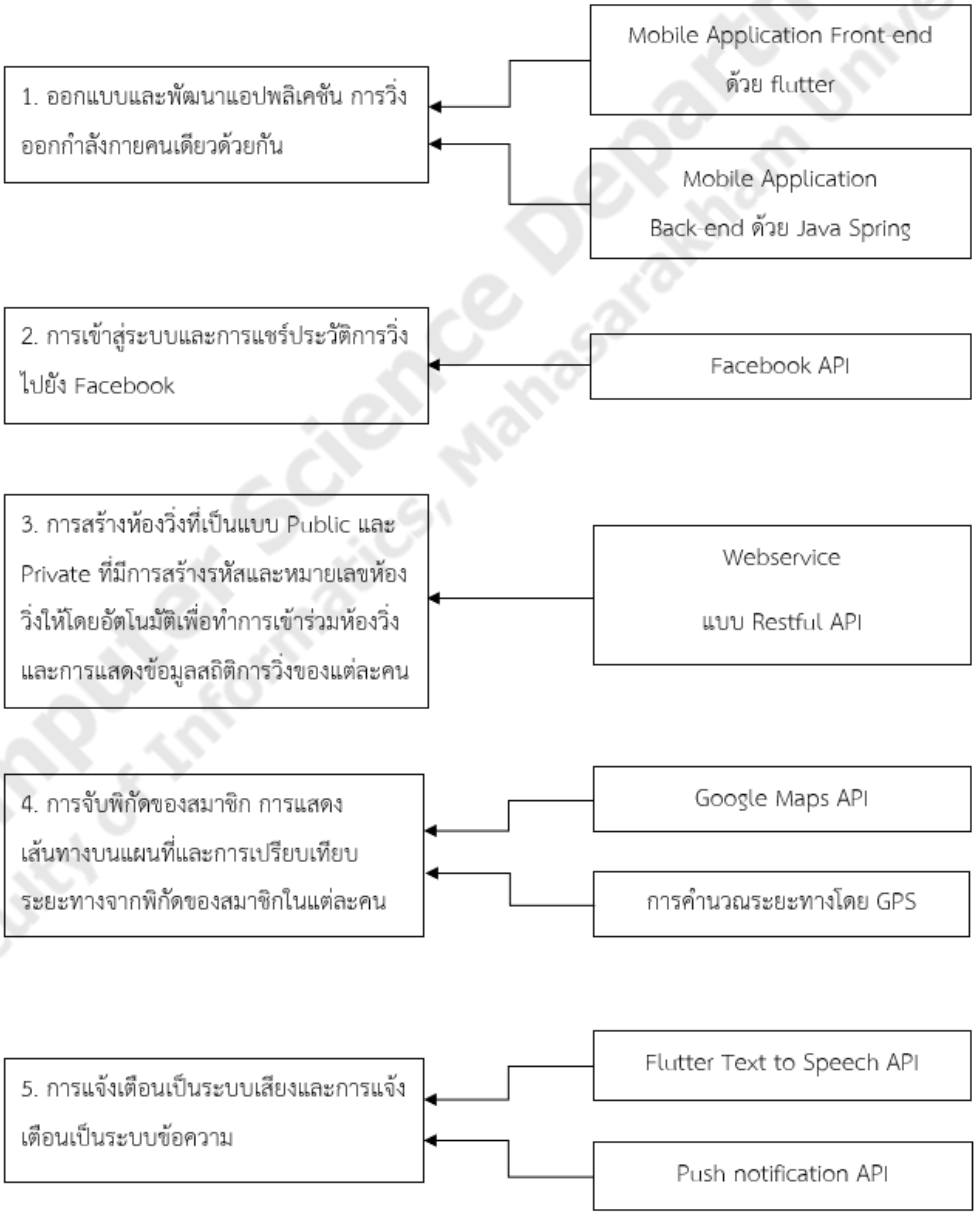
บทที่ 3

ขั้นตอนการดำเนินงานวิจัย

สำหรับบทนี้จะกล่าวถึงขั้นตอนในการดำเนินงานโครงงานปริญญานิพนธ์ ซึ่งจะทำให้ทราบถึงการวิเคราะห์และกระบวนการประมวลผลโดยละเอียดว่ามีแนวทางในการทำงานหรือขั้นตอนในการทำงานของระบบเป็นอย่างไร โดยขั้นตอนในการดำเนินงานมีรายละเอียดดังนี้

3.1 กรอบการดำเนินงาน



ภาพประกอบที่ 3.1 กรอบการดำเนินงาน

3.2 คำอธิบาย

3.2.1 เขียนโปรแกรมออกแบบและพัฒนาแอปพลิเคชัน การวิ่งออกกำลังกายคนเดียวด้วยกัน สำหรับผู้ใช้งานแอปพลิเคชันผ่านสมาร์ทโฟน โดยการสร้างแอปพลิเคชันเป็น 2 ส่วนหลัก ๆ ได้แก่

3.2.1.1 แอปพลิเคชันที่ทำงานอยู่บนสมาร์ทโฟน (Front-end) โดยเครื่องมือที่ใช้ในการพัฒนาแอปพลิเคชันคือ Visual Studio 2019 โดยใช้ Flutter Framework ซึ่งเป็นภาษา Dart

3.2.1.2 แอปพลิเคชันที่ทำงานติดต่อกับฐานข้อมูล (Back-end) โดยเครื่องมือที่ใช้ในการพัฒนาแอปพลิเคชันคือ Eclipse โดยใช้ Java Spring ซึ่งเป็นภาษา Java

3.2.2 การเข้าสู่ระบบและการแชร์ประวัติการวิ่งโดย Facebook ในส่วนนี้จะใช้ Facebook API ซึ่งเชื่อมต่อแอปพลิเคชันกับ Facebook เพื่อติดต่อเรียกใช้ข้อมูลจาก Facebook ทำให้แอปพลิชันสามารถใช้งานการเข้าสู่ระบบผ่านบัญชี Facebook และสามารถแชร์ประวัติการวิ่งผ่าน Facebook

3.2.3 การสร้างห้องวิ่งที่เป็นแบบ Public และ Private ที่มีการสร้างรหัสและหมายเลขห้องวิ่งโดยอัตโนมัติเพื่อทำการเข้าร่วมห้องวิ่งและการแสดงข้อมูลสถิติการวิ่งของแต่ละคนในส่วนนี้คือ Restful API เป็นวิธีในการสร้าง Web Service โดยที่ Restful API อยู่บนพื้นฐานของ HTTP Method เช่น GET, POST, PUT, PATCH, DELETE ในการทำงาน และส่งค่ากลับมาในรูปแบบ JSON หรือ XML

3.2.4 การจับพิกัดของสมาชิก การแสดงเส้นทางบนแผนที่และการเปรียบเทียบระยะทางจากพิกัดของสมาชิกในแต่ละคน ในส่วนนี้จะมีส่วนหลัก ๆ อยู่ 2 ส่วน ได้แก่

3.2.4.1 Google Maps API เป็นชุด API ของ Google สำหรับพัฒนา Web application และ mobile application (Android, iOS) ไว้สำหรับเรียกใช้แผนที่และชุก service ต่าง ๆ ของ Google เพื่อพัฒนา Application โดยการเขียนโปรแกรมด้วย flutter ต้องลงไลบรารีชื่อ google_maps_flutter เพื่อเรียกใช้ Google Maps API

3.2.4.2 GPS คือ ระบบกำหนดตำแหน่งบนพื้นโลก จะทำงานโดยรับสัญญาณ ตัวเครื่องรับสัญญาณ GPS เช่น สมาร์ทโฟน โดยการคำนวณระยะทางโดย GPS ด้วยการเขียนโปรแกรมด้วย Flutter จะต้องลงไลบรารีชื่อ location สำหรับเรียกตำแหน่งสมาร์ทโฟน

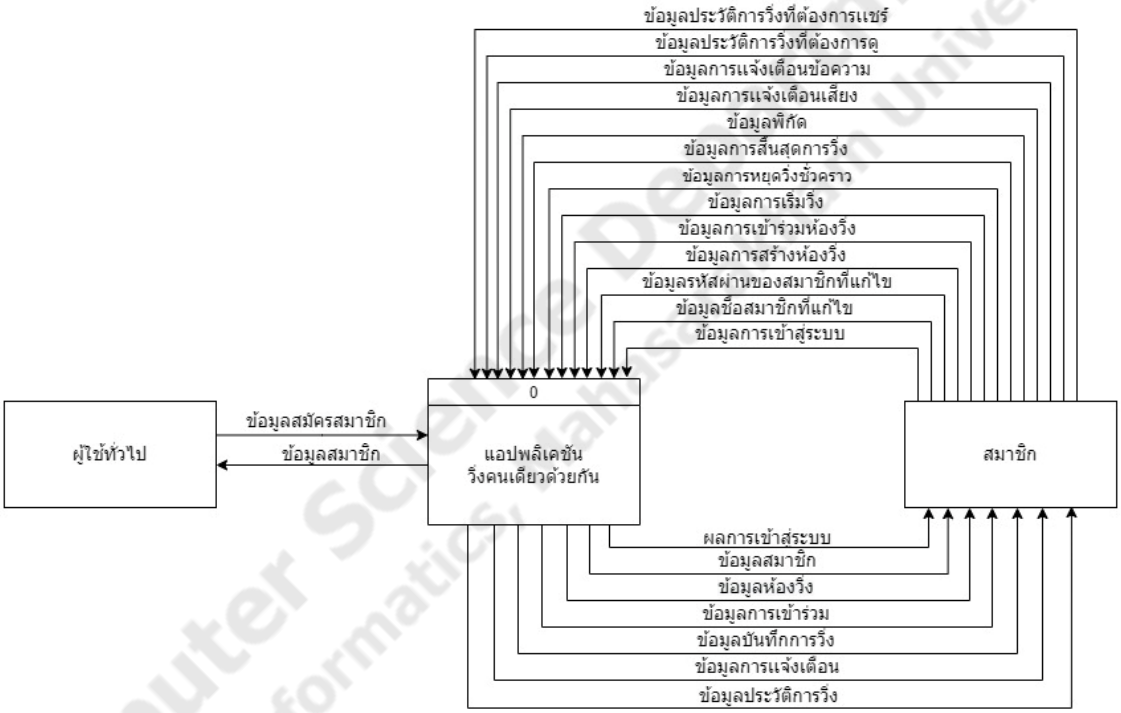
3.2.5 การแจ้งเตือนเป็นระบบเสียงและการแจ้งเตือนเป็นระบบข้อความ ในส่วนนี้มีส่วนหลัก ๆ อยู่ 2 ส่วน ได้แค่

3.2.5.1 Flutter Text to Speech เป็นการแปลกข้อความออกมาเป็นเสียงพูด เป็นส่วนต่อประสานกับโปรแกรมประยุกต์ (API) ที่ใช้ในการแปลงข้อความให้กลายเป็นไฟล์เสียง (TTS) นำมาพัฒนาในส่วนของการทำงานที่มีการแจ้งเตือนเป็นระบบเสียง

3.2.5.2 Push notification API คือการแจ้งเตือนที่แปลพลิเคชันนำข้อมูลมาแส้งในการแจ้งเตือนของระบบปฏิบัติการนั้น ๆ กำหนด นำมาพัฒนาในส่วนของการทำงานที่มีการแจ้งเตือนเป็นระบบข้อความในแอปพลิเคชัน

3.3 แผนภาพบริบท (Context Diagram)

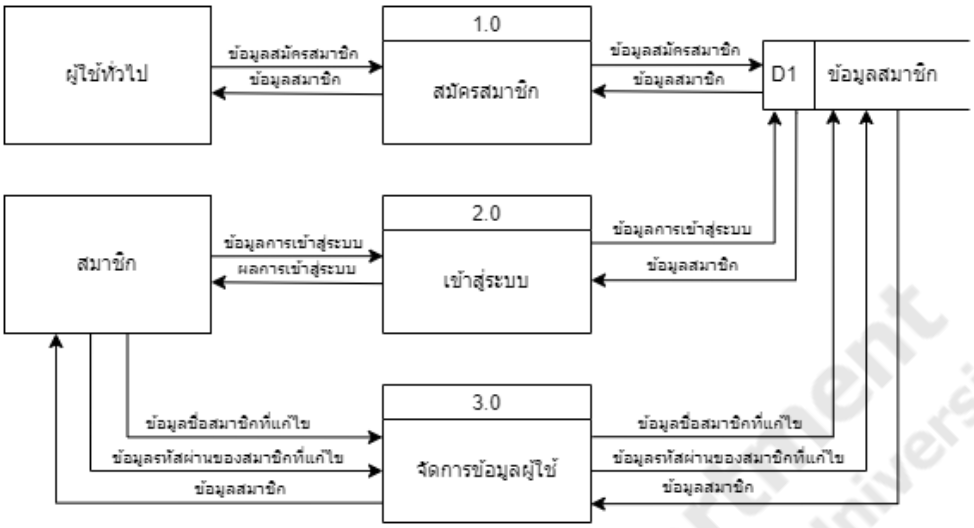
Context Diagram คือ แผนภาพกระแสข้อมูลที่แสดงถึงภาพรวมการทำงานของระบบที่มีความสัมพันธ์กับผู้ใช้งาน แสดงถึงขอบเขตของแอปพลิเคชันวิ่งคนเดียวด้วยกัน (Run Alone Together)



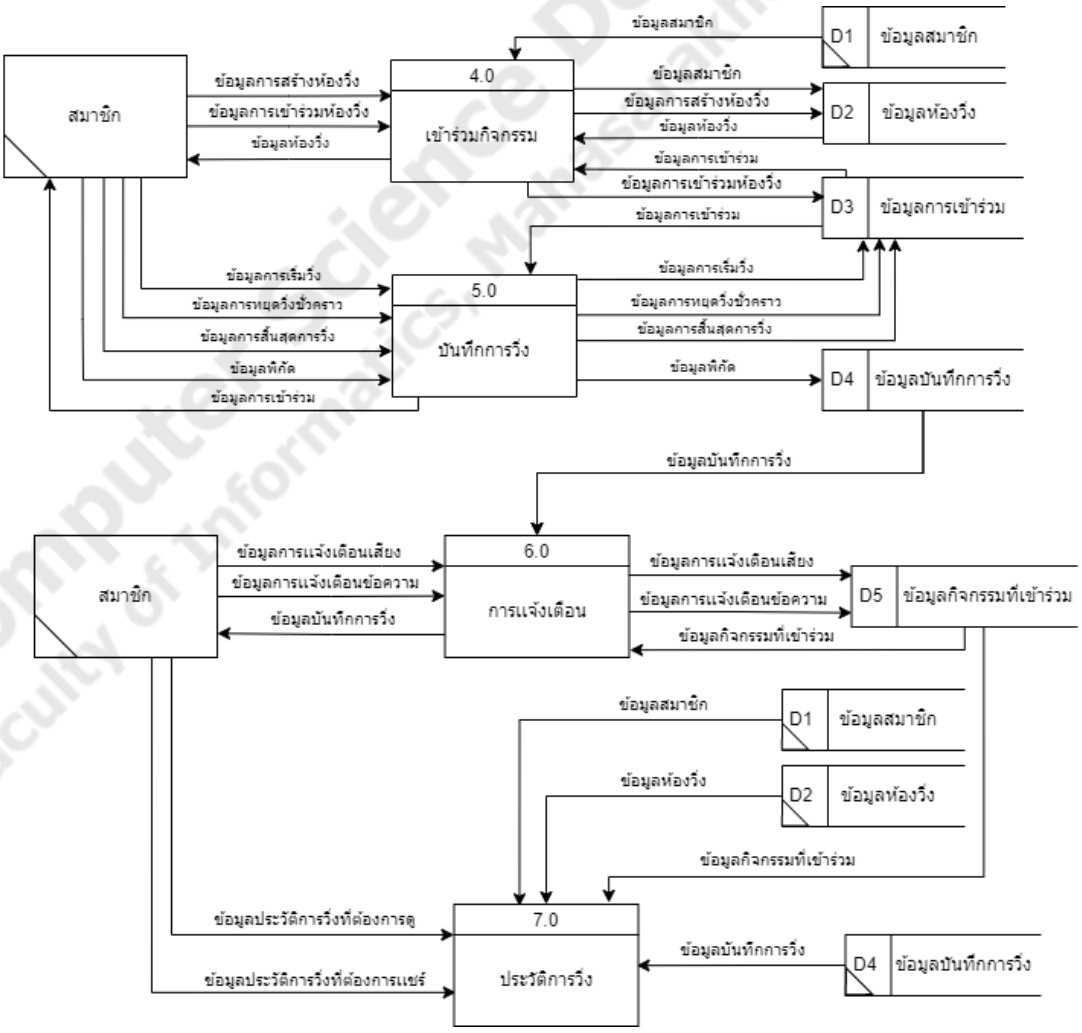
ภาพประกอบที่ 3.2 Context Diagram

3.4 แผนภาพการไหลของข้อมูล (Data Flow Diagram)

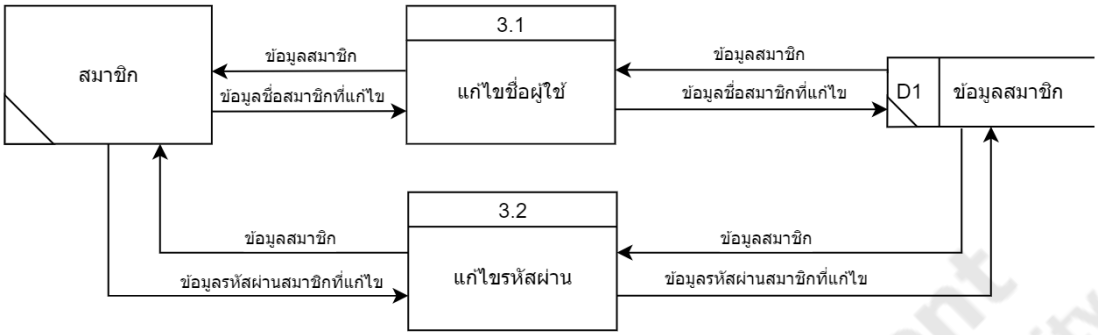
Data Flow Diagram คือ แผนภาพที่บอกถึงรายละเอียดของระบบ โดยเฉพาะข้อมูล ซึ่งทราบถึงการรับ / ส่งข้อมูล การประสานงานระหว่างกิจกรรมต่าง ๆ ในการดำเนินงานของระบบ ซึ่งเป็นแบบจำลองของระบบ แสดงถึงการไหลของข้อมูลทั้งข้อมูลนำเข้าและข้อมูลนำออก ระหว่างระบบกับผู้ใช้งานเอปพลิเคชันวิ่งคนเดียวด้วยกัน (Run Alone Together) ดังภาพประกอบที่ 3.3



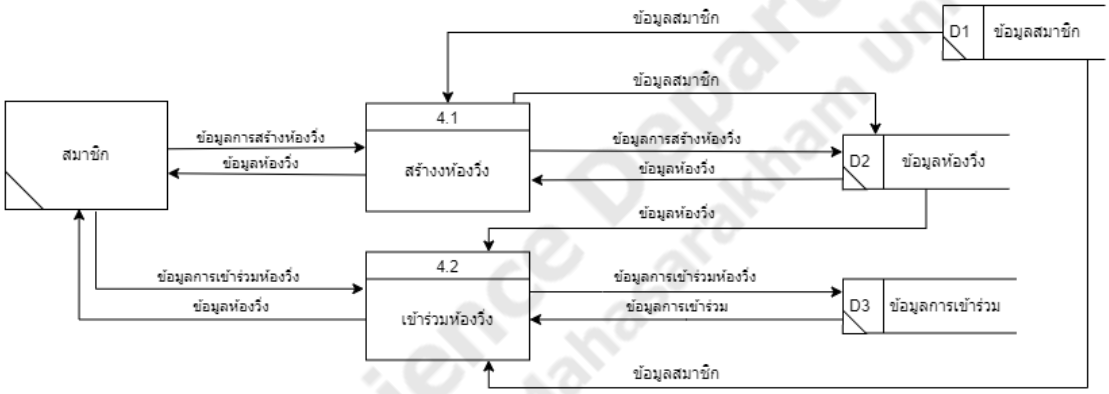
ภาพประกอบที่ 3.3 Data Flow Diagram level 1



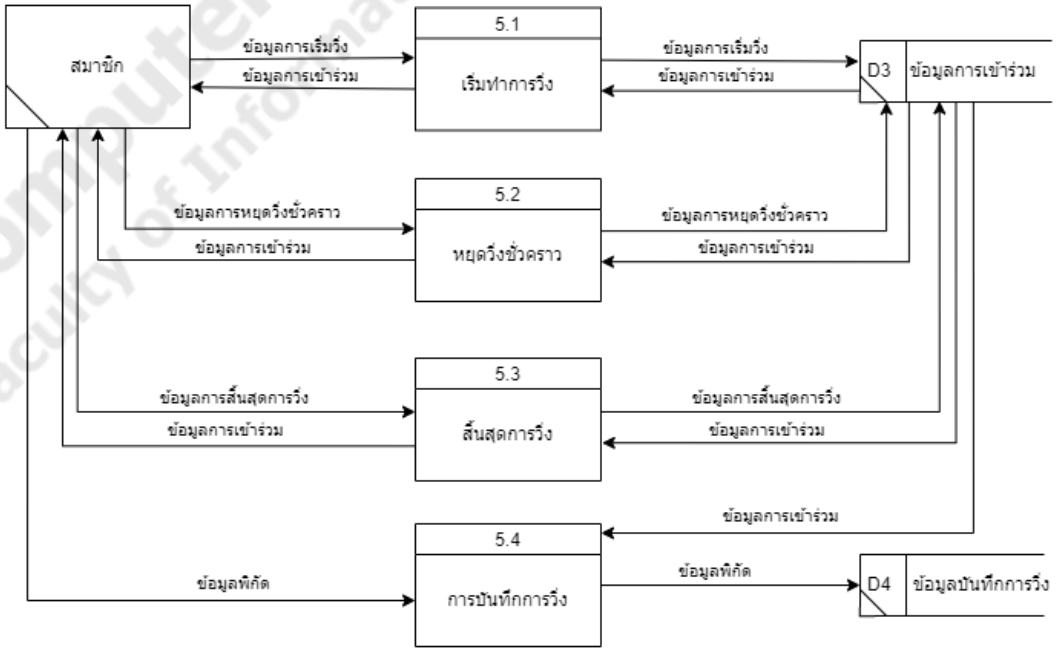
ภาพประกอบที่ 3.3 Data Flow Diagram level 1 (ต่อ)



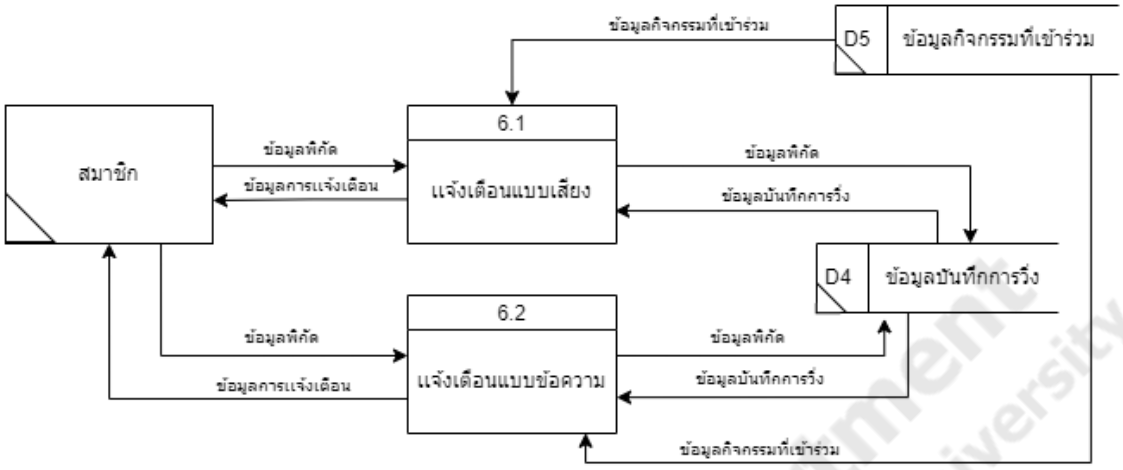
ภาพประกอบที่ 3.4 Data Flow Diagram Level 2 Process 3 จัดการข้อมูลผู้ใช้



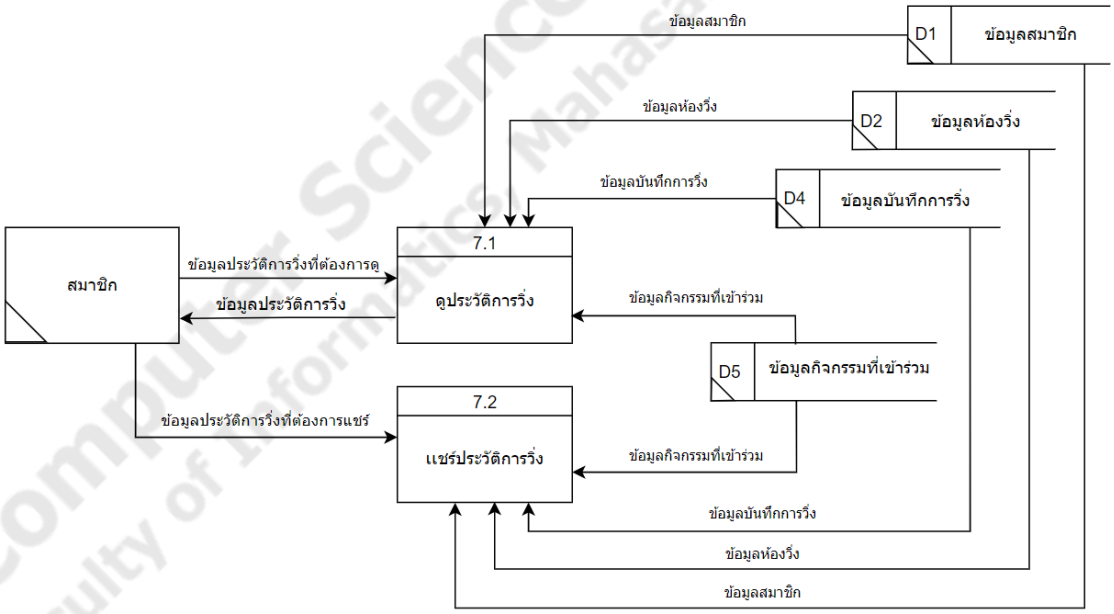
ภาพประกอบที่ 3.5 Data Flow Diagram Level 2 Process 4 เข้าร่วมกิจกรรม



ภาพประกอบที่ 3.4 Data Flow Diagram Level 2 Process 5 บันทึกการวิ่ง



ภาพประกอบที่ 3.6 Data Flow Diagram Level 2 Process 6 การแจ้งเตือน



ภาพประกอบที่ 3.7 Data Flow Diagram Level 2 Process 6 การแจ้งเตือน

3.5 External Entity Description

พจนานุกรมข้อมูลช่วยอธิบายการทานของแผนภาพการไหลของข้อมูล (Data Flow Dictionaries)

ตารางที่ 3.1 External Entity Description

Name	Description	Input Data Flow	Output Data Flow
ผู้ใช้งานทั่วไป	ผู้ใช้งานประเภททั่วไปที่จะต้องทำการสมัครสมาชิกในการใช้งานแอปพลิเคชันวิ่งคนเดียวด้วยกัน	- ข้อมูลสมาชิก	- ข้อมูลสมัครสมาชิก
สมาชิก	ผู้ใช้งานประเภทที่เป็นสมาชิกในการใช้งานแอปพลิเคชันวิ่งคนเดียวด้วยกัน	- ผลการเข้าสู่ระบบ - ข้อมูลสมาชิก - ข้อมูลห้องวิ่ง - ข้อมูลการเข้าร่วม - ข้อมูลบันทึกการวิ่ง - ข้อมูลการแจ้งเตือน	- ข้อมูลการเข้าสู่ระบบ - ข้อมูลชื่อสมาชิกที่แก้ไข - ข้อมูลรหัสผ่านของสมาชิกที่แก้ไข - ข้อมูลการสร้างห้องวิ่ง - ข้อมูลการเข้าร่วมห้องวิ่ง - ข้อมูลการเริ่มวิ่ง - ข้อมูลการหยุดวิ่งชั่วคราว - ข้อมูลการสิ้นสุดการวิ่ง - ข้อมูลการสิ้นสุดการวิ่ง - ข้อมูลพิกัด - ข้อมูลการแจ้งเตือนเสียง - ข้อมูลการแจ้งเตือนข้อความ - ข้อมูลประวัติการวิ่งที่ต้องการดู - ข้อมูลประวัติการวิ่งที่ต้องการแชร์

3.6 Data Flow and Data Structure Description

เป็นขั้นตอนการทำงานของระบบซึ่งทำให้เราทราบถึงการรับ-ส่งข้อมูลแสดงถึงการไหลของข้อมูลทั้ง ข้อมูลเข้า (Input) และข้อมูลส่งออก (Output) ระหว่างข้อมูลต้นทางถึงข้อมูลปลายทางโดยอธิบายข้อมูลและขั้นตอนในการดำเนินงานดังต่อไปนี้

ตารางที่ 3.2 Data Flow and Data Structure Description

Name	Description	Source	Destination	Data Structure
ข้อมูลสมัครสมาชิก	ตรวจสอบการสมัครสมาชิก	ผู้ใช้ทั่วไป	Process 1.0 สมัครสมาชิก	[รหัสผู้ใช้+ชื่อผู้ใช้+ บัญชีอีเมล+ รหัสผ่าน+ (เพศ)+ (รูปโปรไฟล์)+ วันเดือนปีเกิด | บัญชีเฟซบุ๊ก]
			D1 ข้อมูลสมาชิก	
ข้อมูลสมาชิก	เก็บข้อมูลสมาชิกในระบบ	D1 ข้อมูลสมาชิก	Process 1.0 สมัครสมาชิก	รหัสผู้ใช้+ชื่อผู้ใช้+ บัญชีอีเมล+ รหัสผ่าน+ (เพศ)+ (รูปโปรไฟล์)+ วันเดือนปีเกิด+บัญชีเฟซบุ๊ก
		Process 1.0 สมัครสมาชิก	ผู้ใช้ทั่วไป	
		D1 ข้อมูลสมาชิก	Process 2.0 เข้าสู่ระบบ	
		D1 ข้อมูลสมาชิก	Process 3.1 แก้ไขชื่อผู้ใช้	
		Process 3.1 แก้ไขชื่อผู้ใช้	สมาชิก	
		D1 ข้อมูลสมาชิก	Process 3.2 แก้ไขรหัสผ่าน	
		Process 3.2 แก้ไขรหัสผ่าน	สมาชิก	
		D1 ข้อมูลสมาชิก	Process 4.1 สร้างห้องวิ่ง	
		D1 ข้อมูลสมาชิก	Process 4.2 เข้าร่วมห้องวิ่ง	
		Process 4.1 สร้างห้องวิ่ง	D2 ข้อมูลห้องวิ่ง	
		D5 ข้อมูลกิจกรรมที่เข้าร่วม	Process 7.1 ดูประวัติการวิ่ง	

ตารางที่ 3.2 Data Flow and Data Structure Description (ต่อ)

Name	Description	Source	Destination	Data Structure
ข้อมูลสมาชิก	เก็บข้อมูลสมาชิกในระบบ	D1 ข้อมูลสมาชิก	Process 7.2 แชร์ประวัติการวิ่ง	
ข้อมูลการเข้าสู่ระบบ	ตรวจสอบการเข้าสู่ระบบ	สมาชิก	Process 2.0 เข้าสู่ระบบ	รหัสผู้ใช้+บัญชีอีเมล+รหัสผ่าน+เฟซบุ๊กไอดี
		Process 2.0 เข้าสู่ระบบ	D1 ข้อมูลสมาชิก	
ผลการเข้าสู่ระบบ	แสดงผลเมื่อเข้าสู่ระบบ	Process 2.0 เข้าสู่ระบบ	สมาชิก	[เข้าร่วมห้องวิ่ง | สร้างห้องวิ่ง | วิ่งคนเดียว+วิ่งคนเดียว]
ข้อมูลชื่อสมาชิกที่แก้ไข	ข้อมูลชื่อสมาชิกที่จะทำการแก้ไข	สมาชิก	Process 3.1 แก้ไขชื่อผู้ใช้	ชื่อผู้ใช้
		Process 3.1 แก้ไขชื่อผู้ใช	D1 ข้อมูลสมาชิก	
ข้อมูลรหัสผ่านของสมาชิกที่แก้ไข	ข้อมูลสมาชิกที่จะทำการแก้ไข	สมาชิก	Process 3.2 แก้ไขรหัสผ่าน	รหัสผ่านใหม่+รหัสผ่านเดิม
		Process 3.2 แก้ไขรหัสผ่าน	D1 ข้อมูลสมาชิก	
ข้อมูลการสร้างห้องวิ่ง	ข้อมูลที่ใช้ในการสร้างห้องวิ่ง	สมาชิก	Process 4.1 สร้างห้องวิ่ง	รหัสห้องวิ่ง+โค้ดห้องวิ่ง+ระยะทาง+ระยะเวลา
		Process 4.1 สร้างห้องวิ่ง	D2 ข้อมูลห้องวิ่ง	
ข้อมูลห้องวิ่ง	แสดงข้อมูลห้องวิ่ง	D2 ข้อมูลห้องวิ่ง	Process 4.1 สร้างห้องวิ่ง	รหัสห้องวิ่ง+โค้ดห้องวิ่ง+ระยะทาง+ระยะเวลา+จำนวนสมาชิกในห้องวิ่ง+ข้อความแจ้งเตือน+
		Process 4.1 สร้างห้องวิ่ง	สมาชิก	

ตารางที่ 3.2 Data Flow and Data Structure Description (ต่อ)

Name	Description	Source	Destination	Data Structure
ข้อมูลห้องวิ่ง	แสดงข้อมูลห้องวิ่ง	D2 ข้อมูลห้องวิ่ง	Process 4.2 เข้าร่วมห้องวิ่ง	อัตราเฉลี่ย+หยุดชั่วคราว+สิ้นสุดการวิ่ง
		Process 4.2 เข้าร่วมห้องวิ่ง	สมาชิก	
ข้อมูลการเข้าร่วมห้องวิ่ง	ข้อมูลที่ใช้ในการเข้าร่วมห้องวิ่ง	สมาชิก	Process 4.2 เข้าร่วมห้องวิ่ง	รหัสห้องวิ่ง+โค้ดห้องวิ่ง
		Process 4.2 เข้าร่วมห้องวิ่ง	D3 ข้อมูลการเข้าร่วม	
ข้อมูลการเข้าร่วม	เก็บข้อมูลการเข้าร่วมในระบบ	D3 ข้อมูลการเข้าร่วม	Process 4.2 เข้าร่วมห้องวิ่ง	รหัสการเข้าร่วม+ระยะทาง+ระยะเวลา+อัตราเฉลี่ย+วันที่+ชื่อสมาชิกในห้องวิ่ง+(รูปภาพพื้นหลัง)
		D3 ข้อมูลการเข้าร่วม	Process 5.1 เริ่มทำการวิ่ง	
		Process 5.1 เริ่มทำการวิ่ง	สมาชิก	
		D3 ข้อมูลการเข้าร่วม	Process 5.2 หยุดวิ่งชั่วคราว	
		Process 5.2 หยุดวิ่งชั่วคราว	สมาชิก	
		D3 ข้อมูลการเข้าร่วม	Process 5.3 สิ้นสุดการวิ่ง	
		Process 5.3 สิ้นสุดการวิ่ง	สมาชิก	
ข้อมูลการเริ่มวิ่ง	ข้อมูลเมื่อสมาชิกเริ่มทำการวิ่ง	สมาชิก	Process 5.1 เริ่มทำการวิ่ง	รหัสการเข้าร่วม+ระยะทาง+ระยะเวลา+อัตราเฉลี่ย
		Process 5.1 เริ่มทำการวิ่ง	D3 ข้อมูลการเข้าร่วม	

ตารางที่ 3.2 Data Flow and Data Structure Description (ต่อ)

Name	Description	Source	Destination	Data Structure
ข้อมูลการหยุดวิ่งชั่วคราว	ข้อมูลเมื่อสมาชิกทำการหยุดวิ่งชั่วคราว	สมาชิก	Process 5.2 หยุดวิ่งชั่วคราว	รหัสการเข้าร่วม+ระยะทาง+ระยะเวลา+อัตราเฉลี่ย
		Process 5.2 หยุดวิ่งชั่วคราว	D3 ข้อมูลการเข้าร่วม	
ข้อมูลการสิ้นสุดการวิ่ง	ข้อมูลเมื่อสมาชิกทำการสิ้นสุดการวิ่ง	สมาชิก	Process 5.3 สิ้นสุดการวิ่ง	รหัสการเข้าร่วม+ระยะทาง+ระยะเวลา+อัตราเฉลี่ย
		Process 5.3 สิ้นสุดการวิ่ง	D3 ข้อมูลการเข้าร่วม	
ข้อมูลพิกัด	บันทึกข้อมูลพิกัดของสมาชิก	สมาชิก	Process 5. การบันทึกการวิ่ง	รหัสข้อมูลการวิ่ง+รหัสผู้ใช้+ละติจูด+ลองจิจูด+ระยะทาง+ระยะเวลา
		Process 5.4 การบันทึกการวิ่ง	D4 ข้อมูลบันทึกการวิ่ง	
		สมาชิก	Process 6.1 แจ้งเตือนแบบเสียง	
		Process 6.1 แจ้งเตือนแบบเสียง	D4 ข้อมูลบันทึกการวิ่ง	
		สมาชิก	Process 6.2 แจ้งเตือนแบบข้อความ	
		Process 6.2 แจ้งเตือนแบบข้อความ	D4 ข้อมูลบันทึกการวิ่ง	

ตารางที่ 3.2 Data Flow and Data Structure Description (ต่อ)

Name	Description	Source	Destination	Data Structure
ข้อมูลการแจ้งเตือน	ข้อมูลการแจ้งเตือนแบบเสียงและข้อมูลการแจ้งเตือนแบบข้อความ	Process 6.1 แจ้งเตือนแบบเสียง	สมาชิก	รหัสผู้ใช้+ละติจูด+ลองจิจูด+ระยะทาง+ระยะเวลา
		Process 6.2 แจ้งเตือนแบบข้อความ	สมาชิก	
ข้อมูลกิจกรรมที่เข้าร่วม	รายงานข้อมูลกิจกรรมที่เข้าร่วม	D5 ข้อมูลกิจกรรมที่เข้าร่วม	Process 6.1 แจ้งเตือนแบบเสียง	รหัสกิจกรรมที่เข้าร่วม+รหัสผู้ใช้+ระยะทาง+ระยะเวลา+ชื่อสมาชิกในห้องวิ่ง
		D5 ข้อมูลกิจกรรมที่เข้าร่วม	Process 6.2 แจ้งเตือนแบบข้อความ	
		D1 ข้อมูลสมาชิก	Process 7.1 ประวัติการวิ่ง	
		D1 ข้อมูลสมาชิก	Process 7.2 แชร์ประวัติการวิ่ง	
ข้อมูลบันทึกการวิ่ง	บันทึกข้อมูลการวิ่งของสมาชิกในระบบ	D4 ข้อมูลบันทึกการวิ่ง	Process 6.1 แจ้งเตือนแบบเสียง	รหัสข้อมูลการวิ่ง+รหัสผู้ใช้+ละติจูด+ลองจิจูด ระยะทาง+ระยะเวลา+อัตราเฉลี่ย
		D4 ข้อมูลบันทึกการวิ่ง	Process 6.2 แจ้งเตือนแบบข้อความ	
		D4 ข้อมูลบันทึกการวิ่ง	Process 7.1 ดูประวัติการวิ่ง	
		D4 ข้อมูลบันทึกการวิ่ง	Process 7.2 แชร์ประวัติการวิ่ง	

ตารางที่ 3.2 Data Flow and Data Structure Description (ต่อ)

Name	Description	Source	Destination	Data Structure
ข้อมูลบันทึกการวิ่ง	บันทึกข้อมูลการวิ่งของสมาชิกในระบบ	D4 ข้อมูลบันทึกการวิ่ง	Process 6.1 แจ้งเตือนแบบเสียง	รหัสข้อมูลการวิ่ง+รหัสผู้ใช้+ละติจูด+ลองจิจูด ระยะทาง+ระยะเวลา+อัตราเฉลี่ย
		D4 ข้อมูลบันทึกการวิ่ง	Process 6.2 แจ้งเตือนแบบข้อความ	
		D4 ข้อมูลบันทึกการวิ่ง	Process 7.1 ดูประวัติการวิ่ง	
		D4 ข้อมูลบันทึกการวิ่ง	Process 7.2 แชร์ประวัติการวิ่ง	
ข้อมูลประวัติการวิ่งที่ต้องการดู	เรียกดูข้อมูลประวัติการวิ่งของสมาชิกที่ต้องการดู	สมาชิก	Process 7.1 ดูประวัติการวิ่ง	รหัสผู้ใช้+ระยะทาง+ระยะเวลา+อัตราเฉลี่ย+วันที่+ลำดับผลการวิ่ง
ข้อมูลประวัติการวิ่ง	แสดงข้อมูลประวัติการวิ่งของสมาชิกที่ต้องการดู	Process 7.1 ดูประวัติการวิ่ง	สมาชิก	รหัสผู้ใช้+ระยะทาง+ระยะเวลา+อัตราเฉลี่ย+วันที่+ลำดับผลการวิ่ง+(รูปภาพพื้นหลัง)
ข้อมูลประวัติการวิ่งที่ต้องการแชร์	แสดงข้อมูลประวัติการวิ่งของสมาชิกที่ต้องการแชร์	สมาชิก	Process 7.1 ดูประวัติการวิ่ง	รหัสผู้ใช้+ระยะทาง+ระยะเวลา+(รูปภาพพื้นหลัง)
		สมาชิก	Process 7.2 แชร์ประวัติการวิ่ง	

3.7 Data Store Description and Structure Description

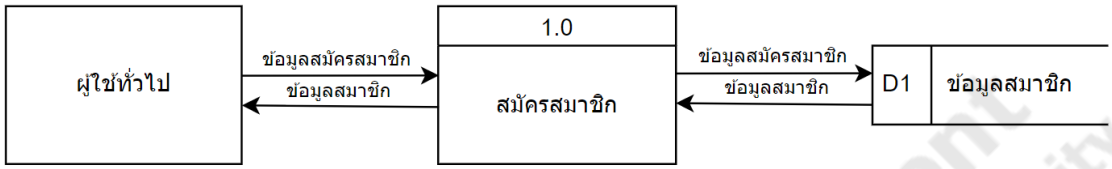
Data Store Description and Structure Description คือ การนำข้อมูลเข้าไปจัดการในฐานข้อมูลโดยมีการแยกออกเป็นแฟ้มข้อมูล

ตารางที่ 3.3 Data Store Description and Structure Description

ID	Data Store	Description	Data Structure
D1	ข้อมูลสมาชิก	เก็บรายละเอียดข้อมูลผู้ใช้งาน	รหัสผู้ใช้+ชื่อผู้ใช้+อีเมล+รหัสผ่าน+เพศ+รูปภาพ+วันเดือนปีเกิด+บัญชี Facebook
D2	ข้อมูลห้องวิ่ง	เก็บรายระเอียดข้อมูลห้องวิ่ง	รหัสห้องวิ่ง+โค้ดห้องวิ่ง+ระยะทาง+ระยะเวลา+จำนวนสมาชิกในห้องวิ่ง+ข้อความแจ้งเตือน+อัตราเฉลี่ย+หยุดชั่วคราว+สิ้นสุดการวิ่ง
D3	ข้อมูลการเข้าร่วม	เก็บรายละเอียดข้อมูลการเข้าร่วมห้องวิ่ง	ระยะทาง+ระยะเวลา+อัตราเฉลี่ย+วันที่+ชื่อ+สมาชิกในห้องวิ่ง+ลำดับผลการวิ่ง+(รูปภาพพื้นหลัง)
D4	ข้อมูลบันทึกการวิ่ง	เก็บรายละเอียดข้อมูลบันทึกการวิ่ง	ละติจูด+ลองจิจูด+ระยะทาง+ระยะเวลา
D5	ข้อมูลกิจกรรมที่เข้าร่วม	เก็บรายละเอียดกิจกรรมที่เข้าร่วม	ระยะทาง+ระยะเวลา+รางวัลที่ได้รับ+อัตราเฉลี่ย+วันที่+รูปภาพ

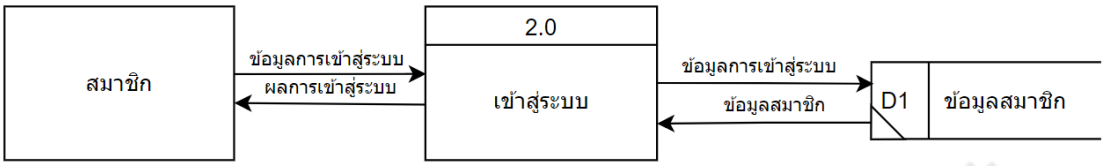
3.8 คำอธิบายการประมวลผล (Process Description)

Process 1.0 สมัครสมาชิก



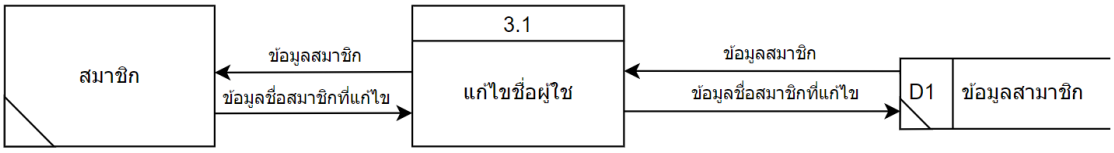
ID	1.0
Name	สมัครสมาชิก
Description	เพื่อทำการเพิ่มข้อมูลสมาชิกเข้าสู่ฐานข้อมูล
Input Data Flows	-ข้อมูลสมัครสมาชิก -ข้อมูลสมาชิก
Output Data Flows	-ข้อมูลสมัครสมาชิก -ข้อมูลสมาชิก
Process Description	เริ่มต้น 1. รับข้อมูลที่ต้องการสมัครสมาชิก 2. ตรวจสอบว่ามีข้อมูลครบถ้วนและถูกต้องหรือไม่ 2.1. ถ้าข้อมูลครบถ้วนและถูกต้องทุกรายการให้ไปทำข้อ (3) 2.2. ถ้าข้อมูลครบถ้วนแต่ไม่ถูกต้องให้แสดงข้อความ “กรุณาป้อนข้อมูลไม่ถูกต้อง” และกลับไปทำข้อ (1) 2.3. ถ้าข้อมูลไม่ครบถ้วนให้แสดงข้อความ “กรุณาป้อนข้อมูลให้ครบ” และกลับไปทำข้อ (1) 3. ตรวจสอบในแฟ้มข้อมูลสมาชิกว่ามีข้อมูลผู้ใช้หรือไม่ 3.1. ถ้ามีชื่อผู้ใช้แล้วให้แสดงข้อความว่า “มีบัญชีนี้อยู่แล้ว” และกลับไปทำข้อ (1) 3.4. ถ้าไม่มีชื่อผู้ใช้ในแฟ้มข้อมูลสมาชิกให้ทำการบันทึกข้อมูลสมาชิกลงในแฟ้มข้อมูลสมาชิก จบการทำงาน

Process 2.0 เข้าสู่ระบบ



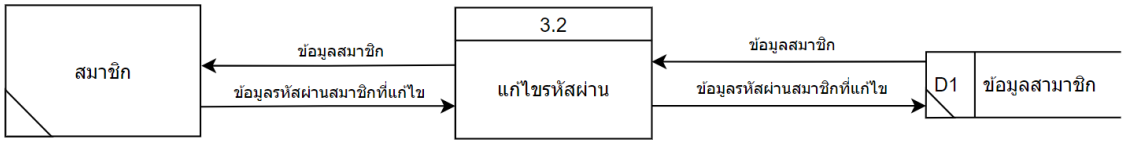
ID	2.0
Name	เข้าสู่ระบบ
Description	การเข้าสู่ระบบ
Input Data Flows	-ข้อมูลการเข้าสู่ระบบ -ข้อมูลสมาชิก
Output Data Flows	-ข้อมูลการเข้าสู่ระบบ -ผลการเข้าสู่ระบบ
Process Description	เริ่มต้น 1.รับข้อมูลการเข้าสู่ระบบ (อีเมล+รหัสผ่าน) 2. ตรวจสอบข้อมูลการเข้าสู่ระบบ 2.1 ถ้าป้อนข้อมูลครบถ้วนและถูกต้องให้ไปทำข้อ (3) 2.2 ถ้าป้อนข้อมูลไม่ถูกต้องให้แสดงข้อความ “ข้อมูลไม่ถูกต้อง กรุณาป้อนข้อมูลให้ถูกต้อง” และกลับไปทำข้อ (1) 3. ตรวจสอบข้อมูลผู้ใช้ในแฟ้มข้อมูลสมาชิก 3.1 ถ้ามีข้อมูลผู้ใช้ในแฟ้มข้อมูลสมาชิกให้แสดงข้อมูลผลการเข้าสู่ระบบให้กับสมาชิก 3.2 ถ้าไม่มีข้อมูลผู้ใช้ในแฟ้มข้อมูลสมาชิกให้แสดงข้อความ “ไม่พบชื่อบัญชีผู้ใช้นี้ กรุณาตรวจสอบอีเมลหรือรหัสผ่านให้ถูกต้อง” จบการทำงาน

Process 3.1 แก้ไขชื่อผู้ใช้



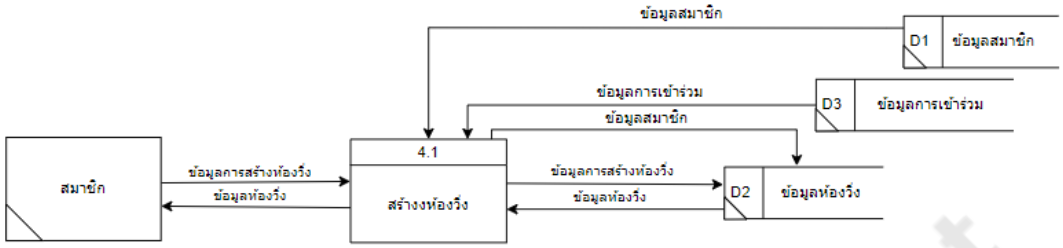
ID	3.1
Name	แก้ไขชื่อผู้ใช้
Description	การแก้ไขชื่อผู้ใช้
Input Data Flows	-ข้อมูลชื่อสมาชิกที่แก้ไข -ข้อมูลสมาชิก
Output Data Flows	-ข้อมูลชื่อสมาชิกที่แก้ไข -ข้อมูลสมาชิก
Process Description	เริ่มต้น 1.สมาชิกป้อนข้อมูลชื่อสมาชิกที่แก้ไข 2. ตรวจสอบว่าข้อมูลครบถ้วนและถูกต้องหรือไม่ 2.1 ถ้าข้อมูลชื่อที่ต้องการแก้ไขครบถ้วน และถูกต้องให้บันทึกข้อมูลชื่อที่ต้องการแก้ไขลงในฐานข้อมูล 2.2 ถ้าข้อมูลไม่ครบถ้วนให้แสดงข้อความ “กรุณาป้อนข้อมูลให้ครบถ้วน” และกลับไปทำข้อ (1) จบการทำงาน

Process 3.2 แก้ไขรหัสผ่าน



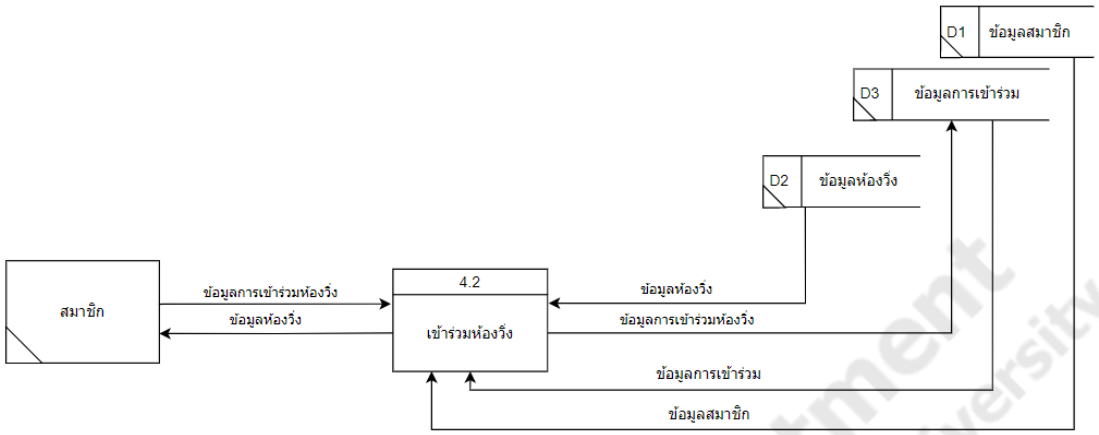
ID	3.2
Name	แก้ไขรหัสผ่าน
Description	การแก้ไขรหัสผ่าน
Input Data Flows	-ข้อมูลรหัสผ่านสมาชิกที่แก้ไข -ข้อมูลสมาชิก
Output Data Flows	-ข้อมูลรหัสผ่านสมาชิกที่แก้ไข -ข้อมูลสมาชิก
Process Description	เริ่มต้น 1. สมาชิกป้อนข้อมูลรหัสผ่านสมาชิกที่แก้ไข 2. สมาชิกตรวจสอบข้อมูลรหัสผ่าน 3. สมาชิกป้อนเพื่อยืนยันรหัสผ่านอีกครั้ง 3.1 ถ้าข้อมูลรหัสผ่านที่สมาชิกยืนยันถูกต้องให้บันทึกข้อมูลสมาชิกที่แก้ไขลงในฐานข้อมูล 3.2 ถ้าข้อมูลรหัสผ่านที่สมาชิกยืนยันไม่ถูกต้องหรือไม่ตรงกับรหัสผ่านที่สมาชิกตั้งให้แสดงข้อ “ข้อมูลไม่ถูกต้อง” และกลับไปทำข้อ (2) จบการทำงาน

Process 4.1 สร้างห้องวิ่งคนเดียว



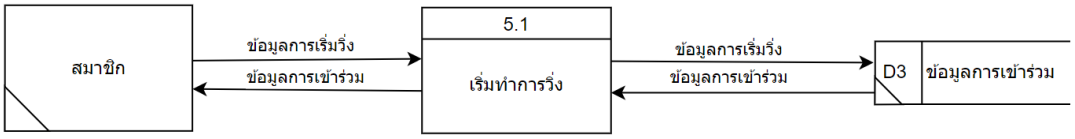
ID	4.1
Name	สร้างห้องวิ่ง
Description	การสร้างห้องเพื่อทำกิจกรรมวิ่ง
Input Data Flows	-ข้อมูลการสร้างห้องวิ่ง -ข้อมูลสมาชิก -ข้อมูลห้องวิ่ง -ข้อมูลการเข้าร่วม
Output Data Flows	-ข้อมูลห้องวิ่ง -ข้อมูลสมาชิก -ข้อมูลการสร้างห้องวิ่ง
Process Description	เริ่มต้น 1. สมาชิกกำหนดระยะทางของกิจกรรม 2. สมาชิกปุ่มเพื่อสร้างห้องกิจกรมม 2.1 ถ้าสมาชิกต้องการที่จะสร้างห้องกิจกรรมเพื่อทำกิจกรรมวิ่งคนเดียวสมาชิกจะต้องกดที่ปุ่ม เริ่ม 2.2 ถ้าสมาชิกต้องการที่จะสร้างห้องกิจกรรมเพื่อทำกิจกรรมวิ่งร่วมกันกับเพื่อนๆ สมาชิกจะต้องกดที่ปุ่ม สร้างห้อง 3. สมาชิกกำหนดระยะเวลาของกิจกรรม 3.1 ถ้าสมาชิกต้องการจะสร้างห้องต่อให้กดที่ปุ่ม OK 3.2 ถ้าสมาชิกต้องการที่จะแก้ไขระยะทางหรือประเภทของห้องกิจกรรมให้กลับไปทำข้อ (1) จบการทำงาน

Process 4.2 สร้างห้องวิ่งคนเดียว



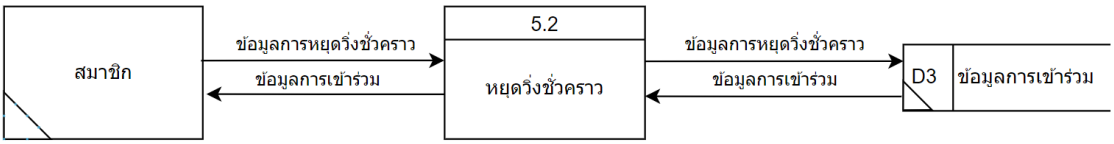
ID	4.2
Name	เข้าร่วมห้องวิ่ง
Description	การเข้าร่วมห้องวิ่ง
Input Data Flows	-ข้อมูลการเข้าร่วมห้องวิ่ง -ข้อมูลสมาชิก -ข้อมูลห้องวิ่ง -ข้อมูลการเข้าร่วม
Output Data Flows	-ข้อมูลห้องวิ่ง -ข้อมูลการเข้าร่วมห้องวิ่ง
Process Description	เริ่มต้น 1. สมาชิกกดที่ปุ่ม Join เพื่อเข้าร่วมห้องวิ่งแบบสาธารณะ 2. รับโค้ดห้องวิ่งที่ได้รับจากสมาชิกที่เป็นคนสร้างห้องวิ่ง 3. ตรวจสอบโค้ดห้องวิ่งที่สมาชิกจะเข้าร่วมห้องวิ่งที่เพื่อนสร้างว่าถูกต้องครบถ้วนหรือไม่ 3.1 ถ้าข้อมูลถูกต้องครบถ้วนสมาชิกจะสามารถดำเนินการเข้าร่วมห้องวิ่งได้ทันที 3.2 ข้อมูลไม่ถูกต้องให้แสดง “กรุณาใส่โค้ดห้องวิ่งใหม่อีกครั้ง” กลับไปทำข้อ (1) จบการทำงาน

Process 5.1 เริ่มทำการวิ่ง



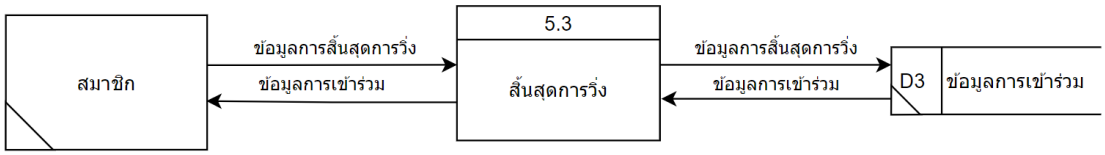
ID	5.1
Name	เริ่มทำการวิ่ง
Description	การเริ่มวิ่ง
Input Data Flows	-ข้อมูลการวิ่ง -ข้อมูลการเข้าร่วม
Output Data Flows	-ข้อมูลการวิ่ง -ข้อมูลการเข้าร่วม
Process Description	เริ่มต้น 1. รับข้อมูลการเริ่มวิ่งหรือการเข้าร่วมห้องวิ่ง 2. ตรวจสอบระยะทางระยะเวลาที่กำหนดและโค้ดห้องวิ่งที่จะเข้าร่วมว่าครบและถูกต้องหรือไม่ 2.1 ถ้าข้อมูลครบถ้วนก็จะสามารถทำการเริ่มทำการวิ่งได้ทันที 2.2 ถ้าข้อมูลไม่ครบถ้วนให้แสดง “กรุณาป้อนข้อมูลให้ครบก่อนเริ่มทำการวิ่ง” กลับไปที่ข้อ (1) จบการทำงาน

Process 5.2 หยุดวิ่งชั่วคราว



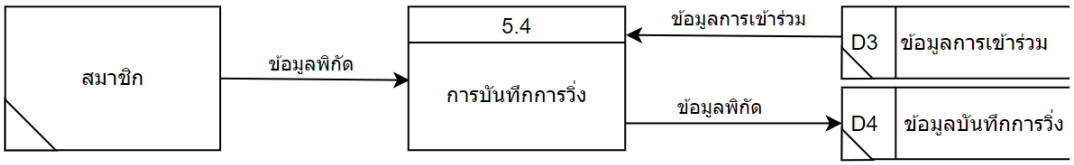
ID	5.2
Name	หยุดวิ่งชั่วคราว
Description	การหยุดวิ่งชั่วคราวระหว่างการทำการวิ่ง
Input Data Flows	-ข้อมูลการหยุดวิ่งชั่วคราว -ข้อมูลการเข้าร่วม
Output Data Flows	-ข้อมูลการหยุดวิ่งชั่วคราว -ข้อมูลการเข้าร่วม
Process Description	เริ่มต้น 1.รับข้อมูลการหยุดวิ่งชั่วคราว 2. ตรวจสอบการหยุดวิ่งชั่วคราวระหว่างทำการวิ่ง 2.1 ถ้าสมาชิกกดหยุดวิ่งชั่วคราวจะทำการหยุดระยะทาง ระยะเวลาและอัตราเฉลี่ยที่ใช้ในการวิ่ง 2.2 ถ้าสมาชิกกดดำเนินการต่อระยะทาง ระยะเวลาและอัตราเฉลี่ยที่ใช้ในการวิ่งก็จะทำการดำเนินต่อไป 1.3 ถ้าสมาชิกไม่กดหยุดวิ่งชั่วคราวระยะทาง ระยะเวลาและอัตราเฉลี่ยที่ใช้ในการวิ่งก็จะทำการดำเนินต่อไป จบการทำงาน

Process 5.3 สิ้นสุดการวิ่ง



ID	5.3
Name	สิ้นสุดการวิ่ง
Description	การสิ้นสุดการวิ่ง
Input Data Flows	-ข้อมูลการสิ้นสุดการวิ่ง -ข้อมูลการเข้าร่วม
Output Data Flows	-ข้อมูลการสิ้นสุดการวิ่ง -ข้อมูลการเข้าร่วม
Process Description	เริ่มต้น 1.รับข้อมูลการสิ้นสุดการวิ่ง 2.ตรวจสอบเมื่อสมาชิกสิ้นสุดการวิ่ง 2.1 ถ้าสมาชิกกดสิ้นสุดการวิ่งก็จะทำการหยุดและสิ้นสุดระยะทาง ระยะเวลาและอัตราเฉลี่ยที่ใช้ในการวิ่ง 2.2 ถ้าสมาชิกวิ่งครบระยะทาง ระยะเวลาที่กำหนดก็จะทำการหยุดและสิ้นสุดระยะทาง ระยะเวลาและอัตราเฉลี่ยที่ใช้ในการวิ่ง 2.3 ถ้าสมาชิกไม่กดหยุดวิ่งชั่วคราวระยะทาง ระยะเวลาและอัตราเฉลี่ยที่ใช้ในการวิ่งก็จะดำเนินการต่อไปจะกว่าจะครบระยะทาง ระยะเวลาที่กำหนดจึงจะทำการสิ้นสุดการวิ่งจบการทำงาน จบการทำงาน

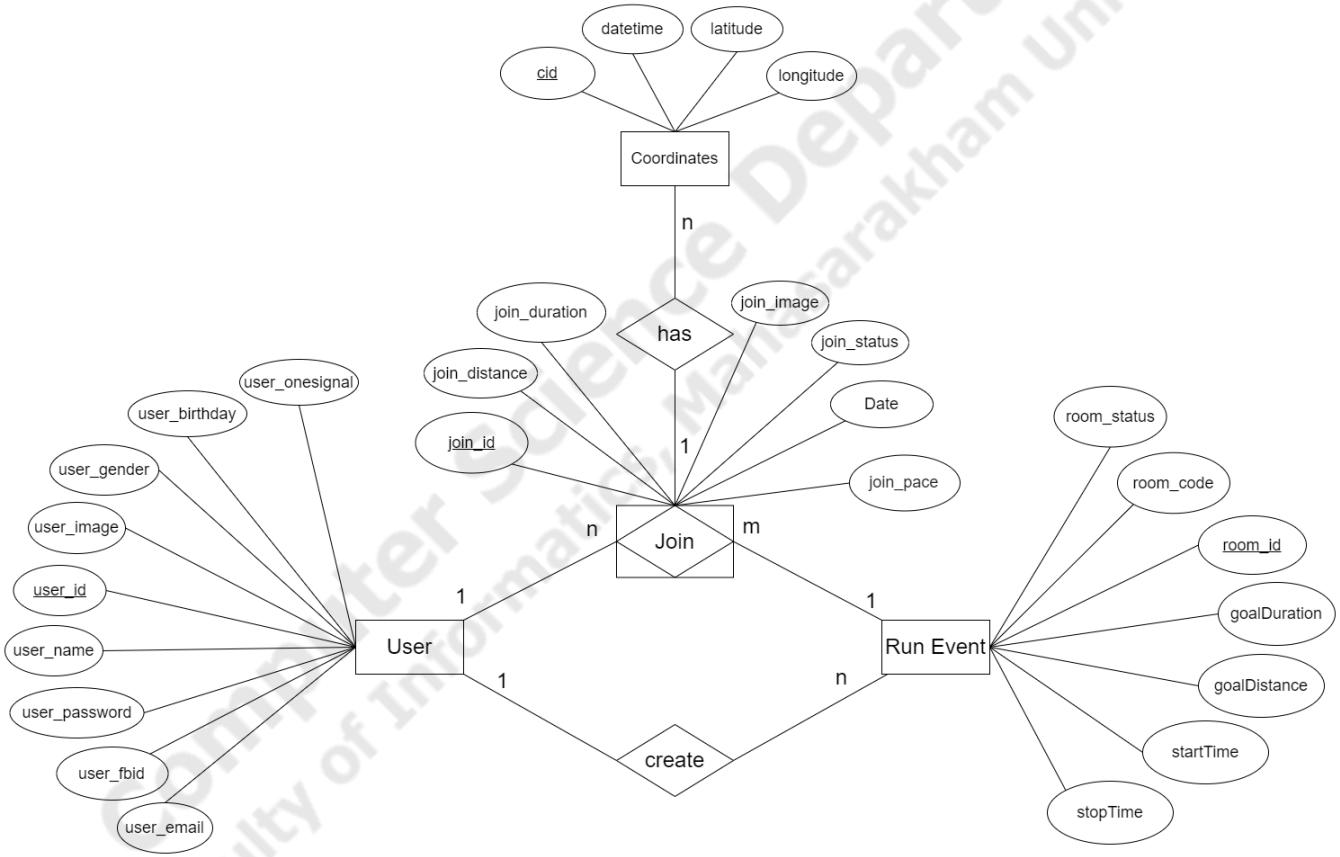
Process 5.4 สิ้นสุดการวิ่ง



ID	5.4
Name	การบันทึกการวิ่ง
Description	การบันทึกการวิ่ง
Input Data Flows	-ข้อมูลพิกัด -ข้อมูลการเข้าร่วม
Output Data Flows	-ข้อมูลพิกัด
Process Description	เริ่มต้น 1.ข้อมูลการเข้าร่วมและข้อมูลพิกัด 2. ตรวจสอบข้อมูลการเข้าร่วมเมื่อจบการวิ่ง 2.1 ถ้าข้อมูลถูกต้องครบถ้วนก็จะทำการบันทึกข้อมูลการวิ่งในระบบ 2.2 ถ้าข้อมูลไม่ถูกต้องครบถ้วนให้แสดง “ไม่พบข้อมูล” กลับไปทำข้อ (1) จบการทำงาน

3.9 แผนภาพ Entity Relationship Diagram (ER-Diagram)

แบบจำลองที่ใช้อธิบายโครงสร้างของฐานข้อมูลที่ใช้ในแอปพลิเคชันซึ่งเขียนออกมาในลักษณะของรูปภาพ การอธิบายโครงสร้างและความสัมพันธ์ของข้อมูล



ภาพประกอบที่ 3.8 Entity Relationship Diagram

3.10 การออกแบบฐานข้อมูล (Database Design)

ตารางที่ 3.4 ตารางข้อมูลสมาชิก (User)

Column	Type	Description	Example	Constraint
user_id	int	รหัสสมาชิก	1	PK
user_name	varchar(100)	ชื่อสมาชิก	นิสา พิมพ์	Not null
user_email	varchar(50)	อีเมล (ใช้เป็น Username เข้าสู่ระบบแอปพลิเคชัน)	nisapim@gmail.com	Not null
user_password	varchar(50)	รหัสผ่าน (Password)	1111	Not null
user_fbid	varchar(255)	ไอดีสำหรับ Facebook	1125478941250145	Not null
user_gender	varchar(50)	เพศ	หญิง	Not null
user_birthday	date	วันเกิด	2000-12-05	Not null
user_image	varchar(255)	ไฟล์รูปภาพ	C:\Users\ASUS\Downloads.jpg	Not null
user_onesignal	varchar(255)	playerID สำหรับข้อความแจ้งเตือน	e725278a-6e30-11ec-a8a9-32f918caeeed	Not null

ตารางที่ 3.5 ตารางการเข้าร่วมกิจกรรมห้องวิ่ง (Join)

Column	Type	Description	Example	Constraint
join_id	int	รหัสการเข้าร่วมห้องวิ่ง	1	PK
join_distance	int(100)	ระยะทางที่ทำการวิ่ง (กม.)	5	Null

ตารางที่ 3.5 ตารางการเข้าร่วมกิจกรรมห้องวิ่ง (Join) (ต่อ)

Column	Type	Description	Example	Constraint
join_duration	time	ระยะเวลาที่ทำกิจกรรม (ชม.)	1:20:00	Null
join_date	date	วันที่ทำกิจกรรม	2021-12-12	Null
join_pace	double	ความเร็วเฉลี่ย (กิโลเมตรต่อนาที)	0.042	Null
join_status	varchar(10)	สถานะห้องวิ่ง	off	Null
join_image	varchar(255)	ไฟล์รูปภาพ(เพิ่มหลังจบกิจกรรม)	C:\Users\ASUS\Downloads.jpg	Null
user_id	int	รหัสสมาชิก	1	FK reference from User(user_id) ON DELETE CASCADE ON UPDATE CASCADE
room_id	int	รหัสห้องวิ่ง	1	FK reference from RunEvent(room_id) ON DELETE CASCADE ON UPDATE CASCADE

ตารางที่ 3.6 ตารางกิจกรรมห้องวิ่ง (RunEvent)

Column	Type	Description	Example	Constraint
room_code	varchar(255)	รหัสห้อง (ที่ใช้สำหรับเข้าร่วมห้องวิ่ง)	7TQArs	Not null
room_id	int	รหัสห้องวิ่ง	1	PK
goalDuration	time	ระยะเวลาของกิจกรรม (ชม.)	3:30:00	Not null
goalDistance	int(100)	ระยะทางของกิจกรรม (กม.)	5	Not null
startTime	datetime	วันเวลาเริ่มกิจกรรม	2021-12-12 15:00:00	Not null
stopTime	datetime	วันเวลาสิ้นสุดกิจกรรม	2021-12-12 18:00:00	Not null
room_status	varchar(255)	สถานะของห้องวิ่ง	public	
user_id	int	รหัสสมาชิก	1	FK reference from User(user_id) ON DELETE CASCADE ON UPDATE CASCADE

ตารางที่ 3.7 ตารางพิกัด (Coordinates)

Column	Type	Description	Example	Constraint
cid	int	รหัสข้อมูลการวิ่ง	1	PK
datetime	datetime	วันเวลา	2022-01-08T09:11:24	Not null
latitude	double	ละติจูด	16.3167	Not null

ตารางที่ 3.7 ตารางพิกัด (Coordinates) (ต่อ)

Column	Type	Description	Example	Constraint
longitude	double	ลองจิจูด	103.3	Not null
join_id	int	รหัสการเข้าร่วมห้องวิ่ง	1	FK reference from Join(join_id) ON DELETE CASCADE ON UPDATE CASCADE

3.11 ทฤษฎีที่เกี่ยวข้องกับการออกแบบ

3.11.1 Web Service

Web Services คือระบบซอฟต์แวร์ที่ออกแบบมา เพื่อสนับสนุนการแลกเปลี่ยนข้อมูลระหว่างเครื่องคอมพิวเตอร์ผ่านระบบเครือข่าย โดยที่ภาษาที่ใช้ในการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ คือ XML เว็บเซอร์วิสมีอินเทอร์เฟส ที่ใช้อธิบายรูปแบบข้อมูลที่เครื่องคอมพิวเตอร์ประมวลผลได้ ลักษณะการให้บริการของ Web Services นั้น จะถูกเรียกใช้งานจาก application อื่นๆ ในรูปแบบ RPC (Remote Procedure Call) ซึ่งการให้บริการจะมีเอกสารที่อธิบายคุณสมบัติของบริการกำกับไว้ โดยภาษาที่ถูกใช้เป็นสื่อในการแลกเปลี่ยนคือ XML ทำให้เราสามารถเรียกใช้ Component ใด ๆ ก็ได้ ใน ระบบ หรือ Platform ใด ๆ ก็ได้ บน Protocol HTTP ซึ่งเป็น Protocol สำหรับ World Wide Web หรืออินเทอร์เน็ต อันเป็นช่องทางที่ได้รับการยอมรับทั่วโลกในการติดต่อสื่อสารกันระหว่าง Application กับ Application ในปัจจุบัน โดยได้นำ web service มาใช้ในการเขียนเพื่อรับ – ส่งข้อมูลหน้าเข้าสู่ระบบของแอปพลิเคชัน ดังรูปภาพ เป็นการเขียนเพื่อ insert ค่าหรือข้อมูลที่รับมาจากผู้ใช้งานและทการบันทึกลงฐานข้อมูลดาต้าเบสโดยผ่าน web service

```
3  public class LogInDTO {
4      private String userEmail;
5      private String userPassword;
6      private String userFbid;
```

ภาพประกอบที่ 3.9 Data Transfer Object (DTO) ส่วนเข้าสู่ระบบ

บรรทัดที่ 3 สร้าง Class LogInDTO เพื่อประกาศตัวแปรไว้เก็บข้อมูลจากระบบหน้าบ้าน

บรรทัดที่ 4-6 ประการตัวแปร userEmail userPassword และ userFbid ที่มีชนิดข้อมูลเป็น String เพื่อเก็บค่าที่รับมาจากระบบหน้าบ้าน

```
 9 public interface UserRepositories extends JpaRepository<User, Integer>{
10     public List<User>findByuserFbid(String fkId);
11     public List<User>findByUserEmailAndUserPassword(
12             String email, String password);
13 }
14
```

ภาพประกอบที่ 3.10 Repositories ของการเข้าสู่ระบบด้วย Facebook และ Email

บรรทัดที่ 9 สร้าง interface UserRepositories เพื่อเก็บคำสั่ง JPQL ส่วน User และกำหนดชนิดข้อมูลให้เป็น Integer ตาม database

```
12 @Service
13 public class LoginServices {
14     @Autowired
15     UserRepositories userRepositories;
16
17     public List <User> getAllUsers() {
18         List<User> users = userRepositories.findAll();
19         return users;
20     }
21
22     public List<User> getUserByFbID(LogInDTO logInDTO) {
23         List<User> users = userRepositories.findByuserFbid(logInDTO.getUserFbid());
24         for (User  user : users) {
25             user.setUserPassword(null);
26         }
27         return users;
28     }
29     public List<User> getUserByEmailAndPassword(LogInDTO logInDTO){
30         return userRepositories.findByUserEmailAndUserPassword(logInDTO.getUserEmail(), logInDTO.getUserPassword());
31
32     }
33 }
```

ภาพประกอบที่ 3.11 ของการเข้าสู่ระบบด้วย Facebook และ Email

บรรทัดที่ 13 สร้าง Service ชื่อ LoginService

บรรทัดที่ 14 ระบุ @Autowired เพื่อเป็นการ Connect ไปยัง database

บรรทัดที่ 15 คือการผูก Repositories เพื่อนำมาใช้ใน Service

บรรทัดที่ 17-20 สร้าง Method getAllUser เพื่อแสดงข้อมูลผู้ใช้งานทั้งหมด

บรรทัดที่ 22-28 สร้าง Method getUserByFbID เพื่อแสดงข้อมูล Facebook ID ของผู้ใช้งาน

บรรทัดที่ 29-33 สร้าง Method getUserByEmailAndPassword เพื่อทำการแสดงข้อมูลของ Email และ Password ของผู้ใช้งาน

```
18 @RestController
19 public class UserController {
20     @Autowired
21     LoginServices loginServices;
22
23     @GetMapping("/users")
24     public List<User> getAllUsers(){
25         return loginServices.getAllUsers();
26     }
27
28     //loginFacebook
29     @PostMapping(value = "/loginByFacebook",
30             consumes = MediaType.APPLICATION_JSON_VALUE,
31             produces = MediaType.APPLICATION_JSON_VALUE)
32     public List<User> getUserByFacebookId(@RequestBody LogInDTO logInDTO){
33         System.out.println(logInDTO.getUserFbid());
34         return loginServices.getUserByFbID(logInDTO);
35     }
```

ภาพประกอบที่ 3.12 Controller การเข้าสู่ระบบด้วย Facebook

บรรทัดที่ 18-19 ใน Class UserController ระบุ @RestController เพื่อให้ทราบว่า Class นี้คือ Restful Controller

บรรทัดที่ 20-21 ระบุ @Autowired loginService เพื่อเป็นการ Connect ไปยัง database

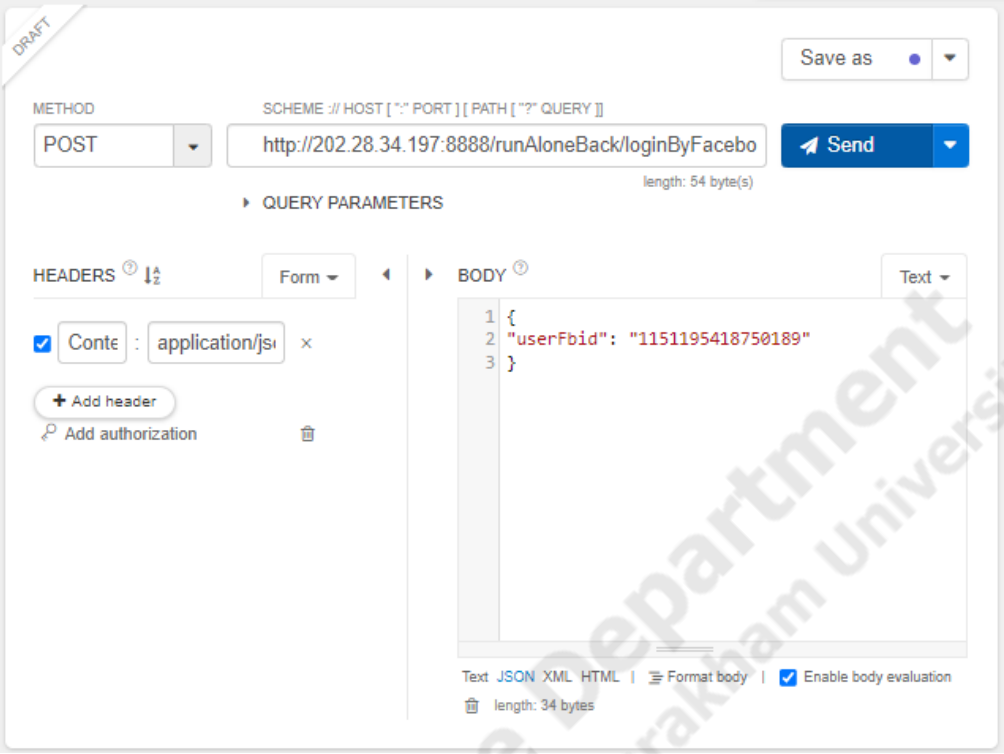
บรรทัดที่ 23 สร้าง Controller โดยใช้ @GetMapping เพื่อระบุ Path ให้กับ Controller

บรรทัดที่ 24-26 สร้าง Method ของ Path ชื่อ getAllUsers เพื่อเป็นการเรียก Method ในไฟล์ LoginService

บรรทัดที่ 29 สร้าง Controller โดยใช้ @PostMapping เพื่อระบุ Path ให้กับ Controller

บรรทัดที่ 30-31 กำหนดชนิดให้ consumes และ produces เป็น JSON เพื่อระบุว่าข้อมูลมีรูปแบบ JSON

บรรทัดที่ 32 เรียกใช้ Method getUserByFacebookId และส่งค่า DTO ให้กับ Method



ภาพประกอบที่ 3.13 ตัวอย่างการเรียกใช้ Service การเข้าสู่ระบบด้วย Facebook

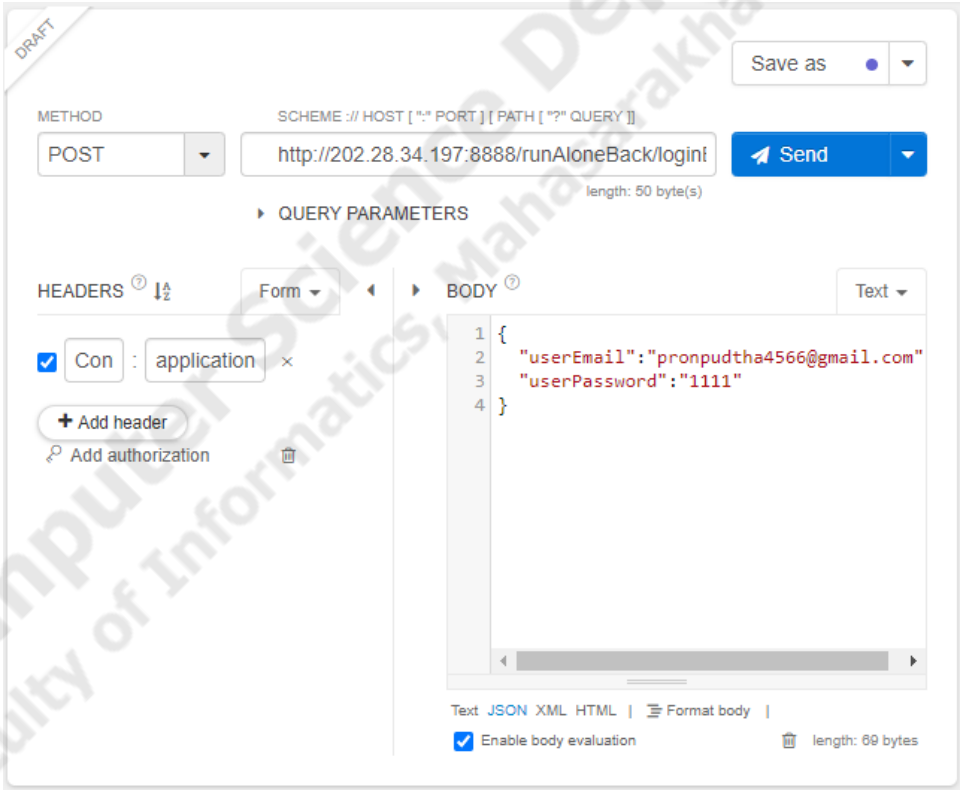
ภาพประกอบที่ 3.14 ตัวอย่างข้อมูลผู้ใช้งานที่ได้จากการเรียกใช้ Service

```
37      //loginEmail And Password
38      @PostMapping(value = "/loginByEmil",
39              consumes = MediaType.APPLICATION_JSON_VALUE,
40              produces = MediaType.APPLICATION_JSON_VALUE)
41      public ResponseEntity<?> getUserByEmailAndPassword(@RequestBody LogInDTO logInDTO){
42          List<User> users = loginServices.getUserByEmailAndPassword(logInDTO);
43          if (users.size() == 0) {
44              return new ResponseEntity<>(HttpStatus.UNAUTHORIZED);
45          }
46          return new ResponseEntity<>(users, HttpStatus.OK);
47      }
48  }
```

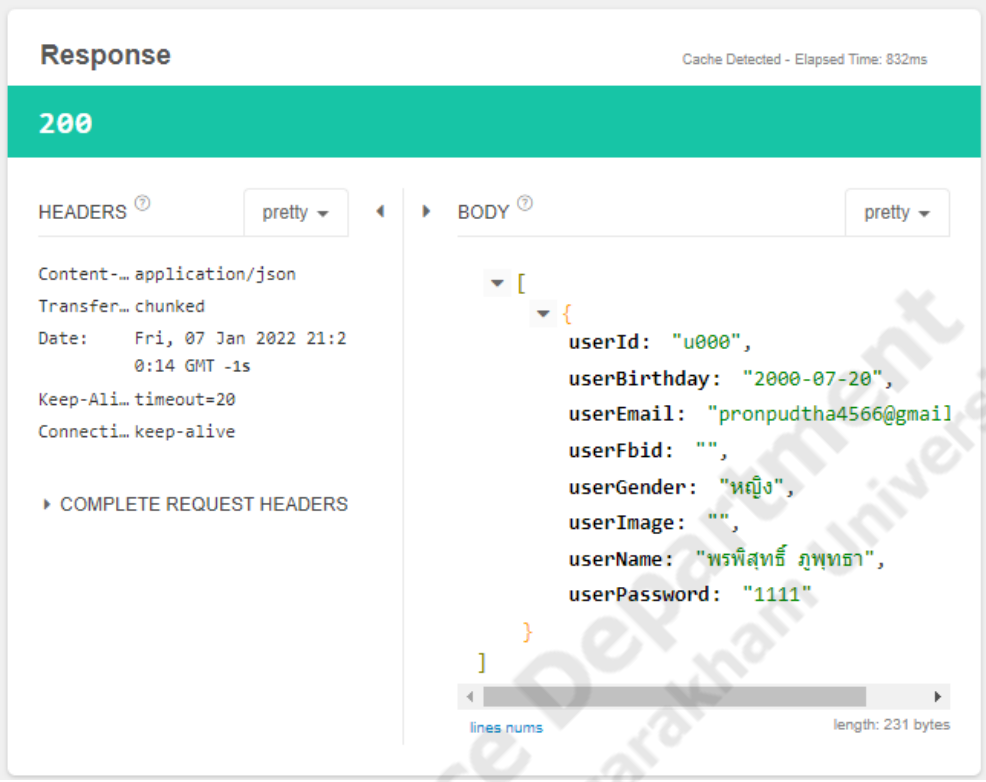
ภาพประกอบที่ 3.15 Controller การเข้าสู่ระบบด้วย Email และ Password

บรรทัดที่ 38 สร้าง Controller โดยใช้ @PostMapping เพื่อระบุ Path ให้กับ Controller

บรรทัดที่ 41 เรียกใช้ Method getUserByEmailAndPassword และส่งค่า DTO ให้กับ Method



ภาพประกอบที่ 3.16 ตัวอย่างการเรียกใช้ Service การเข้าสู่ระบบด้วย Email และ Password



ภาพประกอบที่ 3.17 ตัวอย่างข้อมูลผู้ใช้งานที่ได้จากการเรียกใช้ Service

```
274  Future<void>LoginByFacebook(String fid) async{
275    var json = {"userFbid": fid};
276    String value = await rootBundle.loadString('assets/config/config.yaml');
277    dynamic conf = loadYaml(value);
278    String url = conf['server']['host'];
279    String port = conf['server']['port'];
280    var response =  await http.post(Uri.parse("http://$url:$port/runAloneBack/loginByFacebook"),
281    headers: <String, String>{
282      'Content-Type':'application/json; charset=UTF-8',
283    },
284     body: jsonEncode(json));
285
286    log(response.statusCode.toString());
287    if(response.statusCode == 200){
288      ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("Login Success")));
289      Navigator.push(context, MaterialPageRoute(builder: (context) => LandingPage()));
290    }else{
291      ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("Login Error")));
292    }
293  }
```

ภาพประกอบที่ 3.18 ฟังก์ชันการเชื่อมต่อ Service การเข้าสู่ระบบด้วย Facebook

บรรทัดที่ 274 สร้างฟังก์ชันชื่อ LoginByFacebook เพื่อเชื่อมต่อ service login และประกาศตัวแปร fid เพื่อเอาไว้เก็บค่า Facebook id

บรรทัดที่ 275 ประกาศตัวแปร json เพื่อนำมาเก็บค่า Facebook id ในตัวแปร fid

บรรทัดที่ 276 ประกาศตัวแปร value เพื่อเก็บค่าที่อยู่ใน Path ของไฟล์ config.yaml หรือไฟล์ที่เก็บค่าข้อมูลของ Server แล้วโหลดออกมาเป็น String

บรรทัดที่ 277 ประกาศตัวแปร conf เพื่อดึงค่าจากตัวแปร value แล้วโหลดออกมาเป็นไฟล์ yaml

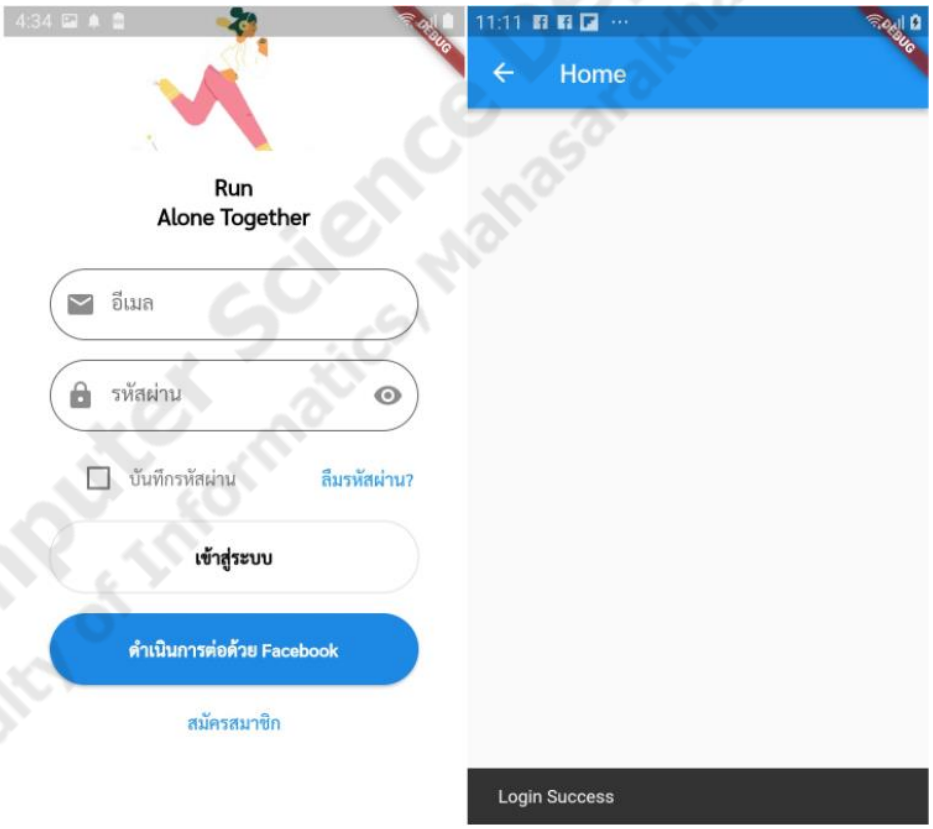
บรรทัดที่ 278 ประกาศตัวแปร url เพื่อดึงคค่าจาก Server host

บรรทัดที่ 279 ประกาศตัวแปร port เพื่อดึงค่าจาก Server port

บรรทัดที่ 280 เรียก Service โดยใช้การ Post ในการส่งข้อมูลที่เป็น Json

บรรทัดที่ 287 ถ้า response มีค่า statusCode เท่ากับ 200 แสดงว่าการเข้าสู่ระบบผ่าน Facebook ID สำเร็จและจะมีข้อความแสดงที่หน้าจอว่า “Login Success” และเข้าสู่หน้า Home ของแอปพลิเคชันวิ่งคนเดียวด้วยกัน

บรรทัดที่ 290 ถ้า response มีค่า statusCode ไม่เท่ากับ 200 แสดงว่าไม่สามารถทำการเชื่อมต่อกับ Service ได้และจะแสดงข้อความ “Login Error” ที่หน้า Login ของแอปพลิเคชัน



ภาพประกอบที่ 3.19 ตัวอย่างการเข้าสู่ระบบด้วย Facebook ที่สำเร็จ

```
295   Future<void>loginByEmail(String  email, String password) async{
296     var json = {"userEmail":email,"userPassword":password};
297     String value = await rootBundle.loadString('assets/config/config.yaml');
298     dynamic conf = loadYaml(value);
299     String url = conf['server']['host'];
300     String port = conf['server']['port'];
301     var response =  await http.post(Uri.parse("http://$url:$port/runAloneBack/loginByEmil"),
302     headers: <String, String>{
303       'Content-Type':'application/json; charset=UTF-8',
304     },
305      body: jsonEncode(json));
306
307      log(response.statusCode.toString());
308     if(response.statusCode == 200){
309       ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("LoginByEmail Success")));
310       Navigator.push(context, MaterialPageRoute(builder: (context) => LandingPage()));
311     }else{
312       ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("LoginByEmail Error")));
313     }
314   }
315 }
```

ภาพประกอบที่ 3.20 ฟังก์ชันการเชื่อมต่อ Service การเข้าสู่ระบบด้วย Email และ Password

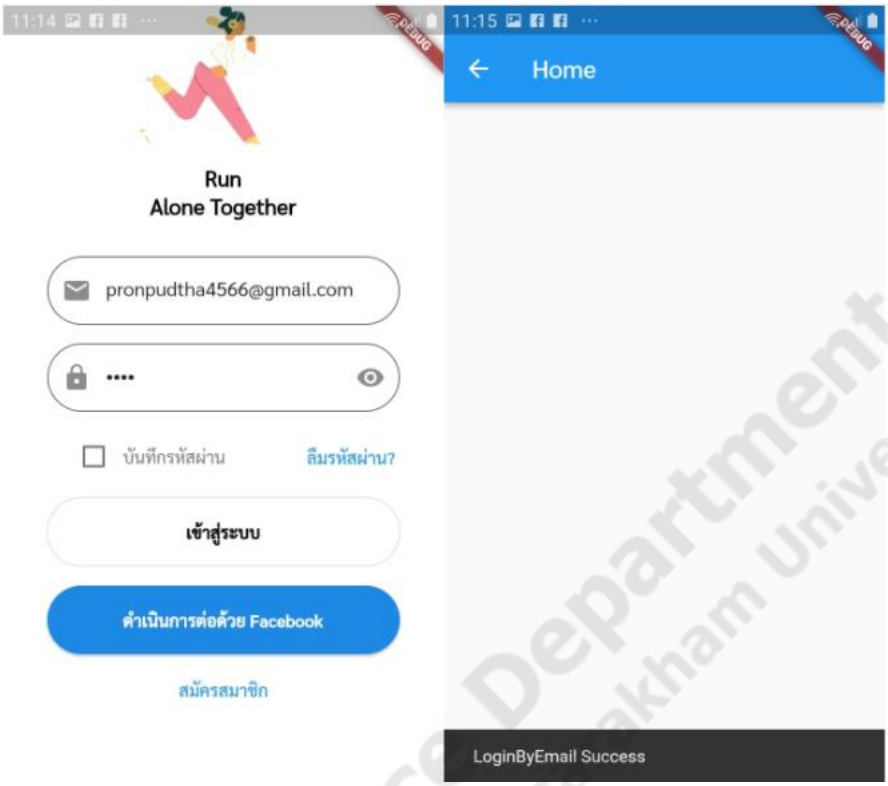
บรรทัดที่ 295 สร้างฟังก์ชันชื่อ LoginByEmail เพื่อเชื่อมต่อ service login และประกาศตัวแปรของ email และ password กำหนดชนิดเป็น String

บรรทัดที่ 296 ประกาศตัวแปร json เพื่อนำมาเก็บค่า email และ password

บรรทัดที่ 297 ประกาศตัวแปร value เพื่อเก็บค่าที่อยู่ Path ของไฟล์ config.yaml หรือไฟล์ที่เก็บค่าข้อมูลของ Server แล้วโหลดออกมาเป็น String

บรรทัดที่ 308 ถ้า response มีค่า statusCode เท่ากับ 200 แสดงว่าการเข้าสู่ระบบผ่าน Email และ Password สำเร็จและจะมีข้อความแสดงที่หน้าจอว่า “LoginByEmail Success” และเข้าสู่หน้า Home ของแอปพลิเคชันวิ่งคนเดียวด้วยกัน

บรรทัดที่ 311 ถ้า response มีค่า statusCode ไม่เท่ากับ 200 แสดงว่าไม่สามารถเชื่อมต่อกับ Service ได้และจะแสดงข้อความ “LoginByEmail Error” ที่หน้า Login ของแอปพลิเคชัน



ภาพประกอบที่ 3.21 ตัวอย่างการเข้าสู่ระบบด้วย Email และ Password ที่สำเร็จ

3.11.2 Google Maps API และ GPS

Google Maps API เป็นชุด API ของ Google สำหรับพัฒนา mobile application ไว้สำหรับเรียกใช้แผนที่และชุก service ต่าง ๆ ของ Google เพื่อพัฒนา Application โดยการเขียนโปรแกรมด้วย flutter ต้องลงไลบรารีชื่อ google_maps_flutter เพื่อเรียกใช้ Google Maps API

GPS คือ ระบบกำหนดตำแหน่งบนพื้นโลก จะทำงานโดยรับสัญญาณ ตัวเครื่องรับสัญญาณ GPS เช่น สมาร์ทโฟน โดยการคำนวณระยะทางโดย GPS ด้วยการเขียนโปรแกรมด้วย Flutter จะต้องลงไลบรารีชื่อ location สำหรับเรียกตำแหน่งสมาร์ทโฟน โดยจะแสดง Code ในส่วนของ Service ของ Google Maps และ GPS ได้ดังนี้

```
9  public class latlngDTO {
10     private int cid;
11     private Date datetime;
12     private double latitude;
13     private double longitude;
14     private int joinRoom;
```

ภาพประกอบที่ 3.22 Data Transfer Object (DTO) ส่วนของ GPS

บรรทัดที่ 9 สร้าง Class latlngDTO เพื่อประกาศตัวแปรไว้เก็บข้อมูลจากระบบหน้าบ้าน

บรรทัดที่ 10 ประการตัวแปร cid ที่มีชนิดข้อมูลเป็น int เพื่อเก็บค่าที่รับมาจากระบบหน้าบ้าน

บรรทัดที่ 11 ประการตัวแปร datetime ที่มีชนิดข้อมูลเป็น Date เพื่อเก็บค่าที่รับมาจากระบบหน้าบ้าน

บรรทัดที่ 12-13 ประการตัวแปร latitude และ longitude ที่มีชนิดข้อมูลเป็น Double เพื่อเก็บค่าที่รับมาจากระบบหน้าบ้าน

บรรทัดที่ 14 ประการตัวแปร joinRoom ที่มีชนิดข้อมูลเป็น int เพื่อเก็บค่าที่รับมาจากระบบหน้าบ้าน

```
13 public interface CoordinateRepositories extends JpaRepository<Coordinate, Integer>{
14
15     List<Coordinate> findByJoinRoom(JoinRoom joinRoom);
16 }
```

ภาพประกอบที่ 3.23 Repositories ของ GPS

บรรทัดที่ 13 สร้าง interface CoordinateRepositories เพื่อเก็บคำสั่ง JPQL ส่วน Coordinate และกำหนดชนิดข้อมูลให้เป็น Integer ตาม database

```
14 @Service
15 public class latlngService {
16     @Autowired
17     CoordinateRepositories coordinateRepositories;
18
19     public List<net.csmsu.runAloneBack.Entity.Coordinate> insertLatLng(latlngDTO dto){
20         Coordinate coordinate = new Coordinate();
21         coordinate.setDatetime(dto.getDatetime());
22         coordinate.setLatitude(dto.getLatitude());
23         coordinate.setLongitude(dto.getLongitude());
24         JoinRoom joinRoom = new JoinRoom();
25         joinRoom.setJoinId(dto.getJoinRoom());
26         coordinate.setJoinRoom(joinRoom);
27
28         coordinateRepositories.save(coordinate);
29         return null;
30     }
```

ภาพประกอบที่ 3.24 Service ของ GPS Method insertLatLng

บรรทัดที่ 15 สร้าง Service ชื่อ latlngService

บรรทัดที่ 16 ระบุ @Autowired เพื่อเป็นการ Connect ไปยัง database

บรรทัดที่ 17 คือการผูก Repositories เพื่อนำมาใช้ใน Service

บรรทัดที่ 19 สร้าง Method insertLatLng เพื่อทำหน้าที่ในการรับข้อมูลเข้ามา และมี paramete เป็น Class latlngDTO

บรรทัดที่ 28 เรียก Method save เพื่อ save ข้อมูล

```
32    public List<Coordinate> getLatLagByJoinID(int joinID){
33        JoinRoom joinRoom = new JoinRoom();
34        joinRoom.setJoinId(joinID);
35
36        List<Coordinate> coord = coordinateRepositories.findByJoinRoom(joinRoom);
37        return coord;
38    }
39 }
```

ภาพประกอบที่ 3.25 Service ของ GPS Method getLatLanByJoinID

บรรทัดที่ 32 สร้าง Method getLatLagByJoinID เพื่อแสดงข้อมูลทั้งหมด โดยประกาศตัวแปร joinID ที่มีชนิดข้อมูลเป็น int เพื่อเก็บข้อมูล

บรรทัดที่ 36 ประกาศตัวแปร coord แล้วเรียกใช้ Method findByJoinRoom ใน coordinate Repositories โดยส่งค่า joinRoom

บรรทัดที่ 37 return ข้อมูลกลับไปที่ CoordinateController Method getLatLagByJoinID

```
13 public interface CoordinateRepositories extends JpaRepository<Coordinate, Integer>{
14
15     List<Coordinate> findByJoinRoom(JoinRoom joinRoom);
16 }
```

ภาพประกอบที่ 3.26 Repositories ส่วนของ GPS

บรรทัดที่ 13 สร้าง interface CoordinateRepositories เพื่อเก็บคำสั่ง JPQL ส่วน Coordinate และกำหนดชนิดข้อมูลให้เป็น Integer ตาม database

บรรทัดที่ 15 สร้าง Method findByJoinRoom เพื่อส่งค่าข้อมูล joinID ไปที่ไฟล์ Service

```
19 @RestController
20 public class CoordinateController {
21     @Autowired
22     latlngService latlngService;
23
24     //Latlng
25     @PostMapping(value = "/coordinate",
26             consumes = MediaType.APPLICATION_JSON_VALUE,
27             produces = MediaType.APPLICATION_JSON_VALUE)
28     public ResponseEntity<?>insertLatLng(@RequestBody latlngDTO latlngDTO){
29         System.out.print(latlngDTO.getLatitude());
30         List<Coordinate> coo = latlngService.insertLatLng(latlngDTO);
31         return new ResponseEntity<>(coo,HttpStatus.OK);
32     }
```

ภาพประกอบที่ 3.27 Controller การรับข้อมูล

บรรทัดที่ 19-20 ใน Class CoordinateController ระบุ @RestController เพื่อให้ทราบว่า Class นี้คือ Restful Controller

บรรทัดที่ 21-22 ระบุ @Autowired latlngService เพื่อเป็นการ Connect ไปยัง database

บรรทัดที่ 25 สร้าง Controller โดยใช้ @PostMapping เพื่อระบุ Path ให้กับ Controller

บรรทัดที่ 26-27 กำหนดชนิดให้ consumes และ produces เป็น JSON เพื่อระบุว่าข้อมูลมีรูปแบบ JSON

บรรทัดที่ 28 ทำการสร้าง Method ของ Path ที่มีชื่อ insertLatLng เพื่อเป็นการเรียก Method insert LatLng ในไฟล์ latlngService

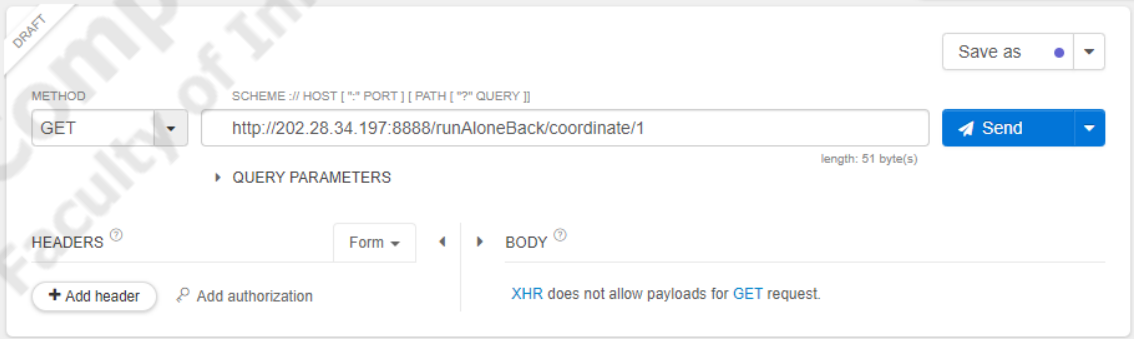
บรรทัดที่ 30 เรียกใช้ Method insertLatLng และส่งค่า DTO ให้กับ Method

```
34  @GetMapping(value = "/coordinate/{joinid}",
35          produces = MediaType.APPLICATION_JSON_VALUE)
36  public ResponseEntity<?>getLatLagByJoinID(
37                              @PathVariable("joinid") int joinid){
38      return new ResponseEntity<>(latlngService.getLatLagByJoinID(joinid),
39              HttpStatus.OK);
40  }
41 }
```

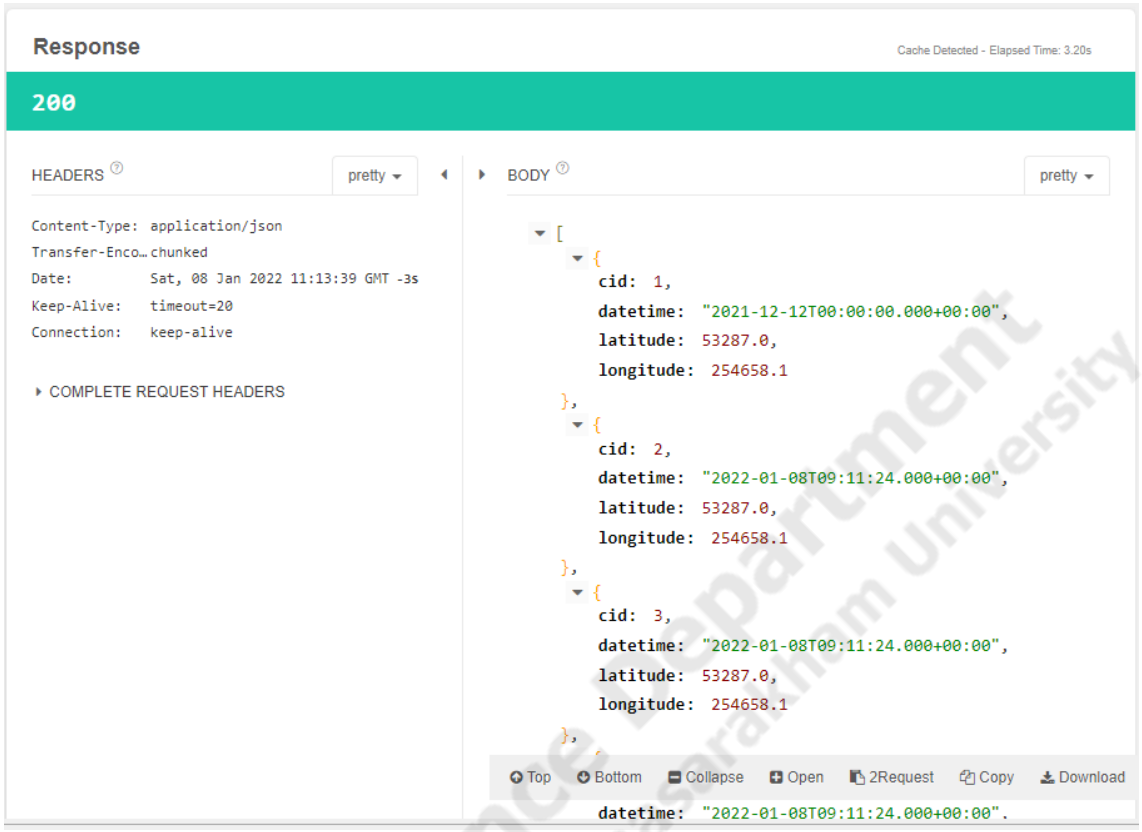
ภาพประกอบที่ 3.28 Controller การเรียกดูข้อมูล

บรรทัดที่ 34 สร้าง Controller โดยใช้ @GetMapping เพื่อระบุ Path ให้กับ Controller โดย {joinid} หมายถึง parameter ของ Path นี้

บรรทัดที่ 36 สร้าง Method ของ Path ชื่อ getLatLagByJoinID เพื่อเป็นการเรียก Method getLatLagByJoinID ในไฟล์ latlngService



ภาพประกอบที่ 3.29 ตัวอย่างการเรียกใช้ Service การเรียกดูข้อมูลในฐานข้อมูล coordinate



ภาพประกอบที่ 3.30 ตัวอย่างข้อมูลที่ได้มาจากการเรียกใช้ Service

```
21  String latitude = '';
22  String longitude = '';
23  String time = '';
24  int joinroom = 1;
25
26  late GoogleMapController mapController;
27  late Timer timer;
```

ภาพประกอบที่ 3.31 การประกาศตัวแปรเพื่อเรียกใช้งาน

บรรทัดที่ 21 ประกาศตัวแปรเพื่อมาเก็บค่าพิกัด latitude

บรรทัดที่ 22 ประกาศตัวแปรเพื่อมาเก็บค่าพิกัด logitude

บรรทัดที่ 23 ประกาศตัวแปรเพื่อมาเก็บค่าวัน เวลา

บรรทัดที่ 22 ประกาศตัวแปรเพื่อมาเก็บค่า join_id

บรรทัดที่ 26 ทำการสร้าง object GoogleMapController เพื่อใช้ในการควบคุมแผนที่

บรรทัดที่ 27 ทำการสร้าง object Timer เพื่อใช้ในการอัปเดตค่าพิกัด

```
43                  locationData('Latitude: ' + latitude),
44                  locationData('Longitude: ' + longitude),
45                  locationData('Time: ' + time),
46                  ElevatedButton(
47                    onPressed: () async {
48                      //await BackgroundLocation.setAndroidConfiguration(1000);
49                      await BackgroundLocation.startLocationService(
50                          distanceFilter: 20);
51                      BackgroundLocation.getLocationUpdates((location) {
52                        setState(() {
53                          latitude = location.latitude.toString();
54                          longitude = location.longitude.toString();
55                          time = DateTime.fromMillisecondsSinceEpoch(
56                                  location.time!.toInt()) // DateTime.fromMillisecondsSinceEpoch
57                              .toString();
58                        });
59                        print('''\n
60                          Latitude:  $latitude
61                          Longitude: $longitude
62                          Time: $time
63                        ''');
64                      });
65                      var json = {
66                        "latitude": latitude,
67                        "longitude": longitude,
68                        "datetime": time,
69                        "joinRoom": joinroom
70                      };
71                      var response = await http.post(Uri.parse(
72                          'http://202.28.34.197:8888/runAloneBack/coordinate'));
73                    },
74                    child: Text('Start Location')), // ElevatedButton
```

ภาพประกอบที่ 3.32 ฟังก์ชันการเริ่มต้นการดึงค่าพิกัดวันเวลา

บรรทัดที่ 43-45 เรียกค่าพิกัดและปี เดือน วัน เวลา ให้แสดงบนหน้าแอปพลิชัน

บรรทัดที่ 47-57 ทำการเรียกใช้ library ของ BackgroundLocation ในการดึงค่าพิกัด latitude longitude และปี เดือน วัน เวลามาเก็บไว้แล้วทำการอัปเดตตลอดเวลาแม้จะ ทำการปิดหน้าแอปพลิเคชั้นลง

บรรทัดที่ 59-63 ทำการแสดงค่าพิกัดเมื่อกดปุ่มเริ่ม

บรรทัดที่ 65-70 ทำการนำค่ามาแปลงเป็น json

บรรทัดที่ 71-73 ทำการเรียก service โดยใช้การ Post ในการส่งข้อมูลที่เป็น Json

```
75                  ElevatedButton(
76                      onPressed: () {
77                        BackgroundLocation.stopLocationService();
78                        timer.cancel();
79                      },
80                      child: Text('Stop Location')), // ElevatedButton
```

ภาพประกอบที่ 3.33 ฟังก์ชันการหยุดรับค่าพิกัดและวันเวลา

บรรทัดที่ 75-80 เมื่อกดปุ่มหยุดค่าพิกัดและวันเวลาจะทำการหยุดส่งทันท

```
83                timer =
84                    Timer.periodic(Duration(seconds: 10), (timer) async {
85                  String formattedDate = DateFormat('yyyy-MM-ddTkk:mm:ss')
86                      .format(DateTime.now());
```

ภาพประกอบที่ 3.34 โค้ดประกาศตัวแปรเพื่อเรียกใช้งาน

บรรทัดที่ 83-84 ทำการอัปเดตส่งค่าพิกัดและวันเวลาที่ขึ้นไปยัง server ทุกๆ 10 วินาที่

```
98                    log(jsonEncode(json));
99                    var response = await http.post(
100                       Uri.parse(
101                           'http://202.28.34.197:8888/runAloneBack/coordinate'),
102                       headers: <String, String>{
103                         'Content-Type': 'application/json; charset=UTF-8',
104                       },
105                       body: jsonEncode(json));
106                   log(jsonEncode(response.statusCode));
107                 }); // Timer.periodic
108               },
109               child: Text('Timer')), // ElevatedButton
```

ภาพประกอบที่ 3.35 ฟังก์ชันการส่งค่าพิกัดและวันเวลาไปยัง Server

บรรทัดที่ 98 ทำการแสดงค่าที่แปลงเป็น json

บรรทัดที่ 99-103 ทำการเรียก service โดยใช้การ Post ในการส่งข้อมูลที่เป็น Json โดยที่ข้อมูลในการส่งจะเป็น ‘Content-Type’

บรรทัดที่105 แสดงค่าที่เป็น json ที่ส่งไปยัง service

บรรทัดที่ 106 แสดงสถานะความสำเร็จของ HTTP ว่าคำขอสำเร็จแล้วจะแสดค่าเป็น 200

```
14    late LocationData currenPosition;
15    late GoogleMapController mapController;
16    Location location = Location();
17    LatLng initialcameraposition = LatLng(16.2466, 103.2519);
18    late Marker marker;
19    List<Marker> markers = <Marker>[];
20    String myLocation = "no";
21    late BitmapDescriptor cuatomIcon = BitmapDescriptor.defaultMarker;
```

ภาพประกอบที่ 3.36 โค้ดประกาศตัวแปรเพื่อเรียกใช้งาน

บรรทัดที่ 14 ทำการสร้างตัวแปร LocationData เพื่อใช้เก็บข้อมูลเกี่ยวกับตำแหน่ง

บรรทัดที่ 15 ทำการสร้าง object GoogleMapController เพื่อใช้ในการควบคุมแผนที่

บรรทัดที่ 16 สร้าง object Location เพื่อจะได้ใช้ฟังก์ชันใน library มาดึงค่าตำแหน่งปัจจุบัน

บรรทัดที่ 17 ทำการสร้างตัวแปลมาเก็บค่าตั้งต้นของ latitude logitude

บรรทัดที่ 18-19 สร้างตัวแปรชนิด Marker เพื่อใช้ตั้งค่าเวลาแสดงมาร์คเกอร์บนแผนที่สร้างลิสต์เพื่อเก็บมาร์คเกอร์

บรรทัดที่ 20-21 สร้างตัวแปร BitmapDescriptor เพื่อใช้ตั้งรูปที่จะนำมาใช้แสดงเป็นมาร์คเกอร์บนแผนที่ ค่าตั้งต้นคือใช้รูป defaultMarker

```
23  createMarker(context) {
24    if (myLocation == "yes") {
25      ImageConfiguration configuration = createLocalImageConfiguration(context);
26      BitmapDescriptor.fromAssetImage(
27              ImageConfiguration(size: Size(24, 24)), 'img/j.png')
28          .then((icon) {
29        setState(() {});
```

ภาพประกอบที่ 3.37 ฟังก์ชันการมาร์คตำแหน่ง

บรรทัดที่ 23-29 ทำการสร้าง Method createMarker เพื่อเปลี่ยนรูปของมาร์คเกอร์

```
34  setMarkers() {
35    createMarker(context);
36    markers.add(
37      Marker(
38        markerId: MarkerId("Home"),
39        position: initialcameraposition,
40        infoWindow: InfoWindow(
41          title: myLocation,
42        ), // InfoWindow
43      ), // Marker
44    );
45    setState(() {});
46  }
```

ภาพประกอบที่ 3.38 ฟังก์ชันการหาตำแหน่งของเครื่อง

บรรทัดที่ 34-46 ทำการสร้าง Method setMarkers() เพื่อตั้งคุณสมบัติให้มาร์คเกอร์ที่จะแสดงบนแผนที่และทำการเรียกฟังชัน createMarker(context) เพื่อเปลี่ยนรูปเมื่อกกดปุ่มหาตำแหน่งเครื่อง

```
71        body: GoogleMap(
72          onMapCreated: _onMapCreated,
73          initialCameraPosition: CameraPosition(
74            target: initialcameraposition,
75            zoom: 5,
76          ), // CameraPosition
77          markers: Set<Marker>.of(markers),
78          mapType: MapType.normal,
```

ภาพประกอบที่ 3.40 โค้ดส่วนของการแสดงแผนที่

บรรทัดที่ 71-78 เป็นการเรียกใช้ Method _onMapCreated ให้แสดงขึ้นบนหน้าแอปพลิเคชัน

```
84  void _onMapCreated(GoogleMapController controller) {
85    mapController = controller;
86    print(
87        "oncreted ${initialcameraposition.latitude} : ${initialcameraposition.longitude}");
88    mapController.animateCamera(
89      CameraUpdate.newCameraPosition(
90        CameraPosition(target: initialcameraposition, zoom: 15),
91      ),
92    );
93    setMarkers();
```

ภาพประกอบที่ 3.41 ฟังก์ชันการสร้างแผนที่บนแอปพลิเคชัน

บรรทัดที่ 84 ทำการสร้าง Method _onMapCreated เพื่อสร้างแผนที่ในแอปพลิเคชัน

บรรทัดที่ 88-90 เป็นการซูมแผนที่ทำให้เห็นตำแหน่งได้ชัดเจนยิ่งขึ้น

บรรทัดที่ 93 ทำการเรียกฟังชัน setMarker() เพื่อตั้งค่ามาร์คเกอร์ที่จะแสดงบนแผนที่ ที่เราได้สร้างไว้

```
102     _serviceEnabled = await location.serviceEnabled();
103     if (!_serviceEnabled) {
104       _serviceEnabled = await location.requestService();
105       if (!_serviceEnabled) {
106         return;
```

ภาพประกอบที่ 3.42 โค้ดส่วนของการเข้าถึงตำแหน่ง

บรรทัดที่ 102-106 ทำการสร้าง Method getLoc() แล้วทำการเช็คว่าเครื่องได้เปิดส่วนของโลเคชั่น

```
109     _permissionGranted = await location.hasPermission();
110     if (_permissionGranted == PermissionStatus.denied) {
111       _permissionGranted = await location.requestPermission();
112       if (_permissionGranted != PermissionStatus) {
113         return;
```

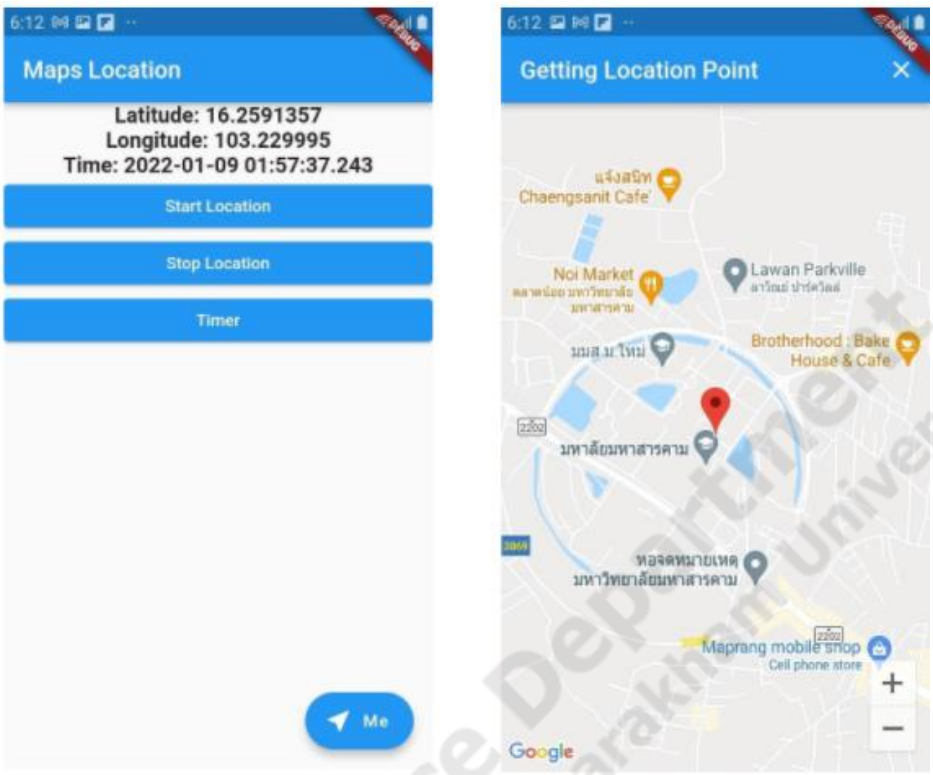
ภาพประกอบที่ 3.43 โค้ดส่วนของการอนุญาตการเข้าถึงตำแหน่ง

บรรทัดที่ 109-113 ทำการเช็คว่าเครื่องว่าได้เปิดอนุญาตให้แอปเข้าถึง ข้อมูลส่วนของโลเคชั่นไว้หรือไม่ในการเข้าถึงตำแหน่งของเครื่องว่ามีค่าพิกัดเท่าไหร่เพื่อนำค่าพิกัดที่ได้นั้นนำไปบันทึกในฐานข้อมูลที่ใช้ในการคำนวณหาระยะทางและแสดงเส้นทางที่ใช้ในการวิ่งบนแผนที่

```
115     currenPosition = await location.getLocation();
116     double? latitude = currenPosition.latitude;
117     double? longitude = currenPosition.latitude;
118
119     location.onLocationChanged.listen((LocationData currentLocation) {
120       setState(() {
121         print(
122             "Current Loc ${currenPosition.latitude} : ${currenPosition.longitude}");
123         initialcameraposition = LatLng(latitude!, longitude!);
124
125         setMarkers();
```

ภาพประกอบที่ 3.44 โค้ดส่วนของการค้นหาตำแน่งของเครื่อง

บรรทัดที่ 115-125 ทำการดึงโลเคชันของเครื่อง และเมื่อตำแหน่งเปลี่ยนจะมีการปรับตำแหน่งกล้องของแผนที่ให้ตรงกับจุดที่ได้มา



ภาพประกอบที่ 3.45 ตัวอย่างการดึงค่าพิกัดและตำแหน่งบนแผนที่

3.11.3 Text to Speech

```
17    TextEditingController textEditingController = TextEditingController();
18    Future speak(String text) async {
19      await flutterTts.setLanguage('th-TH');
20      await flutterTts.setPitch(0.5);
21      await flutterTts.speak(text);
```

ภาพประกอบที่ 3.46 ฟังก์ชันการควบคุมการแปลงข้อความเป็นเสียง

บรรทัดที่ 17 ทำการสร้าง Object เพื่อใช้ควบคุมการนำเข้าข้อความที่จะทำการแปลงเป็นเสียง

บรรทัดที่ 18 สร้าง Future Method speak เพื่อรอเรียกใช้ในการแปลงข้อความเป็นเสียงพูด

บรรทัดที่ 19 เลือกภาษาที่จะแปลงการออกเสียงในแต่ละภาษานั้นให้ถูกสำเนียง

บรรทัดที่ 20 เป็นตัวจัดการระดับเสียงสูงเสียงต่ำซึ่งมีค่าอยู่ระหว่าง 0.5 ถึง 1.5

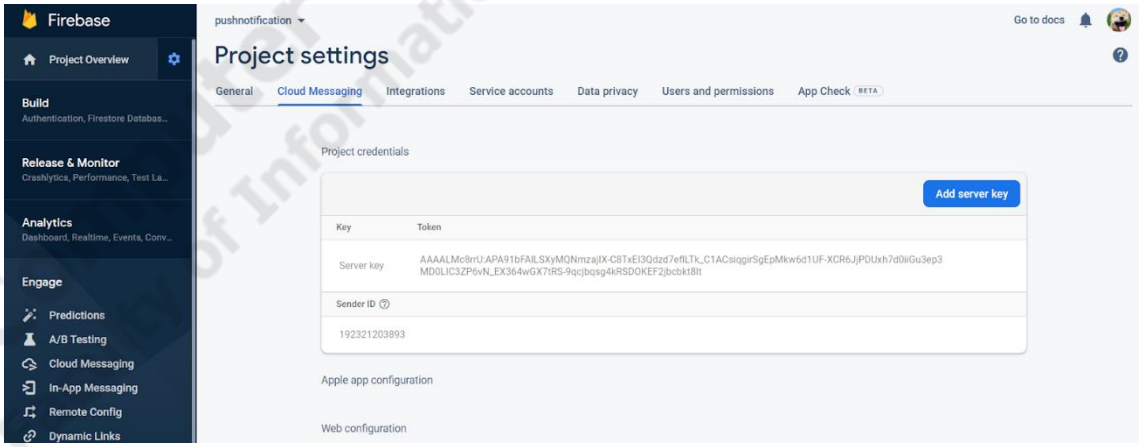
บรรทัดที่ 21 ทำการแปลงข้อความออกมาเป็นเสียงพูด

```
31                TextField(
32                  controller: textEditingController,
33                ), // TextField
34                RaisedButton(
35                  child: Text('Text to speech'),
36                  onPressed: () => speak(textEditingController.text),
```

ภาพประกอบที่ 3.47 การแปลงข้อความเป็นเสียง

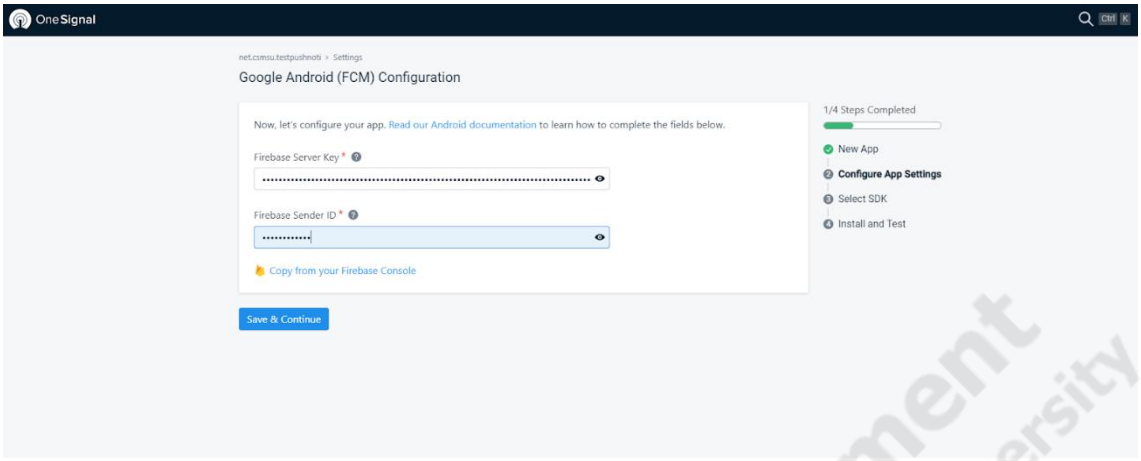
3.11.4 Push Notification

Notification เป็นอีกหนึ่งช่องทางของระบบปฏิบัติการแอนดรอยด์ที่เปิดทำการให้แอปพลิเคชันส่งข้อความให้ผู้ใช้เห็นได้ ในขณะที่หน้าจอทำการปิดอยู่ ผู้ใช้ยังสามารถสั่งงานบางอย่างผ่าน Notification ตัวนั้นๆกลับมาได้อีกด้วยซึ่งเป็นความสามารถของระบบปฏิบัติการแอนดรอยด์ที่เปิดให้นักพัฒนาสามารใช้งานในสถานการณ์ต่างๆ ที่เหมาะสมกับแอปพลิเคชันของเรา โดยการทำงานนั้นจะทำงานของระบบแอนดรอยด์ที่มีไว้ให้แอปต่างๆเรียกใช้งานซึ่งใน Android Framework ก็จะมีการแบ่งการทำงานแยกกันออกไปตามหน้าที่เกี่ยวกับระบบ Notification จากแอปต่างๆก็จะรับข้อมูลจาก Notification Manager ดังนั้นเมื่อใดก็ตามที่เครื่องส่ง Notification ไปให้ Notification Manager ก็จะถูกส่งไปเพื่อแสดงใน Notification Drawer ที่อยู่ใน Status Bar



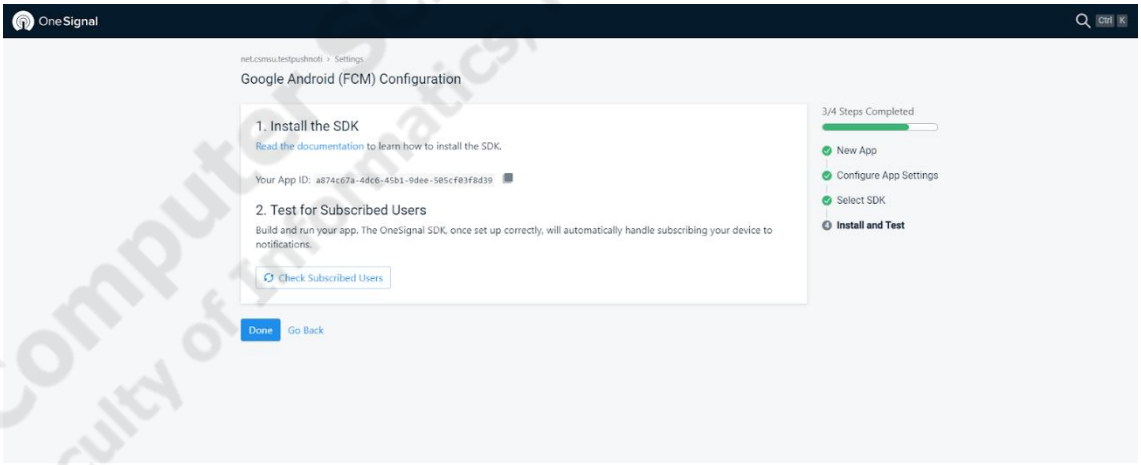
ภาพประกอบที่ 3.48 การล็อคอิน Firebase เพื่อขอ Server key และ Sender ID

เมื่อเข้ามาที่ Firebase แล้วจากนั้นให้ทำการสร้างโปรเจ็คของ Firebase เพื่อขอ Server key และ Sender ID เพื่อที่จะนำไปใส่ใน OneSignal เพื่อขอ key App ID ที่เป็นตัวแทนของแอปพลิเคชัน



ภาพประกอบที่ 3.49 การล็อคอินผ่าน OneSignal

OneSignal เป็นเครื่องมือที่ช่วยในการทำ Push Notification โดยการทำงานของ OneSignal จะทำการสร้าง key มาให้และมี API ให้เราสามารถสั่งการจาก Service ได้ซึ่งถ้าเราต้องการจะส่ง Notification ไปหาผู้ใช้ทุกคนในทุก Platform เราแค่ส่งไปยัง OneSignal ก็จะทำการกระจาย Notification ให้ดองเมื่อเราทำการ login OnSignal แล้วจึงทำการสร้างแอปพลิเคชันหรือเว็บไซต์ขึ้นมา จากนั้นจะมีการกำหนดค่าโดยใช้ Firebase



ภาพประกอบที่ 3.50 รับ key App ID OneSignal SDK

เมื่อเรานำ key App ID ไปใส่แอปจากนั้นก็จะทำการเช็คว่าแอปพลิเคชั้นเรานั้นเชื่อมต่อกับคีย์นี้หรือยังเพื่อที่จะทำการส่งข้อมูลจากแอปพลิเคชันไปยัง OneSignal จากนั้นก็จะเช็คว่าทำการ Config ค่าตัวระบบปฏิบัติการอะไรไว้ จากนั้นก็จะทำการส่งไปยัง Firebase เพื่อที่จะทำการส่งไปยังเครื่องที่กำหนด

```
13    Widget build(BuildContext context) {
14      OneSignal.shared.setAppId("a874c67a-4dc6-45b1-9dee-505cf03f8d39");
```

ภาพประกอบที่ 3.51 การเชื่อมต่อ OneSignal โดย key App ID

บรรทัดที่ 14 ทำการเชื่อมไปยัง OneSignal เพื่อที่จะส่งข้อมูลไปยังเครื่องที่กำหนด

```
22                ElevatedButton(
23                  onPressed: () => _handleSendNotification(),
24                  child: Text('Send Msg')),  // ElevatedButton
25                  Text(msg)],
```

ภาพประกอบที่ 3.52 การส่ง Push Notification

บรรทัดที่ 22-25 เมื่อทำการกดปุ่มจะทำการส่งข้อมูลไปยังเครื่องที่กำหนด

```
30  void _handleSendNotification() async {
31    var deviceState = await OneSignal.shared.getDeviceState();
32
33    if (deviceState == null || deviceState.userId == null) return;
34    var playerId = deviceState.userId!;
35    var imgUrlString =
36        "http://cdn1-www.dogtime.com/assets/uploads/gallery/30-impossibly-cute-puppies/impossibly-cute-puppy-2.jpg";
37
38    var notification = OSCreateNotification(
39        playerIds: [playerId],
40        content: "this is a test from OneSignal's Flutter SDK",
41        heading: "Test Notification",
42        iosAttachments: {"id1": imgUrlString},
43        bigPicture: imgUrlString,
44        buttons: [
45          OSActionButton(text: "test1", id: "id1"),
46          OSActionButton(text: "test2", id: "id2")
47        ]); // OSCreateNotification
48
49    var response = await OneSignal.shared.postNotification(notification);
50  }
```

ภาพประกอบที่ 3.53 ฟังก์ชันการทำงานของ Push Notification

บรรทัดที่ 34 การเชื่อมต่อระหว่าง id ของผู้รับและ id ของเครื่องที่ส่ง

บรรทัดที่ 38-47 เป็นการสร้าง Object ของ Notification และนำ id ของเครื่องปลายทางที่จะทำการโดยใส่ที่ playerIds และข้อความที่จะทำการส่งจะประด้วย content และ heading คือ ส่วนที่เป็นหัวเรื่องและเนื้อหาของข้อความ

บรรทัดที่ 49 ทำการเรียกใช้ OneSignal ให้ส่ง Notification จากนั้นทำการส่ง Object ขึ้นไปยัง Server แล้วทำการตรวจสอบข้อความและหมายเลข id จากนั้นทำการส่งไปยังเครื่องปลายทาง เมื่อทำการส่งเสร็จแล้วสามารถนำ response ไปเช็คได้ว่าส่งไปยังเครื่องปลายทางแล้วกี่คน

3.11.5 การประมวลผลบน Server

1. การหาระยะทางและระยะเวลาระหว่างจุด 2 จุดที่ใช้ในการวิ่งโดยการนำค่าพิกัด Latitude Longitude และวันเวลา ที่ทำการส่งขึ้นไปเก็บยัง Server ในทุกๆระยะเวลา 10 วินาที จากนั้นทำการนำค่าพิกัดที่ได้นั้นมาคำนวณหาระยะทางและระยะเวลาซึ่งจะทำการคำนวณไปเรื่อย ๆ ในขณะผู้ใช้ทำการวิ่งและทำการแสดงระยะทางและระยะเวลาบนหน้าจอแอปพลิเคชันให้ผู้ใช้สามารถเห็นได้ว่าทำการวิ่งไปแล้วระยะทางกี่กิโลเมตรและใช้เวลาเท่าไหร่จนกว่าผู้ใช้จะทำการสิ้นสุดการวิ่ง ซึ่งหาได้จากการคำนวณระยะทางตามแนวเส้นตรง โดยใช้วิธีการคำนวณแบบ haversine ซึ่งคำนวณระยะทางโดยนำ รัศมีของวงกลม (โลก) เป็นปัจจัยในการคำนวณดังสมการ

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \operatorname{atan2}\left(\sqrt{a}, \sqrt{(1-a)}\right)$$

$$d = R \cdot c$$

2. การหาอัตราเฉลี่ยที่ใช้ในการวิ่งก็จะนำค่าของระยะทางและระยะเวลาที่ได้จากค่าระหว่างจุด 2 จุด เพื่อทำการคำนวณหาค่าอัตราเฉลี่ยที่มีหน่วยเป็น นาทีต่อกิโลเมตร คำนวณได้จากสูตร

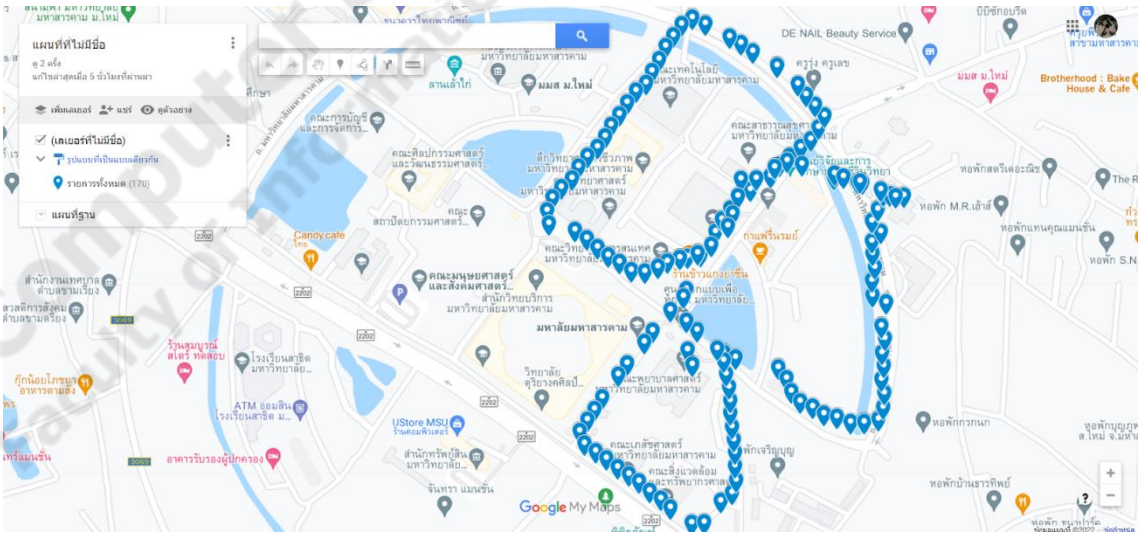
$$\frac{\text{ระยะเวลา}}{\text{ระยะทาง}} = \text{อัตราเฉลี่ย (m/Km.)}$$

3. การแจ้งเตือนเป็นระบบเสียงและระบบข้อความซึ่งการแจ้งเตือนก็มีการเก็บข้อมูลไว้เป็น case ที่มีข้อมูลเป็นข้อความอยู่ข้างในเพื่อรอการตรวจสอบ โดยการนำค่าระยะทางที่ได้จากระหว่างจุด 2 จุดของแต่ละคนในเวลานั้นๆ มาทำการเปรียบเทียบกันจากนั้นทำการตรวจสอบว่าเข้า case ไหนก็จะทำการแจ้งเตือนเป็นระบบเสียงและระบบข้อความ

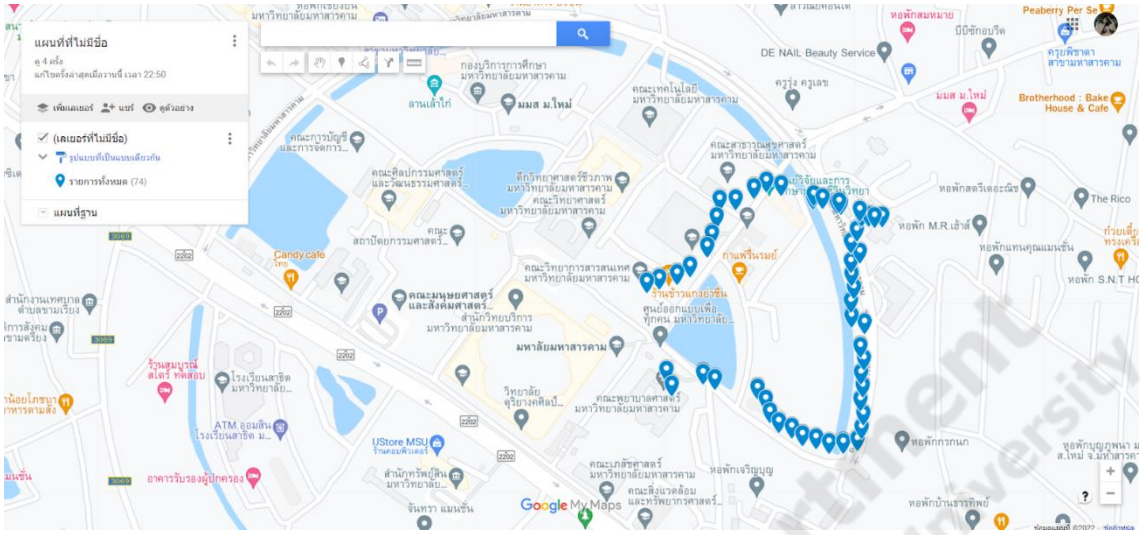
+ Options

	cid	datetime	latitude	longitude	join_id
Edit Copy Delete	186	2022-01-10 00:43:36	16.2465044	103.2521632	2
Edit Copy Delete	187	2022-01-10 00:43:37	16.2465173	103.2522651	1
Edit Copy Delete	188	2022-01-10 00:43:45	16.2460648	103.2522545	5
Edit Copy Delete	189	2022-01-10 00:43:46	16.2465044	103.2521632	2
Edit Copy Delete	190	2022-01-10 00:43:47	16.2465421	103.2524622	1
Edit Copy Delete	191	2022-01-10 00:43:55	16.2457737	103.2520227	5
Edit Copy Delete	192	2022-01-10 00:43:57	16.2468623	103.2526749	1
Edit Copy Delete	193	2022-01-10 00:43:56	16.2464555	103.2519797	2
Edit Copy Delete	194	2022-01-10 00:44:05	16.2454906	103.2517628	5
Edit Copy Delete	195	2022-01-10 00:44:06	16.2464555	103.2519797	2
Edit Copy Delete	196	2022-01-10 00:44:07	16.2470378	103.2527685	1
Edit Copy Delete	197	2022-01-10 00:44:15	16.245334	103.2516446	5
Edit Copy Delete	198	2022-01-10 00:44:17	16.2470378	103.2527685	1
Edit Copy Delete	199	2022-01-10 00:44:16	16.24523	103.2522647	2
Edit Copy Delete	200	2022-01-10 00:44:25	16.2450288	103.2513956	5
Edit Copy Delete	201	2022-01-10 00:44:26	16.2450255	103.2523399	2
Edit Copy Delete	202	2022-01-10 00:44:27	16.2471805	103.2528919	1
Edit Copy Delete	203	2022-01-10 00:44:35	16.2448813	103.2512739	5
Edit Copy Delete	204	2022-01-10 00:44:36	16.2450255	103.2523399	2
Edit Copy Delete	205	2022-01-10 00:44:37	16.2474552	103.2531842	1
Edit Copy Delete	206	2022-01-10 00:44:45	16.2445716	103.2510337	5
Console Edit Copy Delete	207	2022-01-10 00:44:46	16.2451835	103.2527946	2

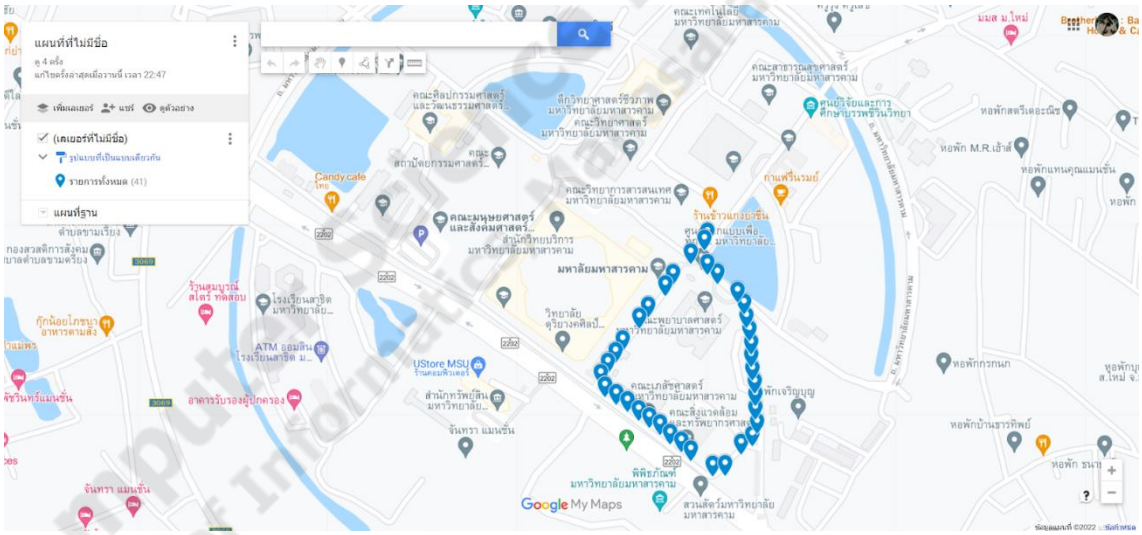
ภาพประกอบที่ 3.54 ตัวอย่างข้อมูลพิกัดที่อยู่ใน Server



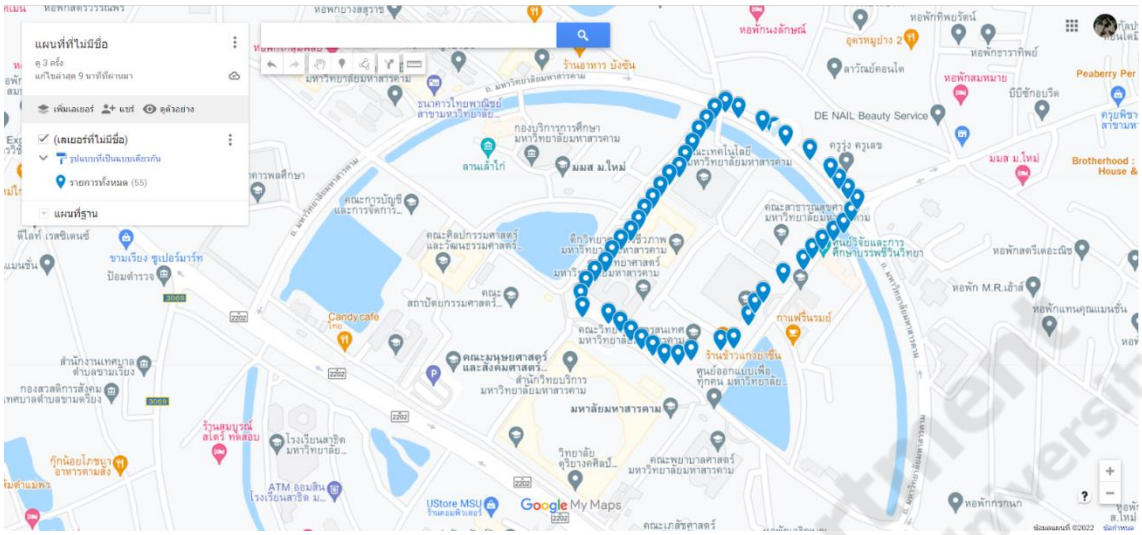
ภาพประกอบที่ 3.55 ตัวอย่างแผนที่จำลองการวิ่งที่ได้จากการนำข้อมูลใน Server มา plot จุด



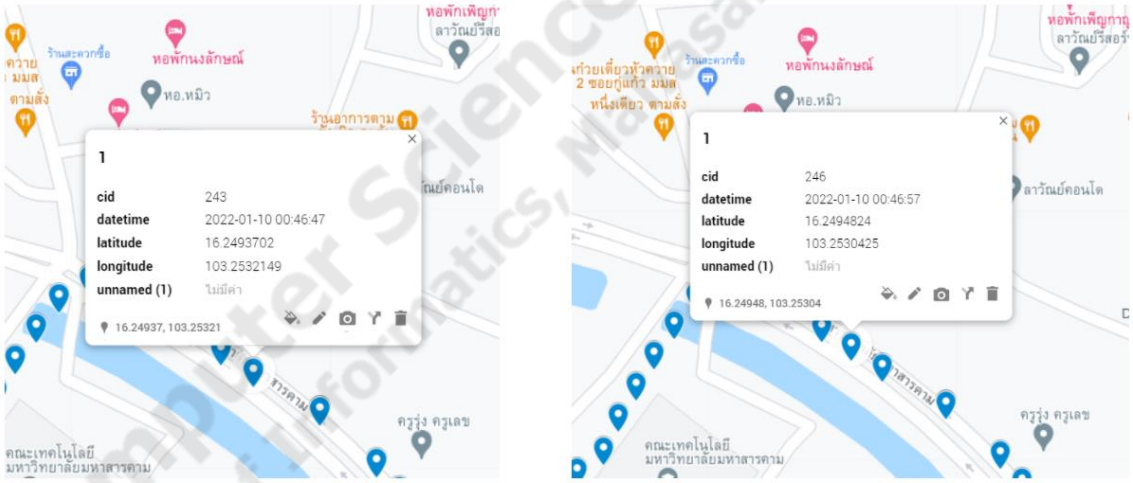
ภาพประกอบที่ 3.56 ตัวอย่างแผนที่จำลองการวิ่งของผู้ใช้งานคนที่ 1



ภาพประกอบที่ 3.57 ตัวอย่างแผนที่จำลองการวิ่งของผู้ใช้งานคนที่ 2

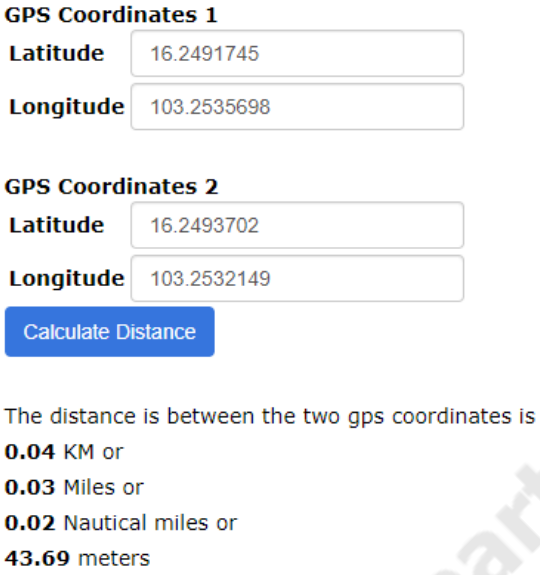


ภาพประกอบที่ 3.58 ตัวอย่างแผนที่จำลองการวิ่งของผู้ใช้งานคนที่ 3

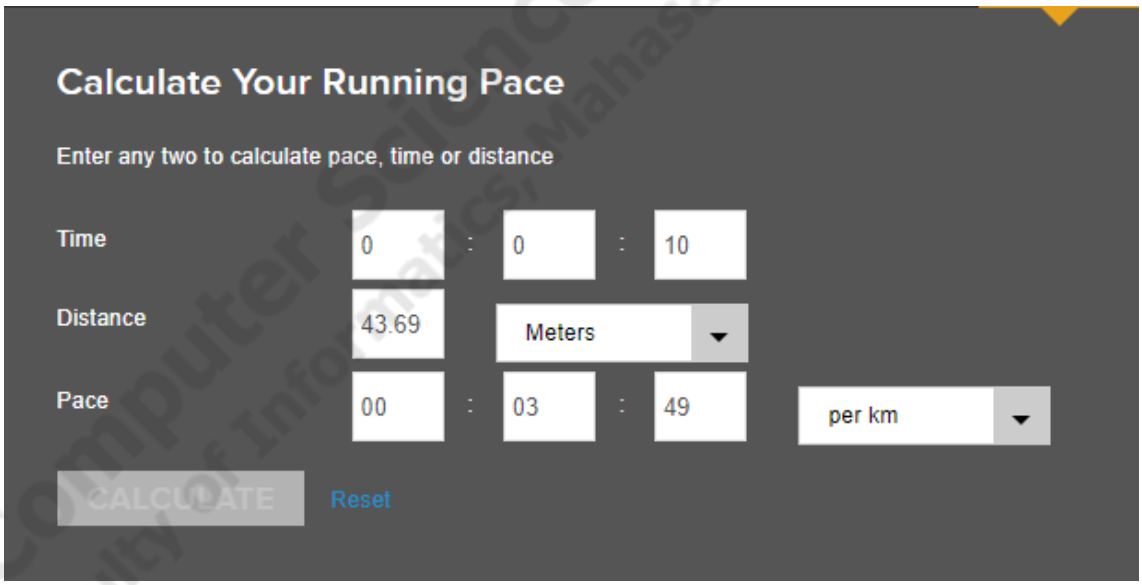


ภาพประกอบที่ 3.59 ตัวอย่างข้อมูลพิกัดของผู้ใช้งานคนที่ 3

พิกัดการวิ่งของผู้ใช้งานคนที่ 3 ยกตัวอย่างพิกัดแต่ละจุด จุดที่ 1 เวลา 00.46.47 วินาที ผู้ใช้งานอยู่ที่พิกัด latitude 16.2493702, longitude 103.2532149 และจุดที่ 2 เวลา 00.46.57 วินาที ผู้ใช้งานอยู่ที่พิกัด latitude 16.2494824, longitude 103.2530425 โดยจุดทั้งสองจุดห่างกันทุก ๆ 10 วินาที



ภาพประกอบที่ 3.60 ตัวอย่างการคำนวณหาระยะทางระหว่างจุดของพิกัด
ที่มา : https://gps-coordinates.org/



ภาพประกอบที่ 3.61 ตัวอย่างการคำนวณอัตราการวิ่ง
ที่มา : https://www.active.com/fitness/calculators/pace