น

2.1.2 โครงสร้างของภาษา PHP

ในภาษา PHP จะใช้เครื่องหมาย semicolon (;) ในการจบการทำงานของแต่ละคำสั่งเช่นเดียวกับภาษา C, C++ หรือ Java ภาษา PHP สามารถเขียนร่วมกับคำสั่ง Tag ของ HTML ได้ ตัวอย่างเช่น

```
1 <html>
2 <head>
3 </head>
4 <body>
5 <?php
6 echo"Hi";
7 ?>
8 </body>
9 </html>
```

รูปที่ 2.1: การจบการทำงานในแต่ละคำสั่งของภาษา PHP

จากตัวอย่างในรูปที่ 2.1 บรรทัดที่ 5 – 7 เป็นส่วนของภาษา PHP ซึ่งเขียนเริ่มต้นด้วย <? php ตามด้วยคำสั่ง และปิดด้วย ?> คำสั่ง echo จะใช้สำหรับแสดงข้อความ ภาษาPHPสามารถ

เปิดและปิดด้วย Tag ของ PHP ได้หลายครั้งดังตัวอย่างในรูป 2.2

```
1  <html>
2  <head>
3  </head>
4  <body>
5  <?php
6   echo"Hi";
7  ?>
8  <?php
9   echo"Hi";
10 ?>
11 </body>
12 </html>
```

รูปที่ 2.2: การสร้างTag PHP

2.1.3 ตัวแปร

การประกาศตัวแปรในภาษา PHP ไม่ต้องกำหนดประเภทของตัวแปร ซึ่งตัวแปรจะขึ้นต้นด้วยเครื่องหมาย Dollar sign ($) และตามด้วยชื่อตัวแปร ชื่อตัวแปรไม่สามารถขึ้นต้นด้วยตัวเลขได้

2.1.4 Srting

การประกาศตัวแปร string สามารถทำได้ 2 แบบโดยการใช้ Double quote ("") หรือ Single quote (') ซึ่งจะให้ผลลัพธ์แตกต่างกัน การใช้ Double quote จะทำให้สามารถแทรกตัวแปรใน string ได้ แต่การประกาศแบบ Single quote จะแสดงผลลัพธ์ของ string ตามที่ได้ประกาศไว้ในตัวแปร การต่อ string คือการนำ string มาต่อรวมกันและได้ผลลัพธ์ใหม่ ในภาษา PHP จะใช้เครื่องหมาย Dot (.) ในการต่อ string เข้าด้วยกัน ดังตัวอย่างในรูป 2.3

```
1 $firstname = "Samu";
2 $lastname = "Rai";
3 $fullname = $firstname.$lastname;
```

รูปที่ 2.3: การต่อ String

2.2 ความรู้พื้นฐาน Laravel Framework

Laravel Framework [2] เป็น PHP Framework สำหรับการพัฒนาเว็บแอปพลิเคชัน ถูกพัฒนาโดย Taylor Otwell และทีมงาน Laravel Framework เป็น open-source framework ทำให้นักพัฒนานำไปใช้ได้อย่างเสรี ทั้งในทางการค้าและไม่ใช่ทางการค้า

2.2.1 การติดตั้ง Laravel

1. ลง composer เพื่อใช้ในการติดตั้ง Laravel Framework
2. เปิด command line เพื่อติดตั้ง Laravel Framework โดยใช้คำสั่ง composer create-project –prefer-dist laravel/laravel blog
3. หลังจากติดตั้งเรียบร้อยแล้วให้เปิดเข้าไปใน Folder และทำการ Run โดยใช้คำสั่ง php artisan serve

2.2.2 โครงสร้างของ Laravel

1. app เป็นโฟลเดอร์ที่ใช้เก็บไฟล์จำพวก Model หรือ Controller ที่ใช้ในการประมวลผลและติดต่อกับฐานข้อมูล
2. database เป็นโฟลเดอร์ที่ใช้เก็บไฟล์จำพวก Migrations และ Seeding เพื่อใช้ในการสร้าง Table หรือใส่ข้อมูลในฐานข้อมูลผ่านคำสั่ง “artisan” ของ laravel
3. resources ใช้เก็บโฟลเดอร์ที่ใช้ในส่วนของการแสดงผลต่างๆ
4. routes เป็นส่วนที่ใช้เก็บไฟล์ในการกำหนด Url ของ web
5. storage เป็นส่วนของคลังพื้นที่จัดเก็บข้อมูลตระกูล Session, caches หรือไฟล์ที่ถูกทาง blade engine ทำการ compiled มาแล้ว
6. ไฟล์ .env เป็นไฟล์ที่ใช้ config laravel กับ ฐานข้อมูล

2.2.3 view

View คือส่วนที่ใช้สำหรับแสดงผลของเว็บ โดยใช้ HTML ในการจัดการส่วนนี้ ซึ่งจะถูกเก็บไว้ใน resources/views สามารถใช้ไฟล์ที่เป็นนามสกุล .html .php และ.blade.php ซึ่งการใช้งาน .blade.php จะเป็นการใช้งาน laravel blade template

2.2.3.1 blade template

การใช้งาน blade template [3] จะต้องตั้งชื่อไฟล์ที่มีนามสกุลเป็น .blade.php การใช้งาน blade template จะทำให้โค้ดดูอ่านง่ายขึ้น ทำให้นักพัฒนาสะดวกยิ่งขึ้น ใน blade template จะใช้เครื่องหมาย ครอบตัวแปร จะเหมือนกับคำสั่ง echo ตัวอย่างเช่น $name จะเป็นการเอาค่า $name ออกมาแสดง

2.2.4 Controller

Controller ทำหน้าที่เป็นส่วนจัดการการทำงานของระบบ โดยการสร้าง Controller จะใช้คำสั่ง php artisan make:controller BlogController ไฟล์ Controller ที่สร้างจะอยู่ที่ app/Http/Controllers ตัวอย่าง Code ใน Controller

```
1 <?php
2 namespace App\Http\Controller;
3 use Iluminate\Http\Request;
4 Class BlogController extends Controller{
5 Public function index(){
6 echo ‘’Hello;
7 }
8 }
```

รูปที่ 2.4: Controller

จากตัวอย่างในรูป 2.4 ถ้าทำการเรียกใช้ BlogController และ function index() จะทำการแสดงผลลัพธ์ว่า Hello โดยการเรียกใช้งานจะกำหนดอยู่ที่ Route

2.2.5 Route

Route เป็นการกำหนดเส้นทางให้กับเว็บว่าให้ไปทำงานที่ controller หรือ แสดงที่ view และยังสามารถส่งพารามิเตอร์ผ่าน Route ได้ ไฟล์ที่ใช้กำหนด Route จะอยู่ใน routes/web.php ตัวอย่างของการกำหนด Route

```
1 Route::get('/', function(){
2 return view('welcome');
3 })
```

รูปที่ 2.5: การกำหนด Route

จากตัวอย่างในรูป 2.5 หมายความว่าเมื่อเปิดหน้าแรกขึ้นมาจะแสดง view welcome

```
1 Route::get('/', 'BlogController@index')
```

รูปที่ 2.6: การเรียกใช้งาน Controller ผ่าน Route

จากตัวอย่างในรูป 2.6จะเป็นการเรียกใช้ Controller ชื่อ BlogController และ Function index() ให้ทำงาน

```
1 Route::get('product/{id}', 'BlogController@test')
```

รูปที่ 2.7: การส่งพารามิเตอร์ผ่าน Route

จากตัวอย่างในรูป 2.7 จะเป็นการส่งพารามิเตอร์ id ไปด้วย โดยพารามิเตอร์ต้องคลุมด้วยเครื่องหมาย เสมอ

2.2.6 Model

Model จะทำหน้าที่สำหรับดึงข้อมูลจากตารางของฐานข้อมูลมาใช้งาน โดยไม่จำเป็นต้องเขียนคำสั่ง SQL โดยปกติชื่อของ Model จะดึงข้อมูลและจัดเก็บข้อมูลจากตารางชื่อเดียวกัน ยกตัวอย่างการสร้าง Model ชื่อ Product จะใช้คำสั่งดังแสดงในรูปที่ 2.8

```
1 php artisan make:model Product
```

รูปที่ 2.8: การสร้าง model

การสร้างข้อมูลหรือการดึงข้อมูลจากตาราง ต้องทำการกำหนดการเข้าถึงของ column ในตารางก่อน โดยการกำหนดการเข้าถึงจะใช้ fillable หรือ guarded ตัวอย่างการใช้งาน

1. Fillable คือการกำหนดชื่อของ column ที่ต้องการอนุญาตให้เข้าถึง protected $fillable = [’name’];
2. Guarded คือการกำหนดชื่อของ column ที่ไม่อนุญาตให้เข้าถึง protected $guarded = [’name’]; protected $guarded = []; การกำหนด guarded เป็นค่าว่างจะหมายถึง column ทุก column สามารถเข้าถึงได้

การ query ข้อมูลจาก Model สามารถ query ข้อมูลได้ทั้งใน Route และ Controller โดยจะต้องทำการ use Model ก่อน ตัวอย่างการ query ข้อมูลใน Controller

```
1 <?php
2 use App\Flight;
3 $flights = Flight::all();
4 foreach ($flights as $flight) {
5 echo $flight->name;
6 }
7 $flights = Flight::where(’id’, 1)->get();
```

รูปที่ 2.9: การ query ข้อมูลจาก Model

จากตัวอย่างในรูป 2.9 $flights = Flight::all(); จะเป็นการดึงข้อมูลทั้งหมดมาเก็บไว้ในตัวแปร $flights ส่วนคำสั่ง $flights = Flight::where(’id’, 1) ->get(); จะเป็นการดึงข้อมูลที่มีค่า id เท่ากับ 1 มาเก็บในตัวแปร $flights

2.3 Guzzle

Guzzle [4] เป็น HTTP Client ที่ใช้สำหรับส่ง HTTP requests และเชื่อมต่อกับ Web services เพื่อแลกเปลี่ยนข้อมูลได้ Guzzle ทำงานได้กับ PHP 5.5.0 ขึ้นไป การติดตั้ง Guzzle จะติดตั้งผ่าน composer โดยใช้คำสั่ง composer require guzzlehttp/guzzle

2.3.1 การใช้งาน Guzzle

การใช้งาน Guzzle จะต้องทำการ use ตัว Guzzle ก่อนโดยต้องประกาศไว้ที่ด้านบนจากนั้นทำการ new Client() และส่ง request ผ่าน method request ดังนี้

```
1 <?php
2 namespace App;
3 use GuzzleHttp\Client;
4 $client = new Client();
5 $response = $client->request($method, $url, $options
    )
```

รูปที่ 2.10: การใช้งาน Guzzle

1. $method คือ HTTP Method ได้แก่ GET,POST,PUT,DELETE
2. $url คือ url ที่จะทำการ request
3. $options เป็น Array สำหรับเก็บ option ต่างๆ เช่น Header และAuthentication เป็น-ต้น

ตัวอย่างการ GET ค่าจาก URL https://maps.googleapis.com/maps/api

```
1 <?php
2 namespace App;
3 use GuzzleHttp\Client;
4 $client = new Client();
5 $response =
6 $client->request‘(’GET’,https://maps.googleapis.com/
    maps/’api);
7 echo $response->getBody();
```

รูปที่ 2.11: ตัวอย่างการใช้งาน Method GET

การส่ง headers ใน Guzzle จะต้องเขียนใน Array ดังตัวอย่างในรูป 2.12

```
1  <?php
2  namespace App;
3  use GuzzleHttp\Client;
4  $client = new Client();
5  $response =
6  $client->request‘(’GET’,https://maps.googleapis.com/
      maps/’api,
7  [‘’
8  headers=>[‘’
9  key => ‘’f40asd5
10 ]
11 ]
12 );
13 echo $response->getBody();
```

รูปที่ 2.12: ตัวอย่างการใช้งาน headers

2.4 Distance Matrix API

Distance Martrix API [5] เป็นหนึ่งในบริการ Web Service ของ google map โดย Distance Martrix API มีความสามารถในการคำนวณระยะทางจากสถานที่ต้นทางไปยังสถานที่ปลายทางและสามารถกำหนดปลายทางได้หลายตำแหน่ง ข้อมูลที่ได้รับกลับจาก Web Service จะอยู่ในรูปแบบ JSON หรือ XML ระบบเว็บไซต์หรือแอปพลิเคชันสามารถนำข้อมูลเหล่านี้ไปแสดงผลหรือประยุกต์ใช้ต่อได้ การใช้งาน Distance Matrix API ต้องส่งคำขอเป็น URL string โดยมีแบบฟอร์มดังในรูปที่ 2.13

```
1  https://maps.googleapis.com/maps/api/distancematrix/
      outputFormat?parameters
```

รูปที่ 2.13: ตัวอย่าง URL ที่เรียกใช้งาน Distance Matrix API

outputFormat สามารถเลือกค่าได้ดังต่อไปนี้

1. json ตัวอย่างผลลัพธ์ที่ถูกส่งกลับมา

```
{
"status": "OK",
"origin_addresses": [
"Vancouver, BC, Canada"
],
"destination_addresses": [
"Victoria, BC, Canada"
],
"rows": [
{
"elements": [
{
"status": "OK",
"duration": {
"value": 24487,
"text": "6 heures 48 minutes"
},
"distance": {
"value": 129324,
"text": "129 km"
}
}
]
}
]
}
```

รูปที่ 2.14: ตัวอย่างผลลัพธ์ในรูปแบบ json

2. xml ตัวอย่างผลลัพธ์ที่ถูกส่งกลับมา

```
{
"status": "OK",
"origin_addresses": [
"Vancouver, BC, Canada"
],
"destination_addresses": [
"Victoria, BC, Canada"
],
"rows": [
{
"elements": [
{
"status": "OK",
"duration": {
"value": 24487,
"text": "6 heures 48 minutes"
},
"distance": {
"value": 129324,
"text": "129 km"
}
}
]
}
]
}
```

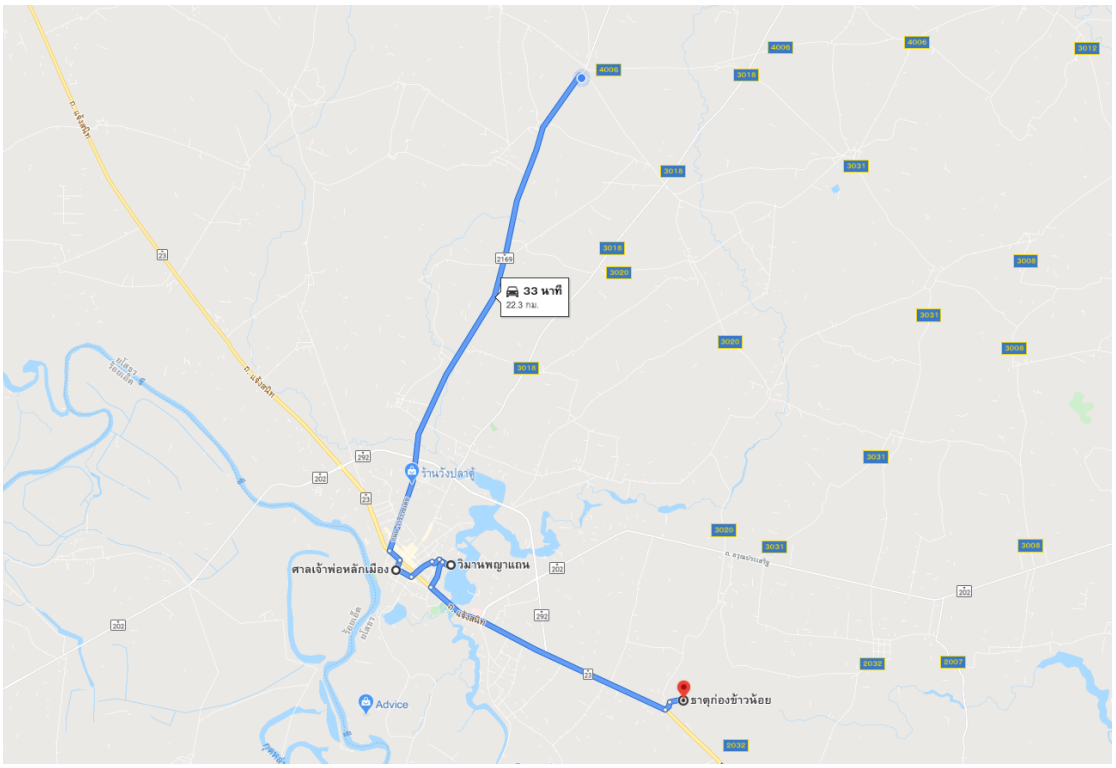
รูปที่ 2.15: ตัวอย่างผลลัพธ์ในรูปแบบ xml

parameters พารามิเตอร์แต่ละตัวจะถูกคั่นด้วยเครื่องหมาย '&' โดยมีพารามิเตอร์ที่จำเป็นดังต่อไปนี้

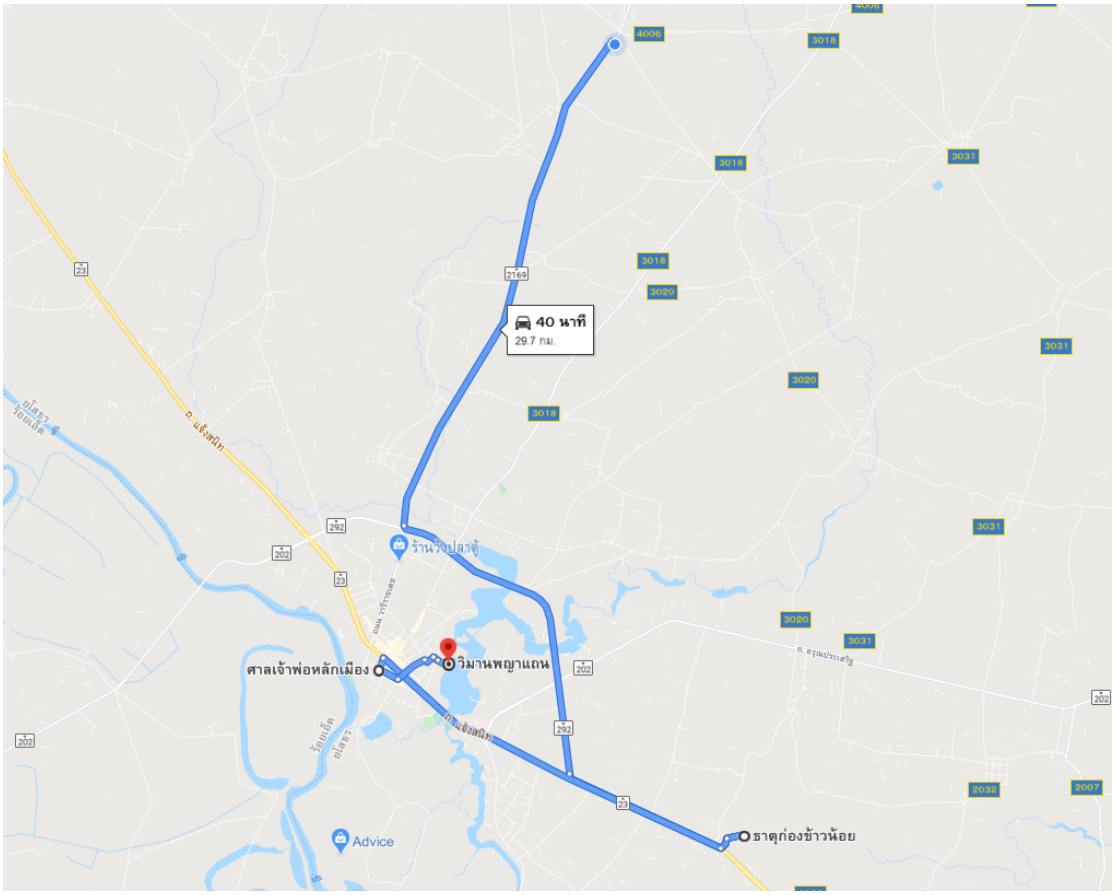
1. origins คือจุดเริ่มต้นสำหรับคำนวณระยะทางโดยสามารถใส่ในรูปแบบของที่อยู่พิกัดละติจูดและลองจิจูด ที่อยู่ หรือรหัสสถานที่ โดยใช้เครื่องหมาย | คั่นกรณีมีหลายตำแหน่ง เช่น origins=41.43206,-81.38992|-33.86748,151.20699
2. destinations คือจุดปลายทางสำหรับการคำนวณโดยสามารถใส่ในรูปแบบของที่อยู่พิกัดละติจูดและลองจิจูด ที่อยู่ หรือรหัสสถานที่ โดยใช้เครื่องหมาย | คั่นกรณีมีหลายตำแหน่ง เช่น destinations= San+Francisco|Vancouver+BC
3. key คือรหัส API ที่ได้ทำการขอใช้งานจาก google

2.5 การหาลำดับการเดินทาง

การหาลำดับการเดินทางของระบบท่องเที่ยวอีสาน จะจัดลำดับโดยการเลือกสถานที่ที่อยู่ใกล้กับตำแหน่งปัจจุบันของผู้ใช้งาน ตัวอย่างเช่น ผู้ใช้งานต้องการไป ธาตุก่องข้าวน้อย ศาลเจ้าพ่อหลักเมือง และวิมานพญาแถน ซึ่งจากตำแหน่งปัจจุบันไปธาตุก่องข้าวน้อย 20กม. ตำแหน่งปัจจุบันไปศาลเจ้าพ่อหลักเมือง 13กม. และตำแหน่งปัจจุบันไปวิมานพญาแถน 14กม. ระบบจะทำการเลือกไปศาลเจ้าพ่อหลักเมืองก่อน จากนั้นจะดูว่าจากศาลเจ้าพ่อหลักเมืองใกล้สถานที่ใดก็จะเลือกสถานที่นั้นต่อไป ในตัวอย่างนี้จะทำการเลือกวิมานพญาแถนและธาตุก่องข้าวน้อยตามลำดับ



รูปที่ 2.16: ตัวอย่างลำดับการเดินทางที่ได้จากระบบ

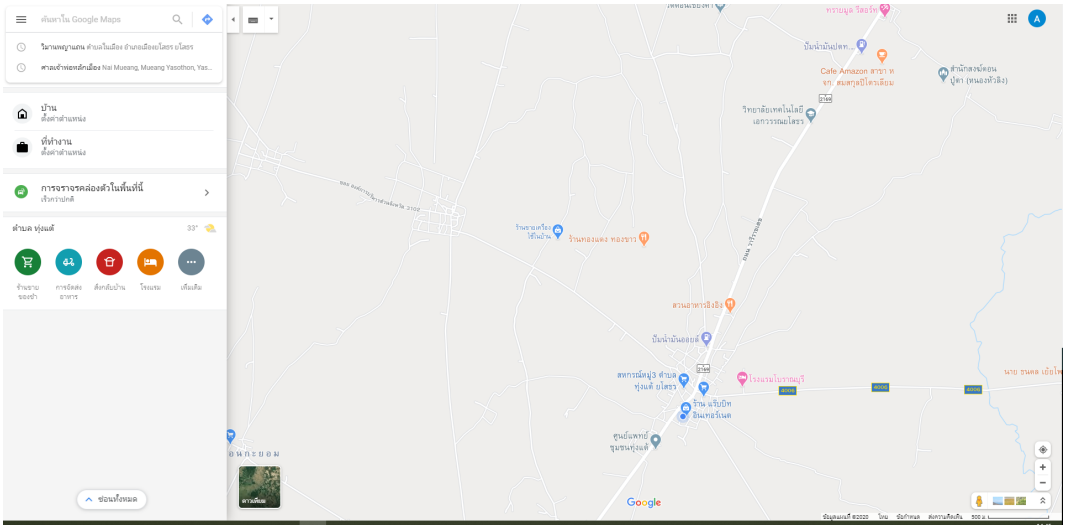


รูปที่ 2.17: ตัวอย่างลำดับการเดินทางที่เรียงลำดับตามสถานที่ที่เลือก

2.6 เอกสารและงานวิจัยที่เกี่ยวข้อง

2.6.1 Google Maps

Google Maps เป็นเว็บไซต์ที่สามารถแสดงเส้นทางการเดินทาง โดยผู้ใช้งานสามารถเลือกสถานที่ที่ต้องการไปได้ สามารถเข้าใช้งานได้ทาง https://www.google.co.th/maps



รูปที่ 2.18: หน้าแรกของ Google Maps

ที่มา : https://www.google.co.th/maps

2.6.2 Thailand Tourism Directory

Thailand Tourism Directory เป็นเว็บไซต์ค้นหาสถานที่ท่องเที่ยว ร้านอาหาร ที่พัก ในประเทศไทย สามารถเข้าใช้งาน ได้ทาง https://thailandtourismdirectory.go.th/th/home

รูปที่ 2.19: หน้าแรกของ Thailand Tourism Directory

ที่มา : https://thailandtourismdirectory.go.th/th/home

บทที่ 3

การวิเคราะห์และออกแบบระบบ

การวิเคราะห์และออกแบบระบบก่อนดำเนินการจริงเป็นอีกหนึ่งขั้นตอนที่มีความสำคัญ เพราะการวิเคราะห์และออกแบบระบบนั้นเป็นการกระทำที่ทำให้ผู้พัฒนาเห็นรายละเอียดส่วนย่อยของงานทั้งหมด เพิ่มประสิทธิภาพในการวางแผน การทำงาน และยังช่วยลดปัญหาที่อาจจะเกิดขึ้นในระหว่างพัฒนา เพื่อให้ระบบมีความสมบูรณ์มากยิ่งขึ้น ในบทนี้จะแบ่งออกเป็น 3 ขั้นตอน เพื่อให้เห็นการดำเนินงานอย่างมีระบบ

3.1 usecase diagram

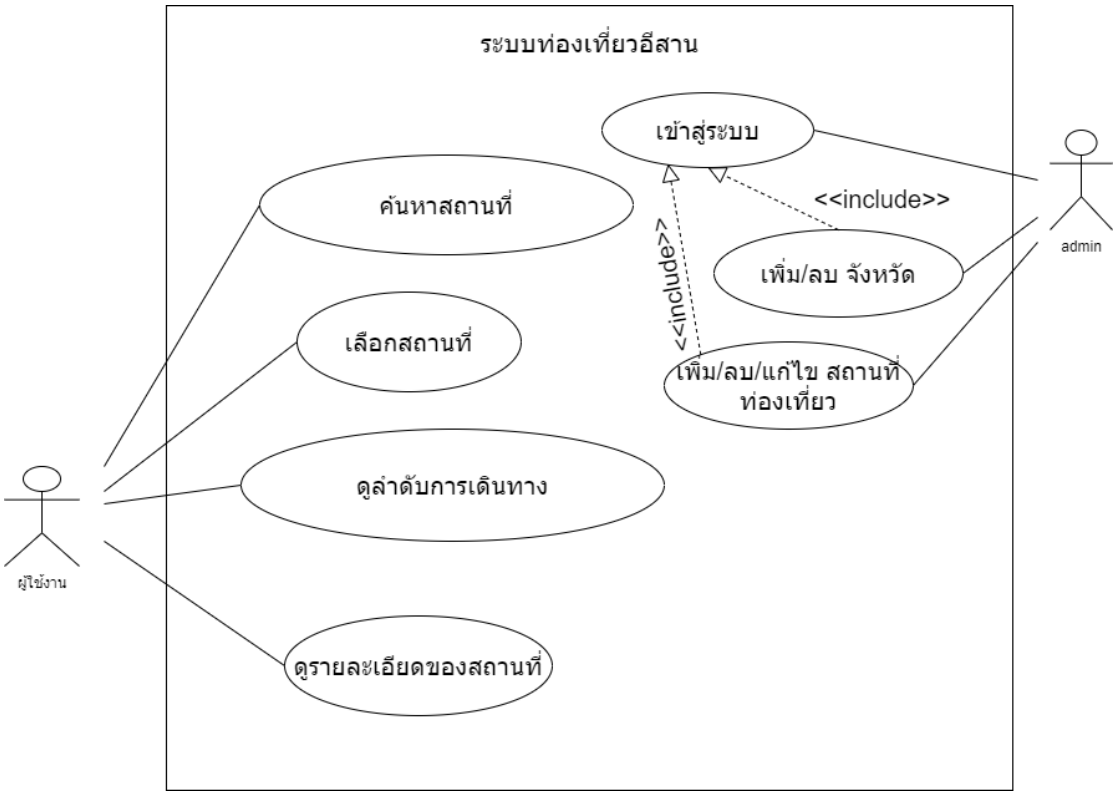
3.2 context diagram

3.3 data flow diagram level 0

3.4 Entity Relationship Diagrams

3.5 Sequence Diagrams

3.1 usecase diagram



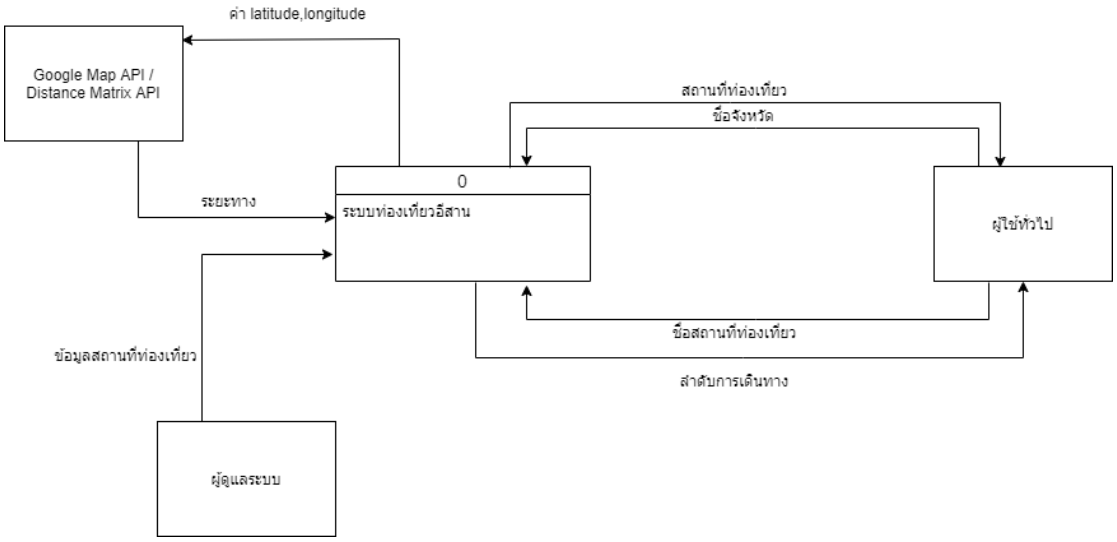
รูปที่ 3.1: usecase diagram ของระบบท่องเที่ยวอีสาน

ตารางที่ 3.1: อธิบาย Use Case หน้าที่ของระบบ ในภาพที่ 3.1

Use Case	คำอธิบาย
ค้นหาสถานที่ท่องเที่ยว	ผู้ใช้งานสามารถค้นหาสถานที่ท่องเที่ยวได้โดยการเลือกจากชื่อจังหวัด
เลือกสถานที่	ผู้ใช้งานเลือกสถานที่ท่องเที่ยวที่สนใจ
ดูลำดับการเดินทาง	ผู้ใช้งานสามารถดูลำดับการเดินทางไปยังสถานที่ท่องเที่ยวที่เลือกได้
ดูรายละเอียดของสถานที่	ผู้ใช้งานสามารถดูรายละเอียดของสถานที่ท่องเที่ยว และตำแหน่งของสถานที่ท่องเที่ยว
เข้าสู่ระบบ	admin ทำการเข้าสู่ระบบก่อน เพื่อที่จะจัดการข้อมูลสถานที่ท่องเที่ยว
เพิ่ม/ลบ จังหวัด	admin สามารถเพิ่มและลบข้อมูลจังหวัด
เพิ่ม/ลบ/แก้ไข	admin สามารถเพิ่ม แก้ไขและลบ ข้อมูลสถานที่ท่องเที่ยว

3.2 context diagram

Context diagram เป็น diagram ที่แสดงถึงขอบเขตของระบบงาน โดยไม่แสดงสัญลักษณ์แหล่งจัดเก็บข้อมูล ทำให้สามารถเห็นภาพรวมของระบบ และขอบเขตการวิเคราะห์ระบบ

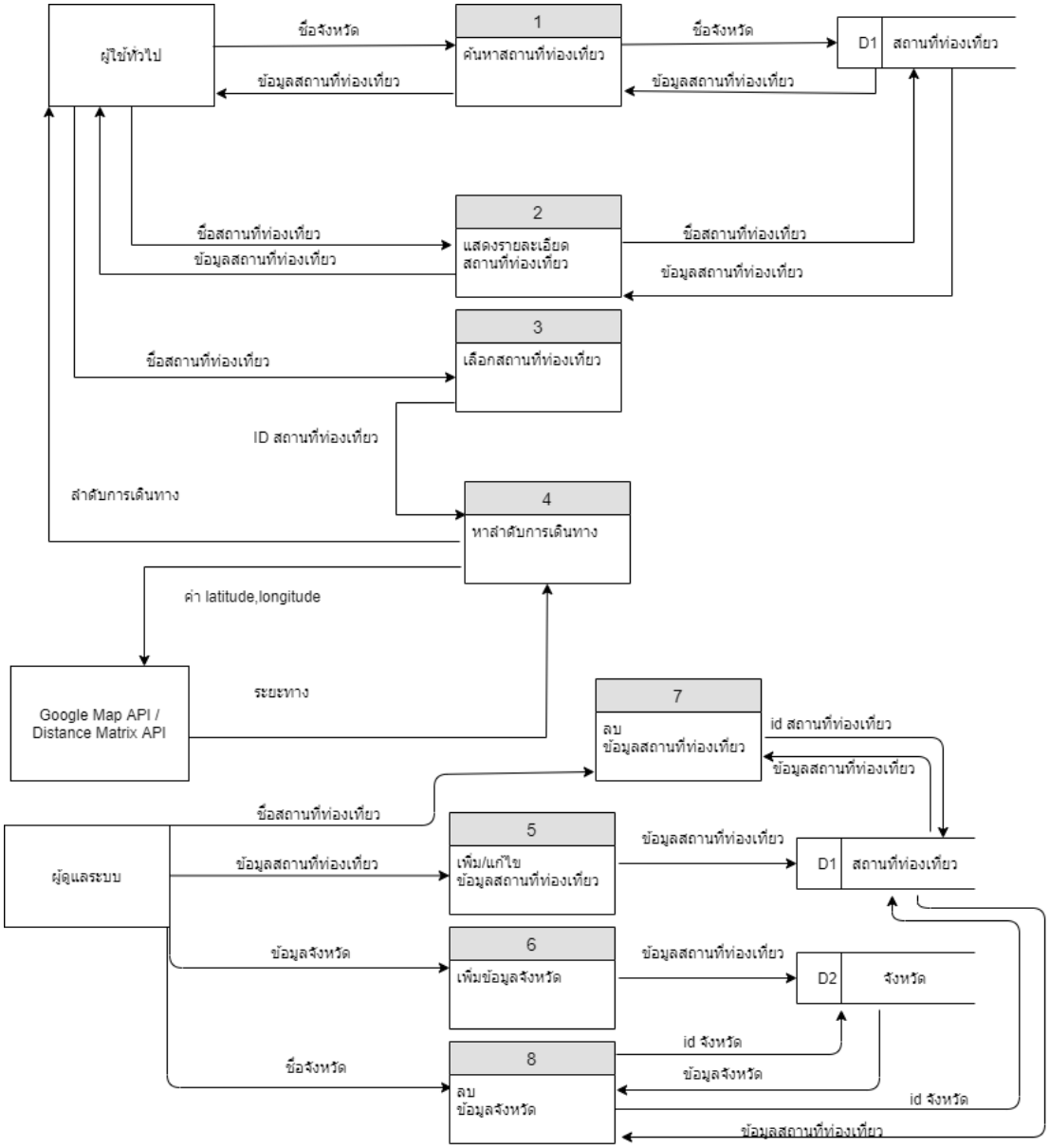


รูปที่ 3.2: context diagram ของระบบท่องเที่ยวอีสาน

จากรูปที่ 3.2 มีส่วนที่เกี่ยวข้อง 3 ส่วน คือ ผู้ใช้ทั่วไป ผู้ดูแลระบบ และgoogle map/ distance martrix api

3.3 data flow diagram level 0

DFD 0 หรือ data flow diagram level 0 เป็นแผนภาพแสดงการไหลของข้อมูลของระบบ เพื่อแสดงวิธีการไหลของข้อมูลจากกระบวนการหนึ่งไปอีกกระบวนการหนึ่ง



รูปที่ 3.3: data flow diagram level 0 ของระบบท่องเที่ยวอีสาน

ตารางที่ 3.2: Process description ค้นหาสถานที่ท่องเที่ยว

Process name :	ค้นหาสถานที่ท่องเที่ยว
Input Data Flows :	ชื่อจังหวัด
Output Data Flows :	ข้อมูลสถานที่ท่องเที่ยว
Data Stored used :	สถานที่ท่องเที่ยว
Description :	เป็นกระบวนการค้นหาสถานที่ท่องเที่ยวจากจังหวัดที่ผู้ใช้สนใจ

ตารางที่ 3.3: Process description แสดงรายละเอียดสถานที่ท่องเที่ยว

Process name :	แสดงรายละเอียดสถานที่ท่องเที่ยว
Input Data Flows :	ชื่อสถานที่ท่องเที่ยว
Output Data Flows :	ข้อมูลสถานที่ท่องเที่ยว
Data Stored used :	สถานที่ท่องเที่ยว
Description :	เมื่อผู้ใช้คลิกดูรายละเอียดสถานที่ ระบบจะส่ง ID สถานที่ท่องเที่ยวเพื่อไปค้นหาข้อมูลจาก Data Stored ที่มีID สถานที่ท่องเที่ยวเหมือนกัน

ตารางที่ 3.4: Process description เลือกสถานที่ท่องเที่ยว

Process name :	เลือกสถานที่ท่องเที่ยว
Input Data Flows :	ชื่อสถานที่ท่องเที่ยว
Output Data Flows :	-
Data Stored used :	-
Description :	เมื่อผู้ใช้ทำการเลือกสถานที่ท่องเที่ยว ระบบจะทำการเก็บ ID สถานที่ท่องเที่ยวที่เลือกไว้ใน Session

ตารางที่ 3.5: Process description หาลำดับการเดินทาง

Process name :	หาลำดับการเดินทาง
Input Data Flows :	ID สถานที่ท่องเที่ยวที่เลือก,ระยะทาง
Output Data Flows :	ลำดับการเดินทาง,ค่าlattitude,longitude
Data Stored used :	สถานที่ท่องเที่ยว
Description :	ระบบจะทำการเอาค่า ID สถานที่ท่องเที่ยวจาก Session เพื่อไปดึงข้อมูลละติจูดกับลองจิจูดไปหาระยะทางด้วย Distance Matrix API เพื่อทำการเรียงลำดับการเดินทาง และส่งให้ผู้ใช้ทั่วไป

ตารางที่ 3.6: Process description เพิ่ม/แก้ไขข้อมูลสถานที่ท่องเที่ยว

Process name :	เพิ่ม/แก้ไขข้อมูลสถานที่ท่องเที่ยว
Input Data Flows :	ข้อมูลสถานที่ท่องเที่ยว
Output Data Flows :	ข้อมูลสถานที่ท่องเที่ยว
Data Stored used :	สถานที่ท่องเที่ยว
Description :	admin ทำการกรอกข้อมูลสถานที่ท่องเที่ยวและทำการบันทึกข้อมูลลง Data Stored D1

ตารางที่ 3.7: Process description เพิ่มข้อมูลจังหวัด

Process name :	เพิ่มข้อมูลจังหวัด
Input Data Flows :	ข้อมูลจังหวัด
Output Data Flows :	ข้อมูลจังหวัด
Data Stored used :	จังหวัด
Description :	admin ทำการ กรอก ข้อมูล จังหวัด และ ทำการ บันทึก ลง Data Stored D2

ตารางที่ 3.8: Process description ลบสถานที่ท่องเที่ยว

Process name :	ลบข้อมูลสถานที่ท่องเที่ยว
Input Data Flows :	ชื่อสถานที่ท่องเที่ยว,ข้อมูลสถานที่ท่องเที่ยว
Output Data Flows :	id สถานที่ท่องเที่ยว
Data Stored used :	สถานที่ท่องเที่ยว
Description :	admin ทำการเลือกสถานที่ท่องเที่ยวที่ต้องการลบ และระบบทำการส่ง id สถานที่ท่องเที่ยวเพื่อดึงข้อมูลจาก Data Stored D1 มาเพื่อทำการลบข้อมูล

ตารางที่ 3.9: Process description ลบข้อมูลจังหวัด

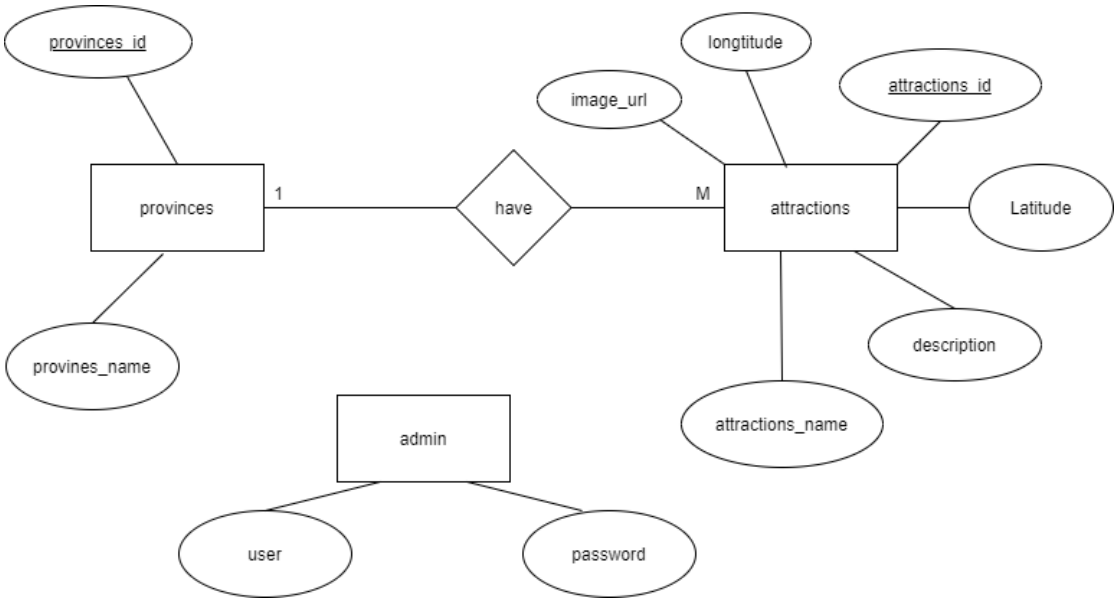
Process name :	ลบข้อมูลจังหวัด
Input Data Flows :	ชื่อจังหวัด,ข้อมูลจังหวัด,ข้อมูลสถานที่ท่องเที่ยว
Output Data Flows :	idจังหวัด
Data Stored used :	สถานที่ท่องเที่ยว,จังหวัด
Description :	admin ทำการเลือกจังหวัดที่ต้องการลบ ระบบจะส่งidจังหวัดนั้นเพื่อไปดึงข้อมูลสถานที่ท่องเที่ยวในจังหวัดนั้นจาก Data Stored D1 มาลบ และทำการลบข้อมูลจังหวัดใน Data Stored D2

3.4 Entity Relationship Diagrams

Entity Relationship Diagrams คือ แบบจำลองที่ใช้อธิบายโครงสร้างของฐานข้อมูลซึ่งเขียนออกมาในลักษณะของรูปภาพ การอธิบายโครงสร้างและความสัมพันธ์ของข้อมูล ประกอบด้วย

1. เอนทิตี้ (Entity) เป็นวัตถุ หรือสิ่งของที่เราสนใจในระบบงานนั้น ๆ
2. แอททริบิว (Attribute) เป็นคุณสมบัติของวัตถุที่เราสนใจ
3. ความสัมพันธ์ (Relationship) คือ ความสัมพันธ์ระหว่างเอนทิตี้

Entity Relationship Diagram ของระบบอธิบายได้ตามภาพที่ดังต่อไปนี้



รูปที่ 3.4: Entity Relationship Diagram ของระบบท่องเที่ยวอีสาน

3.4.1 Data dictionary

ตารางที่ 3.10: ตาราง provinces

Attribute Name	Description	Data Type	Key Type
provinces_id	รหัสจังหวัด	int	PK
provinces_name	ชื่อจังหวัด	varchar	

ตารางที่ 3.11: ตาราง attractions

Attribute Name	Description	Data Type	Key Type
provinces_id	รหัสจังหวัด	int	FK
attractions_id	รหัสสถานที่ท่องเที่ยว	int	PK
attractions_name	ชื่อสถานที่ท่องเที่ยว	varchar	
Latitude	ละติจูด	double	
longitude	ลองจิจูด	double	
description	รายละเอียด	text	
image_url	url รูปภาพ	varchar	

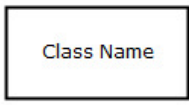
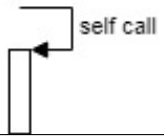
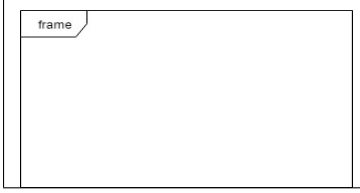
ตารางที่ 3.12: ตาราง admin

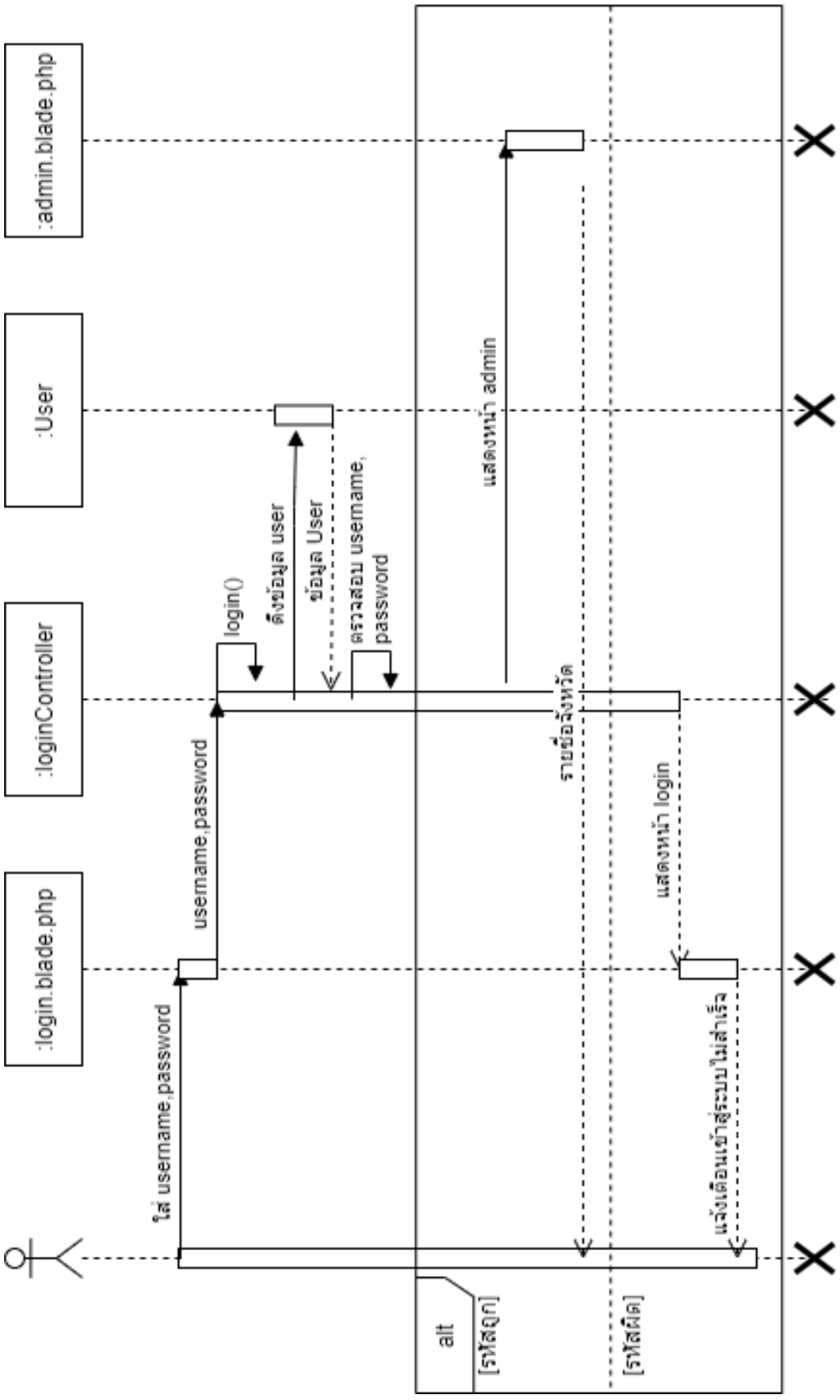
Attribute Name	Description	Data Type	Key Type
username	ชื่อผู้ใช้	varchar	
password	รหัสผู้ใช้	varchar	

3.5 Sequence Diagram

Sequence Diagram เป็น Diagram ที่แสดงขั้นตอนการทำงานของแต่ละ Use Case ระหว่าง Object ต่าง ๆ ที่ส่งข้อความถึงกันและกัน โดย Sequence Diagram จะช่วยให้มองเห็นการทำงานของภาพรวมของระบบ ส่วนประกอบสัญลักษณ์ที่ใช้ในการเขียน Sequence Diagram แสดงดังตารางที่ 3.13

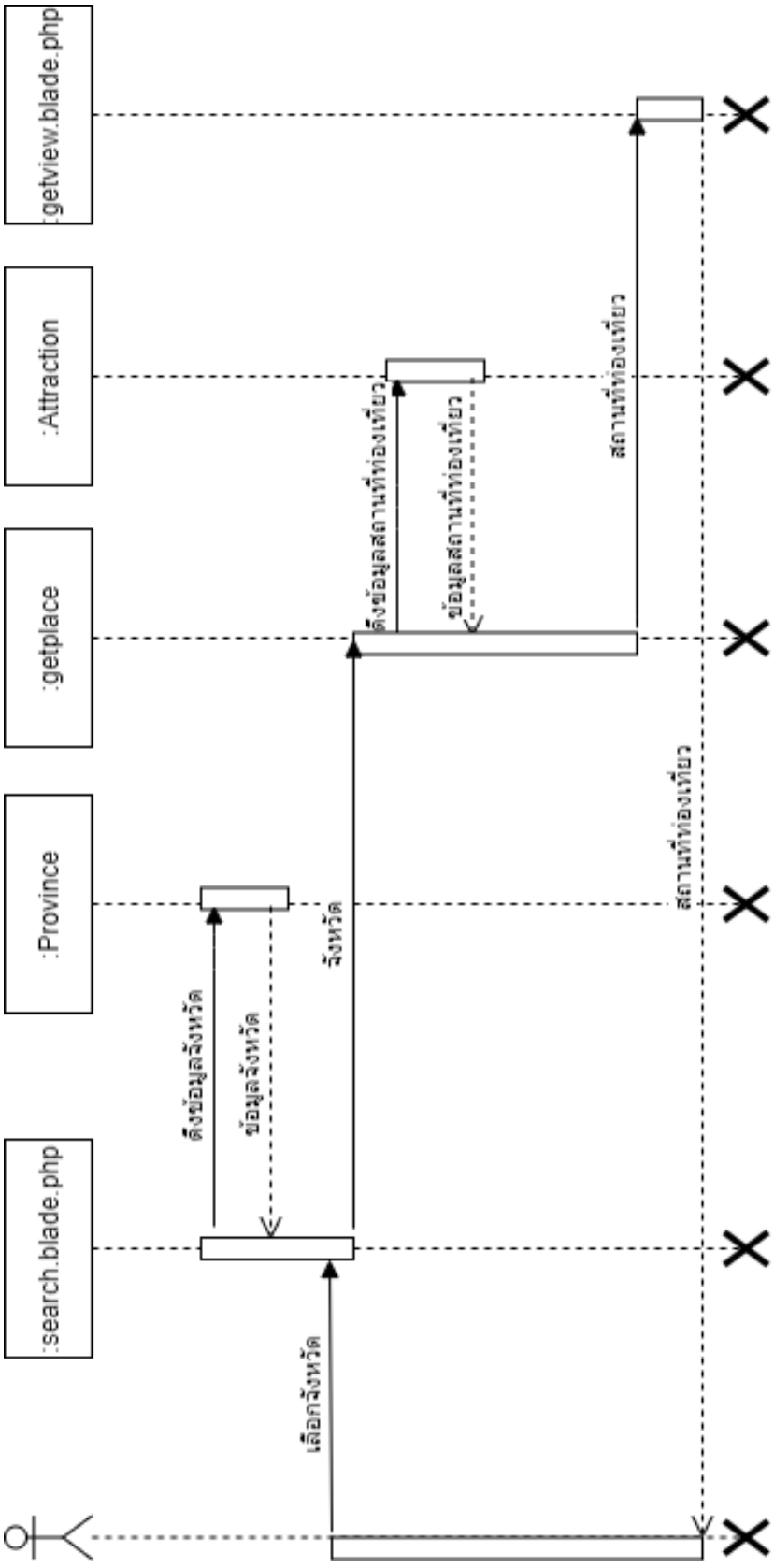
ตารางที่ 3.13: สัญลักษณ์ของ Sequence Diagram

สัญลักษณ์	การใช้งาน
	Object ที่ต้องทำหน้าที่ในการส่งหรือรับข้อมูล
	Lifeline หรือเส้นอายุขัย แสดงช่วงเวลาตั้งแต่เริ่มสร้าง object ในคลาสนั้น จนกระทั่ง object นั้นถูกทำลาย สัญลักษณ์แทนด้วยเส้นประ
	Focus of control หรือจุดควบคุม เป็นจุดควบคุมที่ object ใช้ทำการส่งหรือรับข้อความ สัญลักษณ์แทนด้วยสี่เหลี่ยม
	Message คือ ข้อความที่รับส่งระหว่าง Object สัญลักษณ์แทนด้วยลูกศรและประกอบด้วย 2 ส่วน คือ ข้อมูล (Data) และฟังก์ชัน (Function)
	Return Message เป็นข้อมูลที่ส่งกลับหลังจากทำงานเสร็จ
	Self call เป็นการเรียกฟังชันก์การทำงานภายในตัวเอง
	สร้างกรอบการทำงานของโปรแกรม เพื่อให้รู้ขอบเขตของการทำงานเช่น ลูป(loop)



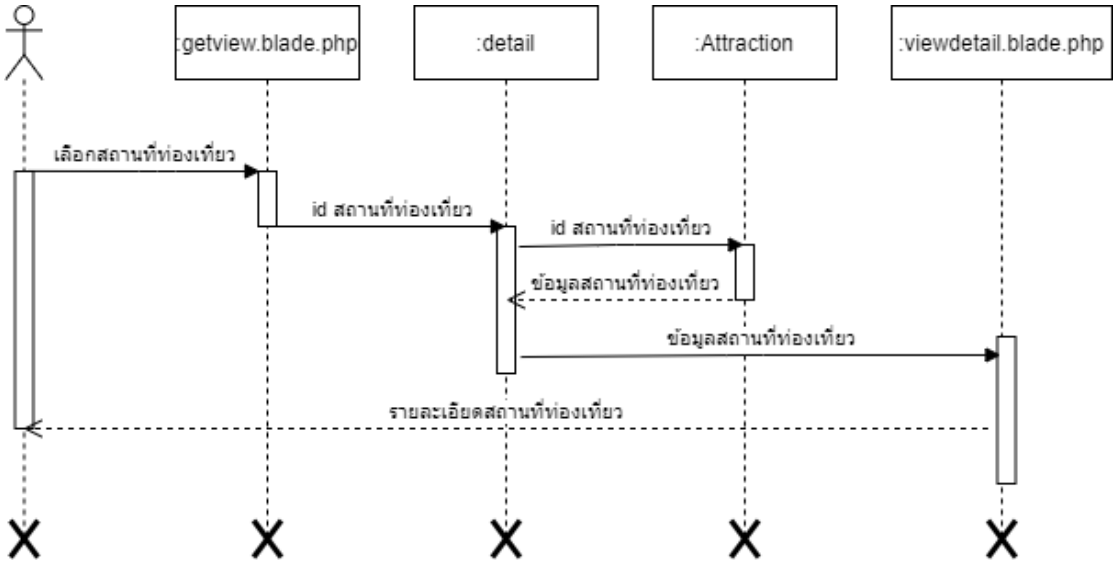
รูปที่ 3.5: Sequence Diagram เข้าสู่ระบบ

จากภาพที่ 3.5 สามารถอธิบายแผนภาพ Sequence Diagram เข้าสู่ระบบ ได้ดังนี้ เมื่อ ผู้ใช้กรอก username,password ระบบจะเรียกใช้เมธอด login() ที่ loginController ระบบจะดึงข้อมูลจาก model User เพื่อมาตรวจสอบว่า username,password ถูกต้องหรือไม่ ถ้าถูกต้องจะไปแสดงหน้า admin.blade.php ถ้าไม่ถูกต้องจะกลับไปแสดงหน้า login พร้อมแจ้งเตือนการเข้าสู่ระบบไม่สำเร็จ



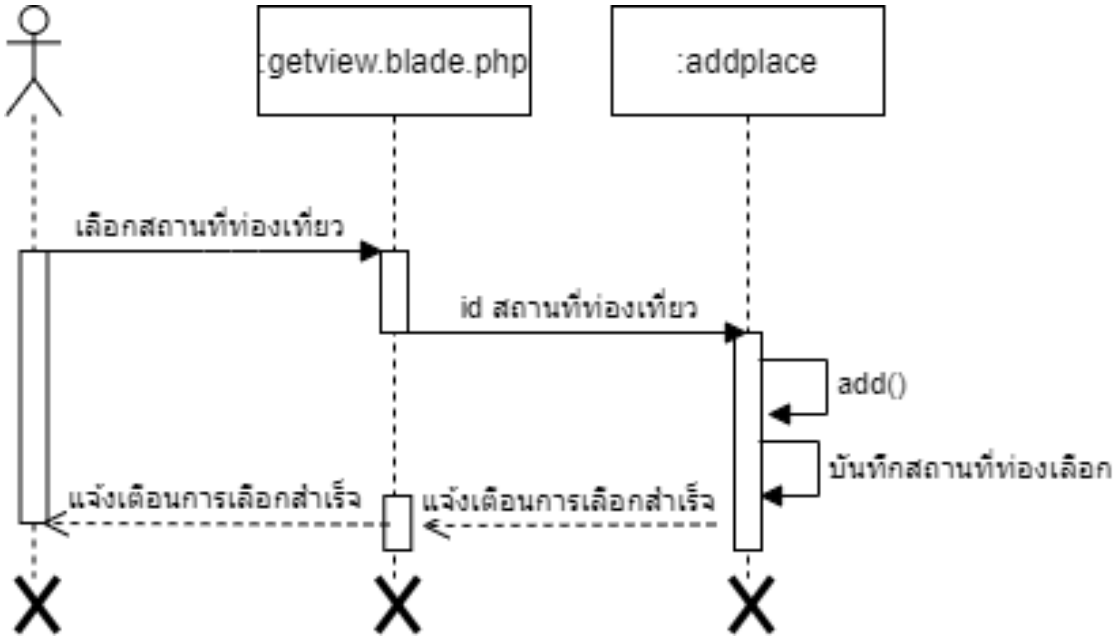
รูปที่ 3.6: Sequence Diagram ค้นหาสถานที่ท่องเที่ยว

จากภาพที่ 3.6 สามารถอธิบายแผนภาพ Sequence Diagram ค้นหาสถานที่ท่องเที่ยว ได้ดังนี้ หน้าsearch.blade.php จะทำการดึงข้อมูลจังหวัดจากโมเดล Province มาเพื่อให้ผู้ใช้งานทำการเลือกจังหวัด เมื่อผู้ใช้งานเลือกจังหวัด จะส่งค่าที่เลือกไปยัง controller getplace จากนั้น getplace จะไปดึงข้อมูลสถานที่ท่องเที่ยวจาก model Attraction และส่งข้อมูลสถานที่ท่องเที่ยวไปยังหน้า getview.blade.php เพื่อแสดงข้อมูลสถานที่ท่องเที่ยว



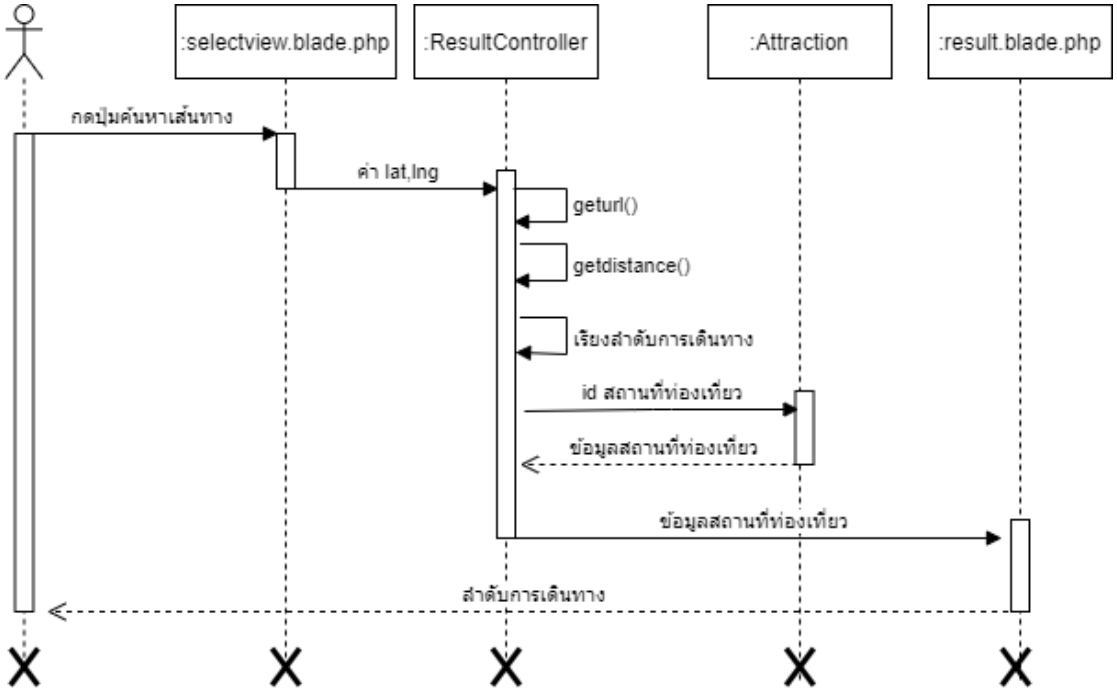
รูปที่ 3.7: Sequence Diagram ดูรายละเอียด

จากภาพที่ 3.7 สามารถอธิบายแผนภาพ Sequence Diagram ดูรายละเอียด ได้ดังนี้ เมื่อผู้ใช้งานทำการเลือกสถานที่ท่องเที่ยวที่หน้า getview.blade.php ระบบจะทำการส่ง idสถานที่ท่องเที่ยวไปยังcontroller detail จากนั้น controller detail จะทำการไปดึงข้อมูลสถานที่ท่องเที่ยวจาก model Attraction และส่งข้อมูลไปยังหน้า viewdetail.blade.php เพื่อแสดงรายละเอียดของสถานที่ท่องเที่ยวให้ผู้ใช้งาน



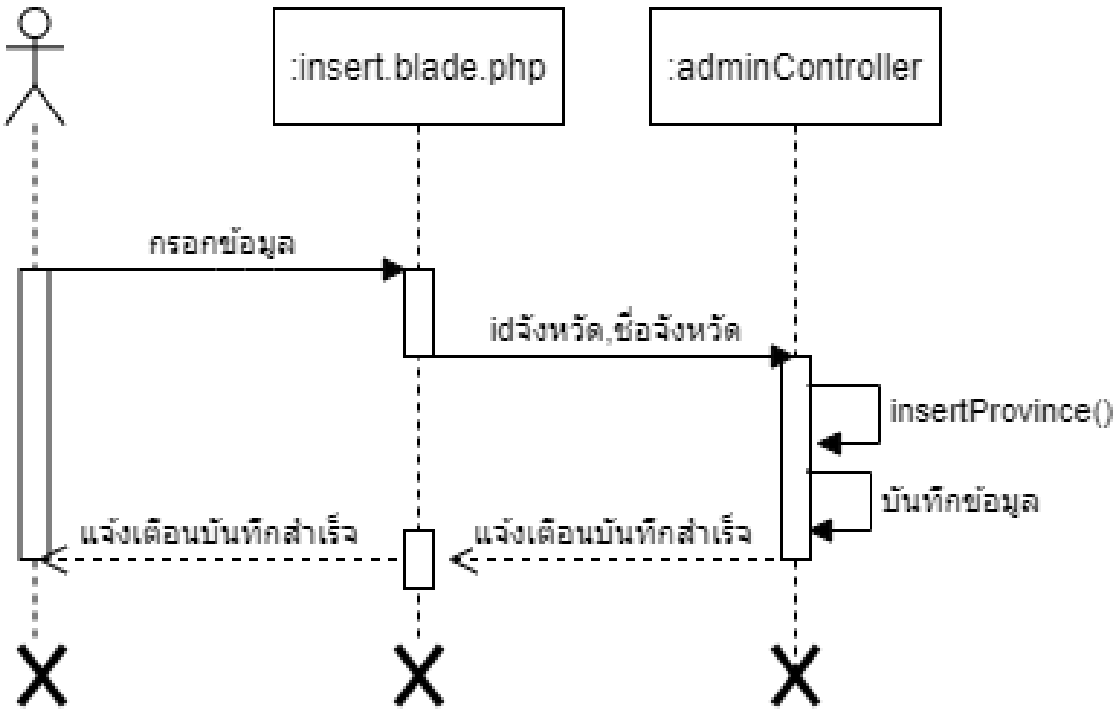
รูปที่ 3.8: Sequence Diagram เลือกสถานที่ท่องเที่ยว

จากภาพที่ 3.8 สามารถอธิบายแผนภาพ Sequence Diagram เลือกสถานที่ท่องเที่ยว ได้ดังนี้ เมื่อผู้ใช้งานทำการเลือกสถานที่ท่องเที่ยวที่หน้า getview.blade.php ระบบจะทำการส่ง idสถานที่ท่องเที่ยวไปยังcontroller addplace และเรียกใช้งานเมธอด add() จากนั้นจะทำการบันทึก idสถานที่ท่องเที่ยวที่เลือก และส่งแจ้งเตือนการเลือกสำเร็จกลับให้ผู้ใช้งาน



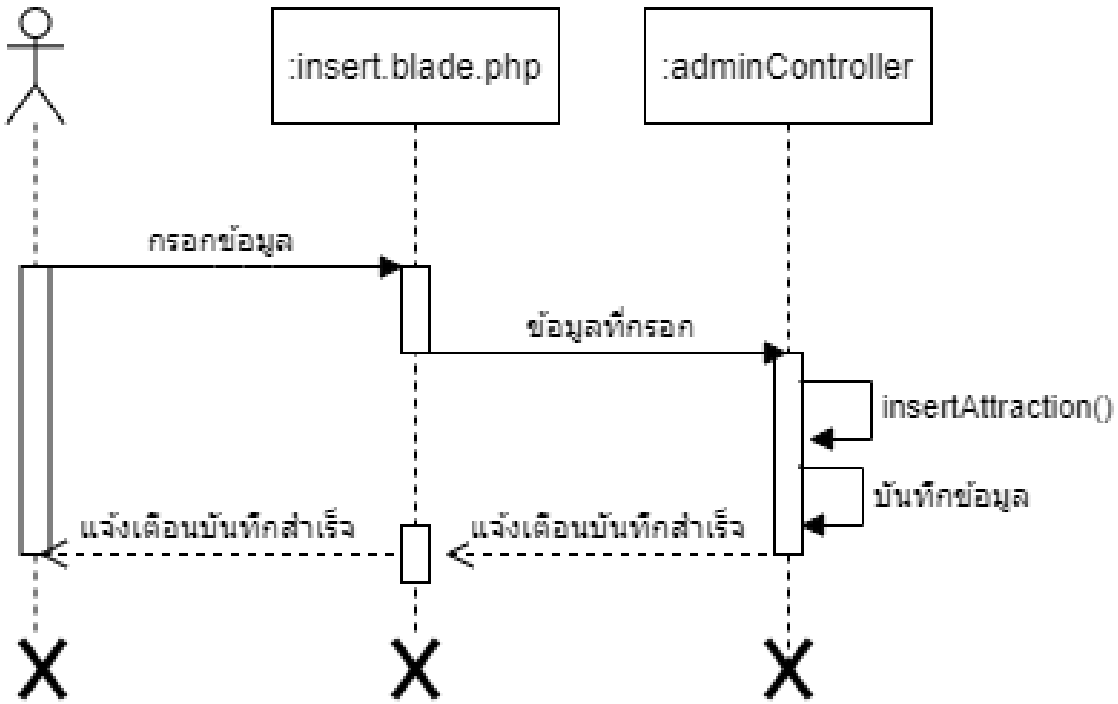
รูปที่ 3.9: Sequence Diagram ดูลำดับการเดินทาง

จากภาพที่ 3.9 สามารถอธิบายแผนภาพ Sequence Diagram ดูลำดับการเดินทาง ได้ดังนี้ เมื่อผู้ใช้งานทำการกดปุ่มค้นหาเส้นทางที่หน้า selectview.blade.php ระบบจะส่งค่า lat,lng ของตำแหน่งปัจจุบันของผู้ใช้งานไปยัง ResultController จากนั้นจะเรียกใช้งานเมธอด geturl() โดยเมธอดนี้จะทำหน้าที่สร้าง url เพื่อส่งไปหาค่าระยะทาง และเรียกใช้เมธอด getdistance() เมธอดนี้มีหน้าที่ส่ง url เพื่อหาระยะทางจาก distance matrix api จากนั้นจะทำการเรียงลำดับการเดินทาง เมื่อเรียงลำดับการเดินทางเสร็จแล้ว จะทำการดึงข้อมูลสถานที่ท่องเที่ยวจาก model Attraction และส่งไปแสดงหน้า result.blade.php เพื่อแสดงลำดับการเดินทางให้ผู้ใช้งาน



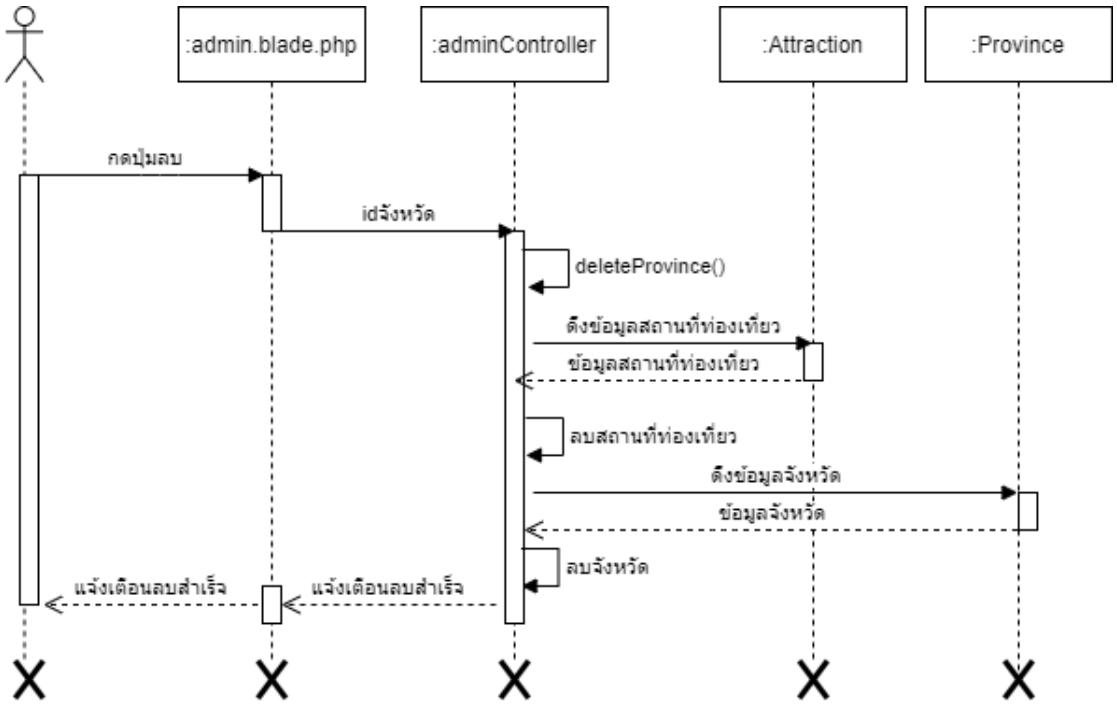
รูปที่ 3.10: Sequence Diagram เพิ่มจังหวัด

จากภาพที่ 3.10 สามารถอธิบายแผนภาพ Sequence Diagram เพิ่มจังหวัด ได้ดังนี้ เมื่อผู้ใช้กรอกข้อมูลที่หน้า insert.blade.php ระบบจะส่งข้อมูลที่กรอกไปยัง adminController และเรียกใช้งานเมธอด insertProvince() จากนั้นจะทำการบันทึกข้อมูลที่กรอกลงฐานข้อมูลและส่งแจ้งเตือนการบันทึกไปยังผู้ใช้



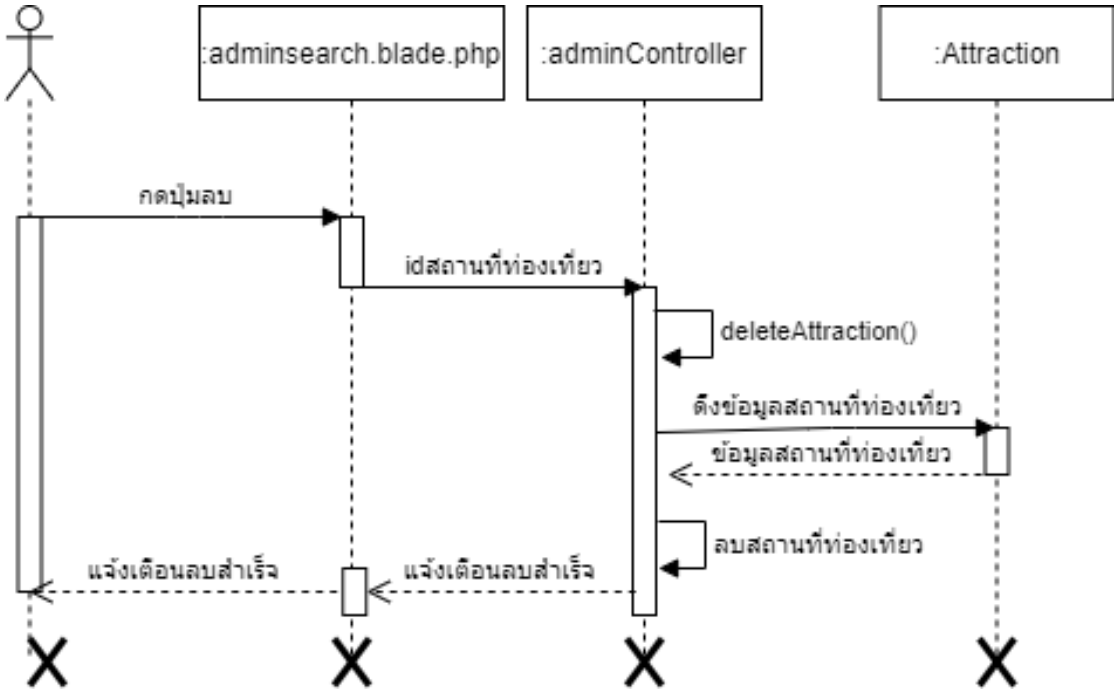
รูปที่ 3.11: Sequence Diagram เพิ่มสถานที่ท่องเที่ยว

จากภาพที่ 3.11 สามารถอธิบายแผนภาพ Sequence Diagram เพิ่มสถานที่ท่องเที่ยว ได้ดังนี้ เมื่อผู้ใช้กรอกข้อมูลที่หน้า insert.blade.php ระบบจะส่งข้อมูลที่กรอกไปยัง adminController และเรียกใช้งานเมธอด insertAttraction() จากนั้นจะทำการบันทึกข้อมูลที่กรอกลงฐานข้อมูลและส่งแจ้งเตือนการบันทึกไปยังผู้ใช้



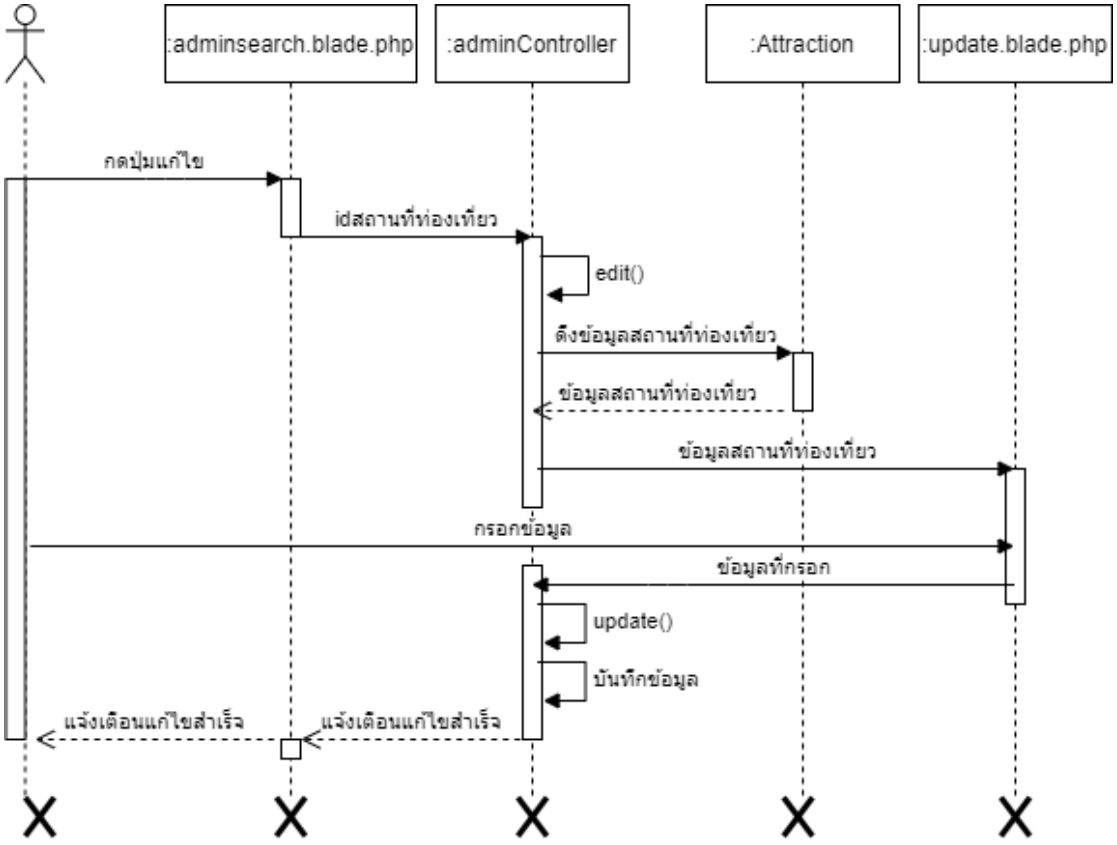
รูปที่ 3.12: Sequence Diagram ลบจังหวัด

จากภาพที่ 3.12 สามารถอธิบายแผนภาพ Sequence Diagram ลบจังหวัด ได้ดังนี้ เมื่อผู้ใช้กดปุ่มลบที่หน้า admin.blade.php ระบบจะส่ง idจังหวัดไปยัง adminController และเรียกใช้งานเมธอด deleteProvince ระบบจะทำการดึงข้อมูลสถานที่ท่องเที่ยวที่มี idจังหวัดที่ต้องการลบจาก model Attraction และทำการลบสถานที่ท่องเที่ยวนั้นออก จากนั้นจะทำการดึงข้อมูลจังหวัดที่ต้องการลบและทำการลบ และส่งแจ้งเตือนการลบไปยังผู้ใช้งาน



รูปที่ 3.13: Sequence Diagram ลบสถานที่ท่องเที่ยว

จากภาพที่ 3.13 สามารถอธิบายแผนภาพ Sequence Diagram ลบสถานที่ท่องเที่ยว ได้ดังนี้ เมื่อผู้ใช้กดปุ่มลบที่หน้า adminsearch.blade.php ระบบจะส่ง idสถานที่ท่องเที่ยวไปยัง adminController และเรียกใช้งานเมธอด deleteAttraction ระบบจะทำการดึงข้อมูลสถานที่ท่องเที่ยวที่มี idสถานที่ท่องเที่ยวที่ต้องการลบจาก model Attraction และทำการลบสถานที่ท่องเที่ยวนั้นออก จากนั้นจะทำการส่งแจ้งเตือนการลบไปยังผู้ใช้งาน



รูปที่ 3.14: Sequence Diagram แก้ไขสถานที่ท่องเที่ยว

จากภาพที่ 3.14 สามารถอธิบายแผนภาพ Sequence Diagram แก้ไขสถานที่ท่องเที่ยว ได้ดังนี้ เมื่อผู้ใช้กดปุ่มแก้ไขที่หน้า adminsearch.blade.php ระบบจะส่ง idสถานที่ท่องเที่ยวไปยัง adminController และเรียกใช้งานเมธอด edit() ระบบจะทำการดึงข้อมูลสถานที่ท่องเที่ยวที่มี idสถานที่ท่องเที่ยวเหมือนกันจาก model Attraction และส่งข้อมูลไปแสดงหน้า update.blade.php จากนั้นผู้ใช้จะทำการกรอกข้อมูลที่ต้องการแก้ไขที่หน้า update.blade.php เมื่อกดปุ่ม submit ระบบจะส่งข้อมูลที่กรอกมายัง adminController และเรียกใช้งานเมธอด update() โดยเมธอดนี้มีหน้าที่อัพเดทข้อมูล และทำการส่งแจ้งเตือนการแก้ไขสำเร็จให้ผู้ใช้

บทที่ 4

การพัฒนาระบบ

หลังจากที่ได้มีการเตรียมความพร้อมสำหรับการพัฒนาในด้านต่าง ไม่ว่าจะเป็นที่มาและความสำคัญของปัญหา เทคโนโลยีที่มีความเหมาะสมกับระบบ และการออกแบบระบบการทำงาน รวมไปถึงโครงสร้างของข้อมูล ในบทนี้จะเป็นการพูดถึงการสร้างระบบที่ได้มีการออกแบบไว้ โดยการพัฒนาระบบท่องเที่ยวอีสาน ถูกพัฒนาขึ้นด้วย Laravel Framework มีรายละเอียดดังนี้

4.1 โครงสร้างของ loginController

```
1  $user = $request->input('username');
2  $pass = $request->input('password');
3  $admin = User::where('username', 'admin')->first()->
      toArray();
4  if ($user == $admin["username"] && $pass == $admin["
      password"]) {
5  session()->put('login', true);
6  return Redirect::to('/admin');
7  } else {
8  session()->flash('loginFail', 'fail');
9  return Redirect::to('/adminlogin');
10 }
```

รูปที่ 4.1: function login

จากภาพที่ 4.1 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการรับค่าอินพุตจาก username มาเก็บในตัวแปร user
- บรรทัดที่ 2 เป็นการรับค่าอินพุตจาก password มาเก็บในตัวแปร pass
- บรรทัดที่ 3 เป็นดึงข้อมูล admin มาเก็บในตัวแปร admin
- บรรทัดที่ 4-6 เป็นการตรวจสอบการ login และอัพเดทสถานการณ์ login
- บรรทัดที่ 7-9 เป็นการอัพเดทสถานะเข้าสู่ระบบไม่สำเร็จ

```
1            if (Session::has('login')) {
2            return Redirect::to('admin');
3            } else
4            return view('login');
```

รูปที่ 4.2: function checklogin

จากภาพที่ 4.2 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-2 เป็นการตรวจสอบการ login ถ้ามีการloginแล้วให้ทำการไปหน้า admin
- บรรทัดที่ 3-4 เป็นการตรวจสอบถ้ายังไม่มีการ login ให้ไปหน้า login

```
1            $request->session()->forget('login');
2            return Redirect::to('/');
```

รูปที่ 4.3: function logout

จากภาพที่ 4.3 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-2 เป็นการลบสถานะการ login และกลับไปหน้าแรก

4.2 โครงสร้างของหน้า login

```
1  <div class="card text-white bg-dark mb-3" style="
       width: 25rem;">
2  <center>
3  <div class="card-header">
4  Login
5  </div>
6  </center>
7  <div class="card-body">
8  @if(Session::has('loginFail'))
9  <div class="alert alert-danger alert-dismissible
       fade show" role="alert">
10 username หรือpassword ไม่ถูกต้อง
11 <button type="button" class="close" data-dismiss="alert" aria-label="
       Close">
12 <span aria-hidden="true">&times;</span>
13 </button>
14 </div>
15 @else
16 @endif
17 <form action="{{route('loginController.login')}}" method="post">
18 {{csrf_field()}}
19 <div class="form-col">
20 <div class="form-group">
21 <label for="username" class="card-text">Username:</label><br>
22 <input type="text" name="username" id="username" class="form-
       control" required>
23 </div>
24 <div class="form-group">
25 <label for="password" class="card-text">Password:</label><br>
26 <input type="password" name="password" id="password" class="form-
       control">
27 </div>
28 <input type="submit" name="submit" class="btn btn-info btn-md"
       value="submit">
29 </div>
30 </form>
31 </div>
32 </div>
```

รูปที่ 4.4: ไฟล์ login.blade.php

จากภาพที่ 4.4 โครงสร้างของไฟล์ login.blade.php สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการสร้างการ์ด
- บรรทัดที่ 3 -5 เป็นการกำหนดส่วนหัวของการ์ด
- บรรทัดที่ 8 -16 เป็นการแสดงข้อความกรณีเข้าสู่ระบบไม่สำเร็จ
- บรรทัดที่ 17 -30 เป็นการสร้างฟอร์มกรอกข้อมูล
- บรรทัดที่ 20 -23 เป็นการสร้างช่องสำหรับกรอก username
- บรรทัดที่ 24 -27 เป็นการสร้างช่องกรอก password
- บรรทัดที่ 28 เป็นการสร้างปุ่ม submit

4.3 โครงสร้างหน้า admin

```
1  <script>
2  $(document).ready(function() {
3  $('.delete').on('submit', function() {
4  if (confirmยืนยันการลบ(" ?")) {
5  return true;
6  } else {
7  return false;
8  }
9  });
10 });
11 </script>
```

รูปที่ 4.5: การยืนยันการลบข้อมูล

จากภาพที่ 4.5 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 3 เป็นการตรวจสอบว่ามีการกดปุ่มหรือไม่
- บรรทัดที่ 4-5 กรณียืนยันการลบจะ return true เพื่อทำงานลบ
- บรรทัดที่ 6-7 กรณีไม่ยืนยันการลบจะ return false

```
1  @if(Session::has('success'))
2  <div class="alert alert-success alert-dismissible
       fade show" role="alert">ลบข้อมูล
       แล้ว
3
4  <button type="button" class="close" data-dismiss="alert" aria-label="
       Close">
5  <span aria-hidden="true">&times;</span>
6  </button>
7  </div>
8  @endif
9  <div class="card">
10 <table style="width:100%" class="table">
11 <tr>
12 <th scope="colรหัสจังหวัด"></th>
13 <th scope="colชื่อจังหวัด"></th>
14 <th></th>
15 </tr>
16 @foreach ($provinces as $province)
17 <tr>
18 <td>{{$province["provinces_id"]}}</td>
19 <td>{{$province["provinces_name"]}}</td>
20 <form class="delete" action="{{ route('adminController.deleteProvince')
       }}" method="post">
21 {{csrf_field()}}
22 <input type="hidden" name="province_id" value={{$province["
       provinces_id"]}}>
23 <td><input type="submit" class="btn btn-danger" valueลบ=""></td>
24 </form>
25 <form action="{{ route('adminController.search') }}" method="get">
26 <input type="hidden" name="province_id" value={{$province["
       provinces_id"]}}>
27 <td><input type="submit" class="btn btn-primary" valueค้นหา=""></td
       >
28 </form>
29 </tr>
30 @endforeach
31 </table>
32 </div>
```

รูปที่ 4.6: โครงสร้างหน้า admin

จากภาพที่ 4.6 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-8 เป็นการแสดงแจ้งเตือนการลบสำเร็จ
- บรรทัดที่ 10-31 เป็นการสร้างตาราง
- บรรทัดที่ 16-30 เป็นการแสดงรายชื่อจังหวัด
- บรรทัดที่ 20-24 เป็นการสร้างปุ่มลบ
- บรรทัดที่ 25-28 เป็นการสร้างปุ่มค้นหา

4.4 โครงสร้างของ adminController

```
1 $province = new Province([
2 'provinces_id' => $request->input('province_id'),
3 'provinces_name' => $request->input('province_name')
4 ]);
5 $province->timestamps = false;
6 $province->save();
7 session()->flash('inserted', 'success');
8 return Redirect::to('/admin/insert');
```

รูปที่ 4.7: function insertProvince

จากภาพที่ 4.7 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-4 เป็นการรับค่าที่อินพุตเข้ามา
- บรรทัดที่ 5 เป็นคำสั่งกำหนดว่าไม่ทำการบันทึกเวลาการเพิ่มข้อมูล
- บรรทัดที่ 6 เป็นการบันทึกข้อมูล
- บรรทัดที่ 7-8 เป็นการส่งแจ้งเตือนการเพิ่มสำเร็จ

```
1 $attraction_id = $request->input('attraction_id');
2 $attractions = Attraction::where('attractions_id',
     $attraction_id)->get()->toArray();
3 return view('update')->with('attractions',
     $attractions);
```

รูปที่ 4.8: function edit

จากภาพที่ 4.8 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการรับค่าจากอินพุต
- บรรทัดที่ 2 เป็นการดึงข้อมูลสถานที่ท่องเที่ยวที่มี id เท่ากับค่าอินพุต
- บรรทัดที่ 3 เป็นการส่งค่าที่ได้ไปแสดงหน้าอัพเดท

```
1 $province_id = $request->input('province_id');
2 $attractions = Attraction::where('provinces_id',
     $province_id)->get()->toArray();
3 return view('adminsearch')
4 ->with('attractions', $attractions);
```

รูปที่ 4.9: function search

จากภาพที่ 4.9 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการรับค่าจากอินพุต
- บรรทัดที่ 2 เป็นการดึงข้อมูลสถานที่ท่องเที่ยวที่มี id เท่ากับค่าอินพุต
- บรรทัดที่ 3 เป็นการส่งค่าที่ได้ไปแสดงหน้า adminsearch

```
1  $attraction_id = $request->input('attraction_id');
2  $data = Attraction::where('attractions_id',
      $attraction_id)->get()->toArray();
3  $data_img = $data[0]['image_url'];
4  $pics = explode("|",$data_img);
5  foreach ($pics as $pic) {
6  if(file_exists(public_path('upload/'.$pic))){
7  unlink(public_path('upload/'.$pic));
8  }}
9  $attraction = Attraction::where('attractions_id',
      $attraction_id);
10 $attraction->delete();
11 session()->flash('success', 'success');
12 return Redirect::back();
```

รูปที่ 4.10: function deleteAttraction

จากภาพที่ 4.10 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการรับค่าจากอินพุต

- บรรทัดที่ 2-3 เป็นการดึงข้อรูปภาพของสถานที่
- บรรทัดที่ 4 เป็นการแยก string
- บรรทัดที่ 5-8 เป็นการลบรูปภาพ
- บรรทัดที่ 9-10 เป็นการลบข้อมูลในฐานข้อมูล

```
1  $this->validate($request,[
2  'image' => 'required',
3  'image.*' => 'image|mimes:jpeg,png,jpg|max:2048'
4  ]);
5  $attraction_id = $request->input('attractions_id');
6  $data = Attraction::where('attractions_id',
       $attraction_id)->get()->toArray();
7  $data_img = $data[0]['image_url'];
8  $pics = explode("|",$data_img);
9  foreach ($pics as $pic) {
10 if(file_exists(public_path('upload/'.$pic))){
11 unlink(public_path('upload/'.$pic));
12 }
13 }
14 if ($request->hasFile('image')) {
15 $files = $request->file('image');
16 foreach($files as $file){
17 $filename=$file->getClientOriginalName();
18 $file->move('upload',$filename);
19 $images[]=$filename;
20 }
21 $data=implode("|",$images);
22 }
23 Attraction::where('attractions_id', $attraction_id)
24 ->update([
25 'attractions_name' => $request->input('
       attractions_name'),
26 'Latitude' => $request->input('lat'),
27 'longitude' => $request->input('lng'),
28 'description' => $request->input('description'),
29 'image_url' => $data
30 ]);
31 session()->flash('success', 'success');
32 return Redirect::back();
```

รูปที่ 4.11: function update

จากภาพที่ 4.11 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-4 เป็นการตรวจสอบไฟล์ที่อินพุตเข้ามา
- บรรทัดที่ 5 เป็นการรับค่าจากอินพุต
- บรรทัดที่ 6 เป็นการดึงข้อมูลสถานที่ท่องเที่ยวจากฐานข้อมูล
- บรรทัดที่ 7 เป็นการดึงรูปภาพจากฐานข้อมูล
- บรรทัดที่ 8 เป็นการแยก string
- บรรทัดที่ 9-13 เป็นการลบรูปภาพใน folder upload
- บรรทัดที่ 14-21 เป็นการเพิ่มรูปภาพใน folder upload
- บรรทัดที่ 23-30 เป็นการอัพเดทข้อมูลลงฐานข้อมูล

```
1  $province_id = $request->input('province_id');
2  $attraction = Attraction::where('provinces_id',
       $province_id)->get()->toArray();
3  for ($i=0; $i < count($attraction); $i++) {
4  $data = $attraction[$i];
5  $data_img = $data['image_url'];
6  $pics = explode("|",$data_img);
7  foreach ($pics as $pic) {
8  if(file_exists(public_path('upload/'.$pic))){
9  unlink(public_path('upload/'.$pic));
10 }
11 }
12 }
13 $attractions = Attraction::where('provinces_id',
       $province_id);
14 $attractions->delete();
15 $province = Province::where('provinces_id',
       $province_id);
16 $province->delete();
17 session()->flash('success', 'success');
18 return Redirect::to('/admin');
```

รูปที่ 4.12: function deleteProvince

จากภาพที่ 4.12 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการรับค่า province_id จากอินพุต

- บรรทัดที่ 2 เป็นการดึงข้อมูลสถานที่ท่องเที่ยวจากฐานข้อมูล
- บรรทัดที่ 3-12 เป็นการลบข้อมูลรูปภาพใน folder upload
- บรรทัดที่ 13-14 เป็นการลบข้อมูลสถานที่ท่องเที่ยวออกจากฐานข้อมูล
- บรรทัดที่ 15-16 เป็นการลบข้อมูลจังหวัดอกจากฐานข้อมูล
- บรรทัดที่ 17-18 เป็นการส่งแจ้งเตือนการลบสำเร็จพร้อมกลับไปยังหน้า admin

```
1  $this->validate($request,[
2  'image' => 'required',
3  'image.*' => 'image|mimes:jpeg,png,jpg,gif,svg|max
       :2048'
4  ]);
5  if ($request->hasFile('image')) {
6  $files = $request->file('image');
7  foreach($files as $file){
8  $filename=$file->getClientOriginalName();
9  $file->move('upload',$filename);
10 $images[]=$filename;
11 }
12 $data=implode("|",$images);
13 }
14 $attraction = new Attraction([
15 'provinces_id' => $request->input('province_id'),
16 'attractions_name' => $request->input('
       attractions_name'),
17 'Latitude' => $request->input('lat'),
18 'longitude' => $request->input('lng'),
19 'description' => $request->input('description'),
20 'image_url' => $data
21 ]);
22 $attraction->timestamps = false;
23 $attraction->save();
24 session()->flash('inserted', 'success');
25 return Redirect::to('/admin/insert');
```

รูปที่ 4.13: function insertAttraction

จากภาพที่ 4.13 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-4 เป็นการตรวจสอบไฟล์ที่อินพุตเข้ามา
- บรรทัดที่ 5-13 เป็นการเพิ่มรูปภาพลงใน folder upload

- บรรทัดที่ 14-21 เป็นการกำหนดค่าที่จะบันทึกลงในฐานข้อมูล
- บรรทัดที่ 22 เป็นการกำหนดค่าไม่บันทึกเวลาลงข้อมูล
- บรรทัดที่ 23 เป็นการบันทึกข้อมูล
- บรรทัดที่ 24-25 เป็นการส่งแจ้งเตือนการบันทึกสำเร็จ

4.5 โครงสร้างของหน้า adminsearch

```
1  <nav class="navbar navbar-dark bg-dark">
2  <a href="/" class="navbar-brand">Travel</a>
3  <form class="form-inline">
4  <div class="dropdown">
5  <button class="btn btn-secondary dropdown-toggle"
     type="button" id="dropdownMenuButton" data-toggle
     ="dropdown" aria-haspopup="true" aria-expanded="
     false">
6  Admin
7  </button>
8  <div class="dropdown-menu" aria-labelledby="
     dropdownMenuButton">
9  <a href="/admin" class="dropdown-itemหน้าแรก"></a>
10 <a class="dropdown-item" href="/admin/insertเพิ่มข้อมูล"></a>
11 <div class="dropdown-divider"></div>
12 <a class="dropdown-item" href="/admin/logout">Log out</a>
13 </div>
14 </div>
15 </form>
16 </nav>
17 @if(Session::has('success'))
18 <div class="alert alert-success alert-dismissible fade show" role="alert
     ">
19 success
20 <button type="button" class="close" data-dismiss="alert" aria-label="
     Close">
21 <span aria-hidden="true">&times;</span>
22 </button>
23 </div>
24 @endif
```

รูปที่ 4.14: การสร้าง navbar หน้า adminsearch

จากภาพที่ 4.15 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-16 เป็นการสร้าง navbar
- บรรทัดที่ 2 เป็นการสร้างลิงค์ไปหน้าค้นหาสถานที่ท่องเที่ยว
- บรรทัดที่ 4-14 เป็นการ dropdown
- บรรทัดที่ 17-24 เป็นการตรวจสอบสถานะการลบ ถ้าลบสำเร็จจะทำการแสดงข้อความ success

```
1  <div class="card">
2  @if($attractions!=null)
3  <table style="width:100%" class="table">
4  <tr>
5  <th scope="colรหัสสถานที่ท่องเที่ยว"></th>
6  <th scope="colชื่อสถานที่ท่องเที่ยว"></th>
7  <th></th>
8  </tr>
9  @foreach ($attractions as $attraction)
10 <tr>
11 <td>{{$attraction["attractions_id"]}}</td>
12 <td>{{$attraction["attractions_name"]}}</td>
13 <form action="{{route('adminController.deleteAttraction')}}" method="
       post">
14 {{csrf_field()}}
15 <input type="hidden" name="attraction_id" value={{$attraction["
       attractions_id"]}}>
16 <td><input type="submit" class="btn btn-danger" valueลบ=""></td>
17 </form>
18 <form action="{{route('adminController.edit')}}" method="get">
19
20 <input type="hidden" name="attraction_id" value={{$attraction["
       attractions_id"]}}>
21 <td><input type="submit" class="btn btn-warning" valueแก้ไข=""></td
       >
22 </form>
23 </tr>
24 @endforeach
25 </table>
26 @else
27 <pไม่มีข้อมูล></p>
28 @endif
29 </div>
```

รูปที่ 4.15: adminsearch.blade.php

จากภาพที่ 4.15 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 2 เป็นการตรวจสอบว่ามีข้อมูลหรือไม่ ถ้าไม่มีข้อมูลจะทำงานบรรทัดที่ 27
- บรรทัดที่ 3-25 เป็นการสร้างตาราง
- บรรทัดที่ 9-24 เป็นการวนลูปเพื่อแสดงชื่อสถานที่ท่องเที่ยวลงในตาราง

- บรรทัดที่ 13-17 เป็นการสร้างปุ่มลบข้อมูลสถานที่ท่องเที่ยว
- บรรทัดที่ 18-22 เป็นการสร้างปุ่มแก้ไขข้อมูลสถานที่ท่องเที่ยว
- บรรทัดที่ 27 เป็นการแสดงข้อความไม่มีข้อมูล

4.6 โครงสร้างของหน้า getview

```
1  <div class="row">
2  @foreach($places as $place)
3  <div class="card text-white bg-secondary mb-3" style
     ="width: 18rem;">
4  <?php
5  $pic= explode("|",$place['image_url']);
6  ?>
7  <img class="card-img-top" src="../upload/{{$pic
     [0]}}" alt="Card image cap">
8  <div class="card-body">
9  <h5 class="card-title">{{$place["attractions_name
     "]}}</h5>
10 <div class="row">
11 <a href="/detail_/{{$place["attractions_id"]}}"
     class="btn btn-primaryราย
     ละเอียด"></a>
12 <form action="{{ route('addplace.add') }}" method="get">
13 <!--{{csrf_field()}}-->
14 <input type= "hidden" name= "id" value="{{$place["attractions_id"]}}">
15 <input onclick="success()" type= "submit" class="btn btn-primary"
     value= เลือก"">
16 </form>
17 </div>
18 </div>
19 </div>
20 @endforeach
21 </div>
```

รูปที่ 4.16: getview.blade.php

จากภาพที่ 4.16 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 2-20 เป็นการวนลูปเพื่อนำข้อมูลมาแสดง
- บรรทัดที่ 4-6 เป็นการแยก string

- บรรทัดที่ 7 เป็นการแสดงรูปภาพ
- บรรทัดที่ 5 เป็นการแสดงชื่อสถานที่ท่องเที่ยว
- บรรทัดที่ 11 เป็นการสร้างปุ่มรายละเอียด
- บรรทัดที่ 12-16 เป็นการสร้างปุ่มเลือก

4.7 โครงสร้างของหน้า insert

```
1  <script type="text/javascript">
2  function setForm(value) {
3  if (value == 'form1') {
4  document.getElementById('form1').style = 'display:
     block;';
5  document.getElementById('form2').style = 'display:
     none;';
6  } else {
7  document.getElementById('form2').style = 'display:
     block;';
8  document.getElementById('form1').style = 'display:
     none;';
9  }
10 }
11 </script>
```

รูปที่ 4.17: การสลับ form กรอกข้อมูล

จากภาพที่ 4.17 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 3-5 เป็นการตรวจสอบถ้าค่าเท่ากับ form1 ให้แสดงform1
- บรรทัดที่ 6-8 เป็นการแสดง form2

```
1 <div class="card-header">
2 <label>เพิ่มข้อมูล</label>
3 <select class="custom-select" onchange="setForm(this.value)" style="
      width: 25rem;">
4 <option value="form1สถานที่ท่องเที่ยว"></option>
5 <option value="form2จังหวัด"></option>
6 </select>
7 </div>
```

รูปที่ 4.18: การสร้างเมนู select หน้า insert

จากภาพที่ 4.18 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 2 เป็นการแสดงข้อความ เพิ่มข้อมูล
- บรรทัดที่ 3-7 เป็นการสร้างปุ่ม select

```
1 <div id="form2" style="display: none">
2 <form action="{{route('adminController.
     insertProvince')}}" method="post">
3 {{csrf_field()}}
4 <div class="col-md-4 mb-3">
5 <label>รหัสจังหวัด</label>
6 <input type="number" class="form-control" name="province_id"
      required></div>
7 <div class="col-md-4 mb-3">
8 <label>ชื่อจังหวัด</label>
9 <input type="text" class="form-control" name="province_name"
      required></div>
10 <input type="submit" name="submit" class="btn btn-info btn-md"
      value="submit"></form></div>
```

รูปที่ 4.19: form2 หน้า insert

จากภาพที่ 4.19 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 4-6 เป็นการสร้างช่องกรอกข้อมูลรหัสจังหวัด
- บรรทัดที่ 7-9 เป็นการสร้างช่องกรอกข้อมูลชื่อจังหวัด
- บรรทัดที่ 10 เป็นการสร้างปุ่ม submit

```
<div id="form1">
<form action="{{route('adminController.
   insertAttraction')}}" method="post" enctype="
   multipart/form-data">
{{csrf_field()}}
<div class="form-col">
<div class="col-md-4 mb-3">
<labelจังหวัด></label>
<select name="province_id" class="form-control">
@foreach ($provinces as $province)
<option value={{$province["provinces_id"]}}>{{$province["
    provinces_name"]}}</option>
@endforeach
</select>
</div>
<div class="col-md-4 mb-3">
<labelชื่อสถานที่ท่องเที่ยว></label>
<input type="text" class="form-control" name="attractions_name"
    required>
</div>
<div class="col-md-4 mb-3">
<label>Latitude</label>
<input type="number" step="any" class="form-control" name="lat"
    required>
</div>
<div class="col-md-4 mb-3">
<label>Longtitud</label>
<input type="number" step="any" class="form-control" name="lng"
    required>
</div>
<div class="col-md-4 mb-3">
<labelรายละเอียด></label>
<textarea class="form-control " name="description" required></
    textarea>
</div>
<div class="col-md-4 mb-3">
<labelรูปภาพ></label>
<input type="file" name="image[]" multiple="multiple" required
    multiple>
</div>
<input type="submit" name="submit" class="btn btn-info btn-md"
    value="submit">
</div>
</form>
</div>
```

รูปที่ 4.20: form1 หน้า insert

จากภาพที่ 4.20 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 5-12 เป็นการสร้างเมนู select
- บรรทัดที่ 8-10 เป็นการสร้าง option แสดงชื่อจังหวัด
- บรรทัดที่ 14-16 เป็นการสร้างช่องกรอกข้อมูลชื่อสถานที่ท่องเที่ยว
- บรรทัดที่ 17-20 เป็นการสร้างช่องกรอกข้อมูลละติจูด
- บรรทัดที่ 21-24 เป็นการสร้างช่องกรอกข้อมูลลองจิจูด
- บรรทัดที่ 25-28 เป็นการสร้างช่องกรอกข้อมูลรายละเอียด
- บรรทัดที่ 29-32 เป็นการสร้างที่เพิ่มรูปภาพ
- บรรทัดที่ 33 เป็นการสร้างปุ่ม submit

4.8 โครงสร้างของหน้า result

```
1  <div class="card">
2  <div class="card-header">ลำดับการเดินทาง
3
4  </div>
5  <ul class="list-group list-group-flush">
6  <?php
7  $p=array('B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K');
8  $i=0;
9  ?>
10 <li class="list-group-item">A ตำแหน่งปัจจุบัน</li>
11 @foreach($dataresults as $data)
12 <li class="list-group-item">{{$p[$i]}} {{$data["attractions_name"]}}</li>
13 <?php $i++; ?>
14 @endforeach
15 </ul>
16 </div>
```

รูปที่ 4.21: การแสดงลำดับการเดินทาง

จากภาพที่ 4.21 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 6-9 เป็นการสร้าง array เพื่อนำไปใช้ในการบอกตำแหน่งของสถานที่ท่องเที่ยว
- บรรทัดที่ 11-14 เป็นการวนลูปเพื่อแสดงข้อมูลลำดับการเดินทาง

```
1  function initMap() {
2  var directionsService = new google.maps.
     DirectionsService();
3  var directionsRenderer = new google.maps.
     DirectionsRenderer();
4  var map = new google.maps.Map(document.
     getElementById('map'), {
5  center: {lat: 13.684164, lng: 100.709522},
6  zoom: 6
7  });
8  directionsRenderer.setMap(map);
9  navigator.geolocation.getCurrentPosition(function(
     position) {
10 var pos = {lat: position.coords.latitude,lng:
     position.coords.longitude};
11 map.setCenter(pos);
12 map.setZoom(15);
13
14 DisplayRoute(directionsService, directionsRenderer,
     pos);
15 });
16 }
```

รูปที่ 4.22: initMap

จากภาพที่ 4.22 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 2-3 เป็นการเรียกใช้งานการค้นหาเส้นทางและการแสดงผลเส้นทาง
- บรรทัดที่ 4-6 เป็นการกำหนดค่าให้แผนที่
- บรรทัดที่ 8 เป็นการสร้างแผนที่ ที่กำหนดค่าเอาไว้
- บรรทัดที่ 9-12 เป็นการหาค่า lat,lng ของตำแหน่งปัจจุบันของผู้ใช้
- บรรทัดที่ 14 เป็นการเรียกใช้งาน DisplayRoute

```
1  function DisplayRoute(directionsService,
       directionsRenderer,pos) {
2  var waypts = [];
3  <?php
4  for ($i=0; $i <= count($dataresults)-2; $i++) {
5  echo "waypts.push({";
6  echo "location: {lat:".$dataresults[$i]["Latitude
       "]."},lng:".$dataresults[$i]["longitude"]."},";
7  echo "stopover: true";
8  echo "});";
9  }
10 ?>
11 var posdes = {lat: <?php echo $dataresults[count(
       $dataresults)-1]["Latitude"]; ?>,lng: <?php echo
       $dataresults[count($dataresults)-1]["longitude"];
        ?>};
12 directionsService.route(
13 {
14 origin: pos,
15 destination: posdes,
16 waypoints: waypts,
17 travelMode: 'DRIVING'
18 },
19 function(response, status) {
20 if (status === 'OK') {
21 directionsRenderer.setDirections(response);
22 } else {
23 window.alert('Directions request failed due to ' +
       status);
24 }
25 });
26 }
```

รูปที่ 4.23: DisplayRoute

จากภาพที่ 4.23 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 3-10 เป็นการนำผลลัพธ์เก็บไว้ใน waypts
- บรรทัดที่ 11 เป็นการนำผลลัพธ์ตำแหน่งสุดท้ายเก็บไว้ในตัวแปร posdes
- บรรทัดที่ 12-25 เป็นการกำหนดค่าเพื่อใช้ในการแสดงเส้นทาง

4.9 โครงสร้างหน้า search

```
1  <?php
2  use App\Province;
3  $provinces = Province::all();
4  ?>
5  <center>
6  <div class="container">
7  <form action="{{ route('getplace.getdata') }}"
       method="get">
8  <div class="form-group">
9  <select name="province" class="form-control">
10 @foreach ($provinces as $province)
11 <option value={{$province["provinces_id"]}} >{{
       $province["provinces_name"]}}</option>
12
13 @endforeach
14 </select>
15 </div>
16 <input type= "submit" class="btn btn-primary" value=
       ค้นหา"">
17 </form>
18 </div>
19 </center>
20 <a class="btn btn-secondary" href="/adminlogin">admin</a>
```

รูปที่ 4.24: search.blade.php

จากภาพที่ 4.24 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-4 เป็นการดึงข้อมูลจังหวัดจากฐานข้อมูลมาเก็บในตัวแปร provinces
- บรรทัดที่ 7-17 เป็นการสร้าง form เพื่อส่งค่าที่ผู้ใช้เลือก
- บรรทัดที่ 9-14 เป็นการสร้างเมนู select
- บรรทัดที่ 10-13 เป็นการกำหนดค่า option
- บรรทัดที่ 16 เป็นการสร้างปุ่มค้นหา

4.10 โครงสร้างหน้า selectview

```
1  <nav class="navbar navbar-dark bg-dark">
2  <a href="/" class="navbar-brand">Travel</a>
3  <div class="form-inline">
4  <a class="btn btn-danger" href="/delAllSelectลบที่เลือก
       ทั้งหมด"></a>
5  <form action="{{ route('ResultController.getresult') }}" method="post">
6  {{csrf_field()}}
7  <input type= "hidden" name= "lat" id= "lat" >
8  <input type= "hidden" name= "lng" id= "lng" >
9  <input type= "submit" class="btn btn-primary" value= ค้นหาเส้นทาง"">
10 </form>
11 </div>
12 </nav>
13 <div class="row">
14 @foreach($selecteds as $selected)
15 <div class="card text-white bg-secondary mb-3" style="width: 18rem
       ;">
16 <img class="card-img-top" src="{{$selected['image_url']}}" alt="Card
       image cap">
17 <div class="card-body">
18 <h5 class="card-title">{{$selected["attractions_name"]}}</h5>
19 </div>
20 <div class="card-body">
21 <form action="{{ route('addplace.del') }}" method="get">
22 <input type= "hidden" name= "id" value="{{$selected["attractions_id
       "]}}">
23 <input type= "submit" class="btn btn-danger" value= ลบ"">
24 </form>
25 </div>
26 </div>
27 @endforeach
28 </div>
```

รูปที่ 4.25: การแสดงสถานที่ท่องเที่ยวที่เลือก

จากภาพที่ 4.26 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-12 เป็นการสร้าง navbar
- บรรทัดที่ 4 เป็นการสร้างปุ่ม ลบที่เลือกทั้งหมด

- บรรทัดที่ 5-10 เป็นการสร้าง form ส่งข้อมูล lat,lng
- บรรทัดที่ 14-27 เป็นการแสดงสถานที่ท่องเที่ยวที่ผู้ใช้เลือก
- บรรทัดที่ 16 เป็นการแสดงรูปภาพ
- บรรทัดที่ 18 เป็นการแสดงชื่อสถานที่ท่องเที่ยวที่เลือก
- บรรทัดที่ 21-24 เป็นการสร้างปุ่มลบ

```
1  <script>
2  var lat = document.getElementById(”lat”);
3  var lng = document.getElementById(”lng”);
4  function getCurrent() {
5  navigator.geolocation.getCurrentPosition(function(
     position) {
6  lat.value = position.coords.latitude;
7  lng.value = position.coords.longitude;
8  });
9  }
10 </script>
11 <script async defer
12 src=”https://maps.googleapis.com/maps/api/js?key=
     AIzaSyAQLj-_PEe0qXFXtqhs_EdE-ZmC5zoReMs&callback=
     getCurrent”>
13 </script>
```

รูปที่ 4.26: การหาค่า lat,lng ของตำแหน่งผู้ใช้งาน

จากภาพที่ 4.26 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 2-3 เป็นการสร้างตัวแปร
- บรรทัดที่ 4-9 เป็นการหาค่า lat,lng ของตำแหน่งผู้ใช้งาน
- บรรทัดที่ 11-13 เป็นการเรียกใช้งานบริการของ google

4.11 โครงสร้างของหน้า update

```
<div class="card">
<div class="card-body">
<form action="{{route('adminController.update')}}"
   method="post" enctype="multipart/form-data">
{{csrf_field()}}
<div class="form-col">
<input type="hidden" name="attractions_id" value="{{
   $attraction['attractions_id']}}">
<div class="col-md-4 mb-3">
<labelชื่อสถานที่ท่องเที่ยว></label>
<input type="text" class="form-control" name="attractions_name"
    placeholder="{{$attraction['attractions_name']}}" value="{{$attraction
    ['attractions_name']}}" required>
</div>
<div class="col-md-4 mb-3">
<label>Latitude</label>
<input type="number" step="any" class="form-control" name="lat"
    placeholder="{{$attraction['Latitude']}}" value="{{$attraction['
    Latitude']}}" required>
</div>
<div class="col-md-4 mb-3">
<label>Longtitud</label>
<input type="number" step="any" class="form-control" name="lng"
    placeholder="{{$attraction['longitude']}}" value="{{$attraction['
    longitude']}}" required>
</div>
<div class="col-md-4 mb-3">
<labelรายละเอียด></label>
<textarea class="form-control " name="description" required>{{
    $attraction['description']}}</textarea>
</div>
<div class="col-md-4 mb-3">
<labelรูปภาพ></label>
<input type="file" name="image[]" multiple="multiple" required
    multiple>
</div>
<input type="submit" name="submit" class="btn btn-info btn-md"
    value="submit">
</div>
</form>
</div>
</div>
```

รูปที่ 4.27: โครงสร้างของหน้า update

จากภาพที่ 4.27 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 7-10 เป็นการสร้างช่องกรอกข้อมูลชื่อสถานที่ท่องเที่ยว
- บรรทัดที่ 11-14 เป็นการสร้างช่องกรอกข้อมูลละติจูด
- บรรทัดที่ 15-18 เป็นการสร้างช่องกรอกข้อมูลลองจิจูด
- บรรทัดที่ 19-22 เป็นการสร้างช่องกรอกข้อมูลรายละเอียด
- บรรทัดที่ 23-26 เป็นการสร้างที่เพิ่มรูปภาพ
- บรรทัดที่ 27 เป็นการสร้างปุ่ม submit

```
1 @if (count($errors) > 0)
2 <div class="alert alert-danger">
3 <ul>
4 @foreach ($errors->all() as $error)
5 <li>{{ $error }}</li>
6 @endforeach
7 </ul>
8 </div>
9 @endif
```

รูปที่ 4.28: การแสดงผล error

จากภาพที่ 4.28 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการตรวจสอบถ้ามี error มากกว่า 0 ให้ทำงานบรรทัดที่ 2-8
- บรรทัดที่ 4-6 เป็นการวนลูปเพื่อแสดงผล error

4.12 โครงสร้างหน้า viewdetail

```
<?php
$pics= explode("|",$detail['image_url']);
?>
<center>
<div class="card">
<div class="card-body">
<div id="carouselExampleControls" class="carousel slide" data-ride="carousel">
<div class="carousel-inner">
<?php
for ($i=0; $i < count($pics) ; $i++) {
if ($i==0) {
echo "<div class='carousel-item active'>";
echo "<img class='d-block w-100' src='../upload/$pics[$i]' >";
echo "</div>";
}else {
echo "<div class='carousel-item'>";
echo "<img class='d-block w-100' src='../upload/$pics[$i]' >";
echo "</div>";
}
} ?>
</div>
<a class="carousel-control-prev" href="#carouselExampleControls" role="button" data-slide="prev">
<span class="carousel-control-prev-icon" aria-hidden="true"></span>
<span class="sr-only">Previous</span>
</a>
<a class="carousel-control-next" href="#carouselExampleControls" role="button" data-slide="next">
<span class="carousel-control-next-icon" aria-hidden="true"></span>
<span class="sr-only">Next</span>
</a>
</div>
</div>
</div>
</center>
```

รูปที่ 4.29: โครงสร้างหน้า viewdetail

จากภาพที่ 4.29 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1-3 เป็นการแยก string
- บรรทัดที่ 9-20 เป็นการแสดงรูปภาพ
- บรรทัดที่ 22-25 เป็นการสร้างปุ่มดูรูปภาพก่อนหน้า
- บรรทัดที่ 26-29 เป็นการสร้างปุ่มดูรูปภาพต่อไป

```
1  <h3 class="text-left">{{$detail['attractions_name
       ']}}</h3>
2  <div class="row">
3  <div id="map"></div>
4  <script>
5  function initMap() {
6  var uluru = {lat: {{$detail["Latitude"]}}, lng:{{
       $detail["longitude"]}} };
7  var map = new google.maps.Map(
8  document.getElementById('map'), {
9  zoom: 16,
10 center: uluru
11 });
12 var marker = new google.maps.Marker({position: uluru
       , map: map});
13 }
14 </script>
15 <script src="https://maps.googleapis.com/maps/api/js
       ?key=AIzaSyAQLj-_PEe0qXFXtqhs_EdE-ZmC5zoReMs&
       callback=initMap"
16 async defer></script>
17 <div class="card">
18 <h3รายละเอียด></h3>
19 <p>{{$detail["description"]}}</p>
20 </div>
21 </div>
```

รูปที่ 4.30: โครงสร้างหน้า viewdetail

จากภาพที่ 4.30 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการแสดงชื่อสถานที่ท่องเที่ยว
- บรรทัดที่ 5-13 เป็นการแสดงตำแหน่งของสถานที่ท่องเที่ยวในแผนที่

- บรรทัดที่ 15-16 เป็นการเรียกใช้งานบริการของ google
- บรรทัดที่ 17-20 เป็นการแสดงรายละเอียดของสถานที่ท่องเที่ยว

```
1 <div class="card" id="cardlocation">
2 <div id="fb-root"></div>
3 <script async defer crossorigin="anonymous" src="
     https://connect.facebook.net/th_TH/sdk.js#xfbml
     =1&version=v6.0"></script>
4 <center><div class="fb-comments" data-href={{Request
     ::url()}} data-width="1000" data-numposts="5"></
     div></center>
5 </div>
```

รูปที่ 4.31: โครงสร้างหน้า viewdetail

จากภาพที่ 4.31 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 3 เป็นการเรียกใช้งาน Comments Plugin
- บรรทัดที่ 4 เป็นการตั้งค่า Comments Plugin

4.13 โครงสร้างของ detail

```
1 function showdetail($id){
2 $details = Attraction::where('attractions_id',$id)->
     get()->toArray();
3 return view('viewdetail')->with('details',$details);
4 }
```

รูปที่ 4.32: function showdetail

จากภาพที่ 4.32 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 2 เป็นการนำข้อมูลสถานที่ท่องเที่ยวที่มีค่า attractions_id เท่ากับ id เก็บในตัวแปร
- บรรทัดที่ 3 เป็นการแสดงหน้า viewdetail พร้อมส่งค่า details ไปด้วย

4.14 โครงสร้างของ getplace

```
1 function getdata(Request $request){
2 $item=$request->input('province');
3 $places = Attraction::where('provinces_id',$item)->
     get()->toArray();
4 return view('getview')
5 ->with('places',$places);
6 }
```

รูปที่ 4.33: function getdata

จากภาพที่ 4.33 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 2 เป็นการนำค่าอินพุตจาก province มาเก็บในตัวแปร item
- บรรทัดที่ 3 แป็นการนำข้อมูลสถานที่ท่องเที่ยวที่มีค่า attractions_id เท่ากับ item เก็บในตัวแปร
- บรรทัดที่ 4 เป็นการแสดงหน้า getview พร้อมส่งค่า places ไปด้วย

4.15 โครงสร้างของ addplace

```
1  $id=$request->input('id');
2  $addPlace = Session::has('attraction_id') ? Session
       ::get('attraction_id') : null;
3  if ($addPlace==null) {
4  $addPlace= array();
5  array_push($addPlace,$id);
6  $request->session()->put('attraction_id',$addPlace);
7  }else {
8  $have=false;
9  for ($i=0; $i < count($addPlace) ; $i++) {
10 if ($addPlace[$i]==$id) {
11 $have=true;
12 break;
13 }
14 }
15 if($have==false){
16 array_push($addPlace,$id);
17 $request->session()->put('attraction_id',$addPlace);
18 }
19 }
20 return Redirect::back();
```

รูปที่ 4.34: function add

จากภาพที่ 4.34 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 1 เป็นการนำค่า id เก็บไว้ในตัวแปร
- บรรทัดที่ 2 เป็นการตรวจสอบว่ามี session attraction_id หรือไม่ ถ้ามีนำค่าไปเก็บในตัวแปร addPlace ถ้าไม่มีเก็บเป็นค่า null
- บรรทัดที่ 3 เป็นการตรวจสอบว่า addPlace มีค่าเท่ากับ null หรือไม่ ถ้ามีจะทำงานบรรทัดที่ 5-7 ถ้าไม่ใช่จะทำงานบรรทัดที่ 9-13
- บรรทัดที่ 4 เป็นการสร้าง array
- บรรทัดที่ 5 เป็นการเพิ่มค่า id ลงใน addPlace
- บรรทัดที่ 6 เป็นการสร้าง session attraction$_i daddPlace 8have false$

– บรรทัดที่ 19-14 เป็นการวนลูปเพื่อนตรวจสอบว่ามีค่าซ้ำกันหรือไม่

– บรรทัดที่ 15-17 เป็นการตรวจสอบดูว่าถ้าไม่มีค่าซ้ำจะทำการเพิ่มค่า id ในตัวแปร addPlace และใส่ค่าใน attraction$_id$

```
1  function showselect(){
2  if (Session::has('attraction_id')) {
3  $selecteds = array();
4  $getselects = Session::get('attraction_id');
5  foreach($getselects as $getselect){
6  $place = Attraction::where('attractions_id',
      $getselect )->get()->toArray();
7  $selecteds=array_merge($selecteds,$place);
8  }
9  return view('selectview')->with('selecteds',
      $selecteds);
10 }else {
11 return view('selectview');
12 }
13 }
```

รูปที่ 4.35: function showselect

จากภาพที่ 4.35 สามารถอธิบายการทำงานได้ดังนี้

– บรรทัดที่ 2 เป็นการตรวจสอบว่ามี session attraction_id หรือไม่ ถ้ามีจะทำงานบรรทัดที่ 3-9 ถ้าไม่มีจะทำงานบรรทัดที่ 11
– บรรทัดที่ 3 เป็นการสร้าง array ชื่อ selecteds
– บรรทัดที่ 4 เป็นการนำค่าจาก session attraction_id เก็บไว้ในตัวแปร getselect
– บรรทัดที่ 5-9 เป็นการวนลูปเพื่อนำค่า id จาก getselect มาดึงข้อมูลสถานที่ท่องเที่ยวเก็บไว้ในตัวแปร place และเพิ่มลงใน array selecteds
– บรรทัดที่ 9 เป็นการ return ไปหน้า selectview พร้อมกับส่งค่า selecteds ด้วย
– บรรทัดที่ 11 เป็นการ return ไปหน้า selectview

```
1  function del(Request $request){
2  $id=$request->input('id');
3  $getselects = Session::get('attraction_id');
4  $newgetselects=array();
5  foreach($getselects as $getselect){
6  if($getselect!=$id){
7  array_push($newgetselects,$getselect);
8  $request->session()->put('attraction_id',
      $newgetselects);
9  }
10 }
11 if ($newgetselects==null) {
12 $request->session()->forget('attraction_id');
13 }
14 return Redirect::to('selectview');
15 //return redirect('del','/search');
16 //return view('selectview')->with('selecteds',
      $selecteds);
17 }
```

รูปที่ 4.36: function del

จากภาพที่ 4.36 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 2 เป็นการนำค่า id เก็บในตัวแปร id
- บรรทัดที่ 3 เป็นการนำค่าจาก session attraction_id เก็บไว้ในตัวแปร getselects
- บรรทัดที่ 4 เป็นการสร้าง array ชื่อ newgetselects
- บรรทัดที่ 5-10 เป็นการวนลูปเพื่อตรวจสอบว่าค่า getselects เท่ากับค่า id หรือไม่ ถ้าไม่เท่ากันจะทำนำข้อมูลจาก getselect เพิ่มลง newgetselects และทำการเปลี่ยนค่า session attraction_id ให้มีค่าเท่ากับ newgetselects
- บรรทัดที่ 11-13 เป็นการตรวจสอบว่า newgetselects เท่ากับ null หรือไม่ ถ้าใช่จะทำการลบ session attraction_id

```
1 function delAllSelect(Request $request){
2 $request->session()->forget('attraction_id');
3 return Redirect::to('/');
4 }
```

รูปที่ 4.37: function delAllSelect

จากภาพที่ 4.37 สามารถอธิบายการทำงานได้ดังนี้

– บรรทัดที่ 2 เป็นการลบ session ชื่อ attraction$_id3redirect$'/'

4.16 โครงสร้างของ ResultController

```
1  public function geturl($lat,$lng,$getselects){
2  $url = "https://maps.googleapis.com/maps/api/
      distancematrix/json?origins=".$lat.",".$lng."&
      destinations=";
3  $i=1;
4  foreach ($getselects as $getselect) {
5  $place = Attraction::where('attractions_id',
      $getselect )->get()->toArray();
6  foreach ($place as $p) {
7  $deslat=$p["Latitude"];
8  $deslng=$p["longitude"];
9  $url .= $deslat.",".$deslng;
10 }
11 if($i!=count($getselects)){
12 $i++;
13 $url .= "|";
14 }else {
15 $url .= "&key=AIzaSyAQLj-_PEe0qXFXtqhs_EdE-
      ZmC5zoReMs";
16 }
17 }
18 return $url;
19 }
```

รูปที่ 4.38: function geturl

จากภาพที่ 4.38 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 2 เป็นการต่อสติงเพื่อใส่ค่าต้นทาง
- บรรทัดที่4-17 เป็นการต่อสติงค่าปลายทาง
- บรรทัดที่ 5 เปแนการดึงข้อสถานที่ท่องเที่ยวจาก database มาเก็บไว้ในตัวแปร place
- บรรทัดที่ 18 ทำการ return ค่า url

```
1  function getresult(Request $request){
2  $getselects = Session::get('attraction_id');
3  $lat=$request->input('lat');
4  $lng=$request->input('lng');
5  $url= $this->geturl($lat,$lng,$getselects);
6  $getselects = $this->getdistance($url,$getselects);
7  while (count($getselects)>1) {
8  $place = Attraction::where('attractions_id',end(
       $this->results) )->get()->toArray();
9  foreach ($place as $p) {
10 $lat=$p["Latitude"];
11 $lng=$p["longitude"];
12 }
13 $url=$this->geturl($lat,$lng,$getselects);
14 $getselects = $this->getdistance($url,$getselects);
15 }
16 if(count($getselects)==1){
17 foreach ($getselects as $getselect) {
18 array_push($this->results,$getselect);
19 }
20 }
21 Session::put('result', 'have');
22 $dataresults=array();
23 foreach($this->results as $result){
24 $dataresult = Attraction::where('attractions_id',
       $result )->get()->toArray();
25 $dataresults=array_merge($dataresults,$dataresult);
26 }
27 return view('result')->with('dataresults',
       $dataresults);
28 }
```

รูปที่ 4.39: function getresult

จากภาพที่ 4.39 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 2 เป็นการนำค่าจาก session attraction$_i dgetselects 3-4$

- บรรทัดที่ 5 เป็นการเรียกใช้งานฟังก์ชัน geturl
- บรรทัดที่ 6 เป็นการเรียกใช้งานฟังก์ชัน getdistance
- บรรทัดที่ 7-15 เป็นการวนลูปเพื่อเรียกใช้งานฟังก์ชัน geturl และ getdistance
- บรรทัดที่ 6-20 เป็นการเพิ่มข้อมูลใน getselects ใส่ใน result
- บรรทัดที่ 23-26 เป็นการนำค่าจาก result ไปดึงข้อมูลจาก database มาเก็บใน dataresult
- บรรทัดที่ 27 เป็นการ return ไปหน้า result พร้อมส่งค่า dataresult ด้วย

```
1  public function getdistance($url,$getselects){
2  $mindistance = null;
3  $index=null;
4  $client = new \GuzzleHttp\Client();
5  $response = $client->request('GET', $url);
6  $data = $response->getBody();
7  $get_rows = json_decode($data, true);
8  $get_elements = $get_rows["rows"];
9  $get_distance = $get_elements[0];
10 $get_value = $get_distance["elements"];
11 for ($i=0; $i < count($get_value); $i++) {
12 $text = $get_value[$i]["distance"]["text"];
13 $split= explode(" ",$text);
14 $distance = floatval($split[0]);
15 if($mindistance==null){
16 $mindistance = $distance;
17 $index=$i;
18 }else {
19 if ($mindistance>=$distance) {
20 $mindistance = $distance;
21 $index=$i;
22 }
23 }
24 }
25 $i=0;
26 foreach ($getselects as $getselect) {
27 if($i==$index){
28 $index=array_search($getselect, $getselects);
29 array_push($this->results,$getselects[$index]);
30 unset($getselects[$index]);
31 }
32 $i++;
33 }
34 return $getselects;
35 }
```

รูปที่ 4.40: function getdistance

จากภาพที่ 4.40 สามารถอธิบายการทำงานได้ดังนี้

- บรรทัดที่ 4-5 เป็นการส่ง request เพื่อหาระยะทาง
- บรรทัดที่ 6-10 เป็นการนำผลลัพธ์ระยะทางมาเก็บในตัวแปร

- บรรทัดที่ 11-24 เป็นการหาปลายทางที่ใกล้ที่สุด
- บรรทัดที่ 12 เป็นการนำค่าระยะทางที่เป็น string เก็บในตัวแปร $text
- บรรทัดที่ 13 เป็นการแยก string
- บรรทัดที่ 14 เป็นการแปลงตัวอักษรเป็นตัวเลข
- บรรทัดที่ 15-23 เป็นการเก็บค่าระยะทางที่สั้นที่สุดในตัวแปร mindistance
- บรรทัดที่ 26-33 เป็นการเพิ่มปลายทางที่ใกล้ที่สุดใส่ในตัวแปร result

บทที่ 5

การทดสอบระบบ

การทดสอบการทำงานของระบบ โดยทำการทดสอบในลักษณะ Black-box Testing ซึ่งเป็นการทดสอบแบบที่ไม่สนใจการทำงานภายในของโปรแกรมว่าทำงานอย่างไร แต่จะเน้นไปที่ Input และ Result ซึ่งการทดสอบได้ผลดังนี้

5.1 ทดสอบหน้า login

ตารางที่ 5.1: ผลการทดสอบหน้า login

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้า login	กดที่ช่องกรอกข้อมูล	ระบบ แสดง ตัว กระ พริบ (cursor) เพื่อชี้ให้รู้ตำแหน่งของการพิมพ์อักขระ
	พิมพ์อักขระ	ระบบ แสดง ผล อัก ระ ที่ ถูก พิมพ์
	กดปุ่ม submit กรณียังไม่ได้กรอก username,password	ระบบ แสดง ข้อความ โปรด กรอกข้อมูล
	กด ปุ่ม summit กรณี username,password ไม่ถูกต้อง	แสดง ข้อความ username หรือ password ไม่ถูกต้อง
	กด ปุ่ม summit กรณี username,password ถูกต้อง	ระบบแสดงหน้า admin

5.2 ทดสอบหน้า admin

ตารางที่ 5.2: ผลการทดสอบหน้า admin

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้า admin	กดปุ่ม travel	กลับไปหน้าค้นหาสถานที่ท่องเที่ยว
	กดปุ่ม admin	แสดง ตัว เลือก หน้า แรก,เพิ่มข้อมู,logout
	กดปุ่มลบ	ระบบแสดงให้ยืนยันการลบ
	กดปุ่ม ค้นหา	แสดงรายชื่อสถานที่ท่องเที่ยวทั้งหมดในจังหวัดที่ค้นหา
	กดตัวเลือก หน้าแรก	ระบบแสดงหน้า admin
	กดตัวเลือก เพิ่มข้อมูล	ระบบแสดงหน้าเพิ่มข้อมูล
	กดตัวเลือก log out	ระบบ ทำงาน ออก จาก ระบบ พร้อมกลับไปหน้าค้นหาสถานที่ท่องเที่ยว

5.3 ทดสอบหน้า adminsearch

ตารางที่ 5.3: ผลการทดสอบหน้า adminsearch

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้า adminsearch	กดปุ่มลบ	ระบบ ทำการ ลบ สถาน ที่ ท่อง เที่ยวที่เลือก
	กดปุ่มแก้ไข	ระบบแสดงหน้าอัพเดทข้อมูล
	กดตัวเลือก หน้าแรก	ระบบแสดงหน้า admin
	กดตัวเลือก เพิ่มข้อมูล	ระบบแสดงหน้าเพิ่มข้อมูล
	กดตัวเลือก log out	ระบบ ทำงาน ออก จาก ระบบ พร้อมกลับไปหน้าค้นหาสถาน ที่ท่องเที่ยว
	กดปุ่ม travel	กลับไปหน้าค้นหาสถานที่ท่อง เที่ยว
	กดปุ่ม admin	แสดง ตัว เลือก หน้า แรก,เพิ่ม ข้อมู,logout

5.4 ทดสอบหน้าแสดงรายละเอียดสถานที่ท่องเที่ยว

ตารางที่ 5.4: ผลการทดสอบหน้าแสดงรายละเอียดสถานที่ท่องเที่ยว

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้าแสดงรายละเอียดสถานที่ ท่องเที่ยว	กดปุ่มเลือกจังหวัด	แจ้งเตือนว่าเลือกสำเร็จ
	กดปุ่ม travel	กลับไปหน้าค้นหาสถานที่ท่อง เที่ยว

5.5 ทดสอบหน้าเพิ่มข้อมูล

ตารางที่ 5.5: ผลการทดสอบหน้าเพิ่มข้อมูล

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้าเพิ่มข้อมูล	กดที่ช่องกรอกข้อมูล	ระบบ แสดง ตัว กระ พริบ (cursor) เพื่อ ชี้ ให้ รู้ ตำแหน่ง ของการพิมพ์อักขระ
	พิมพ์อักขระ	ระบบ แสดง ผล อัก ระ ที่ ถูก พิมพ์
	กดปุ่ม admin	แสดง ตัว เลือก หน้า แรก,เพิ่ม ข้อมู,logout
	กดตัวเลือก หน้าแรก	ระบบแสดงหน้า admin
	กดตัวเลือก เพิ่มข้อมูล	ระบบแสดงหน้าเพิ่มข้อมูล
	กดตัวเลือก log out	ระบบ ทำงาน ออก จาก ระบบ พร้อมกลับไปหน้าค้นหาสถาน ที่ท่องเที่ยว
	กดปุ่ม travel	กลับไปหน้าค้นหาสถานที่ท่อง เที่ยว
	กดปุ่ม submit	ระบบ ทำการ แจ้ง เตือน success
	ไม่กรอกข้อมูล	ระบบแจ้งเตือนให้กรอกข้อมูล

5.6 ทดสอบหน้าแสดงลำดับการเดินทาง

ตารางที่ 5.6: ผลการทดสอบหน้าแสดงลำดับการเดินทาง

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้าแสดงสถานที่ท่องเที่ยวที่เลือก	กดปุ่มลบที่เลือกทั้งหมด	ลบ สถาน ที่ ท่อง เที่ยว ที่ เลือก ทั้งหมด พร้อม กลับ ไป หน้า ค้นหาสถานที่ท่องเที่ยว

5.7 ทดสอบหน้าค้นหาสถานที่ท่องเที่ยว

ตารางที่ 5.7: ผลการทดสอบหน้าค้นหาสถานที่ท่องเที่ยว

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้าค้นหาสถานที่ท่องเที่ยว	กดเลือกจังหวัด	แสดงชื่อจังหวัดที่เลือก
	กดค้นหา	แสดงสถานที่ท่องในจังหวัดที่เลือก
	กดปุ่ม admin	แสดงหน้า login

5.8 ทดสอบหน้าแสดงสถานที่ท่องเที่ยวที่เลือก

ตารางที่ 5.8: ผลการทดสอบหน้าแสดงสถานที่ท่องเที่ยวที่เลือก

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้าแสดงสถานที่ท่องเที่ยวที่เลือก	กดปุ่มลบที่เลือกทั้งหมด	ลบ สถาน ที่ ที่ เลือก ทั้งหมด พร้อมกลับไปหน้าค้นหาสถานที่ท่องเที่ยว
	กดปุ่ม travel	กลับไปหน้าค้นหาสถานที่ท่องเที่ยว
	กดปุ่มลบ	ลบสถานที่ที่ทำการลบ
	กดปุ่มค้นหาเส้นทาง	แสดง หน้า ลำดับ การ เดิน ทาง พร้อมเส้นทาง

5.9 ทดสอบหน้าอัพเดทข้อมูล

ตารางที่ 5.9: ผลการทดสอบหน้าอัพเดทข้อมูล

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้าอัพเดทข้อมูล	กดที่ช่องกรอกข้อมูล	ระบบแสดงข้อมูลเดิม
	พิมพ์อักขระ	ระบบ แสดง ผล อัก ระ ที่ ถูก พิมพ์
	กดปุ่ม admin	แสดง ตัว เลือก หน้า แรก,เพิ่ม ข้อมู,logout
	กดตัวเลือก หน้าแรก	ระบบแสดงหน้า admin
	กดตัวเลือก เพิ่มข้อมูล	ระบบแสดงหน้าเพิ่มข้อมูล
	กดตัวเลือก log out	ระบบ ทำงาน ออก จาก ระบบ พร้อมกลับไปหน้าค้นหาสถานที่ท่องเที่ยว
	กดปุ่ม travel	กลับไปหน้าค้นหาสถานที่ท่องเที่ยว
	กดปุ่ม submit	ระบบ ทำการ แจ้ง เตือน success

5.10 ทดสอบหน้าแสดงรายละเอียดสถานที่ท่องเที่ยว

ตารางที่ 5.10: ผลการทดสอบหน้าแสดงรายละเอียดสถานที่ท่องเที่ยว

การทำงาน	เงื่อนไขการทดสอบ	ผลการทดสอบ
หน้าแสดงรายละเอียดสถานที่ท่องเที่ยว	กดปุ่มเลือกจังหวัด	แจ้งเตือนว่าเลือกสำเร็จ
	กดปุ่ม travel	กลับไปหน้าค้นหาสถานที่ท่องเที่ยว

บทที่ 6

สรุปและข้อเสนอแนะ

ในการพัฒนาระบบท่องเที่ยวอีสาน พบว่าระบบสามารถทำงานได้ ตามที่วิเคราะห์และออก-แบบไว้แต่ก็พบปัญหาและอุปสรรคระหว่างการพัฒนา ในบทนี้ผู้พัฒนาจึง ขอสรุปความสามารถของระบบ ชี้แจงปัญหาและอุปสรรค พร้อมเสนอแนวทางในการพัฒนาระบบ เช็คชื่อด้วยคิวอาร์โค้ดต่อตามลำดับ

6.1 สรุปความสามารถของระบบ

ระบบท่องเที่ยวอีสานสามารถสรุปความสามารถที่ระบบทำได้ดังนี้

1. ผู้ใช้งานทั่วไป
    - ดูรายละเอียดสถานที่ท่องเที่ยว
    - เลือกสถานที่ท่องเที่ยว
    - ค้นหาสถานที่ท่องเที่ยว
    - ดูลำดับการเดินทาง
2. ผู้ดูแลระบบ
    - เข้าสู่ระบบได้
    - เพิ่มข้อมูลสถานที่ท่องเที่ยว
    - ลบข้อมูลสถานที่ท่องเที่ยว
    - แก้ไขข้อมูลสถานที่ท่องเที่ยว
    - เพิ่มข้อมูลจังหวัด
    - ลบข้อมูลจังหวัด

6.2 ปัญหาและอุปสรรคในการพัฒนา

1. เมื่อผู้ใช้กดค้นหาเส้นทางแต่ผู้ใช้ไม่อนุญาตไม่ทราบตำแหน่งปัจจุบันจะไม่สามารถหาลำดับการเดินทางได้

แนวทางการแก้ไข : ทำการใช้ตำแหน่งสถานที่ท่องเที่ยวที่เลือกลำดับแรก เป็นจุดเริ่มต้นในการหาลำดับการเดินทาง

6.3 แนวทางการพัฒนาต่อ

1. ออกแบบเว็บให้ดูน่าสนใจมากขึ้น
2. สามารถค้นหาสถานที่ท่องเที่ยวได้ทั้งประเทศไทย
3. สามารถแสดงเวลาที่ใช้ในการเดินทาง

บรรณานุกรม

[1] marcuscode (2559). โครงสร้างของภาษา php [ออนไลน์]. สืบค้นเมื่อ จาก http:// marcuscode.com/lang/php/program-struct .

[2] Krissanawat Kaewsanmuang (). คู่มือการใช้งาน laravel [ออนไลน์]. สืบค้นเมื่อ จาก https://leanpub.com/thailaraveldoc/read#leanpub-auto-installation .

[3] benext (2563). การใช้งาน blade template [ออนไลน์]. สืบค้นเมื่อ จาก https:// www.itoffside.com/laravel-ep10-blade-template/ .

[4] wp63 (2562). ใช้ guzzle ในการเชื่อมต่อ api แทน curl [ออนไลน์]. สืบค้นเมื่อ จาก https:// wp63.co/2019/10/guzzle-php-http-client/ .

[5] google maps platform (2563). Distance matrix api [ออนไลน์]. สืบค้นเมื่อ จาก https:// developers-dot-devsite-v2-prod.appspot.com/ maps/ documentation/ distance-matrix/intro#DistanceMatrixRequests .

ภาคผนวก

ภาคผนวก ก

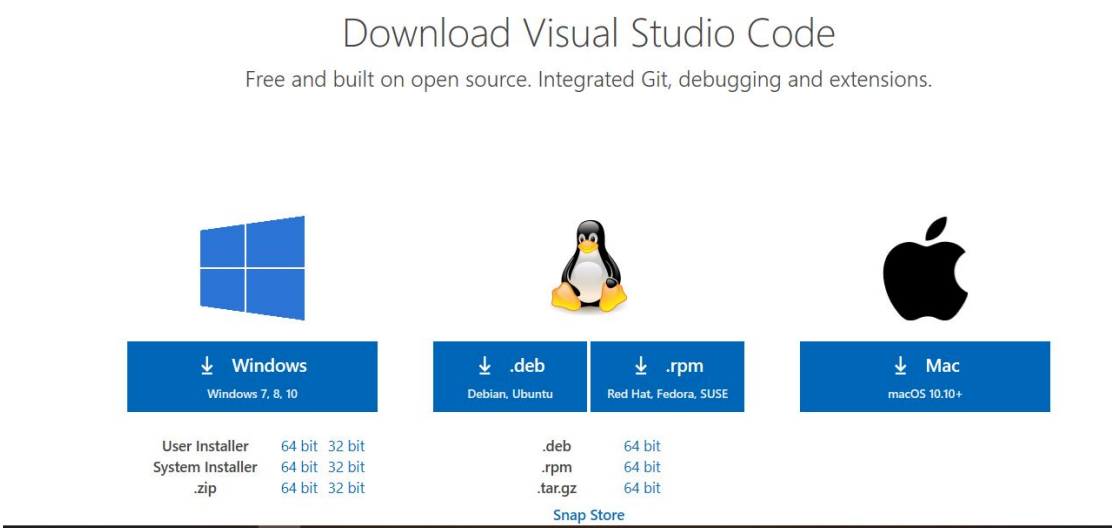
การติดตั้งเครื่องมือที่ใช้พัฒนาโปรแกรม

การติดตั้งเครื่องมือที่ใช้ในการพัฒนาเว็บแนะนำและสร้างแผนการท่องเที่ยว มีโปรแกรมที่จำเป็นในการพัฒนาระบบดังต่อไปนี้

- การติดตั้ง VisualStudioCode
- การติดตั้ง XAMPP

ก.1 การติดตั้ง Visual Studio Code

1. สามารถดาวน์โหลด VisualStudioCodeได้ที่https://code.visualstudio.com/download ดังแสดงในรูปที่ ก.1

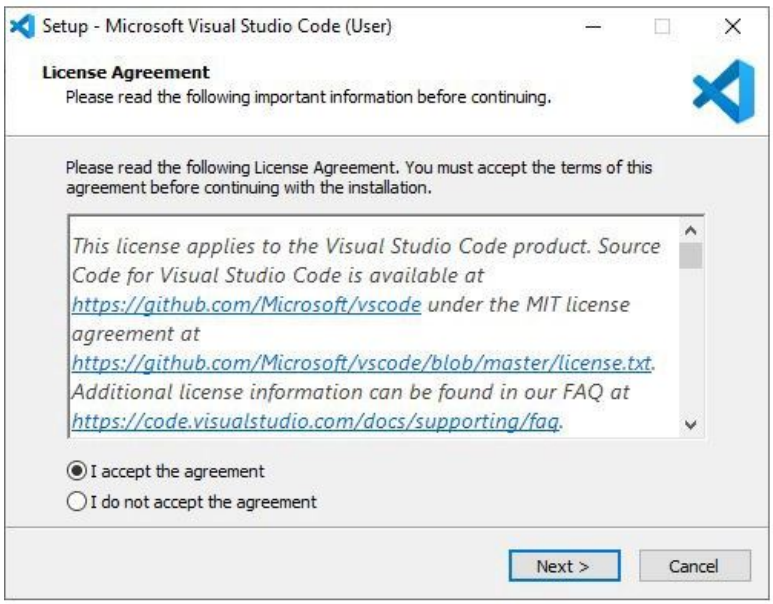


รูปที่ ก.1: หน้าเว็บดาวน์โหลด Visual Studio Code

2. แสดงหน้าต่างต้อนรับของ Visual Studio Codeทำการกด Next เพื่อเริ่มกระบวนการติดตั้ง ดังแสดงในรูปที ก.2

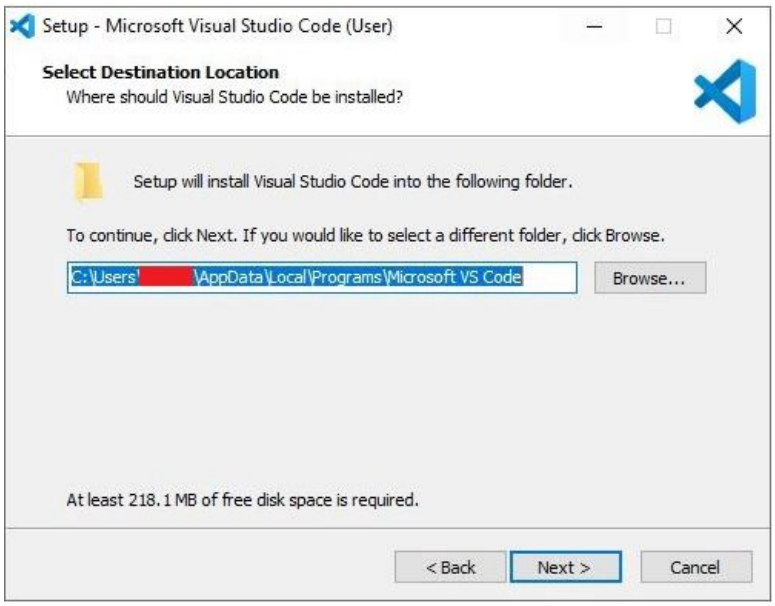
รูปที่ ก.2: หน้าต่างต้อนรับของ Visual Studio Code

3. แสดงหน้าต่างข้อตกลงการใช้งาน Visual Studio Code ทำการกด IAgree ดังแสดงในรูปที่ ก.3



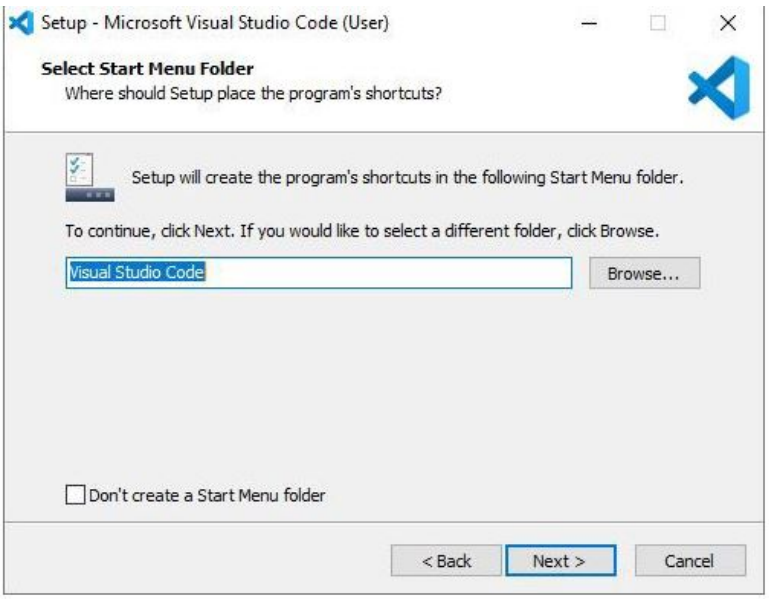
รูปที่ ก.3: หน้าต่างข้อตกลงการใช้งาน Visual Studio Code

4. แสดงหน้าต่างที่จัดเก็บไฟล์ต่างๆของ Visual Studio Codeทำการกด Next ดังแสดงในรูปที ก.4



รูปที่ ก.4: หน้าต่างที่จัดเก็บไฟล์ต่างๆของ Visual Studio Code

5. แสดงหน้าต่างเริ่มทำการติดตั้งทำการกด Install ดังแสดงในรูปที ก.5



รูปที่ ก.5: หน้าต่างติดตั้งโปรแกรม Visual Studio Code

6. แสดงหน้าต่างผลการติดตั้ง Visual Studio Code ดังแสดงในรูปที ก.6

รูปที่ ก.6: หน้าต่างผลการติดตั้ง Android Studio

ก.2 การติดตั้ง XAMPP

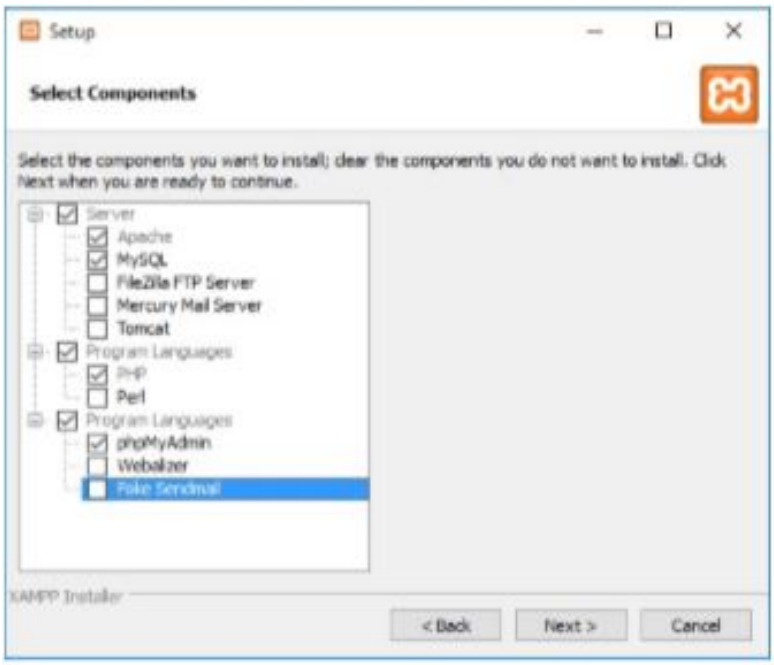
1. สามารถดาวน์โหลด XAMPP ได้ที่ https://www.apachefriends.org/index.html ดังแสดง ในรูปที ก.7

รูปที่ ก.7: หน้าเว็บดาวน์โหลด XAMPP

2. แสดงหน้าต่างติดตั้ง (setup) ของ XAMPP ทำการกด Next เพื่อเริ่มกระบวนการติดตั้งดังแสดงในรูปที ก.8

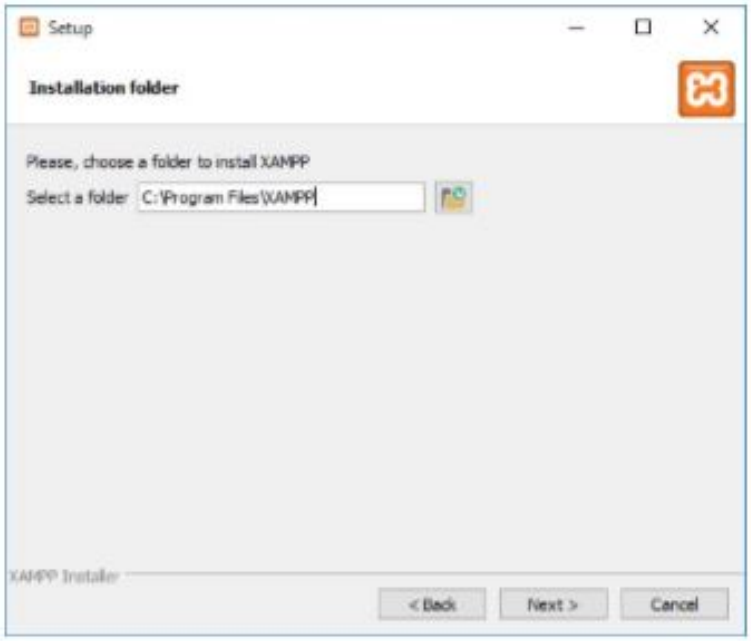
รูปที่ ก.8: หน้าสำหรับติดตั้ง XAMPP

3. แสดงหน้าต่างทำการเลือก Components ที่เราต้องการใช้งาน XAMPP และกด Next ดังแสดงในรูปที่ ก.9



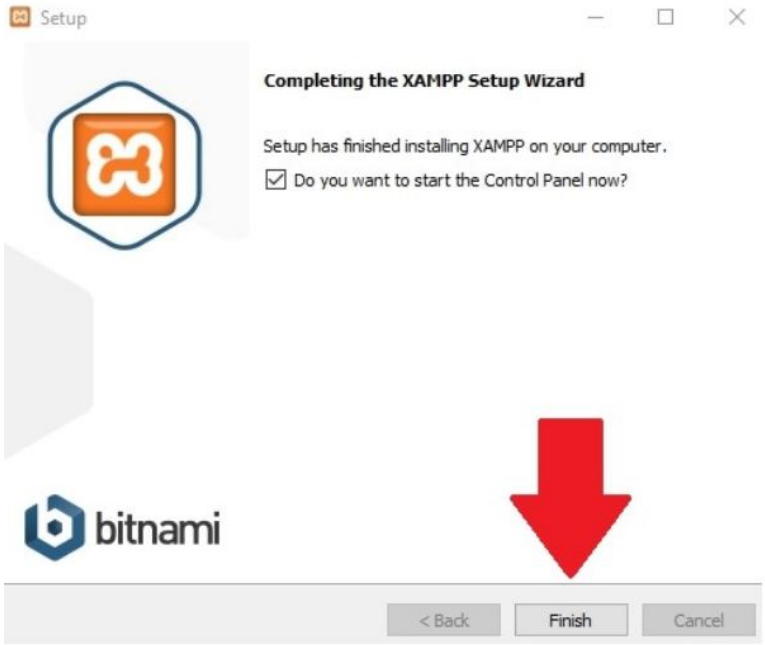
รูปที่ ก.9: หน้าต่างเลือก Components ที่ต้องการใช้งาน XAMPP

4. แสดงหน้าต่างเลือกโฟลเดอร์ที่จะทำการติดตั้งและกด Next ดังแสดงในรูปที ก.10



รูปที่ ก.10: หน้าต่างเลือกโฟลเดอร์ที่จะทำการติดตั้ง XAMPP

5. แสดงหน้าต่างติดตั้งเสร็จสิ้นและทำการกด Finish เพื่อจบการติดตั้งดังแสดงในรูปที ก.11

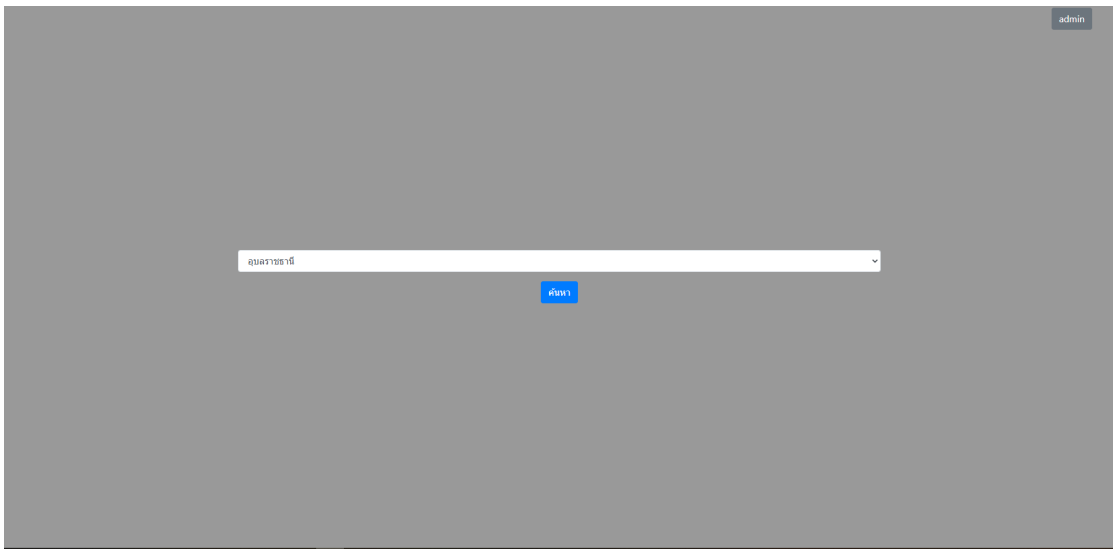


รูปที่ ก.11: หน้าต่างเสร็จสิ้นการติดตั้ง XAMPP

ภาคผนวก ข

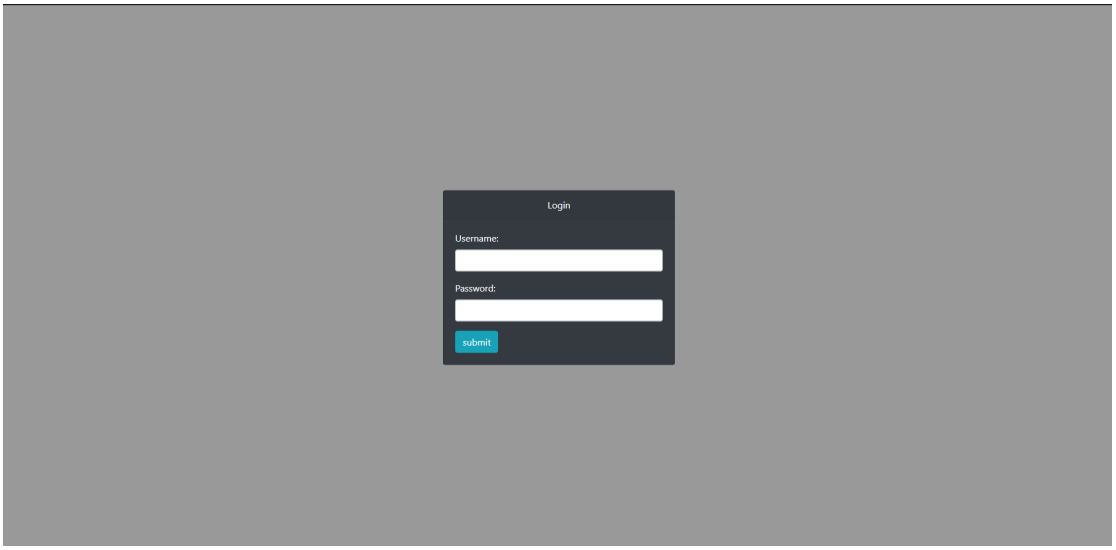
คู่มือการใช้งานระบบ

คู่มือการใช้งานระบบท่องเที่ยวอีสานมีรายละเอียดดังนี้



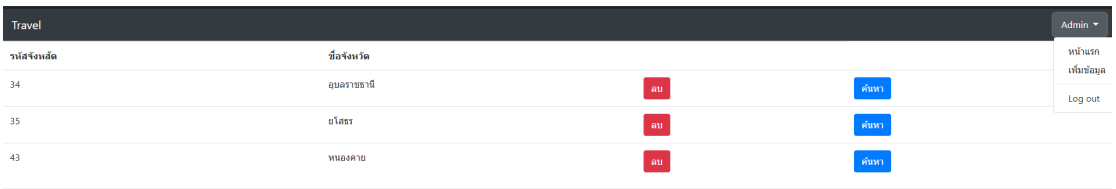
รูปที่ ข.1: หน้าจอหลักของระบบ

จากรูปที่ ข.1 แสดงหน้าจอหลัก ผู้ใช้งานต้องเลือกจังหวัดและกดปุ่ม ค้นหา เพื่อดูสถานที่ท่องเที่ยวในจังหวัดนั้น เมื่อกดปุ่ม admin จะเข้าสู่หน้า login สำหรับ admin



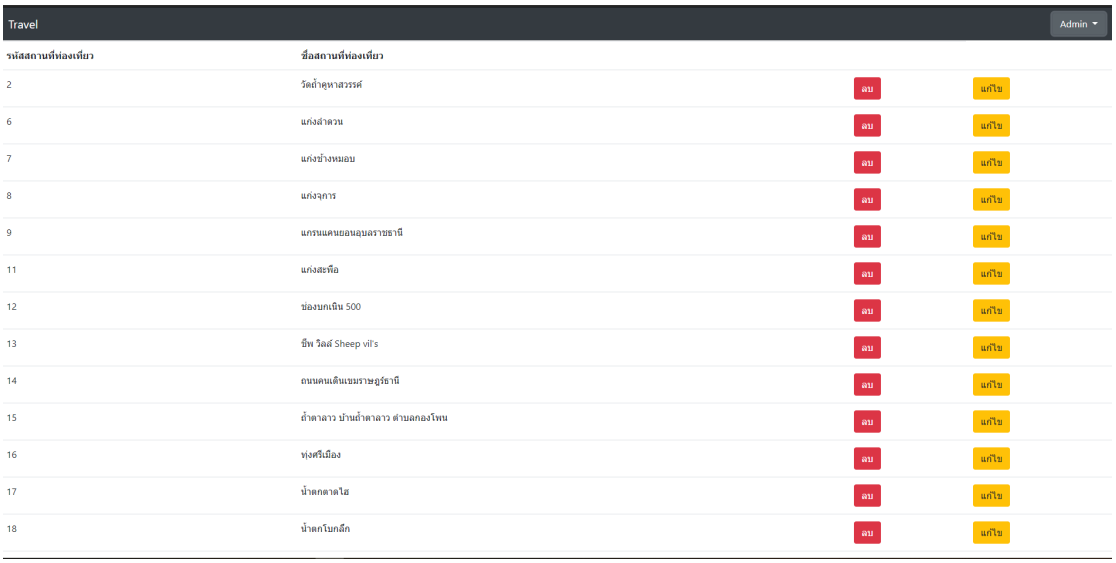
รูปที่ ข.2: หน้าlogin

จากรูป ข.2 เป็นหน้าเข้าสู่ระบบของ admin โดยจะต้องกรอกข้อมูลให้ถูกต้องและกดปุ่ม submit



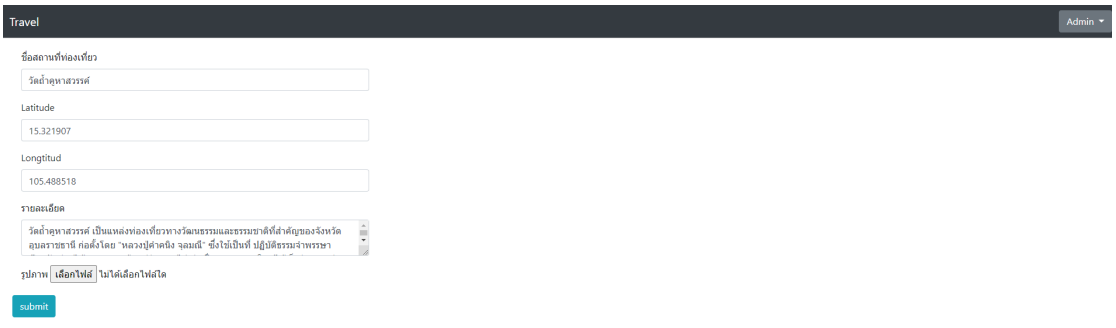
รูปที่ ข.3: หน้าadmin

เมื่อเข้าสู่ระบบสำเร็จจะแสดงหน้า admin ดังรูปที่ ข.3 กดปุ่ม ลบ จะทำการลบข้อมูลสถานที่ท่องเที่ยวในจังหวัดนั้นทั้งหมดออก กดปุ่ม ค้นหา เพื่อดูสถานที่ท่องเที่ยวในจังหวัดนั้น กดปุ่ม เพิ่มข้อมูล เพื่อไปหน้าเพิ่มข้อมูลอ กดปุ่ม Log out เพื่อออกจากระบบและกลับไปแสดงหน้าจอหลัก



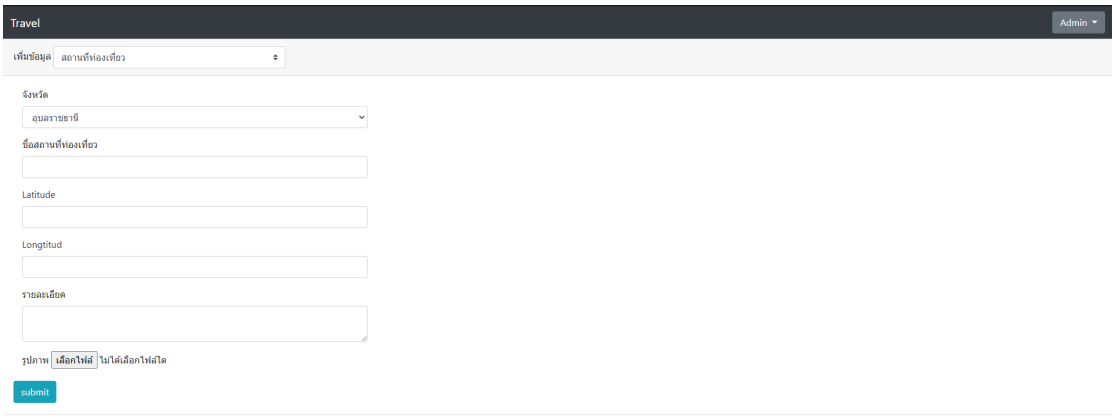
รูปที่ ข.4: หน้า adminsearch

เมื่อกดปุ่ม ค้นหา จากรูป ข.3 จะแสดงหน้าดังรูป ข.4 กดปุ่ม ลบ เพื่อลบข้อมูลสถานที่ท่องเที่ยวนั้น กดปุ่ม แก้ไข เพื่อไปยังหน้าแก้ไขข้อมูลสถานที่ท่องเที่ยว



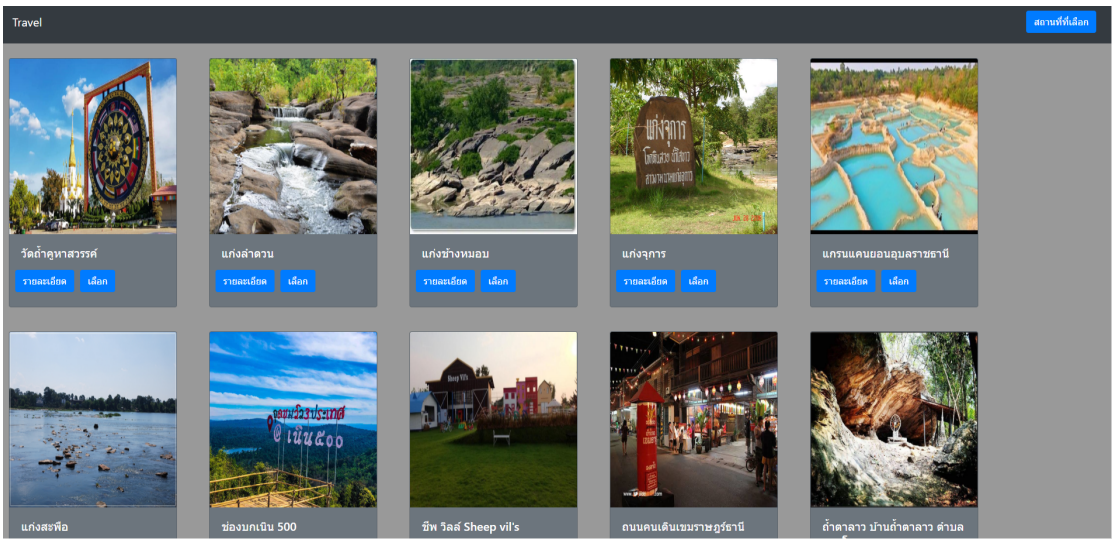
รูปที่ ข.5: หน้าแก้ไขข้อมูล

เมื่อกดปุ่ม แก้ไข จากรูป ข.3 จะแสดงหน้าให้กรอกข้อมูลดังในรูป ข.5 เมื่อให้การกรอกข้อมูลครบ กดปุ่ม submit เพื่อบันทึกข้อมูล



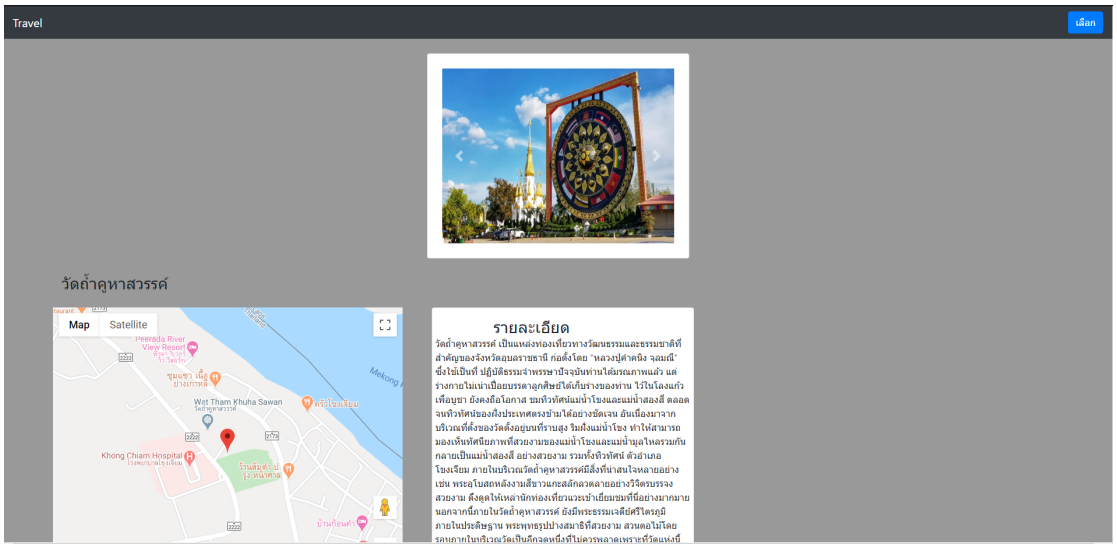
รูปที่ ข.6: หน้าเพิ่มข้อมูล

เมื่อกดปุ่ม เพิ่มข้อมูล ตรงnavbar จะแสดงหน้าดังในรูป ข.6 เมื่อให้การกรอกข้อมูลครบกดปุ่ม submit เพื่อบันทึกข้อมูล



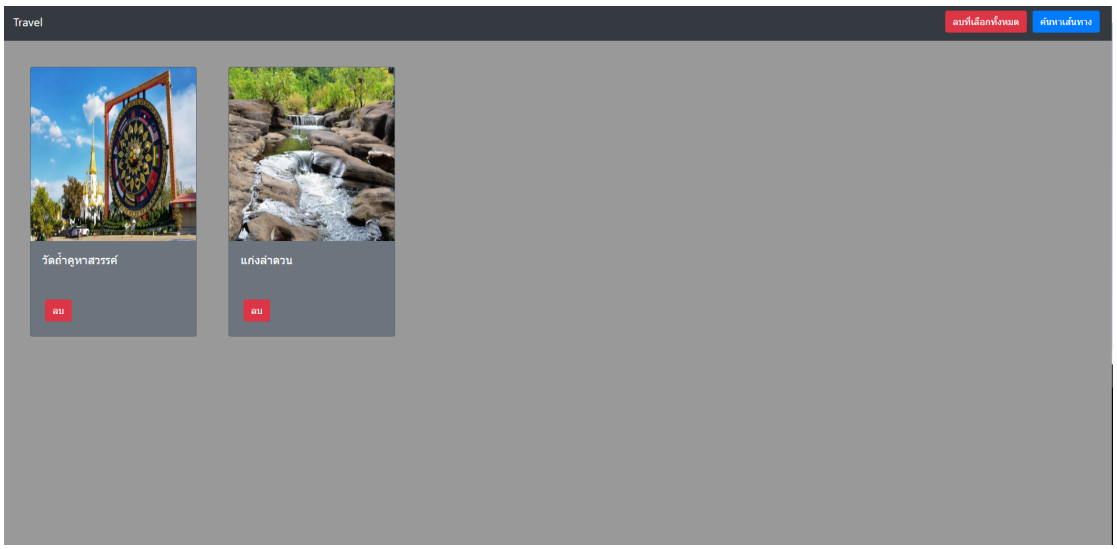
รูปที่ ข.7: หน้าแสดงสถานที่ท่องเที่ยว

เมื่อกดปุ่ม ค้นหา จากรูป ข.1 จะแสดงหน้าแสดงสถานที่ท่องเที่ยวดังรูป ข.7 เมื่อกดปุ่ม รายละเอียด จะแสดงหน้ารายละเอียดดังรูป ข.8 กดปุ่ม เลือก เพื่อเลือกสถานที่ท่องเที่ยวที่สนใจ กดปุ่ม สถานที่ที่เลือก เพื่อดูสถานที่ท่องเที่ยวที่ผู้ใช้กดเลือก



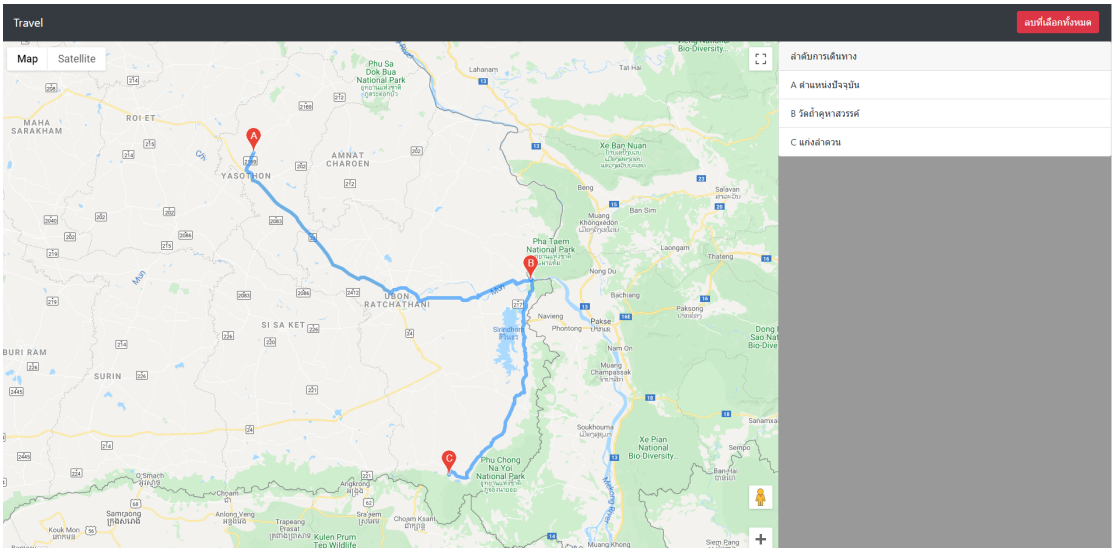
รูปที่ ข.8: หน้าแสดงรายละเอียด

เมื่อกดปุ่ม เลือก จากรูป ข.8 จะทำการเลือกสถานที่ท่องเที่ยวนี้



รูปที่ ข.9: หน้าแสดงสถานที่ท่องเที่ยวที่เลือก

เมื่อกดปุ่ม สถานที่ที่เลือก จากรูป ข.7 จะแสดงสถานที่ท่องเที่ยวที่ผู้ใช้กดเลือกดังรูป ข.9 เมื่อกดปุ่ม ลบที่เลือกทั้งหมด จะทำการลบสถานที่ที่ผู้ใช้เลือก



รูปที่ ข.10: หน้าแสดงลำดับการเดินทาง

เมื่อกดปุ่ม ค้นหาเส้นทาง จากรูป ข.9 จะแสดงลำดับการเดินทางดังรูป ข.10

ประวัติผู้เขียน

ชื่อ-สกุล: นายธนภูมิ สุปันนุช

รหัสประจำตัวนักศึกษา: 5711403250

วัดเกิด: 7 05 2539

ที่อยู่ที่สามารถติดต่อได้: 239 หมู่1 บ้านทุ่งแต้ ตำบลทุ่งแต้ อำเภอเมือง จังหวัดยโสธร 35000

เบอร์โทรศัพท์: (+66) 93 320 2389

อิเมลล์: thanaphum.su.57@ubu.ac.th

ระดับมัธยมต้น: โรงเรียนยโสธรพิทยาคม จังหวัดยโสธร

ระดับมัธยมปลาย: โรงเรียนยโสธรพิทยาคม จังหวัดยโสธร

ระดับอุดมศึกษา: ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ สาขาวิทยาการ คอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี