ชื่อเรื่องภาษาไทย	RoadSafe Alert : แอปพลิเคชันเตือนภัยจากจุดเสี่ยงอุบัติเหตุบนถนน
ชื่อเรื่องภาษาอังกฤษ	RoadSafe Alert: Road Accident Alert Mobile Application
ผู้วิจัย	นางสาวสุชานาฏ มโนทัย
ที่ปรึกษาวิทยานิพนธ์	ผศ.ดร.กัมปนาท ปิยะอำรงชัย
ประเภทสารนิพนธ์	วิทยานิพนธ์ วิทยาศาสตรบัณฑิต สาขาวิชาภูมิศาสตร์, มหาวิทยาลัยนเรศวร, 2567

(ผู้ช่วยศาสตราจารย์ ดร.กัมปนาท ปิยะอำรงชัย)	(อาจารย์ชัญญาลัคน์ จันทร์สมบัติ)	(ผู้ช่วยศาสตราจารย์ ว่าที่ ร.ต.ดร.รังสรรค์ เกตุอ๊อด)
อาจารย์ที่ปรึกษา	ประธานบริหารหลักสูตร	หัวหน้าภาควิชา
วิทยานิพนธ์ระดับปริญญาตรี	วิทยาศาสตรบัณฑิต สาขาภูมิศาสตร์	ทรัพยากรธรรมชาติและสิ่งแวดล้อม

ABSTRACT

Road accidents significantly impact drivers and road users. The RoadSafe Alert app reduces this risk by alerting drivers when entering areas with frequent accidents. Using GIS technology, it displays accident hotspots based on historical data. Users can set an alert radius for real-time notifications when approaching high-risk areas. Developed with PostgreSQL, Flutter, Dart, and Node.js, the app accurately identifies accident-prone zones and has potential for use in public and transport industries to enhance road safety.

คำสำคัญ: การเตือนภัย, จุดเสี่ยง, อุบัติเหตุบนท้องถนน, ระบบสารสนเทศภูมิศาสตร์ (GIS), การแจ้งเตือนแบบเรียลไทม์

บทคัดย่อ

อุบัติเหตุบนท้องถนนเป็นปัญหาสำคัญที่ส่งผลต่อชีวิตและทรัพย์สิน แอป RoadSafe Alert พัฒนาขึ้นเพื่อลดความเสี่ยงนี้โดยแจ้งเตือนผู้ขับขี่เมื่อเข้าสู่พื้นที่ที่มีประวัติอุบัติเหตุบ่อยครั้ง ใช้เทคโนโลยี GIS ในการวิเคราะห์และแสดงผลจุดอุบัติเหตุจากข้อมูลประวัติศาสตร์ ผู้ใช้สามารถกำหนดระยะการแจ้งเตือนเพื่อรับการเตือนแบบเรียลไทม์เมื่อเข้าใกล้พื้นที่เสี่ยง พัฒนาด้วย PostgreSQL, Flutter, Dart และ Node.js ผลการทดสอบแสดงว่าแอปมีศักยภาพและอาจต่อยอดใช้ในระบบขนส่งสาธารณะหรืออุตสาหกรรมขนส่งเพื่อเพิ่มความปลอดภัย

กิตติกรรมประกาศ

การพัฒนาแอปพลิเคชัน RoadSafe Alert แอปพลิเคชันเตือนภัยจากจุดเสี่ยงอุบัติเหตุบนถนน สำเร็จลุล่วงไปได้ด้วยการสนับสนุนและความช่วยเหลือจากหลายฝ่าย ข้าพเจ้าขอแสดงความขอบคุณอย่างสูงต่อทุกท่านที่มีส่วนร่วมในการทำวิจัยครั้งนี้ก่อนอื่น ข้าพเจ้าขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.กัมปนาท ปิยะธำรงชัย อาจารย์ที่ปรึกษาหลัก ที่ได้ให้คำแนะนำอันทรงคุณค่าและความช่วยเหลือที่มีความสำคัญอย่างยิ่งในการพัฒนาโครงการนี้ ท่านได้มอบข้อคิดเห็นเชิงวิชาการที่ช่วยให้ข้าพเจ้าแก้ไขปัญหาและพัฒนาแอปพลิเคชันนี้ได้อย่างมีประสิทธิภาพ ท่านยังได้ชี้แนะแนวทางการวิจัยและให้การสนับสนุนตลอดทุกขั้นตอนจนโครงการนี้เสร็จสมบูรณ์ ข้าพเจ้าขอขอบคุณคณาจารย์และบุคลากร ทุกท่านในสาขาวิชาภูมิศาสตร์ ที่ให้คำปรึกษาและคำแนะนำในด้านการวิจัยตลอดจนการใช้ทรัพยากรและเทคโนโลยีต่างๆ ซึ่งมีความสำคัญต่อการพัฒนาแอปพลิเคชันนี้

นอกจากนี้ ข้าพเจ้าขอขอบคุณ หน่วยงานและองค์กร ที่ให้การสนับสนุนในด้านข้อมูลเกี่ยวกับอุบัติเหตุ รวมถึงการให้ข้อมูลและการใช้ฐานข้อมูล PostgreSQL ซึ่งเป็นส่วนสำคัญในการวิเคราะห์และพัฒนาแอปพลิเคชันนี้ การสนับสนุนดังกล่าวมีบทบาทสำคัญในการทำให้โครงการนี้สำเร็จลุล่วงตามเป้าหมายที่ตั้งไว้ด้วยความขอบคุณอย่างยิ่ง

นางสาวสุชานาฏ มโนทัย

ที่มาและความสำคัญ

ปัจจุบันมีการขับขี่บนท้องถนนเพิ่มขึ้นอย่างมาก ซึ่งนำไปสู่การเพิ่มขึ้นของอุบัติเหตุทางถนน ความเสี่ยงที่เกิดขึ้นนี้ได้สร้างความสูญเสียอย่างมหาศาลทั้งในด้านชีวิตและทรัพย์สิน เพื่อรับมือกับปัญหานี้ จึงได้มีแนวโน้มการเกิดอุบัติเหตุจราจรเป็นปัญหาสำคัญที่ทุกประเทศกำลังเผชิญอยู่ โดยในแต่ละปีมีผู้เสียชีวิตและบาดเจ็บจากอุบัติเหตุจราจรเพิ่มสูงขึ้น องค์การอนามัยโลก (WHO) รายงานว่าในแต่ละปีมีประชากรโลกประมาณ 1.3 ล้านคนเสียชีวิตจากอุบัติเหตุจราจร และอีก 20 ถึง 50 ล้านคนได้รับบาดเจ็บหรือพิการ โดยร้อยละ 93 ของการเสียชีวิตเกิดขึ้นในประเทศที่มีรายได้ต่ำถึงปานกลาง (WHO, 2022) สำหรับประเทศไทย อัตราการเสียชีวิตจากอุบัติเหตุจราจรต่อประชากร 100,000 คน อยู่ที่ 32.7 ซึ่งสูงที่สุดเป็นอันดับเก้าของโลก (WHO, 2018) ตามรายงานจากศูนย์ข้อมูลกลางด้านการบาดเจ็บ กรมควบคุมโรค กระทรวงสาธารณสุข ระหว่างปี พ.ศ. 2554-2564 พบว่าประเทศไทยมีผู้เสียชีวิตจากอุบัติเหตุจราจรถึง 223,545 ราย ซึ่งส่งผลกระทบอย่างมากต่อเศรษฐกิจและสังคม (IDCC, 2022)

การศึกษาข้อมูลด้านวิทยาการระบาดเป็นกิจกรรมสำคัญที่ช่วยบ่งชี้ถึงสถานการณ์และแนวโน้มการบาดเจ็บและเสียชีวิตจากอุบัติเหตุจราจร ระบบสารสนเทศทางภูมิศาสตร์ (GIS) เป็นเครื่องมือที่มีประสิทธิภาพสูงในการจัดการ วิเคราะห์ และแสดงผลข้อมูลเชิงพื้นที่ เช่น การศึกษาเกี่ยวกับจุดเสี่ยงการเกิดอุบัติเหตุจราจรในฮ่องกง ลอนดอน รัฐมินนิโซตา สหรัฐอเมริกา และฮานอย (Loo & Yao, 2013; Anderson, 2009; Thakali et al., 2015; Le et al., 2019) ซึ่งใช้เทคนิคการวิเคราะห์ความหนาแน่นเชิงพื้นที่แบบเคอร์เนล (Kernel Density Estimation) ในการระบุจุดที่มีความเสี่ยงสูงจากปัญหาดังกล่าว

แอปพลิเคชัน RoadSafe Alert ถูกพัฒนาขึ้นเพื่อลดความเสี่ยงจากอุบัติเหตุบนท้องถนน โดยนำข้อมูลอุบัติเหตุย้อนหลังจากกระทรวงคมนาคมในปี พ.ศ. 2566 มาวิเคราะห์และระบุตำแหน่งที่เกิดอุบัติเหตุบ่อยครั้ง ระบบจะทำการแจ้งเตือนผู้ขับขี่แบบเรียลไทม์เมื่อเข้าใกล้จุดเสี่ยง โดยผู้ใช้สามารถปรับแต่งรัศมีการแจ้งเตือนได้ตามความต้องการ รวมถึงเข้าถึงข้อมูลเชิงลึก เช่น วันเวลา สถิติการเกิดอุบัติเหตุ และข้อมูลผู้บาดเจ็บหรือเสียชีวิตในพื้นที่นั้น และการพัฒนาแอปนี้อาศัยเทคโนโลยีระบบสารสนเทศภูมิศาสตร์ (GIS) ในการวิเคราะห์ข้อมูลเชิงพื้นที่และข้อมูลอุบัติเหตุ แอปได้รับการออกแบบให้ใช้งานง่าย มีฟีเจอร์สำคัญ เช่น การแจ้งเตือนแบบเรียลไทม์ การแสดงข้อมูลจุดอุบัติเหตุย้อนหลัง

และการเลือกรัศมีแจ้งเตือน เพื่อเพิ่มความปลอดภัยในการขับขี่และสนับสนุนการตัดสินใจของผู้ใช้ในการหลีกเลี่ยงพื้นที่เสี่ยง

วัตถุประสงค์ในการศึกษา

1. เพื่อพัฒนาแอปพลิเคชันที่สามารถตรวจจับจุดเกิดอุบัติเหตุที่เกิดขึ้นในอดีตทางถนน

2. เพื่อพัฒนาแอปพลิเคชันที่สามารถแจ้งเตือนการเกิดอุบัติเหตุทางถนน

ความสำคัญของการพัฒนาระบบ

RoadSafe Alert คือ การเพิ่มความปลอดภัยบนท้องถนน โดยระบบจะแจ้งเตือนผู้ขับขี่เมื่อเข้าสู่พื้นที่เสี่ยงอุบัติเหตุ ผ่านการวิเคราะห์ข้อมูลเชิงพื้นที่ (GIS) เพื่อให้การแจ้งเตือนแม่นยำและทันเวลา ช่วยลดโอกาสการเกิดอุบัติเหตุ ระบบยังเก็บข้อมูลอุบัติเหตุเพื่อวิเคราะห์แนวโน้มและช่วยหน่วยงานที่เกี่ยวข้องปรับปรุงความปลอดภัย ทั้งนี้ แอปถูกออกแบบให้ใช้งานง่ายเพื่อให้ผู้ขับขี่สามารถตอบสนองได้ทันที

นิยามศัพท์เฉพาะ

การเตือนภัย หมายถึง การแจ้งเตือนล่วงหน้าผ่านระบบในแอปพลิเคชัน RoadSafe Alert ที่มุ่งเน้นให้ผู้ใช้งานมีความปลอดภัยจากการขับขี่ โดยตรวจจับและส่งการแจ้งเตือนเมื่อผู้ใช้เข้าใกล้จุดเสี่ยงที่เกิดอุบัติเหตุบ่อยครั้ง ระบบนี้ใช้การประมวลผลตามตำแหน่งแบบเรียลไทม์เพื่อแสดงผลข้อมูลอย่างทันท่วงที การเขียนโค้ดในส่วนนี้ครอบคลุมทั้งฟังก์ชันที่เรียกข้อมูลจากฐานข้อมูลอุบัติเหตุ และการสร้างเงื่อนไขการแจ้งเตือนในหน้า Notification โดยตั้งเกณฑ์ให้มีการแจ้งเตือนเมื่อจำนวนจุดเสี่ยงที่ใกล้เกิน 5 จุด ผู้ใช้สามารถปรับการแจ้งเตือนและรับข้อมูลจุดเสี่ยงได้ในหน้า Map ซึ่งจะช่วยให้ผู้ใช้ทราบล่วงหน้าและมีเวลาในการเตรียมตัว

อุบัติเหตุบนท้องถนน หมายถึง ข้อมูลอุบัติเหตุที่ได้รวบรวมจากแหล่งข้อมูลภาครัฐ โดยเฉพาะกระทรวงคมนาคม ข้อมูลนี้นำมาใช้วิเคราะห์เพื่อตรวจสอบพื้นที่เสี่ยง ระบบได้ออกแบบให้สามารถบันทึกรายละเอียดต่าง ๆ เช่น วันที่ เวลา สถานที่ และสถิติการเสียชีวิตหรือบาดเจ็บของเหตุการณ์นั้น ๆ ผ่านการเชื่อมต่อกับฐานข้อมูล PostgreSQL และการประมวลผลตำแหน่งเพื่อให้ข้อมูลที่ปรากฏในแอปสามารถอ้างอิงข้อมูลจริงได้ แอปยังมีส่วนช่วยให้ผู้ใช้เข้าถึงข้อมูลย้อนหลังเพื่อช่วยในการประเมินความเสี่ยงขณะขับขี่ได้ดียิ่งขึ้น การเขียนโค้ดในส่วนนี้เน้นให้สามารถเชื่อมต่อกับข้อมูลที่อัปเดตและรองรับการค้นหาย้อนหลังตามคำสั่งจากผู้ใช้งาน

ระบบสารสนเทศภูมิศาสตร์ (GIS) หมายถึง การนำเทคโนโลยี GIS เข้ามาประมวลผลข้อมูลเชิงพื้นที่ในระบบ RoadSafe Alert โดยใช้ในการวิเคราะห์ตำแหน่งและการกระจายตัวของอุบัติเหตุที่เกิดขึ้น โดยการเขียนโค้ดในส่วนนี้จะเชื่อมต่อกับ API เพื่อเรียกใช้ข้อมูล GIS และคำนวณระยะรัศมีรอบตำแหน่งผู้ใช้แบบเรียลไทม์ เช่น รัศมี 1,000, 1,500, หรือ 2,000 เมตร ซึ่งสามารถเลือกปรับได้ตามความต้องการ การใช้ GIS ช่วยให้แอปสามารถระบุตำแหน่งของจุดเสี่ยงและแสดงข้อมูลบนแผนที่ได้แม่นยำ ระบบนี้ยังต้องทำงานร่วมกับ PostgreSQL เพื่อจัดการฐานข้อมูลที่มีความซับซ้อนให้รวดเร็ว การใช้ GIS ช่วยเพิ่มความสามารถของแอปในการตอบสนองต่อการแจ้งเตือนภัยโดยลดเวลาประมวลผล และแสดงข้อมูลที่สอดคล้องกับความจริงเชิงภูมิศาสตร์

จุดเสี่ยง (Hazardous Zones) หมายถึง พื้นที่ที่มีสถิติการเกิดอุบัติเหตุบ่อยครั้ง และถูกระบุในระบบเพื่อแจ้งเตือนผู้ใช้แอป RoadSafe Alert จุดเสี่ยงเหล่านี้มาจากการวิเคราะห์ข้อมูลอุบัติเหตุในอดีตโดยระบบสารสนเทศภูมิศาสตร์ (GIS) และใช้ฐานข้อมูล PostgreSQL ในการจัดเก็บข้อมูลที่แม่นยำ เพื่อการแสดงตำแหน่งบนแผนที่ การพัฒนาฟังก์ชันในโค้ดจะครอบคลุมการประมวลผลจุดเสี่ยงที่เข้ามาใกล้ผู้ใช้งานและสามารถแสดงข้อมูลรายละเอียด เช่น จำนวนอุบัติเหตุที่เกิดขึ้นในรัศมีที่เลือก วันเวลา และประเภทของอุบัติเหตุ การแจ้งเตือนจะเกิดขึ้นเมื่อผู้ใช้เข้ามาใกล้บริเวณจุดเสี่ยงตามรัศมีที่กำหนด เป็นการเพิ่มความปลอดภัยให้ผู้ขับขี่บนท้องถนน

การแจ้งเตือนแบบเรียลไทม์ หมายถึง ความสามารถของแอปพลิเคชันในการส่งข้อมูลแจ้งเตือนผู้ใช้เมื่อเข้าใกล้จุดเสี่ยงโดยอัตโนมัติ การพัฒนาโค้ดในส่วนนี้จะใช้ทั้งการตรวจสอบตำแหน่งและการแจ้งเตือนผ่านหน้า Notification และการตั้งค่ารัศมีในหน้า Map ผู้ใช้จะได้รับการแจ้งเตือนเมื่ออยู่ในบริเวณที่มีอุบัติเหตุสูง โดยแอปจะวิเคราะห์ตำแหน่งผู้ใช้และจุดเสี่ยงอย่างรวดเร็ว และให้ข้อมูลในรูปแบบที่อ่านง่าย การแจ้งเตือนนี้จะเป็นประโยชน์ในระหว่างการขับขี่ซึ่งผู้ใช้จะได้รับการอัปเดต อย่างทันท่วงทีเมื่อต้องผ่านจุดที่มีความเสี่ยงสูง ลดโอกาสการเกิดอุบัติเหตุซ้ำซ้อน

การรายงานแบบเรียลไทม์ (Real-Time Reporting) หมายถึง ฟังก์ชันที่ผู้ใช้สามารถส่งข้อมูลอุบัติเหตุหรือเหตุการณ์ที่พบเจอโดยตรงเข้าไปในระบบ ซึ่งข้อมูลนี้จะถูกจัดเก็บในฐานข้อมูลพร้อมบันทึกตำแหน่ง วัน เวลา สาเหตุ และสามารถแนบรูปถ่ายเพื่อเป็นหลักฐาน ผู้ใช้ในพื้นที่ใกล้เคียงจะได้รับการแจ้งเตือนถึงเหตุการณ์นี้ในทันที การพัฒนาโค้ดในส่วนนี้มุ่งให้ผู้ใช้สามารถส่งรายงานที่อัปเดตแบบเรียลไทม์ เพิ่มความสามารถในการติดตามสถานการณ์ปัจจุบันของผู้ใช้งาน และให้เจ้าหน้าที่สามารถรับทราบข้อมูลอย่างรวดเร็ว

ข้อตกลงเบื้องต้น

งานวิจัยนี้มุ่งพัฒนาแอปพลิเคชัน RoadSafe Alert เพื่อช่วยลดความเสี่ยงจากอุบัติเหตุบนท้องถนน โดยใช้เทคโนโลยี GIS และข้อมูลอุบัติเหตุในอดีตเพื่อแจ้งเตือนผู้ใช้เมื่อเข้าใกล้จุดเสี่ยง แอปได้รับการออกแบบเพื่อให้ใช้งานได้จริงและเพิ่มความปลอดภัยในการขับขี่ โดยการวิจัยนี้จะประเมินความสามารถของแอปในการลดความเสี่ยงและความสะดวกในการใช้งาน

เอกสารงานวิจัยที่เกี่ยวข้อง

H Maulid , W Nurhidayat and S J Priyono.(2020) **“SafeDri: A mobile-based application for safety driving”** แอปพลิเคชันมือถือสำหรับการขับขี่อย่างปลอดภัย คือแอปที่ออกแบบเพื่อช่วยลดอุบัติเหตุทางถนนโดยการรวบรวมและวิเคราะห์ข้อมูลจากการขับขี่ เช่น ความเร็ว การเร่งและเบรกหนัก และการใช้เชื้อเพลิง แอปนี้พัฒนาด้วยเทคโนโลยี Android และ Firebase โดยใช้ข้อมูลจากเซ็นเซอร์ในสมาร์ทโฟน, OBD-II Dongle, และข้อมูลภายนอก เช่น แผนที่และข้อมูลการจราจร เพื่อให้ผู้ขับขี่ได้รับข้อมูลที่เป็นประโยชน์และการแจ้งเตือนเมื่อเกิดการละเมิดข้อกำหนด ผู้ใช้สามารถเรียนรู้การขับขี่อย่างปลอดภัย ตั้งเป้าหมายและรับข้อเสนอแนะเกี่ยวกับพฤติกรรมการขับขี่ แอปนี้ได้ผ่านการทดสอบการใช้งานและได้รับข้อเสนอแนะแบบจริงจังเพื่อการปรับปรุง เช่น การตรวจจับอาการเหนื่อยล้า การตรวจสอบเส้นทาง และการป้องกันการชน โดยมุ่งเน้นที่การปรับปรุงประสบการณ์การขับขี่ให้ดีขึ้นในบริบทของการใช้แอปในประเทศต่างๆ

Mary Ann E. Ignaco.(2021) **“Mobile Application for Incident Reporting”** งานวิจัยนี้นำเสนอการพัฒนาแอปพลิเคชันสำหรับการรายงานเหตุการณ์ในฟิลิปปินส์ โดยมุ่งเป้าสำหรับการรายงานการละเมิดกฎหมายและเหตุการณ์อาชญากรรมอย่างรวดเร็วและแม่นยำ แอปฯ นี้ใช้ฟังก์ชัน GPS เพื่อลงทะเบียนพิกัดตำแหน่งของผู้รายงานอัตโนมัติหรือให้ผู้ใช้ปักหมุดตำแหน่งด้วยตนเอง พร้อมส่งข้อมูลประเภทเหตุการณ์ รายละเอียด และภาพถ่ายไปยังเจ้าหน้าที่ท้องถิ่น เจ้าหน้าที่สามารถขอความช่วยเหลือจากหน่วยกู้ภัยหรือส่งรายงานไปยังสถานีตำรวจใกล้เคียงได้ แอปฯ ยังมีระบบการจัดการข้อมูลและสร้างรายงานสถิติ การพัฒนาใช้วิธี Agile เพื่อเพิ่มความยืดหยุ่นและการตอบสนองต่อการเปลี่ยนแปลง ใช้เทคนิคการพัฒนาแอปพลิเคชันแบบ Cross-Platform เพื่อให้เข้าถึง

อุปกรณ์หลายประเภท และการประมวลผลภาพเพื่อเพิ่มความแม่นยำในการจัดการข้อมูล แอปฯ ได้รับการประเมินตามมาตรฐาน ISO 25010 และได้รับผลตอบรับที่ดีจากการทดสอบ

Elizabeth Liñan Espinoza ,Erich Jordy Echevarria Carpio ,Laberiano Andrade-Arenas .(2021) **"Immediate Notification of Traffic Accidents through a Mobile Application"** เป็นการพัฒนาแอป ฯ มือถือที่ใช้ Scrum Methodology เพื่อแจ้งเตือนอุบัติเหตุจราจรทันที แอปมีปุ่มฉุกเฉินที่ช่วยผู้บาดเจ็บและติดต่อศูนย์ช่วยเหลือใกล้เคียง เช่น ตำรวจ, หน่วยดับเพลิง และโรงพยาบาล วิธีการ Scrum ช่วยจัดการโครงการซอฟต์แวร์อย่างเป็นระบบ โดยมีบทบาทหลักคือ Product Owner, Scrum Master และ Development Team ซึ่งช่วยลดความซับซ้อนและปรับตัวตามความต้องการที่เปลี่ยนแปลงได้ แอปใช้ GPS เพื่อตรวจจับตำแหน่งและส่งการแจ้งเตือนไปยังศูนย์ช่วยเหลือที่ใกล้ที่สุด การพัฒนาต้นแบบแอปได้รับการวางแผนอย่างละเอียดผ่าน Sprint Planning และการประเมินจากการสำรวจผู้ใช้ 30 คนในเปรู เพื่อให้แน่ใจว่าแอปตอบสนองความต้องการและช่วยลดการเสียชีวิตจากอุบัติเหตุโดยการให้ความช่วยเหลืออย่างรวดเร็วและมีประสิทธิภาพ

Supanit Angsirikul ,Narawan Kaewthawee ,Pornthip Ploensil ,Janjuree Netsawang ,Saranthum Phurijaruyangkun ,Sumana Kasemsawasdi.(2023) **"Development of Notification System for Traffic Accidents in Thailand"** การพัฒนาแอปพลิเคชันสำหรับแจ้งเตือนอุบัติเหตุจราจรได้รับความสนใจอย่างมากในการวิจัยล่าสุด เพื่อเพิ่มประสิทธิภาพในการช่วยเหลือผู้ประสบเหตุ แนะนำการใช้ปุ่มฉุกเฉินที่สามารถแจ้งเตือนญาติและศูนย์ช่วยเหลืออย่างรวดเร็ว นำเสนอระบบการแจ้งเตือนอัตโนมัติในรถยนต์ที่สามารถส่งข้อมูลอุบัติเหตุไปยังบริการฉุกเฉิน และเน้นการใช้เทคโนโลยีเพื่อรวบรวมข้อมูลอุบัติเหตุจากช่องทางต่าง ๆ เช่น GPS และ Wi-Fi สำหรับการช่วยเหลือฉุกเฉิน กล่าวถึงการใช้เทคโนโลยีติดตามอัตราการเต้นของหัวใจเพื่อลดความเสี่ยงจากการขับขี่ที่ง่วงนอน เน้นการใช้แอปพลิเคชันในการติดตามพฤติกรรมการขับขี่เพื่อป้องกันอุบัติเหตุ แนะนำการใช้สมาร์ทโฟนและเซ็นเซอร์ในการตรวจสอบและแจ้งเตือนความเสี่ยง แอปพลิเคชันล่าสุดได้พัฒนาด้วยเทคโนโลยี Flutter และ Firebase เพื่อให้การแจ้งเตือนและการช่วยเหลือ

Sadda Bharath Reddy ,Pulakandla Vivek Reddy ,KavaliIndra Reddy ,Kalpure Devraj ,Jajimogga Sravanthi ,Khristina Maksudovna Vafaeva.(2024) **"Android accident detection and alert system"**การตรวจจับอุบัติเหตุและระบบแจ้งเตือนด้วยแอป Android

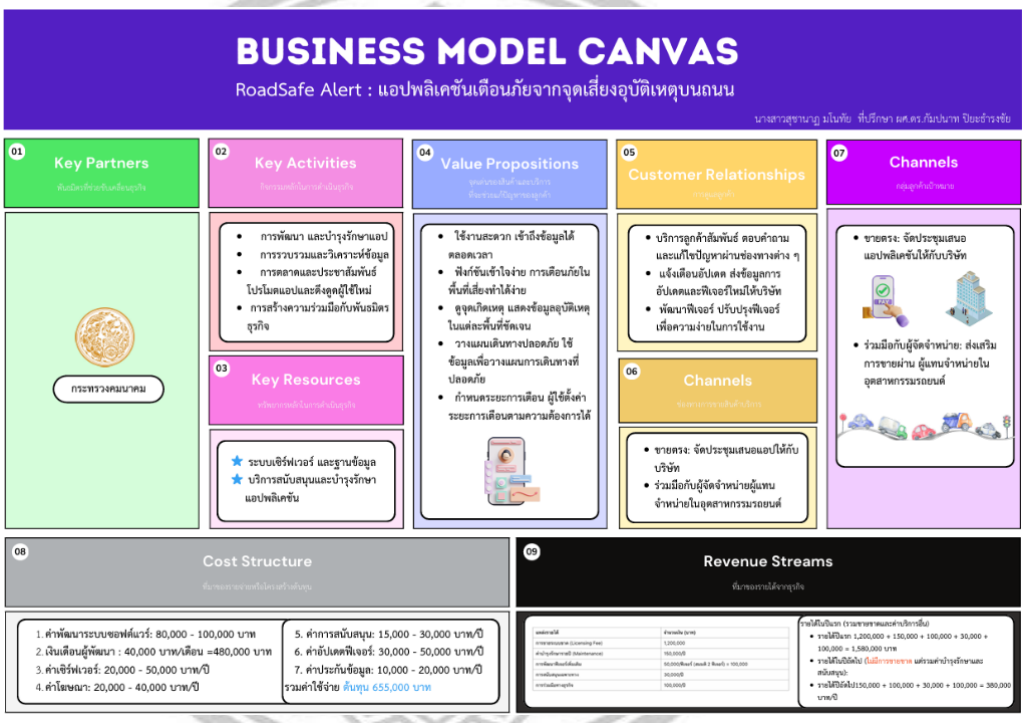
ที่พัฒนาขึ้นเพื่อการตรวจจับอุบัติเหตุและการแจ้งเตือนอัตโนมัติจะดึงข้อมูลตำแหน่ง GPS ทันทีเมื่อเกิดการชน พร้อมเปิดหน้าจอเตือนภัยและโทรหาหมายเลขฉุกเฉินท้องถิ่น ผู้ใช้สามารถขอความช่วยเหลือฉุกเฉินได้ด้วยการแตะเพียงครั้งเดียว แอปจะส่งข้อความเตือนภัยที่มีข้อมูลเกี่ยวกับตำแหน่งที่เกิดอุบัติเหตุและรายละเอียดอื่นๆ ไปยังผู้ติดต่อที่กำหนดไว้ การพัฒนาแอปนี้มีความท้าทาย เช่น การลดการแจ้งเตือนผิดพลาดและการรักษาความปลอดภัยไซเบอร์ ระบบใช้เซนเซอร์เร่งความเร็วและการหมุนเพื่อตรวจจับการชน ข้อมูลตำแหน่งจาก GPS จะช่วยระบุตำแหน่งที่เกิดอุบัติเหตุอย่างแม่นยำ โมดูล GSM หรือ LTE ช่วยในการส่งข้อความและการเชื่อมต่อไร้สาย โดยรวมแล้ว ระบบนี้มีศักยภาพในการลดเวลาตอบสนองฉุกเฉินและช่วยชีวิตได้

การวางแผนและแนวคิดทางธุรกิจ

การวางแผนและแนวคิดทางธุรกิจสำหรับแอปพลิเคชัน RoadSafe Alert มีการปรับปรุงเพื่อมุ่งเน้นการขายเซิร์ฟเวอร์และบริการให้กับบริษัทผู้ผลิตรถยนต์โดยตรง โดยบริษัทเหล่านี้จะติดตั้งระบบ RoadSafe Alert ในรถยนต์ที่มีการจำหน่าย ซึ่งรวมถึงการแถมแอปพลิเคชันสำหรับมือถือให้กับผู้ซื้อรถยนต์ในแพ็กเกจด้วย แอปพลิเคชัน RoadSafe Alert มุ่งเน้นการให้บริการเตือนภัยบนถนนที่แม่นยำและเป็นมิตรกับผู้ใช้งาน โดยใช้ข้อมูลจากสถิติอุบัติเหตุและระบบ GIS (Geographic Information System) เพื่อแสดงพื้นที่เสี่ยงและช่วยผู้ใช้หลีกเลี่ยงเส้นทางที่มีความเสี่ยงสูง แอปพลิเคชันจะทำการแจ้งเตือนเมื่อผู้ใช้เข้าใกล้พื้นที่ที่มีประวัติอุบัติเหตุบ่อยครั้ง

ฟีเจอร์สำหรับผู้ใช้งานทั่วไป สำหรับผู้ใช้งานทั่วไปที่ไม่ใช่รถยนต์ที่มีการติดตั้งระบบ RoadSafe Alert ฟีเจอร์จะถูกลดลงจาก 4 หน้าเหลือ 2 หน้า ซึ่งรวมถึง หน้าแสดงแผนที่และตำแหน่งอุบัติเหตุ แสดงแผนที่พร้อมกับจุดที่มีประวัติอุบัติเหตุ โดยมีการแจ้งเตือนเมื่อเข้าใกล้จุดอันตราย หน้าตั้งค่ารัศมีการแจ้งเตือน ให้ผู้ใช้ตั้งค่ารัศมีสำหรับการแจ้งเตือนเมื่อเข้าใกล้พื้นที่ที่มี อุบัติเหตุบ่อยครั้ง โดยมีกลุ่มเป้าหมาย ประกอบด้วย ผู้ใช้รถยนต์และรถจักรยานยนต์ทั่วไปที่ให้ความสำคัญกับความปลอดภัยในการเดินทาง บริษัทประกันภัยที่ต้องการลดอัตราการเคลมจากอุบัติเหตุ ผู้ให้บริการขนส่งที่ต้องการบริหารจัดการเส้นทางการขนส่งอย่างปลอดภัย และบริษัทผู้ผลิตรถยนต์ที่ต้องการเพิ่มฟังก์ชันความปลอดภัยให้กับยานพาหนะ โครงสร้างต้นทุนและรายได้ RoadSafe Alert จะต้องลงทุนในระบบจัดเก็บและประมวลผลข้อมูลเชิงพื้นที่ และการใช้ API แผนที่สำหรับแสดงตำแหน่งในแอป ค่าใช้จ่ายที่สำคัญรวมถึง ทีมพัฒนาซอฟต์แวร์และการบำรุงรักษาระบบ การปรับปรุงและอัปเดตฟังก์ชันใหม่ ๆ สำหรับรายได้หลักสามารถมาจาก การขายเซิร์ฟเวอร์และบริการให้กับบริษัทผลิตรถยนต์ การร่วมมือ

กับบริษัทประกันภัย เพื่อเป็นโซลูชันเสริมในการช่วยลดอัตราการเคลม การดึงดูดลูกค้า การเปลี่ยนแปลงของเทคโนโลยีและการเพิ่มขึ้นของจำนวนผู้ขับขี่ที่หันมาใช้ยานพาหนะไฟฟ้า (EV) และเทคโนโลยีขับขี่อัตโนมัติจะเพิ่มความสนใจในแอปพลิเคชันที่ช่วยลดความเสี่ยงจากอุบัติเหตุ RoadSafe Alert จะเป็นที่ต้องการเพิ่มขึ้นเรื่อย ๆ เนื่องจากสามารถปรับตัวเข้ากับแนวโน้มของตลาดและตอบสนองต่อความต้องการด้านความปลอดภัยบนท้องถนนได้อย่างมีประสิทธิภาพ



รูปที่ 1 Business Model

แนวคิดและทฤษฎีในการพัฒนาระบบ

ในการศึกษาวิจัยและพัฒนาแอปพลิเคชัน RoadSafe Alert สำหรับแจ้งเตือนจุดเสี่ยงอุบัติเหตุบนท้องถนน ผู้วิจัยได้ศึกษาแนวคิดและทฤษฎีที่เกี่ยวข้องโดยละเอียด มีเนื้อหาดังนี้

1. เครื่องมือข้อมูลที่เกี่ยวข้องที่ช่วยในการพัฒนาเว็บแอปพลิเคชัน

1.1 ภาษาที่ใช่ในการเขียนโปรแกรม

1.1.1 ภาษา Dart

1.1.2 ภาษา JavaScript

1.2 โปรแกรมใช้ในการพัฒนาแอปพลิเคชัน

1.2.1 Android Studio

1.2.2 PostgreSQL/PostGIS

1.2.3 Visual Studio Code

1.3 เฟรมเวิร์กในการพัฒนา

1.3.1 Flutter

1.3.2 Node.js

1.4 API การเชื่อมต่อและดึงข้อมูลอุบัติเหตุในรูปแบบเรียลไทม์จากเซิร์ฟเวอร์ไปยังแอปพลิเคชันที่กำลังพัฒนา มีการใช้ API เพื่อเชื่อมต่อข้อมูล โดยอธิบายถึงการใช้เครื่องมือดังนี้

Flutter SDK เป็นเฟรมเวิร์กที่ช่วยในการพัฒนาแอปพลิเคชันแบบข้ามแพลตฟอร์มที่รองรับทั้ง Android และ iOS ช่วยให้สามารถเขียนโค้ดเพียงครั้งเดียวและใช้งานได้กับหลายระบบ ซึ่งในงานวิจัยนี้ Flutter ใช้เพื่อสร้างอินเทอร์เฟซผู้ใช้ที่สามารถแสดงข้อมูลอุบัติเหตุแบบเรียลไทม์จากเซิร์ฟเวอร์ได้อย่างมีประสิทธิภาพ

http Package ในการเชื่อมต่อกับ API สำหรับการดึงข้อมูลอุบัติเหตุเรียลไทม์จากเซิร์ฟเวอร์ไปยังแอปพลิเคชัน ได้ใช้แพ็กเกจ http ซึ่งเป็นแพ็กเกจมาตรฐานของ Dart และ Flutter ที่ช่วยให้สามารถทำคำร้องขอ HTTP ไปยังเซิร์ฟเวอร์เพื่อดึงข้อมูลได้อย่างง่ายดาย โดยเรียกใช้ HTTP method เช่น GET เพื่อดึงข้อมูล และ POST สำหรับการอัปโหลดข้อมูล

Node.js และ Express.js เซิร์ฟเวอร์ของ API พัฒนาด้วย Node.js และ Express.js ซึ่งทำหน้าที่เป็น backend ที่จัดการข้อมูลและการเชื่อมต่อกับฐานข้อมูล ข้อมูลที่เกี่ยวข้องกับอุบัติเหตุจะถูกจัดเก็บและส่งผ่าน API เพื่อให้แอปสามารถเรียกใช้ได้ในรูปแบบเรียลไทม์

PostgreSQL ฐานข้อมูลที่ใช้สำหรับจัดเก็บข้อมูลอุบัติเหตุถูกพัฒนาและจัดการด้วย ซึ่งเป็นระบบจัดการฐานข้อมูลที่มีประสิทธิภาพสูง เหมาะสำหรับการจัดเก็บข้อมูลในปริมาณมากและรองรับการสืบค้นข้อมูลแบบ geospatial ที่ซับซ้อนได้

API Endpoint ในการดึงข้อมูลอุบัติเหตุในรูปแบบเรียลไทม์ แอปพลิเคชันใช้ endpoint เช่น http://119.59.99.46:3000/api/imgupload/accidentrecords ซึ่งถูกกำหนดในเซิร์ฟเวอร์สำหรับให้ข้อมูลอุบัติเหตุที่อัปเดตล่าสุดแก่แอปพลิเคชัน การกำหนด endpoint ที่ชัดเจนทำให้แอปพลิเคชันสามารถรับข้อมูลที่ต้องการได้อย่างมีประสิทธิภาพและปลอดภัย

Geolocation and Mapping Tools (เช่น flutter_map หรือ Google Maps)

ข้อมูลที่ดึงมาในแอปพลิเคชันสามารถนำไปแสดงผลบนแผนที่เพื่อให้ผู้ใช้เห็นตำแหน่งอุบัติเหตุที่เกิดขึ้น การใช้เครื่องมือเหล่านี้ช่วยให้แอปสามารถแสดงผลข้อมูลในรูปแบบเชิงพื้นที่ ซึ่งช่วยให้ผู้ใช้สามารถมองเห็นแนวโน้มและระบุตำแหน่งได้ง่ายขึ้น

1.5 โปรแกรม Figma ใช้ในการออกแบบหน้าแอปพลิเคชัน

เป็นเครื่องมือออกแบบ UI/UX ที่นิยมใช้ในการพัฒนาแอปพลิเคชันและเว็บไซต์ โดยมีคุณสมบัติเด่นในการสนับสนุนการทำงานร่วมกันแบบเรียลไทม์ นักออกแบบสามารถสร้างไฟล์ออกแบบใหม่และกำหนดขนาดของเฟรมตามอุปกรณ์ที่ต้องการได้ โดยใช้เครื่องมือวาดรูปต่างๆ เช่น Rectangle, Circle, และ Line เพื่อสร้างส่วนประกอบของหน้าแอปพลิเคชัน

Figma มีระบบจัดการเลเยอร์ที่มีประสิทธิภาพ และสนับสนุนการใช้สไตล์และคอมโพเนนต์ซึ่งช่วยให้การออกแบบมีความสอดคล้องและสามารถนำกลับมาใช้ใหม่ได้ นอกจากนี้ ฟีเจอร์การสร้างโปรโตไทป์ช่วยให้นักออกแบบสามารถสร้างลิงก์ระหว่างหน้าเพื่อทดสอบการใช้งานได้อย่างง่ายดาย

ด้วยฟังก์ชันการส่งออกที่รองรับหลายรูปแบบไฟล์ เช่น PNG, JPG, และ SVG Figma จึงเป็นเครื่องมือที่เหมาะสมสำหรับนักออกแบบที่ต้องการพัฒนาแอปพลิเคชัน

1.6 ข้อมูลที่เกี่ยวข้อง

กระทรวงคมนาคม (Ministry of Transport)

สำนักงานสถิติแห่งชาติ (National Statistical Office)

2. เครื่องมือที่ช่วยในการพัฒนาแอปพลิเคชัน

2.1 ภาษาที่ใช้ในการเขียนโปรแกรม

2.1.1 ภาษา Dart

Dart เป็นภาษาการเขียนโปรแกรมที่ถูกพัฒนาโดย Google ในปี พ.ศ. 2554 โดยมีเป้าหมายหลักเพื่อใช้ในการพัฒนาแอปพลิเคชันที่รองรับหลายแพลตฟอร์ม (Cross-Platform) ด้วยความสามารถในการทำงานร่วมกับเฟรมเวิร์ก Flutter จึงเป็นที่นิยมในกลุ่มนักพัฒนาแอปพลิเคชัน Dart ได้รับการออกแบบให้เป็นภาษาที่มีประสิทธิภาพสูง โครงสร้างภาษามีความเรียบง่าย ทำให้โค้ดอ่านง่ายและสะดวกต่อการบำรุงรักษา

Dart มีลักษณะเป็นภาษาแบบ Object-Oriented และรองรับการใช้งาน Asynchronous Programming ได้อย่างมีประสิทธิภาพ จึงเหมาะสมสำหรับการพัฒนาแอปพลิเคชันที่ต้องการตอบสนองรวดเร็วและทำงานแบบเรียลไทม์ นอกจากนี้ การทำงานแบบ Asynchronous ของ Dart ช่วยให้การประมวลผลหลายคำสั่งพร้อมกัน (Concurrency) เป็นไปอย่างราบรื่นโดยไม่ส่งผลต่อประสิทธิภาพของแอป การจัดการหน่วยความจำของ Dart ถูกออกแบบมาเพื่อให้รองรับการทำงานของอุปกรณ์พกพาเป็นพิเศษ จึงมีความสามารถในการจัดสรรทรัพยากรได้อย่างมีประสิทธิภาพ ภาษา Dart ยังมาพร้อมกับชุดไลบรารีมาตรฐานที่ครอบคลุมการใช้งานทั่วไป เช่น การเชื่อมต่อเครือข่าย การจัดการไฟล์ และการเข้าถึงฐานข้อมูล ทำให้นักพัฒนาสามารถเขียนโค้ดได้รวดเร็วและลดความซับซ้อนของการพัฒนาแอปพลิเคชัน Dart มีฟีเจอร์ Hot Reload ซึ่งเป็นเครื่องมือสำคัญที่ช่วยให้นักพัฒนาสามารถทดสอบการเปลี่ยนแปลงโค้ดได้ทันทีโดยไม่ต้องเริ่มแอปใหม่ ทำให้ประหยัดเวลาและเพิ่มประสิทธิภาพในกระบวนการพัฒนา

2.1.2 ภาษา JavaScript

JavaScript เป็นภาษาโปรแกรมเชิงวัตถุที่มีความยืดหยุ่นและใช้งานได้หลากหลายซึ่งถูกออกแบบมาเพื่อเพิ่มความสามารถในการโต้ตอบและการทำงานของเว็บไซต์ โดยเริ่มต้นจากการใช้ในเบราว์เซอร์เพื่อสร้างเว็บไซต์ที่มีความโต้ตอบ เช่น การตอบสนองต่อการคลิก การแสดงข้อมูลแบบเรียลไทม์ และการจัดการฟอร์มการส่งข้อมูล โดยการเขียนโค้ด JavaScript สามารถทำได้ทั้งในฝั่งเซิร์ฟเวอร์ด้วย Node.js และในฝั่งไคลเอนต์ผ่านเบราว์เซอร์ นอกจากนี้ JavaScript ยังมีฟีเจอร์ที่สนับสนุนการเขียนโปรแกรมแบบเชิงฟังก์ชันและเชิงวัตถุ ซึ่งทำให้ผู้พัฒนาสามารถสร้างโค้ดที่มีความสามารถในการนำกลับมาใช้ใหม่ได้ดี และสามารถจัดการกับข้อมูลที่ซับซ้อนได้อย่างมีประสิทธิภาพ

อีกทั้งยังมีไลบรารีและเฟรมเวิร์กมากมาย เช่น React, Angular, และ Vue.js ที่ช่วยให้การพัฒนาแอปพลิเคชันเว็บมีความสะดวกและรวดเร็วขึ้น รวมถึงเครื่องมือการพัฒนาที่มีอยู่ในเบราว์เซอร์ที่ช่วยในการดีบักและปรับแต่งโค้ด JavaScript ทำให้สามารถพัฒนาแอปพลิเคชันที่มีประสิทธิภาพและสามารถทำงานได้ทั้งบนอุปกรณ์มือถือและเดสก์ท็อป นอกจากนี้ การสนับสนุนจากชุมชนที่กว้างขวางทำให้ JavaScript ยังคงเป็นหนึ่งในภาษาโปรแกรมที่ได้รับความนิยมสูงสุดในโลกการพัฒนาเว็บและแอปพลิเคชันต่าง ๆ

2.2 โปรแกรมที่ใช้ในการสร้างแอปพลิเคชัน

2.2.1 Android Studio

เป็นสภาพแวดล้อมในการพัฒนาซอฟต์แวร์ (Integrated Development Environment หรือ IDE) ที่พัฒนาโดย Google ซึ่งออกแบบมาเพื่อใช้สร้างแอปพลิเคชันบนระบบปฏิบัติการ Android อย่างมีประสิทธิภาพและสะดวกสบาย Android Studio ได้รับการพัฒนาโดยอ้างอิงจาก IntelliJ IDEA ซึ่งเป็น IDE ที่ได้รับความนิยม โดย Android Studio เพิ่มเติมเครื่องมือและฟีเจอร์ที่ช่วยอำนวยความสะดวกในการพัฒนาแอป Android ให้มีประสิทธิภาพยิ่งขึ้น ฟีเจอร์ที่สำคัญของ Android Studio ประกอบด้วย Android Emulator ซึ่งช่วยให้นักพัฒนาสามารถทดสอบแอปพลิเคชันในหลากหลายรูปแบบของอุปกรณ์เสมือน ทั้งในด้านขนาดหน้าจอ ความละเอียด และเวอร์ชันของ Android ทำให้สามารถตรวจสอบความเข้ากันได้ของแอปพลิเคชันบนอุปกรณ์ต่างๆ ได้อย่างรวดเร็วและครอบคลุม นอกจากนี้ Android Studio ยังมี Layout Editor ซึ่งเป็นเครื่องมือที่ช่วยในการออกแบบและปรับแต่งอินเทอร์เฟซของแอปพลิเคชัน ทำให้สามารถสร้างอินเทอร์เฟซที่ตอบสนองต่อการใช้งานจริงได้อย่างรวดเร็วโดยไม่จำเป็นต้องเขียนโค้ดทั้งหมด

นอกจากนี้ Android Studio ยังรองรับ Gradle ซึ่งเป็นเครื่องมือในการจัดการการสร้างโครงสร้างโปรเจกต์ (Build System) ที่ยืดหยุ่น ช่วยในการจัดการการตั้งค่า การพึ่งพา (Dependencies) และการจัดการเวอร์ชันของไลบรารีที่ใช้ภายในโปรเจกต์ได้อย่างมีประสิทธิภาพ ทั้งนี้ Gradle ช่วยให้นักพัฒนาสามารถกำหนดการสร้างโปรเจกต์สำหรับหลายเวอร์ชันได้ภายในโปรเจกต์เดียวกัน ซึ่งช่วยลดเวลาและความยุ่งยากในการจัดการโปรเจกต์ขนาดใหญ่ Android Studio ยังมีฟีเจอร์ Code Editor ที่มีเครื่องมือช่วยเขียนโค้ด เช่น การเติมคำอัตโนมัติ (Auto-complete) การตรวจสอบความถูกต้องของโค้ด (Linting) และการดีบัก (Debugging) ที่ช่วยให้นักพัฒนาสามารถค้นหาและแก้ไขข้อผิดพลาดได้รวดเร็วขึ้น อีกทั้งยังมี Firebase ซึ่งเป็นชุดเครื่องมือที่ช่วยในการจัดการฟีเจอร์ต่างๆ เช่น การแจ้งเตือนแบบ Push Notification การเก็บข้อมูลแบบเรียลไทม์ และการวิเคราะห์พฤติกรรมผู้ใช้งาน โดยสามารถเชื่อมต่อ Firebase เข้ากับโปรเจกต์ได้โดยตรงจาก Android Studio ด้วยคุณสมบัติและเครื่องมือต่างๆ ของ Android Studio นักพัฒนาสามารถพัฒนาแอปพลิเคชันบนระบบปฏิบัติการ Android ได้อย่างครบถ้วน ตั้งแต่การออกแบบอินเทอร์เฟซไปจนถึงการทดสอบและดีบัก ทำให้ Android Studio เป็นเครื่องมือที่มีประสิทธิภาพและได้รับความนิยมอย่างแพร่หลายในกลุ่มนักพัฒนา Android

2.2.2 PostgreSQL

เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์แบบโอเพนซอร์สที่มีความสามารถสูง (Open-Source Relational Database Management System - RDBMS) ซึ่งถูกพัฒนาขึ้นครั้งแรกในปี พ.ศ. 2539 โดยมุ่งเน้นให้มีความสามารถในการทำงานร่วมกับข้อมูลเชิงโครงสร้างและการประมวลผลข้อมูลเชิงซับซ้อน PostgreSQL รองรับการทำงานกับโครงสร้างข้อมูลที่มีความซับซ้อน เช่น ข้อมูลเชิงพื้นที่ (Geospatial Data) และข้อมูลเชิงการวิเคราะห์ขั้นสูง จึงเป็นระบบฐานข้อมูลที่เหมาะสมกับการพัฒนาแอปพลิเคชันที่ต้องการจัดเก็บข้อมูลจำนวนมากและมีความหลากหลายในลักษณะการประมวลผลข้อมูล

หนึ่งในจุดเด่นของ PostgreSQL คือการสนับสนุนมาตรฐาน SQL (Structured Query Language) และยังมีการขยายความสามารถโดยการเพิ่มประเภทข้อมูล (Data Types) ใหม่ๆ และฟังก์ชันการทำงานที่หลากหลาย เช่น การจัดการข้อมูล JSON และ JSONB ซึ่งช่วยให้นักพัฒนาสามารถทำงานกับข้อมูลที่ไม่มีโครงสร้าง (Unstructured Data) ได้อย่างยืดหยุ่น PostgreSQL ยังมีฟังก์ชันการทำงานสำหรับการจัดการข้อมูลเชิงพื้นที่ (GIS - Geographic Information System) ผ่าน PostGIS ซึ่งเป็นส่วนขยายที่ช่วยในการจัดการข้อมูลเชิงตำแหน่ง (Geospatial Data) อย่างมีประสิทธิภาพ เช่น การสร้างบัฟเฟอร์ (Buffer), การคำนวณระยะทาง และการวิเคราะห์ข้อมูลเชิงพื้นที่ขั้นสูง PostgreSQL ยังรองรับฟังก์ชันการทำงานที่ช่วยในการจัดการธุรกรรม (Transaction Management) แบบ ACID (Atomicity, Consistency, Isolation, Durability) ซึ่งช่วยรับประกันความถูกต้องและความสมบูรณ์ของข้อมูลในกรณีที่เกิดความผิดพลาดระหว่างการประมวลผล ทำให้เหมาะสำหรับการใช้งานในแอปพลิเคชันที่มีการทำธุรกรรมที่มีความสำคัญสูง เช่น ระบบการเงิน การจัดการคลังสินค้า และระบบการจองห้องพัก เป็นต้น

ในแง่ของการทำงานร่วมกัน PostgreSQL รองรับการเชื่อมต่อกับภาษาการเขียนโปรแกรมหลากหลาย เช่น Python, Java, และ Node.js ซึ่งช่วยให้นักพัฒนาสามารถสร้างแอปพลิเคชันที่เชื่อมโยงฐานข้อมูลกับระบบอื่นๆ ได้อย่างง่ายดาย นอกจากนี้ PostgreSQL ยังมีฟีเจอร์ที่ช่วยในการจัดการผู้ใช้ (User Management) และการกำหนดสิทธิ์การเข้าถึงข้อมูลในระดับที่ละเอียดอ่อน (Granular Access Control) เพื่อเพิ่มความปลอดภัยในการใช้งานด้วยความสามารถในการจัดการข้อมูลที่ซับซ้อน ฟีเจอร์ที่รองรับการทำงานเชิงวิเคราะห์ขั้นสูง และการสนับสนุนการทำงานร่วมกับ

เทคโนโลยีอื่นๆ ทำให้ PostgreSQL เป็นฐานข้อมูลที่ได้รับความนิยมในกลุ่มนักพัฒนาและองค์กรที่ต้องการระบบฐานข้อมูลที่มีประสิทธิภาพ ยืดหยุ่น และรองรับการขยายตัวได้

2.2.3 Visual Studio Code

Visual Studio Code (VS Code) เป็นโปรแกรมแก้ไขข้อความ (Text Editor) ที่พัฒนาโดย Microsoft ซึ่งได้รับความนิยมในหมู่นักพัฒนาซอฟต์แวร์อย่างรวดเร็วตั้งแต่เปิดตัวในปี พ.ศ. 2557 ด้วยความสามารถในการใช้งานที่หลากหลายและประสิทธิภาพสูง VS Code จึงถูกออกแบบมาเพื่อตอบสนองความต้องการของนักพัฒนาที่ต้องการเครื่องมือที่มีความยืดหยุ่นและใช้งานง่ายในการเขียนโค้ดในหลากหลายภาษาโปรแกรม

หนึ่งในคุณสมบัติที่โดดเด่นของ Visual Studio Code คือ การสนับสนุนหลายภาษา (Multi-language Support) ซึ่งช่วยให้สามารถพัฒนาโค้ดในภาษาโปรแกรมที่หลากหลาย เช่น JavaScript, Python, Java, C++, และอื่นๆ โดยมีการปรับแต่งที่ง่ายดายผ่านการติดตั้งส่วนขยาย (Extensions) ซึ่งสามารถดาวน์โหลดและติดตั้งจากตลาดของ VS Code (Marketplace) โดย Extensions เหล่านี้จะช่วยเพิ่มฟีเจอร์ต่างๆ เช่น การเติมคำอัตโนมัติ (Auto-completion), การวิเคราะห์โค้ด (Code Analysis), และการดีบัก (Debugging) VS Code ยังมี ฟังก์ชันการควบคุมเวอร์ชัน (Version Control) ที่รองรับ Git อย่างครบวงจร ซึ่งช่วยให้ผู้ใช้สามารถจัดการกับโค้ดที่พัฒนาในหลายเวอร์ชันได้อย่างสะดวก โดยสามารถตรวจสอบสถานะการเปลี่ยนแปลง (Changes) การสร้างการตรวจสอบ (Commits) และการจัดการกับ Branches ได้จากภายในตัวโปรแกรม โดยไม่จำเป็นต้องใช้คำสั่งในเทอร์มินัล (Terminal) ฟีเจอร์ Live Share เป็นอีกหนึ่งคุณสมบัติที่สำคัญของ Visual Studio Code ซึ่งช่วยให้นักพัฒนาสามารถทำงานร่วมกันแบบเรียลไทม์ โดยสามารถแชร์เซสชันการพัฒนาให้กับผู้ใช้อื่นๆ ได้ ทำให้สามารถแก้ไขโค้ดร่วมกันและสนับสนุนการทำงานแบบทีมได้อย่างมีประสิทธิภาพ นอกจากนี้ VS Code ยังมี Integrated Terminal ซึ่งช่วยให้ผู้ใช้สามารถเข้าถึงเทอร์มินัลภายในโปรแกรมได้ทันที โดยไม่จำเป็นต้องสลับไปยังโปรแกรมอื่น ซึ่งช่วยเพิ่มประสิทธิภาพในการทำงาน

นอกจากนี้ VS Code ยังได้รับการออกแบบให้มี UI ที่เรียบง่ายและใช้งานง่าย ทำให้ผู้ใช้สามารถปรับแต่งธีม (Themes) และการตั้งค่าต่างๆ ได้ตามความต้องการ เช่น การจัดเรียงแถบเครื่องมือ (Toolbars) การปรับขนาดตัวอักษร (Font Size) และการเลือกฟอนต์ (Font Family) เพื่อให้เหมาะสมกับรูปแบบการทำงานและความสะดวกในการพัฒนา ด้วยความสามารถในการปรับแต่งที่หลากหลาย ความสะดวกในการใช้งาน และการสนับสนุนฟีเจอร์ที่ครบครัน ทำให้ Visual

Studio Code เป็นเครื่องมือที่เหมาะสมสำหรับนักพัฒนาทุกระดับ ตั้งแต่นักพัฒนามือใหม่ไปจนถึงนักพัฒนามืออาชีพที่ต้องการสร้างสรรค์โปรเจกต์ที่มีความซับซ้อนและมีคุณภาพสูง

2.3 เฟรมเวิร์กในการพัฒนา

2.3.1 Flutter

เป็นเฟรมเวิร์กโอเพ่นซอร์สที่พัฒนาโดย Google สำหรับการสร้างแอปพลิเคชันที่สามารถทำงานข้ามแพลตฟอร์มได้ เช่น บนระบบปฏิบัติการ iOS, Android, เว็บ และเดสก์ท็อป โดย Flutter ใช้ภาษา Dart เป็นภาษาหลักในการพัฒนา ซึ่งมีลักษณะเฉพาะในการทำงานที่รวดเร็วและประสิทธิภาพสูง เนื่องจากสามารถคอมไพล์โค้ดลงในภาษาเครื่อง (machine code) ได้โดยตรง ทำให้การทำงานของแอปพลิเคชันที่พัฒนาด้วย Flutter มีความลื่นไหลและตอบสนองได้ดี นอกจากนี้ Flutter ยังมาพร้อมกับชุดเครื่องมือและไลบรารีสำหรับสร้าง User Interface ที่มีความยืดหยุ่น และปรับแต่งได้ง่าย ผู้พัฒนาจึงสามารถสร้าง UI ที่มีความสวยงามและสอดคล้องกับแนวทางการออกแบบ Material Design และ Cupertino ได้อย่างครบถ้วน

สำหรับงานวิจัยนี้ Flutter ถูกเลือกใช้ในการพัฒนาแอปพลิเคชัน RoadSafe Alert เนื่องจากความสามารถในการพัฒนาแอปพลิเคชันแบบข้ามแพลตฟอร์มที่สามารถรองรับ Android ได้จากชุดโค้ดเดียว ซึ่งช่วยลดเวลาและทรัพยากรที่ต้องใช้ในการพัฒนาเมื่อเทียบกับการพัฒนาระบบแยกตามแพลตฟอร์ม นอกจากนี้ Flutter ยังรองรับการทำงานแบบ real-time และสามารถเชื่อมต่อกับ API หรือฐานข้อมูลภายนอกได้ง่าย ซึ่งเป็นคุณสมบัติสำคัญที่ช่วยให้แอปพลิเคชันสามารถดึงข้อมูลอุบัติเหตุและส่งการแจ้งเตือนให้กับผู้ใช้ในทันทีเมื่อตรวจจับตำแหน่งที่ผู้ใช้ใกล้ถึงจุดเสี่ยงได้อย่างแม่นยำ

2.3.2 Node.js

Node.js และ Express.js เป็นเครื่องมือสำคัญในการพัฒนาเว็บแอปพลิเคชันและ RESTful API ซึ่งได้รับความนิยมอย่างแพร่หลายในหมู่นักพัฒนา เนื่องจากความยืดหยุ่นและประสิทธิภาพในการประมวลผลข้อมูลในระบบที่รองรับการทำงานแบบ Asynchronous ซึ่งเหมาะสมอย่างยิ่งกับการพัฒนาระบบที่ต้องการการตอบสนองรวดเร็วและรองรับการทำงานหลายคำสั่งพร้อมกัน (Concurrency) ได้อย่างมีประสิทธิภาพ

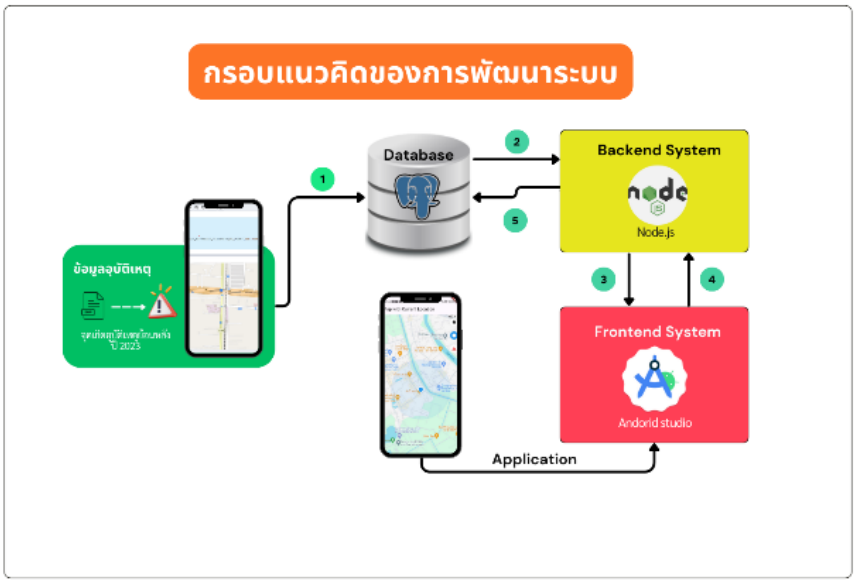
Node.js คือแพลตฟอร์มที่ถูกพัฒนาขึ้นจาก JavaScript Engine V8 ของ Google โดยมีความสามารถในการประมวลผลคำสั่งแบบ Non-blocking ซึ่งช่วยให้สามารถจัดการคำสั่งหลายคำสั่ง

พร้อมกันได้อย่างรวดเร็ว ทำให้ Node.js เป็นที่นิยมในการพัฒนาเซิร์ฟเวอร์และแอปพลิเคชันที่ต้องการการตอบสนองที่รวดเร็ว โดยเฉพาะแอปพลิเคชันที่มีการส่งและรับข้อมูลจากผู้ใช้จำนวนมาก เช่น ระบบการส่งข้อความ แอปพลิเคชันที่ใช้ข้อมูลเรียลไทม์ และแอปพลิเคชันที่มีการทำงานกับฐานข้อมูลบ่อยครั้ง

Express.js เป็นเฟรมเวิร์กของ Node.js ที่ถูกพัฒนาขึ้นเพื่อช่วยให้การพัฒนาแอปพลิเคชันมีความรวดเร็วและสะดวกสบายยิ่งขึ้น โดยมีเครื่องมือและฟังก์ชันที่ช่วยในการจัดการคำขอและการตอบสนองของเซิร์ฟเวอร์ Express.js ช่วยในการกำหนดเส้นทาง (Routing) และจัดการข้อมูลในรูปแบบ JSON ได้อย่างมีประสิทธิภาพ ทั้งนี้ Express.js มีโครงสร้างที่ง่ายต่อการใช้งานและรองรับการพัฒนา RESTful API อย่างเต็มรูปแบบ ซึ่งเป็นโครงสร้างที่มีความสำคัญสำหรับการพัฒนาแอปพลิเคชันที่มีการติดต่อและแลกเปลี่ยนข้อมูลกับฐานข้อมูล หรือเชื่อมต่อกับบริการภายนอกผ่าน API

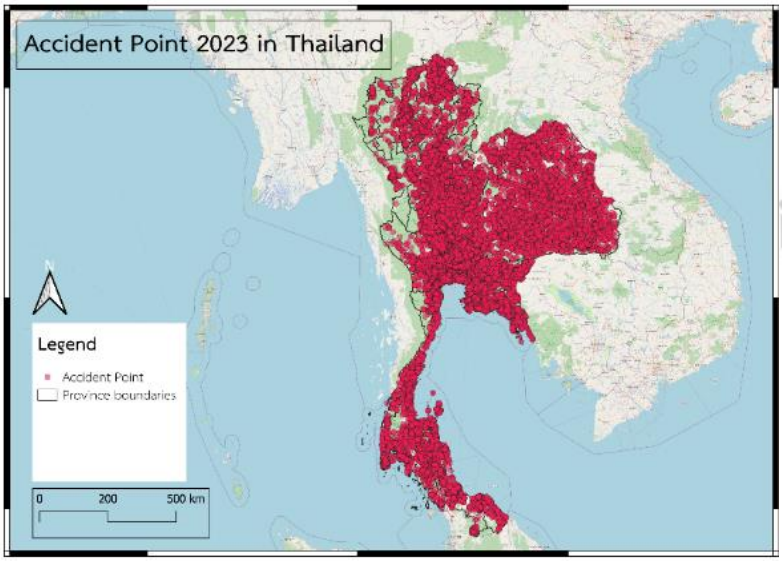
การรวมกันระหว่าง Node.js และ Express.js จึงทำให้เกิดโซลูชันที่มีประสิทธิภาพสูงสำหรับการพัฒนาเว็บแอปพลิเคชันที่ทันสมัย ซึ่งสามารถรองรับการใช้งานบนหลายแพลตฟอร์ม อีกทั้งยังเพิ่มความคล่องตัวในการพัฒนาและการบำรุงรักษาโค้ด ช่วยให้นักพัฒนาสามารถสร้างแอปพลิเคชันที่สามารถตอบสนองต่อผู้ใช้จำนวนมากและให้ประสบการณ์การใช้งานที่ดี

กรอบแนวความคิด



รูปที่ 2 กรอบแนวคิดของการพัฒนาระบบ

การพัฒนาแอปพลิเคชัน RoadSafe Alert มุ่งเน้นที่การสร้างระบบแจ้งเตือนเพื่อความปลอดภัยบนท้องถนนโดยใช้งานข้อมูลอุบัติเหตุในอดีตมาวิเคราะห์เพื่อคาดการณ์และแจ้งเตือนผู้ขับขี่เมื่อเข้าสู่จุดที่มีความเสี่ยงในการเกิดอุบัติเหตุสูง ระบบนี้ประกอบด้วย 2 ส่วนหลักคือระบบ Backend และระบบ Frontend โดยการออกแบบโครงสร้างระบบมีรายละเอียดดังนี้



รูปที่ 3 จุดเกิดอุบัติเหตุในประเทศไทย ช่วงปี ค.ศ.2023

1. **ข้อมูลอุบัติเหตุในอดีตปี 2023**

ที่ได้จากหน่วยงานคมนาคมเพื่อรวบรวมสถิติและระบุตำแหน่งที่มีการเกิดอุบัติเหตุบ่อยครั้ง ข้อมูลนี้จะถูกเก็บในฐานข้อมูลและนำมาวิเคราะห์ในขั้นตอนการประมวลผลของระบบ โดยการประมวลผลนี้จะช่วยสร้าง “จุดเสี่ยง” บนแผนที่ที่สามารถแจ้งเตือนผู้ขับขี่ได้

2. **ระบบ Backend**

2.1 ฐานข้อมูล (Database) ใช้ PostgreSQL ในการจัดเก็บข้อมูลอุบัติเหตุย้อนหลัง รวมถึงข้อมูลจุดเสี่ยงที่ได้จากการประมวลผล ระบบฐานข้อมูลจะต้องสามารถรองรับการเข้าถึงข้อมูลได้อย่างรวดเร็วและมีประสิทธิภาพ โดยเฉพาะในการดึงข้อมูลที่เกี่ยวข้องกับตำแหน่งเพื่อแสดงผลในระบบแจ้งเตือน

2.2 ระบบประมวลผล (Processing System) การพัฒนาในส่วน Backend จะใช้ Node.js ซึ่งทำหน้าที่เป็นเซิร์ฟเวอร์เพื่อประมวลผลข้อมูลที่เกี่ยวข้องกับจุดเกิดเหตุ การคำนวณระยะห่างจากผู้ใช้งานกับจุดเสี่ยง และการส่งข้อมูลไปยัง Frontend เพื่อแจ้งเตือนแบบเรียลไทม์ การประมวลผลนี้ยังรวมถึงการเชื่อมต่อกับ API ที่ดึงข้อมูลแบบสดจาก GPS ของผู้ใช้งานเพื่อตรวจสอบตำแหน่งปัจจุบัน

3. **ระบบ Frontend**

การพัฒนาแอปพลิเคชัน (Application Development) ใช้ Flutter ในการพัฒนา Frontend ของแอปพลิเคชันเพื่อให้รองรับการทำงานทั้งบนระบบปฏิบัติการ Android และ iOS โดย Flutter ช่วยสร้างหน้าตาของแอปที่สวยงามและตอบสนองได้ดี ซึ่งเป็นส่วนสำคัญในการใช้งานของผู้ขับขี่ เช่น การเลือกเปิดปิดการแจ้งเตือนและการปรับตั้งค่าระยะรัศมีที่ต้องการรับการแจ้งเตือน

Android Studio ใช้เป็นเครื่องมือในการทดสอบและพัฒนาแอปสำหรับอุปกรณ์ Android โดยเฉพาะ

4. **ระบบแจ้งเตือน (Alert System)**

แอปพลิเคชันจะดึงข้อมูลที่คำนวณได้จาก Backend เพื่อแจ้งเตือนผู้ใช้ผ่านระบบการแจ้งเตือนแบบเรียลไทม์เมื่อเข้าสู่บริเวณที่มีอุบัติเหตุบ่อยครั้ง ผู้ใช้งานสามารถปรับรัศมีของการแจ้งเตือนให้เหมาะสมกับความต้องการ เช่น การตั้งค่าให้แจ้งเตือนเมื่อเข้าใกล้จุดเสี่ยงในระยะ กำหนดค่ารัศมีเบื้องต้น 600 เมตร เพื่อให้ผู้ขับขี่สามารถเตรียมตัวและเพิ่มความระมัดระวัง

5. ผลการวิเคราะห์และการนำเสนอข้อมูล (Data Analysis and Visualization)

ด้วยการใช้ Flutter และระบบสารสนเทศภูมิศาสตร์ (GIS) แอปพลิเคชัน RoadSafe Alert สามารถแสดงผลข้อมูลจุดเสี่ยงได้ในรูปแบบแผนที่ ผู้ใช้งานจะเห็นจุดที่มีการเกิดอุบัติเหตุบ่อยครั้งและสถิติที่เกี่ยวข้อง ซึ่งช่วยเพิ่มความเข้าใจในการใช้ถนนและทำให้ตัดสินใจได้อย่างปลอดภัยมากขึ้น โครงสร้างของแอปพลิเคชันนี้ถูกออกแบบให้มีการประมวลผลที่รวดเร็ว ข้อมูลที่แม่นยำ และการแสดงผลที่เข้าใจง่าย เพื่อให้ผู้ขับขี่สามารถนำไปใช้งานจริงในการเพิ่มความปลอดภัยบนท้องถนน โดยการทดลองเบื้องต้นแสดงให้เห็นว่าแอปมีศักยภาพในการช่วยลดความเสี่ยงจากอุบัติเหตุ

รูปที่ 4 กรอบแนวคิดการใช้งานระบบ

กรอบแนวคิดการใช้งานระบบแอปพลิเคชัน RoadSafe Alert นี้อาศัยการผสานเทคโนโลยีหลายประเภท เช่น Flutter, PostgreSQL, Node.js, และ GIS เพื่อสร้างระบบแจ้งเตือนสำหรับความปลอดภัยบนท้องถนน โดยดึงข้อมูลย้อนหลังเกี่ยวกับจุดเกิดอุบัติเหตุและวิเคราะห์เพื่อการแจ้งเตือนแก่ผู้ขับขี่เมื่อเข้าสู่พื้นที่เสี่ยง ในการพัฒนาระบบนี้มีหลักการและขั้นตอนสำคัญดังนี้

1. การเชื่อมต่อและจัดเก็บข้อมูลอุบัติเหตุในฐานข้อมูล (Database Connection and Data Storage)

แอปพลิเคชันใช้ PostgreSQL ในการจัดเก็บข้อมูลจุดเกิดอุบัติเหตุ โดยมีข้อมูลหลักที่ใช้ได้แก่พิกัดจุดเกิดอุบัติเหตุ (ละติจูดและลองจิจูด) รวมถึงข้อมูลอุบัติเหตุ เช่นจำนวนครั้งที่เกิดและระดับความรุนแรง

โดยใช้คำสั่ง SQL ในการสร้างและจัดการตารางข้อมูล ซึ่งส่วนของโค้ดที่ใช้เชื่อมต่อกับฐานข้อมูลในเซิร์ฟเวอร์ Node.js จะทำงานผ่านการเชื่อมต่อกับ pg library

2. การสร้างรัศมีการแจ้งเตือน (Buffer Creation)

สำหรับการแจ้งเตือนเมื่อผู้ใช้งานเข้าใกล้จุดเสี่ยง ระบบจะสร้างรัศมีโดยใช้คำสั่ง ST_Buffer ใน PostgreSQL ในการสร้าง Buffer สำหรับพื้นที่แจ้งเตือน ซึ่งสามารถปรับระยะ (เช่น 1000, 1500, หรือ 2000 เมตร) ตามการตั้งค่าของผู้ใช้งาน โค้ด SQL สำหรับการสร้าง Buffer โดยข้อมูล buffer_geom ที่ได้จะถูกส่งไปยังแอปพลิเคชันฝั่ง Flutter เพื่อใช้ในการวิเคราะห์เชิงพื้นที่กับตำแหน่งปัจจุบันของผู้ขับขี่

3. การตรวจสอบตำแหน่งผู้ใช้ภายในรัศมี (Using ST_Within for Proximity Check)

ฟังก์ชัน ST_Within ถูกใช้ในการตรวจสอบว่าตำแหน่งปัจจุบันของผู้ขับขี่อยู่ในรัศมี Buffer ของจุดเกิดเหตุหรือไม่ ซึ่งโค้ดส่วนนี้ทำงานในฝั่งเซิร์ฟเวอร์ Node.js โดยใช้การ Query ข้อมูลจาก PostgreSQL ซึ่งจะดึงข้อมูลจาก current_location (ตำแหน่งผู้ใช้งาน) และ buffer_geom (ตำแหน่งจุดเกิดเหตุ) ที่คำนวณไว้ หากพบว่าผู้ใช้เข้าสู่พื้นที่เสี่ยง แอปจะเริ่มกระบวนการแจ้งเตือน

4. การเรียกใช้ Query ในการแสดงผลข้อมูล (Executing Query for Data Retrieval and Display)

เซิร์ฟเวอร์ Node.js จะทำหน้าที่ Query ข้อมูลจุดเกิดเหตุและส่งข้อมูลไปยัง Flutter ผ่าน API Flutter จะดึงข้อมูลที่ได้มาจาก API เพื่อนำไปแสดงผลเป็นจุดบนแผนที่และให้ผู้ใช้งานเห็นข้อมูลจุดเสี่ยงในพื้นที่ที่กำลังขับขี่

5. การแจ้งเตือน (Notification System)

เมื่อผู้ใช้งานเข้าสู่รัศมีของจุดเสี่ยงที่มีข้อมูลมากกว่า 5 จุด ระบบจะทำการแจ้งเตือนผ่านโค้ด Flutter โดยใช้ flutter_local_notifications package ในการสร้างการแจ้งเตือนแบบเรียลไทม์ เมื่อเซิร์ฟเวอร์ระบุว่าผู้ใช้เข้าสู่พื้นที่ Buffer ระบบจะแจ้งเตือนให้ผู้ใช้ระมัดระวัง โดยแสดงข้อความเตือนที่อิงจากข้อมูลจุดเสี่ยงในบริเวณนั้น

6. การแสดงผลในแอปพลิเคชัน (Frontend System in Flutter)

แอปพลิเคชัน RoadSafe Alert พัฒนาใน Flutter โดยแสดงข้อมูลจุดอุบัติเหตุที่เก็บไว้ในฐานข้อมูลแบบเรียลไทม์บนแผนที่ ผู้ใช้สามารถเห็นจุดเสี่ยงจากอุบัติเหตุและการแจ้งเตือน โดยใช้ flutter_map ในการแสดงผลข้อมูลแผนที่และจุดต่างๆ ที่ดึงมาจาก API ตัวอย่างโค้ดในการดึงข้อมูลจาก API โดยรวมโครงสร้างระบบนี้ได้ออกแบบเพื่อให้ผู้ใช้ได้รับการแจ้งเตือนตามเวลาจริง ช่วยเพิ่มความปลอดภัยในการเดินทาง

7. การแสดงผลแผนที่และข้อมูลภาพ (Photomap Display in Flutter)

ในหน้า photomap.dart ของแอปพลิเคชัน RoadSafe Alert จะทำการแสดงผลข้อมูลจุดเสี่ยงและภาพประกอบที่เกี่ยวข้องเพื่อให้ผู้ใช้เข้าใจสภาพแวดล้อมของจุดเกิดอุบัติเหตุได้ชัดเจนยิ่งขึ้น โดย photomap.dart ใช้ flutter_map สำหรับการแสดงแผนที่ และดึงข้อมูลภาพจากฐานข้อมูลหรือ API เซิร์ฟเวอร์เพื่อแสดงผลควบคู่ไปกับข้อมูลจุดเกิดเหตุ ตัวอย่างโค้ดการแสดงข้อมูลภาพใน Flutter นอกจากนี้ ผู้ใช้ยังสามารถแตะที่จุดเสี่ยงบนแผนที่เพื่อเปิดดูภาพประกอบที่แสดงรายละเอียดเกี่ยวกับสถานที่หรือสภาพแวดล้อมที่จุดเสี่ยงนั้น ๆ ขณะขับขี่ ช่วยเพิ่มมุมมองเชิงพื้นที่และความเข้าใจของผู้ใช้ในการหลีกเลี่ยงจุดอันตราย

ขั้นตอนการดำเนินการ

การพัฒนาแอปพลิเคชัน RoadSafe Alert เพื่อแจ้งเตือนจุดเสี่ยงอุบัติเหตุบนท้องถนน ดำเนินการผ่านขั้นตอนต่าง ๆ อย่างเป็นระบบ ตั้งแต่การวางแผน การพัฒนา ไปจนถึงการทดสอบและประเมินผล โดยรายละเอียดของขั้นตอนการพัฒนาแอปพลิเคชันมีดังนี้

1. การวิเคราะห์และวางแผนโครงการ

ในระยะแรกของการพัฒนา แอปพลิเคชัน RoadSafe Alert ถูกกำหนดวัตถุประสงค์เพื่อพัฒนาแอปที่ช่วยแจ้งเตือนผู้ใช้เมื่อเข้าใกล้บริเวณที่มีความเสี่ยงสูงต่อการเกิดอุบัติเหตุ เป้าหมายสำคัญคือการให้ข้อมูลอุบัติเหตุในพื้นที่นั้น ๆ และแจ้งเตือนผู้ใช้ในระยะที่กำหนด จึงมีการกำหนดกลุ่มเป้าหมายที่คาดว่าจะได้รับประโยชน์จากแอปพลิเคชัน รวมถึงวิเคราะห์ฟีเจอร์ที่ต้องการ เช่น การ

แสดงข้อมูลบนแผนที่ การแจ้งเตือนตามระยะห่าง และการเชื่อมต่อกับฐานข้อมูลที่มีข้อมูลสถิติอุบัติเหตุ

ระยะเวลา: 1 มิถุนายน – 16 กรกฎาคม 2567

2. การออกแบบระบบและโครงสร้างข้อมูล

ข้อมูลอุบัติเหตุที่จำเป็นถูกวางแผนเพื่อนำเข้าจากฐานข้อมูล เช่น PostgreSQL ที่มีข้อมูลตำแหน่งและสถิติอุบัติเหตุจากหน่วยงานต่าง ๆ เช่น กระทรวงคมนาคม (Ministry of Transport) สำนักงานสถิติแห่งชาติ (National Statistical Office) โครงสร้างของฐานข้อมูลและการเชื่อมต่อข้อมูลได้รับการออกแบบเพื่อรองรับการใช้งานตามขนาดของข้อมูลและความเร็วในการประมวลผล

ระยะเวลา: 16 กรกฎาคม – 15 สิงหาคม 2567

3. การออกแบบส่วนติดต่อผู้ใช้ (UI/UX Design)

รูปแบบของแอปพลิเคชันได้รับการออกแบบผ่าน Figma เพื่อสร้างประสบการณ์ผู้ใช้ที่ใช้งานง่ายและมีประสิทธิภาพ ส่วนการออกแบบ UI มุ่งเน้นความเรียบง่ายและฟีเจอร์ที่สำคัญ เช่น แผนที่ (Map View) ที่สามารถแสดงข้อมูลจุดเสี่ยงได้อย่างชัดเจน รวมถึงปุ่มต่าง ๆ ที่ช่วยให้ผู้ใช้เลือกระยะที่ต้องการรับการแจ้งเตือน

ระยะเวลา: 16 สิงหาคม 2567

4. การพัฒนาและเขียนโปรแกรม

การพัฒนาแอปพลิเคชัน RoadSafe Alert เริ่มต้นโดยการพัฒนา Backend ด้วย Node.js และ Express.js เพื่อเชื่อมต่อกับฐานข้อมูล PostgreSQL สำหรับการจัดการและดึงข้อมูลจุดเสี่ยงอุบัติเหตุ ส่วน Frontend ใช้ Flutter และภาษา Dart ในการสร้างอินเทอร์เฟซของแอป การใช้ Flutter ช่วยในการพัฒนาแอปที่สามารถทำงานได้ทั้งบนระบบ Android และ iOS นอกจากนี้ยังมีการพัฒนาระบบแจ้งเตือนผู้ใช้ผ่านการเรียกใช้ API ที่กำหนดการแจ้งเตือนตามระยะห่างจากจุดเสี่ยง

ระยะเวลา: 17 สิงหาคม – 15 ตุลาคม 2567

5. การเชื่อมต่อข้อมูลและทดสอบระบบ

การพัฒนาแอปพลิเคชัน RoadSafe Alert ต้องอาศัยการเชื่อมต่อข้อมูลจากฐานข้อมูล PostgreSQL เพื่อดึงข้อมูลจุดเสี่ยงอุบัติเหตุขึ้นมาแสดงบนแผนที่ในแอปพลิเคชัน การทดสอบระบบประกอบด้วยการตรวจสอบการทำงานของฟีเจอร์แผนที่ การคำนวณระยะ และการแจ้งเตือน โดยมีการทดสอบทั้งในสภาพแวดล้อมจำลองและบนอุปกรณ์จริงเพื่อให้มั่นใจว่าแอปสามารถทำงานได้อย่างถูกต้องและมีเสถียรภาพ

6. การประเมินผลและปรับปรุง

เมื่อการทดสอบเสร็จสมบูรณ์ ผลการทดสอบและความคิดเห็นของผู้ใช้นำมาใช้เพื่อปรับปรุงระบบ ทั้งในด้าน UI และการทำงานของระบบ เช่น ปรับการคำนวณระยะการแจ้งเตือนให้แม่นยำยิ่งขึ้น และเพิ่มประสิทธิภาพการประมวลผลข้อมูลจุดเสี่ยง การทดสอบและปรับปรุงนี้มีความสำคัญอย่างยิ่งเพื่อให้แอปพลิเคชัน RoadSafe Alert มีความสมบูรณ์และพร้อมใช้งานจริง การจัดทำเอกสารนำเสนอมีความสำคัญสำหรับการสื่อสารผลการทำงานแก่ผู้เกี่ยวข้อง รวมถึงการสนับสนุนความเข้าใจและการนำแอปพลิเคชันไปใช้งานจริง

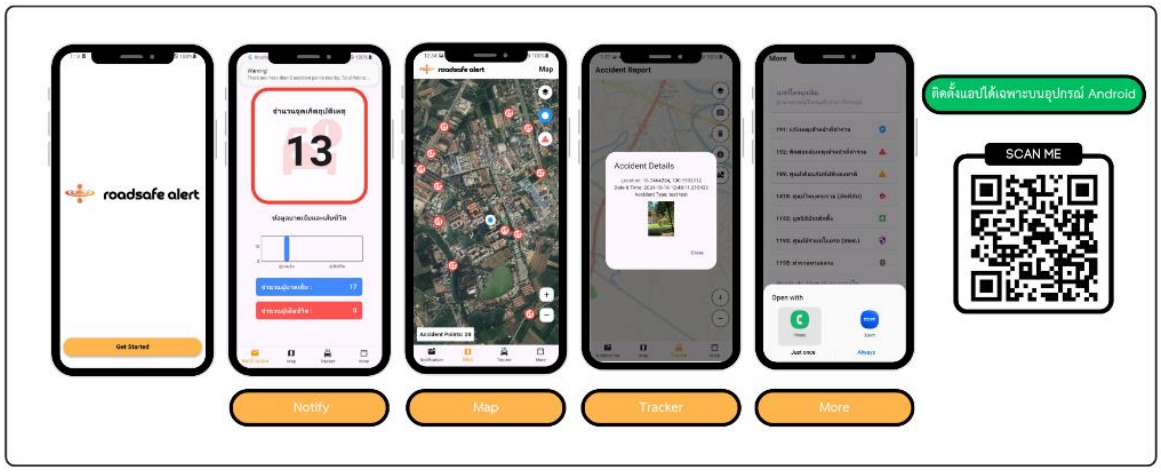
ระยะเวลา: 16 ตุลาคม - 18 ตุลาคม 2567

7. สรุปผลการพัฒนาแอปพลิเคชันพร้อมจัดทำรูปเล่มฉบับสมบูรณ์

เมื่อการพัฒนาและทดสอบแอปพลิเคชัน RoadSafe Alert เสร็จสมบูรณ์ ขั้นตอนสุดท้ายคือการสรุปผลการพัฒนาแอปพลิเคชันและจัดทำเอกสารเพื่อนำเสนอผลลัพธ์ของโครงการ เอกสารนี้ครอบคลุมรายละเอียดการออกแบบ การพัฒนา ฟังก์ชันการทำงานหลัก การทดสอบระบบ และปัญหาที่พบระหว่างการพัฒนา รวมถึงวิธีการแก้ไข นอกจากนี้ยังมีการจัดทำข้อเสนอแนะเพื่อพัฒนาต่อยอดในอนาคต

ระยะเวลา: 19 ตุลาคม – 31 ตุลาคม 2567

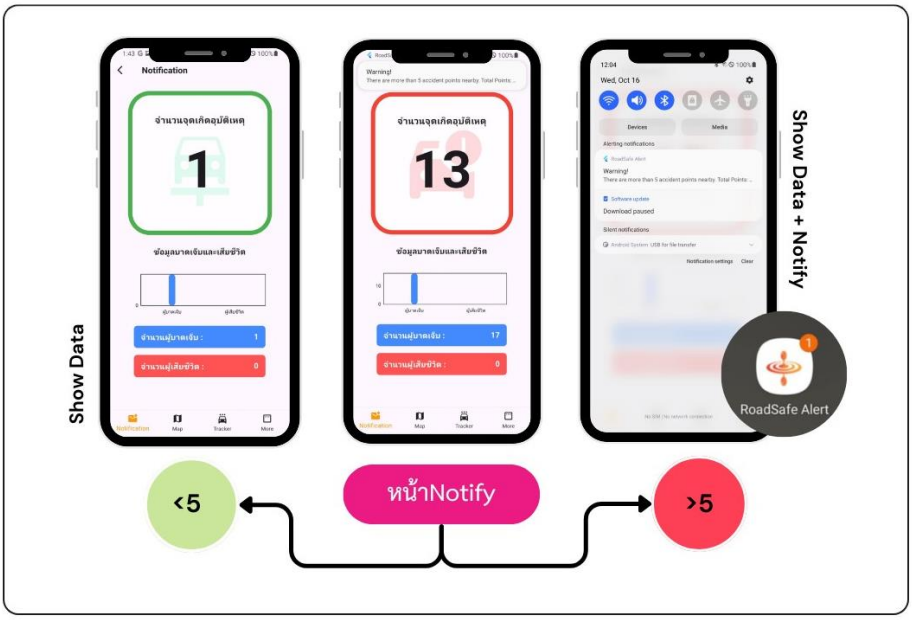
ขั้นตอนการพัฒนาระบบ



รูปที่ 5 การแสดงผลหน้าแอปพลิเคชัน

1. การออกแบบหน้า Notification

แอปพลิเคชันถือเป็นส่วนสำคัญในการแจ้งเตือนผู้ใช้ถึงพื้นที่ที่มีอุบัติเหตุสูงโดยใช้การแสดงผลแบบโต้ตอบและการใช้ทรัพยากรจากระบบเพื่อแจ้งเตือนผ่านการสั่นหรือการแจ้งเตือนข้อความในอุปกรณ์ ข้างล่างนี้เป็นขั้นตอนเชิงวิชาการของการทำงานในหน้า Notification พร้อมโค้ดที่ได้รับ



รูปที่ 6 การแสดงผลหน้า Notification

1.1 การเตรียมข้อมูลและการตั้งค่าการแจ้งเตือน

โครงสร้างของคลาส `_Notification1State` ภายใต้คลาส `Notification1` มีหน้าที่ในการกำหนดค่าการแจ้งเตือนและการตรวจจับตำแหน่งปัจจุบันของผู้ใช้ การเตรียมข้อมูลดังกล่าวมีขั้นตอนดังนี้

1.1.1 การสร้างอินสแตนซ์ของ `ApiService` ใช้เพื่อเรียกข้อมูลอุบัติเหตุที่อยู่ใกล้เคียงจาก API ที่ระบุไว้ใน URL ซึ่งถูกตั้งค่าให้เป็น``http://119.59.99.46:3000/api/accapp'`

```
class _Notification1State extends State<Notification1> {
  late ApiService apiService;
  Position? currentLocation;
  late FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin;

  @override
  void initState() {
    super.initState();
    apiService = ApiService('http://119.59.99.46:3000/api/accapp');
    flutterLocalNotificationsPlugin = FlutterLocalNotificationsPlugin();
    var initializationSettingsAndroid = AndroidInitializationSettings('@mipmap/ic_launcher');
    var initializationSettings = InitializationSettings(android: initializationSettingsAndroid);
    flutterLocalNotificationsPlugin.initialize(initializationSettings);
    getLocation(); // เรียกฟังก์ชันเพื่อดึงตำแหน่งปัจจุบัน
  }
```

รูปที่ 7 API ที่ระบุไว้ใน URL

การเชื่อมโยง API กับหน้า Notification

เมื่อผู้ใช้เปิดหน้า Notification แอปพลิเคชันจะทำการเรียกใช้ API ผ่านคำสั่ง GET ซึ่งกำหนดไว้ในเซิร์ฟเวอร์ Express เช่น app.get('/getaccidentpoints/:lat/:lng/:dist', ...) เพื่อดึงข้อมูลจุดอุบัติเหตุที่อยู่ในระยะที่ผู้ใช้กำหนด โดยจะส่งพารามิเตอร์ที่จำเป็น เช่น ละติจูด (latitude), ลองจิจูด (longitude) และระยะทาง (distance) ไปยัง API เพื่อให้ได้ข้อมูลตามตำแหน่งของผู้ใช้

1) /getbufferzone/:lat/:lng/:dist

รับพารามิเตอร์ lat, lng, และ dist เพื่อสร้างเขตการคุ้มครองรอบจุดที่กำหนด

ใช้ ST_Buffer เพื่อสร้างโซนและส่งข้อมูล GeoJSON กลับไป

2) /getaccidentpoints/:lat/:lng/:dist

รับพารามิเตอร์ lat, lng, และ dist เพื่อค้นหาจุดอุบัติเหตุภายในเขตที่กำหนด

ใช้ ST_Within เพื่อตรวจสอบจุดอุบัติเหตุและส่งข้อมูล เช่น จำนวนผู้บาดเจ็บและตำแหน่งกลับไปยังผู้ใช้

```
app.get('/getbufferzone/:lat/:lng/:dist',async (req,res)=>{
    const latitude = req.params.lat;
    const longitude = req.params.lng;
    const distance = req.params.dist;

    let data
    const sql = `select st_asgeojson(ST_Transform(ST_Buffer(ST_Transform(ST_POINT(${longitude},${latitude}, 4326), 32647),${distance}), 4326)) as gjson
`
    const result = await pool.query(sql)
    data = result.rows
    console.log(data);
    djson = JSON.parse(data[0].gjson)
    console.log(djson);
    res.send(djson)
})
app.get('/getaccidentpoints/:lat/:lng/:dist', (req, res) => {
    const latitude = parseFloat(req.params.lat);
    const longitude = parseFloat(req.params.lng);
    const distance = parseFloat(req.params.dist);
    const accident = parseFloat(req.params.acc);

    const sql = `
    SELECT
        a.gid,
        a.injured,
        a.dead,
        a.udead,
        a.accplace,  -- Add the accplace column here
        ST_AsGeoJSON(a.geom) as geom
    FROM
        accroad as a
    WHERE
        ST_Within(
            ST_Transform(a.geom, 4326),
            ST_Transform(
                ST_Buffer(
                    ST_Transform(
                        ST_SetSRID(ST_Point(${longitude}, ${latitude}), 4326),
                        32647
                    ),
                    ${distance}
                ),
                4326
            )
        )
    `;
```

รูปที่ 8 ทำงานร่วมกันเพื่อดึงข้อมูลอุบัติเหตุจากฐานข้อมูลตามพิกัดที่กำหนดโดยผู้ใช้

โครงสร้างของข้อมูลข้อมูลนี้เป็นลิสต์ของอุบัติเหตุ โดยแต่ละอุบัติเหตุมีคุณสมบัติที่สำคัญ ได้แก่ gid: หมายเลขประจำตัวของอุบัติเหตุ (Unique Identifier) เพื่อใช้ในการระบุแต่ละอุบัติเหตุในฐานข้อมูล

injured: จำนวนผู้ที่ได้รับบาดเจ็บในอุบัติเหตุนี้ โดยใช้ค่าเป็นสตริง

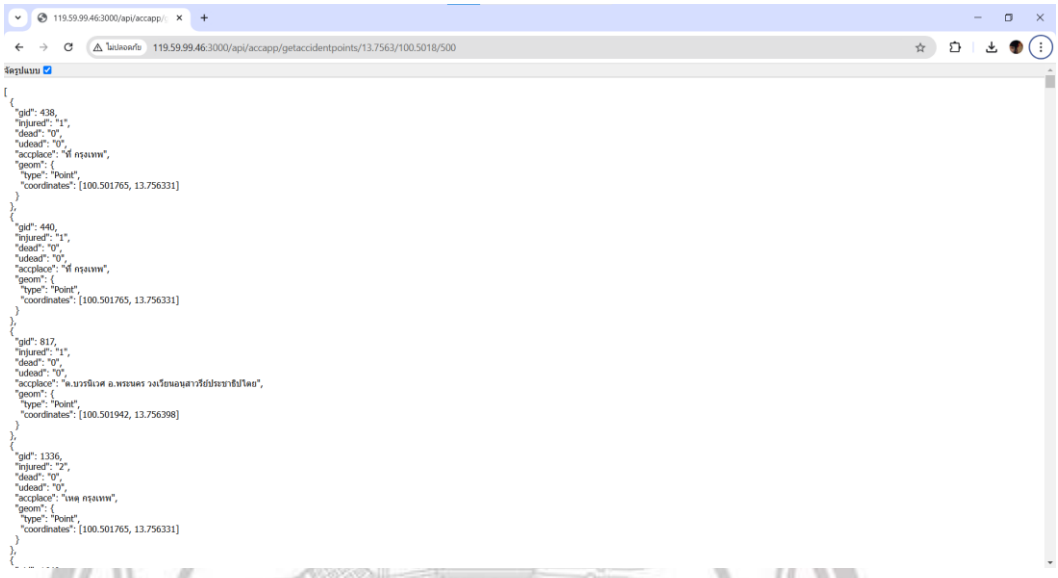
dead: จำนวนผู้ที่เสียชีวิตจากอุบัติเหตุนี้

udead: จำนวนผู้ที่บาดเจ็บซึ่งไม่เข้ารับการรักษา (undocumented dead)

accplace: สถานที่เกิดอุบัติเหตุ โดยในที่นี้มีการระบุเป็นตำแหน่งที่ตั้ง (เช่น "จ กท" หรือ "ที่ กรุงเทพ")

geom: ข้อมูลทางภูมิศาสตร์ของอุบัติเหตุ ประกอบด้วย type ประเภทของข้อมูลเชิงพื้นที่ (ในที่นี้คือ "Point" หมายถึงจุด)

coordinates พิกัดทางภูมิศาสตร์ของอุบัติเหตุในรูปแบบ [longitude, latitude]



รูปที่ 9 ข้อมูลที่มีให้มาคือข้อมูล JSON ที่แสดงถึงจุดที่เกิดอุบัติเหตุ

ตำแหน่งการดึงข้อมูลจุดอุบัติเหตุจาก API นี้อยู่ในฟังก์ชัน _loadAccidents โดยใช้คำสั่ง http.get เพื่อส่งคำขอไปยัง URL ของ API ที่ให้มา เพื่อนำข้อมูลที่ได้มาสร้าง Marker บนแผนที่ ด้วย final response = await http.get(Uri.parse("http://119.59.99.46:3000/api/accapp/getaccidentpoints/$lat/$lng/$radius"));

1.2 การกำหนดค่า `FlutterLocalNotificationsPlugin` ใช้เพื่อแสดงการแจ้งเตือนในอุปกรณ์ โดยมีการตั้งค่า `AndroidInitializationSettings` และ `InitializationSettings` ซึ่งจะใช้ไอคอนจาก `@mipmap/ic_launcher` ในการแสดงผล


```
class _Notification1State extends State<Notification1> {
  late ApiService apiService;
  Position? currentLocation;
  late FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin;

  @override
  void initState() {
    super.initState();
    apiService = ApiService('http://119.59.99.46:3000/api/accapp');
    flutterLocalNotificationsPlugin = FlutterLocalNotificationsPlugin();
    var initializationSettingsAndroid = AndroidInitializationSettings('@mipmap/ic_launcher');
    var initializationSettings = InitializationSettings(android: initializationSettingsAndroid);
    flutterLocalNotificationsPlugin.initialize(initializationSettings);
    getLocation(); // เรียกฟังก์ชันเพื่อดึงตำแหน่งปัจจุบัน
  }
```

รูปที่ 10 กำหนดค่าเพื่อแสดงการแจ้งเตือน

1.3 การตรวจจับตำแหน่งปัจจุบัน ใช้ `Geolocator` ในการรับตำแหน่งที่ถูกต้องของผู้ใช้ พร้อมทั้งตรวจสอบสิทธิ์การใช้งานการเข้าถึงตำแหน่ง

```
Future<void> getLocation() async {
  bool serviceEnabled;
  LocationPermission permission;

  serviceEnabled = await Geolocator.isLocationServiceEnabled();
  if (!serviceEnabled) {
    return Future.error('Location services are disabled.');
  }

  permission = await Geolocator.checkPermission();
  if (permission == LocationPermission.denied) {
    permission = await Geolocator.requestPermission();
    if (permission == LocationPermission.denied) {
      return Future.error('Location permissions are denied');
    }
  }

  if (permission == LocationPermission.deniedForever) {
    return Future.error('Location permissions are permanently denied.');
  }

  Position position = await Geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.best);
  setState(() {
    currentLocation = position;
  });
```

รูปที่ 11 เพื่ออัปเดตพิกัดปัจจุบันของผู้ใช้และแจ้งให้หน้าจอเพื่อแสดงข้อมูลตำแหน่งล่าสุด

1.3.1 การดึงข้อมูลจุดอุบัติเหตุ

หลังจากได้ตำแหน่งของผู้ใช้ผ่านฟังก์ชัน `getLocation` จะมีการเรียกฟังก์ชัน `_getAccidentPoints(double radius)` เพื่อค้นหาจุดอุบัติเหตุภายในรัศมีที่กำหนด ข้อมูลดังกล่าวถูกจัดเก็บในรูปแบบของ `AccidentPoint` ซึ่งประกอบด้วยข้อมูลสำคัญ ได้แก่

1) จำนวนผู้บาดเจ็บ (`injured`) หมายถึง จำนวนผู้ที่ได้รับบาดเจ็บจากอุบัติเหตุในจุดเกิดเหตุที่อยู่ในรัศมีที่กำหนด โดยข้อมูลนี้ช่วยให้ทราบถึงความรุนแรงของอุบัติเหตุในแต่ละพื้นที่ ซึ่งมีผลต่อการประเมินความเสี่ยงและการเตรียมมาตรการป้องกัน
2) จำนวนผู้เสียชีวิต (`dead`) หมายถึง จำนวนผู้ที่เสียชีวิตจากอุบัติเหตุในพื้นที่ที่สนใจ ข้อมูลนี้มีความสำคัญต่อการวิเคราะห์และระบุจุดที่มีอัตราการเสียชีวิตสูง อันจะเป็นข้อมูลพื้นฐานที่ช่วยในการกำหนดแนวทางเพื่อปรับปรุงความปลอดภัยบนถนน

```dart
Future<void> _getAccidentPoints(double radius) async {
  if (currentLocation != null) {
    try {
      List<AccidentPoint> points = await apiService.fetchAccidentPoints(
        currentLocation!.latitude,
        currentLocation!.longitude,
        radius,
      );

      if (mounted) {
        int tinf = 0;
        int tdead = 0;

        for(var i in points){
          tinf += i.injured;
          tdead += i.dead;
        }
```

รูปที่ 12 การเรียกข้อมูลจุดเกิดอุบัติเหตุภาย ในรัศมีที่กำหนดและคำนวณจำนวนผู้บาดเจ็บและเสียชีวิต

ข้อมูลเหล่านี้ถูกสะสมเพื่อนำมาแสดงในแอปพลิเคชันโดยใช้ Provider (`AccidentProvider`) เพื่ออัปเดตค่าอุบัติเหตุในรัศมีของผู้ใช้ตามจำนวนที่ดึงมาได้จาก API นอกจากนี้ มีการตั้งค่าให้แสดงการแจ้งเตือนหากจำนวนจุดอุบัติเหตุในพื้นที่ใกล้เคียงมีมากกว่า 5 จุดขึ้นไป โดยแสดงทั้งข้อความและการสั่นของอุปกรณ์

1.4 การแสดงการแจ้งเตือน

ฟังก์ชัน `_showNotification(int points)` มีการใช้ `FlutterLocalNotificationsPlugin` เพื่อแสดงผลการแจ้งเตือนผ่านช่องทางที่มีการตั้งค่าไว้ นอกจากนี้ยังมีการใช้ `Vibration` สำหรับการสั่นของอุปกรณ์เพื่อดึงดูดความสนใจของผู้ใช้ ข้อมูลที่แจ้งเตือนมีรายละเอียดเกี่ยวกับจำนวนจุดอุบัติเหตุใกล้เคียง โดยมีเงื่อนไขการสั่นที่กำหนดเป็นรูปแบบการสั่นแบบต่อเนื่อง


```dart
    // สั่นอุปกรณ์
    bool? hasVibrator = await Vibration.hasVibrator();
    if (hasVibrator != null && hasVibrator) {
      Vibration.vibrate(pattern: [500, 1000, 500, 2000]); // รูปแบบการสั่น
    }
  }
```

รูปที่ 13 ฟังก์ชันสั่น

1.5 การแสดงผลบนหน้าจอ Notification

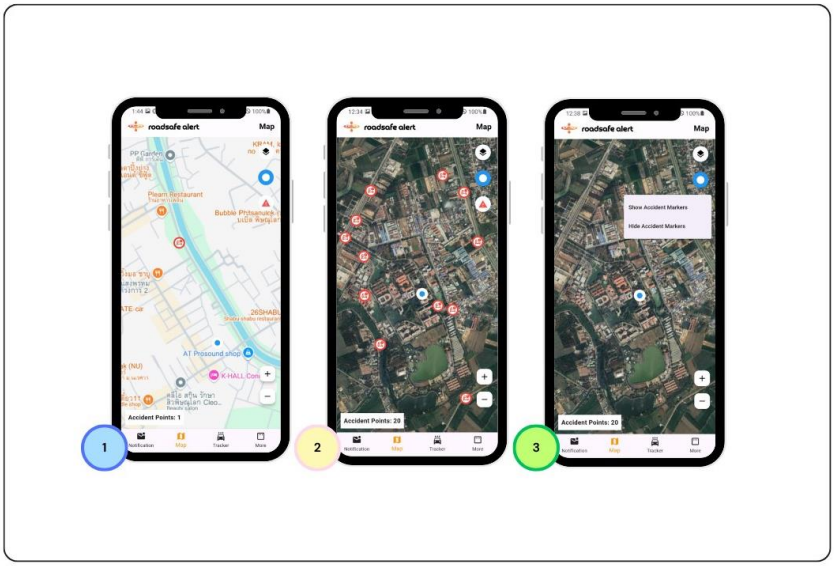
การแสดงผลหน้าจอ Notification ออกแบบโดยใช้ `Scaffold` ซึ่งในส่วนของ `body` ถูกจัดวางให้มีการแสดงผลจำนวนจุดเกิดอุบัติเหตุในกราฟิกวงกลม รวมถึงแถบสีที่เปลี่ยนแปลงตามจำนวนอุบัติเหตุ นอกจากนี้ยังมีการใช้ `Stack` ในการจัดตำแหน่งเพื่อแสดงรายละเอียดของการบาดเจ็บและการเสียชีวิตในบริเวณที่เป็นไปตามรัศมีที่ผู้ใช้อยู่ ณ ขณะนั้น ซึ่งรายละเอียดเพิ่มเติมประกอบด้วย

1.5.1 ตัวเลขแสดงจำนวนจุดเกิดอุบัติเหตุ แสดงในรูปแบบของจำนวนตัวเลขตรงกลางวงกลม

1.5.2 รายละเอียดผู้บาดเจ็บและเสียชีวิต ใช้ `BarChart` แสดงจำนวนผู้บาดเจ็บ (`injured`) และจำนวนผู้เสียชีวิต (`dead`) แยกออกเป็นแท่งกราฟตามลำดับ ทั้งนี้เพื่อช่วยให้ผู้ใช้มองเห็นแนวโน้มอุบัติเหตุในพื้นที่ได้อย่างชัดเจนยิ่งขึ้น

2. การออกแบบหน้า Map

การออกแบบหน้า Map เป็นส่วนสำคัญของแอปพลิเคชันที่ช่วยให้ผู้ใช้สามารถเห็นตำแหน่งของตนเองและข้อมูลจุดอุบัติเหตุบนแผนที่ได้อย่างชัดเจน โดยเน้นความเรียบง่ายใช้งานสะดวก และสามารถปรับการแสดงข้อมูลต่าง ๆ บนแผนที่ได้ ซึ่งมีรายละเอียดการออกแบบดังนี้



รูปที่ 14 การแสดงผลหน้า Map

2.1 ส่วนแสดงแผนที่หลัก

2.1.1 โครงสร้างใช้ `flutter_map` ในการแสดงแผนที่ โดยกำหนดตำแหน่งเริ่มต้นตามตำแหน่งปัจจุบันของผู้ใช้ ซึ่งแผนที่สามารถเลื่อน ซูมเข้าและออกได้

```
children: [
  FlutterMap(
    mapController: _mapController,
    options: MapOptions(
      center: LatLng(currentLocation!.latitude, currentLocation!.longitude),
      zoom: 17.0,  //15ก็ได้
      //interactiveFlags: InteractiveFlag.pinchZoom | InteractiveFlag.doubleTapZoom,
      // ให้สามารถซูมได้ด้วยการ pinch และ double tap แต่จะล็อกการ drag map
      interactiveFlags: InteractiveFlag.pinchZoom | InteractiveFlag.doubleTapZoom | InteractiveFlag.drag,
      // การตั้งค่าการโต้ตอบเพิ่มเติม

    ),  // MapOptions
```

รูปที่ 15 ตั้งค่าให้เริ่มต้นที่ตำแหน่งปัจจุบันของผู้ใช้พร้อมความสามารถในการซูมและเลื่อนแผนที่

2.1.2 ชั้นแผนที่ (Map Layers) ผู้ใช้สามารถเลือกชั้นแผนที่ที่ต้องการแสดง เช่น Google Maps, OpenStreetMap, หรือ Satellite โดยใช้ Popup Menu ให้เลือกได้ง่าย

```
TileLayer getTileLayer() {
  switch (selectedLayerIndex) {
    case 0:
      return TileLayer(
        urlTemplate: 'http://mt0.google.com/vt/lyrs=r&hl=en&x={x}&y={y}&z={z}',
        subdomains: ['mt0', 'mt1', 'mt2', 'mt3'],
        userAgentPackageName: 'com.example.app',
      );
    case 1:
      return TileLayer(
        urlTemplate: "https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
        subdomains: ['a', 'b', 'c'],
        userAgentPackageName: 'com.example.app',
      );
    case 2:
      return TileLayer(
        urlTemplate: 'http://mt0.google.com/vt/lyrs=s&hl=en&x={x}&y={y}&z={z}',
        subdomains: ['mt0', 'mt1', 'mt2', 'mt3'],
        userAgentPackageName: 'com.example.app',
      );
    default:
      return TileLayer(
        urlTemplate: 'http://mt0.google.com/vt/lyrs=r&hl=en&x={x}&y={y}&z={z}',
        subdomains: ['mt0', 'mt1', 'mt2', 'mt3'],
        userAgentPackageName: 'com.example.app',
      );
  }
}
```

รูปที่ 16 ชั้นแผนที่

2.1.3 การออกแบบฟังก์ชันการซูมและการย้ายแผนที่ เมื่อผู้ใช้เคลื่อนที่ แผนที่จะเลื่อนไปตามตำแหน่งของผู้ใช้ และซูมไปยังจุดปัจจุบันโดยอัตโนมัติ ทำให้สามารถติดตามตำแหน่งได้แบบเรียลไทม์

2.2 การแสดงตำแหน่งผู้ใช้

2.2.1 ตัวบ่งชี้ตำแหน่ง (Location Indicator) ตำแหน่งผู้ใช้จะแสดงเป็นจุดสีน้ำเงินขนาดเล็กที่มีวงกลม ซึ่งช่วยให้ผู้ใช้เห็นตำแหน่งของตนเองบนแผนที่ได้ง่าย

```
getTileLayer(), // เลือก TileLayer ตามชั้นที่เลือก
if (showAccidentMarkers) // แสดงจุดอุบัติเหตุถ้า showAccidentMarkers เป็น true
  MarkerLayer(
    markers: accidentMarkers,
  ),  // MarkerLayer
if (radiusSize > 0) // แสดงวงกลมถ้า radiusSize มากกว่า 0
  CircleLayer(
    circles: [
      CircleMarker(
        point: LatLng(currentLocation!.latitude, currentLocation!.longitude),
        radius: radiusSize.toDouble(),  // แปลงเป็นกิโลเมตรถ้าต้องการ (ถ้า radiusSize เป็นเมตร)
        color: Colors.blue.withOpacity(0.0),
        borderStrokeWidth: 0.1, // ความหนาของขอบ
        borderColor: Colors.blueAccent, // สีขอบ
      ),  // CircleMarker
    ],
  ),  // CircleLayer
```

รูปที่ 17 ตำแหน่งผู้ใช้

2.3 การแสดงข้อมูลจุดอุบัติเหตุ (Accident Markers)

2.3.1 จุดอุบัติเหตุ แสดงข้อมูลอุบัติเหตุที่เกิดขึ้นในพื้นที่ใกล้เคียงด้วย icon สีแดง รูปรถชน ซึ่งทำให้เห็นได้ชัดเจนบนแผนที่ และเข้าใจความเสี่ยงของพื้นที่นั้นๆ ได้ง่าย

2.3.2 การคำนวณจุดอุบัติเหตุในรัศมี ระบบคำนวณและแสดงจำนวนอุบัติเหตุภายในรัศมีของผู้ใช้ โดยผลลัพธ์จะแสดงในกล่องข้อมูลที่มุมล่างของหน้าจอ ช่วยให้ผู้ใช้เข้าใจความหนาแน่นของอุบัติเหตุในพื้นที่นั้นๆ

```
        setState(() {
          accidentMarkers = points.map((point) => Marker(
            point: LatLng(point.latitude, point.longitude),
            builder: (context) => Container(
              width: 20.0, // ขนาดของวงกลมครอบ
              height: 20.0, // ขนาดของวงกลมครอบ
              decoration: BoxDecoration(
                color: Colors.white, // สีของวงกลมครอบ
                shape: BoxShape.circle, // รูปแบบวงกลม
                border: Border.all( // เพิ่มขอบวงกลมสีแดง
                  color: Colors.red, // สีของขอบ
                  width: 4.0, // ความหนาของขอบ
                ),  // Border.all
              ),  // BoxDecoration
              child: Center(
                child: Icon(
                  Icons.car_crash_outlined, // ไอคอนรถชน
                  color: Colors.red, // สีของไอคอน
                  size: 18.0, // ขนาดของไอคอน
                ),  // Icon
              ),  // Center
            ),  // Container
          )).toList();  // Marker
        });
      } catch (e) {
        print('Error loading accident points: $e');
      }
    }
}
```

รูปที่ 18 ข้อมูลอุบัติเหตุที่เกิดขึ้นในพื้นที่ใกล้เคียงด้วย icon สีแดง รูปรถชน

2.4 ตัวเลือกการปรับแต่งรัศมี (Radius Selection)

2.4.1 การเลือกขนาดรัศมี ผู้ใช้สามารถเลือกขนาดรัศมีที่ต้องการจากเมนูปุ่ม Popup เช่น 1000 เมตร หรือ 1500 เมตร ซึ่งจะมีผลต่อจำนวนจุดอุบัติเหตุที่แสดง และขนาดของวงกลมรอบตำแหน่งผู้ใช้

2.4.2 เมนู Popup ที่เป็นปุ่มวงกลม เมนูนี้ออกแบบให้เป็นปุ่มวงกลม เพื่อให้การเลือกขนาดรัศมีทำได้ง่ายและรวดเร็ว อีกทั้งยังสอดคล้องกับการเลือกชั้นแผนที่ ทำให้ใช้งานได้สะดวก

```dart
SizedBox(height: 10),
PopupMenuButton<double>(
  icon: CircleAvatar(
    backgroundColor: Colors.blue,
    child: Icon(Icons.circle, color: Colors.white),
  ),  // CircleAvatar
  onSelected: (double radius) {
    _showRadiusSelectionDialog(radius);
  },
  itemBuilder: (BuildContext context) => [
    PopupMenuItem<double>(
      value: 1000.0,
      child: Text('1000 meters'),
    ),  // PopupMenuItem
    PopupMenuItem<double>(
      value: 1500.0,
      child: Text('1500meters'),
    ),  // PopupMenuItem
    PopupMenuItem<double>(
      value: 2000.0,
      child: Text('2000 meters'),
    ),  // PopupMenuItem
  ],
),  // PopupMenuButton
```

รูปที่ 19 ปรับแต่งรัศมี (Radius Selection)

2.5 ส่วนการนำทางและควบคุม

2.5.1 ปุ่มเปลี่ยนชั้นแผนที่ ตั้งอยู่ที่มุมขวาบนของแผนที่ เพื่อให้ผู้ใช้สามารถเลือกชั้นแผนที่ได้อย่างรวดเร็ว

2.5.2 ปุ่มแสดงจุดอุบัติเหตุ ปุ่มเพื่อเปิด/ปิดการแสดงจุดอุบัติเหตุได้ตามต้องการ ช่วยให้หน้าจอไม่หนาแน่นและข้อมูลไม่ทับซ้อนกันมากเกินไป

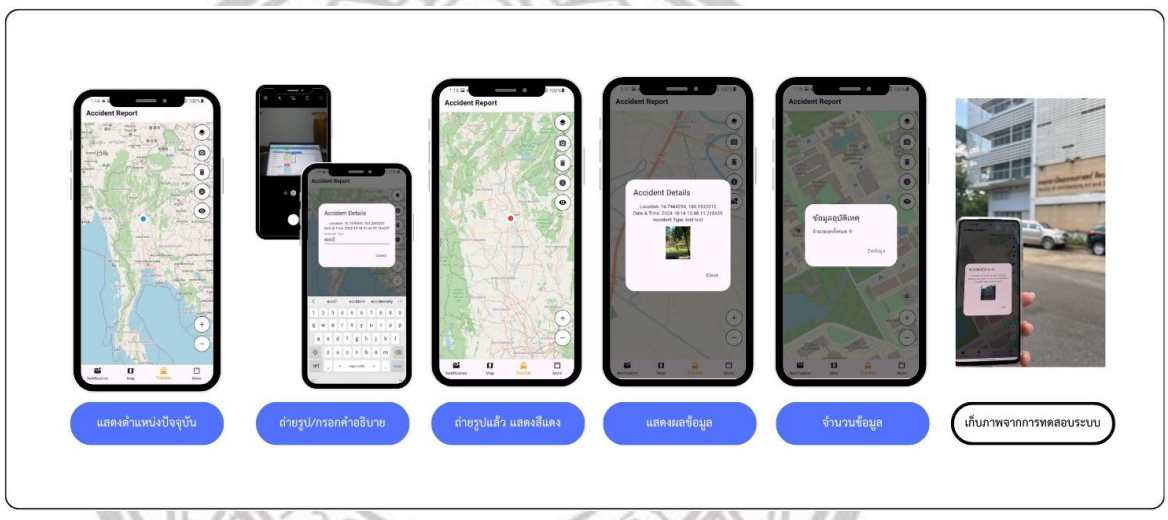
```dart
PopupMenuButton<bool>(
  icon: CircleAvatar(
    backgroundColor: Colors.white,
    child: Icon(Icons.warning, color: Colors.redAccent),
  ),  // CircleAvatar
  onSelected: (bool show) {
    setState(() {
      showAccidentMarkers = show;
      if (show) {
        _loadAccidents(); // โหลดจุดอุบัติเหตุเมื่อเลือกแสดง
      } else {
        accidentMarkers = []; // ซ่อนจุดอุบัติเหตุ
      }
    });
  },
  itemBuilder: (BuildContext context) => [
    PopupMenuItem<bool>(...),  // PopupMenuItem
    PopupMenuItem<bool>(
      value: false,
      child: Text('Hide Accident Markers'),
    ),  // PopupMenuItem
```

```dart
child: Column(
  children: [
    PopupMenuButton<int>(
      icon: CircleAvatar(
        backgroundColor: Colors.white,
        child: Icon(Icons.layers, color: Colors.black),
      ),  // CircleAvatar
      onSelected: (int result) {
        setState(() {
          selectedLayerIndex = result;
        });
      },
      itemBuilder: (BuildContext context) => [
        PopupMenuItem<int>(
          value: 0,
          child: Text('Google Maps'),
        ),  // PopupMenuItem
        PopupMenuItem<int>(
          value: 1,
          child: Text('OpenStreetMap'),
        ),  // PopupMenuItem
        PopupMenuItem<int>(
          value: 2,
          child: Text('Satellite'),
        ),  // PopupMenuItem
      ],
    ),  // PopupMenuButton
```

รูปที่ 20 เปลี่ยนชั้นแผนที่ และปุ่มเพื่อเปิด/ปิดการแสดงจุดอุบัติเหตุ

3. การออกแบบหน้า Tracker

Tracker หรือ PhotoMap ช่วยให้ผู้ใช้เห็นตำแหน่งอุบัติเหตุและตำแหน่งปัจจุบันแบบเรียลไทม์ เพิ่มความปลอดภัยในการขับขี่ในพื้นที่เสี่ยง โดยผู้ใช้สามารถถ่ายภาพและรายงานข้อมูลอุบัติเหตุได้ทันที ช่วยสนับสนุนการเก็บข้อมูลเพื่อวิเคราะห์และป้องกันอุบัติเหตุ พร้อมทั้งช่วยในการวิเคราะห์พื้นที่ `PhotoMap` ใน Flutter นี้ มีองค์ประกอบสำคัญ อธิบายในงานวิจัยได้ ดังนี้



รูปที่ 21 การแสดงผลหน้า Tracker

3.1 การแสดงตำแหน่งปัจจุบันของผู้ใช้ (Current Location Display)

โค้ดมีการใช้ตัวแปร `_currentLocation` ในการแสดงตำแหน่งปัจจุบันของผู้ใช้ โดยใช้ไอคอนที่ระบุไว้ใน `MarkerLayer` ช่วยให้ผู้ใช้สามารถเห็นตำแหน่งของตนบนแผนที่ได้ทันที ซึ่งเป็นองค์ประกอบสำคัญสำหรับแอปที่เน้นการนำทางหรือให้ข้อมูลตามสถานที่จริง


```
@override
void initState() {
  super.initState();
  _mapController = MapController();
  _getCurrentLocation(); // เรียกฟังก์ชันเพื่อรับตำแหน่งปัจจุบัน
  _loadAccidentData(); // เรียกฟังก์ชันเพื่อโหลดข้อมูลอุบัติเหตุ 15/10 _loadAccidentData
}

//15/10_loadAccidentData
Future<void> _loadAccidentData() async {
  List<AccidentRecord> accidentRecords = await _getAccidentRecords(); // ดึงข้อมูลอุบัติเหตุ
  _showAllAccidentPoints(accidentRecords); // แสดงอุบัติเหตุบนแผนที่
}


Future<void> _getCurrentLocation() async {
  LocationPermission permission = await Geolocator.checkPermission();
  if (permission == LocationPermission.denied) {
    permission = await Geolocator.requestPermission();
  }

  if (permission == LocationPermission.whileInUse || permission == LocationPermission.always) {
    Position position = await Geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.high);
    setState(() {
      _currentLocation = LatLng(position.latitude, position.longitude);
      _mapController.move(_currentLocation!, _zoomLevel); // ย้ายแผนที่ไปยังตำแหน่งปัจจุบันทันที
    });
```

รูปที่ 22 การแสดงตำแหน่งปัจจุบันของผู้ใช้

3.2 การจัดการมาร์กเกอร์อุบัติเหตุ (Accident Marker Management)

3.2.1 โค้ดใช้ `_accidentRecords` สำหรับเก็บข้อมูลจุดอุบัติเหตุที่ดึงมาจากฐานข้อมูล ซึ่งใช้เพื่อแสดงมาร์กเกอร์บนแผนที่ Flutter โดยสามารถควบคุมการแสดงผลมาร์กเกอร์ผ่านตัวแปร `showAccidentMarkers`

3.2.2 การเลือกวิธีการจัดการข้อมูลใน `_accidentRecords` หรือ `_markers` แสดงให้เห็นถึง ทำให้ระบบสามารถจัดการข้อมูลได้

```
//15/10
void _showAllAccidentPoints(List<AccidentRecord> accidentRecords) {
  for (var record in accidentRecords) {
    _addMarker(record);
  }
}
//15/10
void _addMarker(AccidentRecord record) {
  setState(() {
    _markers.add(
      Marker(
        point: LatLng(record.location.latitude, record.location.longitude),
        builder: (context) => GestureDetector(
          onTap: () {
            // แสดง Popup เมื่อคลิกที่ Marker
            _showAccidentDetails(record);
          },
          child: Icon(Icons.location_pin, color: Colors.red),
        ),  // GestureDetector
      ),  // Marker
    );
  });
}
```

รูปที่ 23 เก็บข้อมูลจุดอุบัติเหตุที่ดึงมาจากฐานข้อมูล

3.3 การควบคุมการแสดงผลด้วยเงื่อนไข (Conditional Rendering)

โค้ดใช้เงื่อนไข `if` สำหรับการแสดงผล เช่น การใช้ `if (_currentLocation != null)` และ `if (showAccidentMarkers)` เพื่อควบคุมการแสดงมาร์กเกอร์ การใช้เงื่อนไขนี้ช่วยให้โค้ดมีความยืดหยุ่น ลดความซับซ้อน และสามารถนำมาใช้เพื่อให้ข้อมูลแสดงเฉพาะเมื่อจำเป็น

3.4 ฟังก์ชันสำหรับโหลดข้อมูลอุบัติเหตุ (Accident Data Loading Functionality)

ฟังก์ชัน `_loadAccidentData` เป็นการดึงข้อมูลอุบัติเหตุจากฐานข้อมูล API โดยตรง ซึ่งมีบทบาทสำคัญในการแสดงข้อมูลอุบัติเหตุในแผนที่ ทำให้ระบบสามารถทำงานได้แบบเรียลไทม์ เพิ่มความน่าสนใจและความถูกต้องของข้อมูลให้แก่ผู้ใช้

```dart
@override
void initState() {
  super.initState();
  _mapController = MapController();
  _getCurrentLocation(); // เรียกฟังก์ชันเพื่อรับตำแหน่งปัจจุบัน
  _loadAccidentData(); // เรียกฟังก์ชันเพื่อโหลดข้อมูลอุบัติเหตุ 15/10 _loadAccidentData
}

//15/10_loadAccidentData
Future<void> _loadAccidentData() async {
  List<AccidentRecord> accidentRecords = await _getAccidentRecords(); // ดึงข้อมูลอุบัติเหตุ
  _showAllAccidentPoints(accidentRecords); // แสดงอุบัติเหตุบนแผนที่
}
```

รูปที่ 24 สำหรับโหลดข้อมูลอุบัติเหตุ

3.5 ฟังก์ชัน `_uploadImage`

ในโค้ดนี้มีหน้าที่ในการอัปโหลดภาพไปยังเซิร์ฟเวอร์ผ่าน APIโดยเริ่มจากการตรวจสอบว่ามีภาพถูกเลือกหรือไม่ หากไม่มีภาพ ฟังก์ชันจะหยุดทำงานทันที หลังจากนั้นจะสร้าง URI สำหรับ API ที่จะใช้ในการอัปโหลดฟังก์ชันจะสร้างคำขอแบบ POST เพื่อส่งข้อมูลไปยังเซิร์ฟเวอร์ และแนบไฟล์ภาพที่ถูกเลือกเข้ากับคำขอ นอกจากนี้ยังมีการแนบข้อมูลสำคัญ เช่น ละติจูด ลองจิจูด หมายเลขบัญชี และเวลาเกิดเหตุไปกับคำขอด้วย

```dart
// Method to upload the image to the API
Future<void> _uploadImage(lat,  lng, acct,  dtime) async {
  if (_image == null) return;

  // Prepare the request
  final uri = Uri.parse('http://119.59.99.46:3000/api/imgupload/upload');
  var request = http.MultipartRequest('POST', uri);

  // Attach the image file from the XFile path
  var imageFile = await http.MultipartFile.fromPath('image', _image!.path);
  request.files.add(imageFile);
  request.fields['lat'] = lat.toString();
  request.fields['lng'] = lng.toString();
  request.fields['acct'] = acct;
  request.fields['dtime'] = dtime.toString();
```

รูปที่ 25 อัปโหลดภาพไปยังเซิร์ฟเวอร์ผ่าน API

3.6 การจัดการข้อผิดพลาดในการเรียก API (API Error Handling)

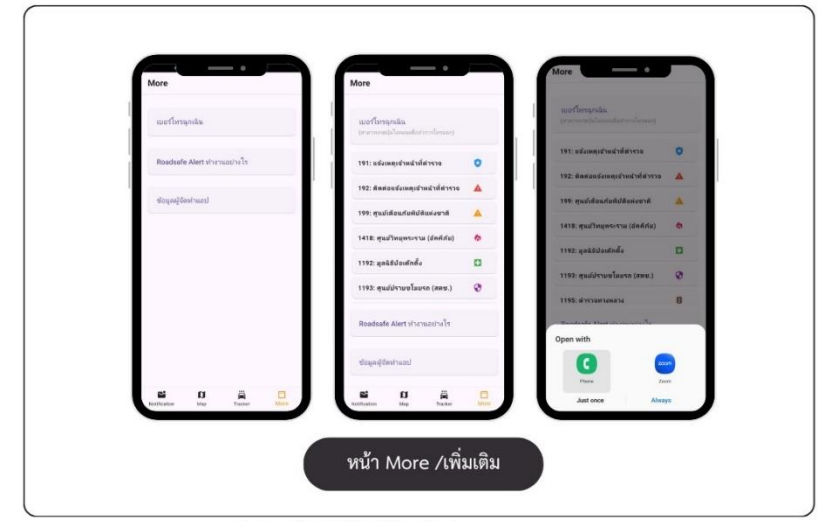
การจัดการข้อผิดพลาดในฟังก์ชัน `_uploadImage` และ `_getAccidentRecords` แสดงถึงการเตรียมพร้อมให้ระบบมีความเสถียร ลดข้อผิดพลาดที่อาจเกิดจากปัญหาเครือข่ายหรือข้อมูลที่ไม่ถูกต้อง ซึ่งมีความสำคัญสำหรับแอปที่ต้องใช้ข้อมูลจากภายนอกในการทำงาน

```dart
// ฟังก์ชันดึงข้อมูลอุบัติเหตุจากฐานข้อมูล
// ฟังก์ชันดึงข้อมูลอุบัติเหตุจากฐานข้อมูล
Future<List<AccidentRecord>> _getAccidentRecords() async {
  final uri = Uri.parse('http://119.59.99.46:3000/api/imgupload/accidentrecords'); // เปลี่ยน URL ให้ตรงกับ endpoint ใหม่

  try {
    final response = await http.get(uri); // เรียกข้อมูลจาก API

    if (response.statusCode == 200) {
      // หากสถานะโค้ดเป็น 200 แปลว่าข้อมูลถูกส่งกลับมาอย่างถูกต้อง
      final List<dynamic> jsonResponse = json.decode(response.body);
      print(jsonResponse);
      // ตรวจสอบว่า jsonResponse ไม่ใช่ค่าว่างหรือ null
      if (jsonResponse != null && jsonResponse.isNotEmpty) {
        return jsonResponse.map((record) => AccidentRecord.fromJson(record)).toList();
      } else {
        throw Exception('ไม่มีข้อมูลอุบัติเหตุ');
      }
    } else {
      throw Exception('ไม่สามารถโหลดข้อมูลอุบัติเหตุได้ สถานะโค้ด: ${response.statusCode}');
    }
  } catch (error) {
    print('เกิดข้อผิดพลาดในการเรียกข้อมูล: $error');
    throw Exception('เกิดข้อผิดพลาดในการเรียกข้อมูล');
  }
}
```

รูปที่ 26 จัดการข้อผิดพลาดในการเรียก API



รูปที่ 27 การแสดงผลหน้า More

4. การออกแบบหน้า More

มีความสำคัญในการจัดการข้อมูลสำคัญ เช่น รายชื่อผู้ติดต่อฉุกเฉินและประวัติการใช้งาน ช่วยให้ผู้ใช้เข้าถึงข้อมูลได้อย่างรวดเร็วและสะดวกสบาย ฟีเจอร์การเปิดหรือปิดการแสดงข้อมูลเพิ่มความยืดหยุ่นในการใช้งาน และการออกแบบที่เป็นมิตรกับผู้ใช้ช่วยให้การนำทางมีประสิทธิภาพยิ่งขึ้นในเวลาที่ต้องการข้อมูลสำคัญ

4.1 การจัดการสถานะและการแสดงผล

หน้า More ถูกออกแบบโดยใช้ StatefulWidget เพื่อให้สามารถจัดการสถานะการแสดงผลของข้อมูลต่าง ๆ ได้อย่างมีประสิทธิภาพ โดยใช้ตัวแปร Boolean เช่น isLoading, showContacts, showHistory, และ showDeveloperInfo เพื่อควบคุมการแสดงผลของรายการต่าง ๆ ภายในหน้าจอ ซึ่งช่วยให้ผู้ใช้สามารถเข้าถึงข้อมูลได้อย่างรวดเร็ว

4.2 การแสดงรายการผู้ติดต่อฉุกเฉิน

หน้านี้มีการแสดงรายชื่อผู้ติดต่อฉุกเฉินที่ประกอบด้วยชื่อหน่วยงานและหมายเลขโทรศัพท์ โดยมีฟังก์ชัน _makePhoneCall ที่รองรับการโทรออกเมื่อผู้ใช้กดที่ไอคอนของผู้ติดต่อแต่ละคน การออกแบบนี้ช่วยให้ผู้ใช้สามารถติดต่อหน่วยงานที่เกี่ยวข้องในกรณีฉุกเฉิน

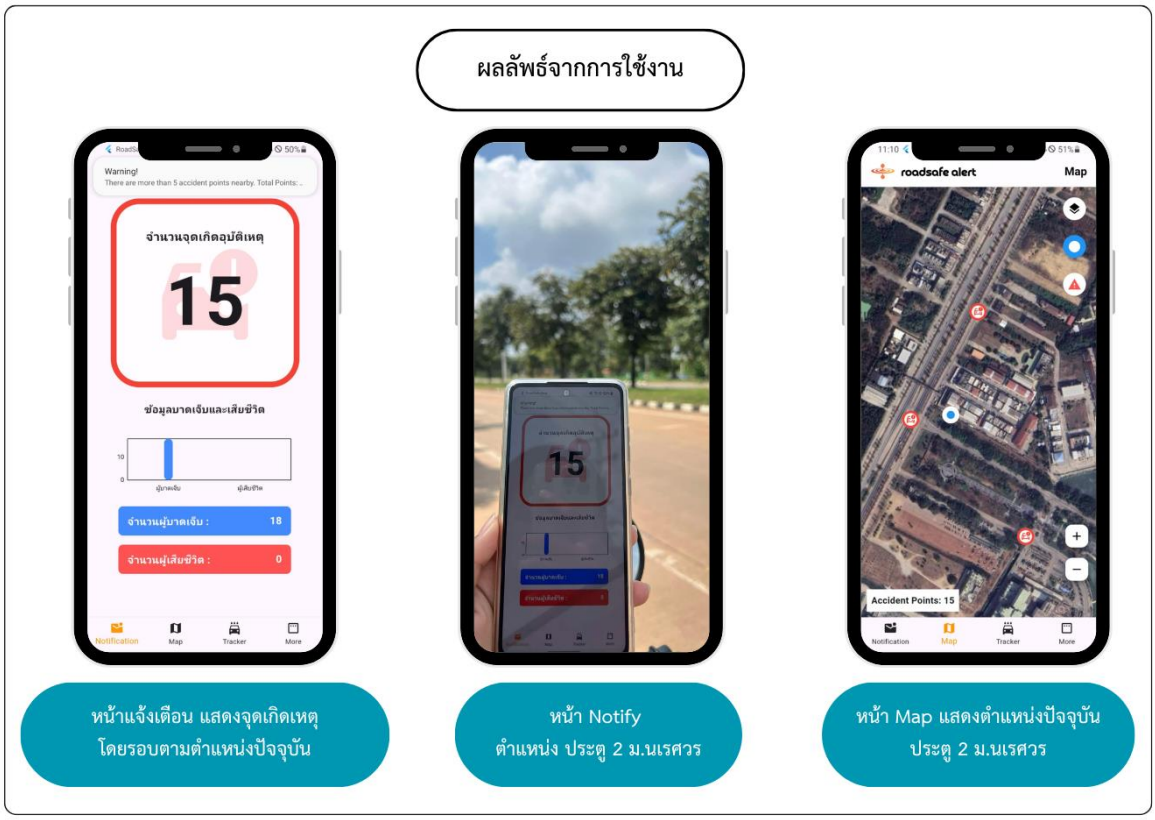
```dart
// ฟังก์ชันเพื่อทำการโทร
Future<void> _makePhoneCall(String phoneNumber) async {
  // ขอ permission CALL_PHONE
  if (await Permission.phone.request().isGranted) {
    final Uri launchUri = Uri(
      scheme: 'tel',
      path: phoneNumber,
    );
    try {
      // Launch the phone dialer
      await launchUrl(launchUri);
    } catch (e) {
      // แสดงข้อผิดพลาดที่เกิดขึ้นเมื่อพยายามโทร
      print('Error launching phone call: $e');
    }
  } else {
    // แสดงข้อความเมื่อ permission ถูกปฏิเสธ
    print('Permission denied');
    // Optionally, show a dialog or a Snackbar to inform the user
  }
}
```

รูปที่ 28 การแสดงรายชื่อผู้ติดต่อฉุกเฉิน

4.3 ปุ่มเปิด-ปิดการแสดงข้อมูล

ระบบมีปุ่มสำหรับเปิดและปิดการแสดงข้อมูลต่าง ๆ ได้แก่ รายชื่อผู้ติดต่อฉุกเฉิน, ประวัติการทำงานของแอป, และข้อมูลนักพัฒนา ซึ่งช่วยให้ผู้ใช้สามารถควบคุมการแสดงผลได้ตามความต้องการ ทำให้การใช้งานแอปพลิเคชันมีความยืดหยุ่นและตอบสนองต่อความต้องการของผู้ใช้ได้ดียิ่งขึ้น

ผลลัพธ์



รูปที่ 29 ผลลัพธ์การใช้งานแสดงผลการแจ้งเตือนตามจำนวนจุดเกิดเหตุและตำแหน่งปัจจุบัน

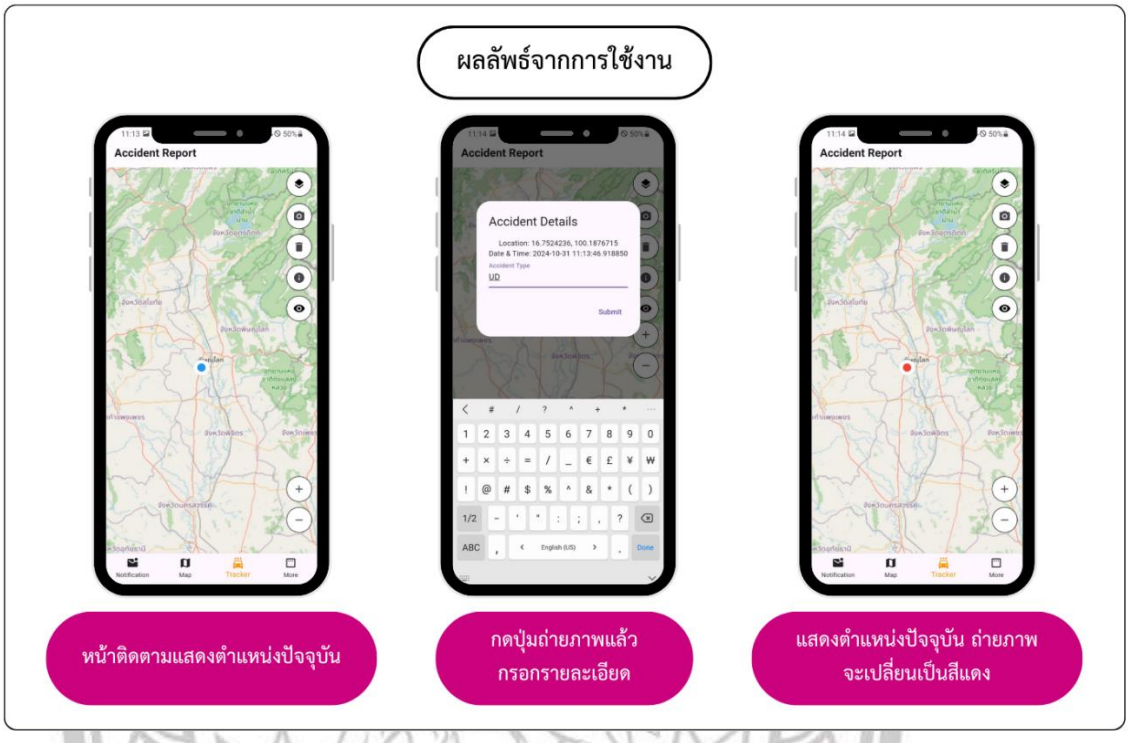
ผลลัพธ์จากการใช้งานแอป เมื่อผู้ใช้เดินทางถึงตำแหน่ง ประตู 2 มหาวิทยาลัยนเรศวร ขณะเปิดแอป RoadSafe Alert ระหว่างขับรถ แอปจะทำการแจ้งเตือนด้วยการสั่นโดยอัตโนมัติ เมื่อพบว่ามีจุดเกิดอุบัติเหตุในบริเวณที่มีการกำหนดเงื่อนไขเป็นจุดเสี่ยง และเมื่อจำนวนจุดเกิดเหตุเกิน 5 จุด แอปจะทำการแจ้งเตือนทันที เพื่อให้ผู้ใช้ระมัดระวังขณะขับขี่

ในหน้าจอแจ้งเตือน แอปจะแสดงข้อมูลของจุดเกิดอุบัติเหตุทั้งหมดในรูปแบบ ListView โดยประกอบด้วยตำแหน่งการเกิดเหตุ, วันที่เกิดเหตุ, จำนวนผู้บาดเจ็บ และจำนวนผู้เสียชีวิต สำหรับตำแหน่ง ประตู 2 ระบบจะแจ้งว่ามีจำนวนจุดเกิดอุบัติเหตุทั้งหมด 15 จุดในบริเวณนี้ เมื่อต้องการดูรายละเอียดเพิ่มเติม สามารถเปิดดูในหน้าแผนที่ (Map) โดยกดปุ่มไอคอนการแจ้งเตือนสีแดง เมื่อเปิดใช้งานแล้ว ระบบจะแสดงตำแหน่งปัจจุบันของผู้ใช้ด้วยไอคอนรถ พร้อมกับแสดงจุดเกิดอุบัติเหตุ ที่เกิดขึ้นในบริเวณนั้นอย่างชัดเจนผ่าน Markers โดยผู้ใช้สามารถเห็นได้ว่าแต่ละจุดเกิดขึ้น

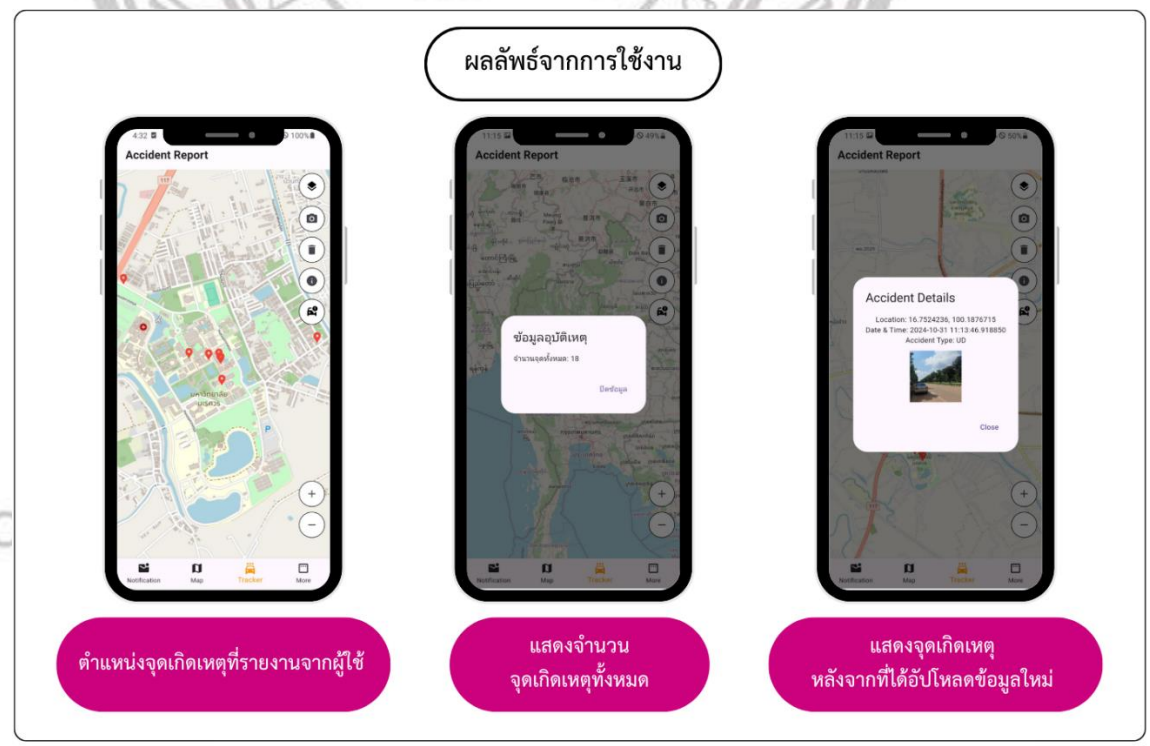
ในตำแหน่งใดบ้าง และหน้าแผนที่ (Map) ของแอป RoadSafe Alert ถูกออกแบบมาเพื่อช่วยเพิ่มความปลอดภัย โดยผู้ใช้สามารถเลือกชั้นข้อมูลต่างๆ ได้ เช่น Google Maps, OpenStreetMap หรือ Satellite ซึ่งช่วยให้ผู้ใช้สามารถเลือกมุมมองแผนที่ที่ต้องการเพื่อความเหมาะสมและความชัดเจนมากยิ่งขึ้น ฟีเจอร์นี้ช่วยให้ผู้ใช้สามารถติดตามสถานการณ์และเข้าใจความเสี่ยงในพื้นที่ได้อย่างรวดเร็ว

รูปที่ 30 ผลลัพธ์การใช้งานภาพถ่ายแสดงผลตำแหน่งปัจจุบันที่ใกล้จุดเกิดเหตุ

นอกจากนี้ ยังสามารถเลือกระยะในการดูข้อมูลอุบัติเหตุได้ตามต้องการ โดยจำนวนจุดอุบัติเหตุภายในระยะที่เลือกจะแสดงในกรอบสี่เหลี่ยมที่ด้านล่างของหน้าจอ ฟังก์ชันเหล่านี้ทำให้ผู้ใช้สามารถตัดสินใจในการเดินทางได้อย่างปลอดภัย โดยมีข้อมูลสนับสนุน ในการประเมินความเสี่ยงจากอุบัติเหตุได้ดียิ่งขึ้น



รูปที่ 31 ผลลัพธ์การใช้งานแสดงผลการรายงานสถานการณ์จากการถ่ายภาพในตำแหน่งปัจจุบัน



รูปที่ 32 ผลลัพธ์การใช้งานแสดงผลการรายงานสถานการณ์ตำแหน่งปัจจุบัน

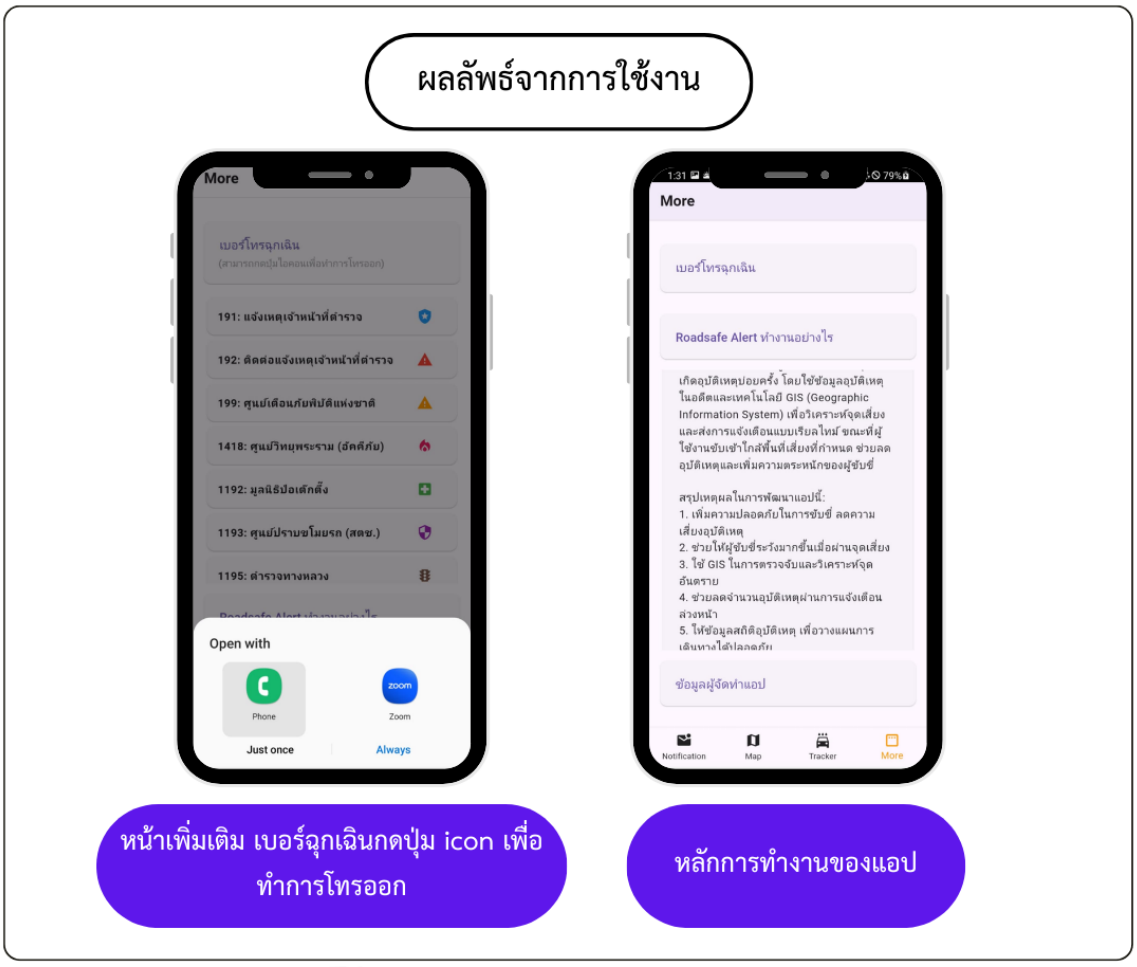
รูปที่ 33 ผลลัพธ์การใช้งานภาพถ่ายแสดงผลการรายงานตำแหน่งปัจจุบันที่ใกล้จุดเกิดเหตุ

ในหน้า ติดตาม (Tracker) ของแอป RoadSafe Alert ผู้ใช้สามารถทดลองใช้งานได้โดยสมมุติการถ่ายภาพจากตำแหน่ง ประตู 2 มหาวิทยาลัยนเรศวร เพื่อทดสอบระบบการรายงานจุดเกิดอุบัติเหตุ ณ ตำแหน่งปัจจุบัน เมื่อเปิดหน้า ติดตาม แอปจะแสดงตำแหน่งปัจจุบัน ของผู้ใช้เป็นปุ่มสีฟ้า เพื่อให้ระบุได้อย่างชัดเจนว่าผู้ใช้อยู่จุดใด

เมื่อต้องการรายงานอุบัติเหตุ ผู้ใช้สามารถกดที่ไอคอนถ่ายภาพ (รูปกล้อง) เพื่อถ่ายภาพเหตุการณ์ เช่น รถชน หรือจุดที่เกิดอุบัติเหตุ เมื่อถ่ายภาพแล้ว ปุ่มสีฟ้าซึ่งแสดงตำแหน่งปัจจุบันจะเปลี่ยนเป็นสีแดงทันที เพื่อบ่งบอกว่ามีการถ่ายภาพเพื่อรายงานอุบัติเหตุในพื้นที่นั้นแล้ว หลังจากถ่ายภาพเสร็จ ผู้ใช้สามารถกรอกรายละเอียดเพิ่มเติมเกี่ยวกับเหตุการณ์อุบัติเหตุได้ เช่น ลักษณะของอุบัติเหตุ ความเสียหาย หรือข้อคิดเห็น จากนั้นกด Submit เพื่ออัปเดตข้อมูลเข้าสู่ระบบ หากต้องการดูรายงานสถานการณ์อุบัติเหตุทั้งหมดในพื้นที่ ผู้ใช้สามารถกดที่ไอคอนรูปตาด้านขวามือของหน้าจอ ซึ่งจะแสดงรายการจุดเกิดอุบัติเหตุทั้งหมดในพื้นที่ โดยแสดงเป็นพิน (Pin) ตำแหน่งแต่ละจุดพร้อมข้อมูลเพิ่มเติม ทั้งภาพถ่ายและรายละเอียดที่ผู้ใช้รายงานเข้ามา ทำให้ผู้ใช้สามารถติดตามสถานการณ์

โดยรอบได้ชัดเจน รวมถึงสามารถตรวจสอบข้อมูลจากผู้ใช้คนอื่นๆ ได้หน้าติดตาม (Tracker) ช่วยให้ผู้ใช้งานสามารถอัปโหลดและดูรูปภาพที่เกี่ยวข้องกับจุดเสี่ยงหรืออุบัติเหตุได้ โดยสามารถเลือก และอัปโหลดรูปภาพที่ช่วยเพิ่มข้อมูลและความเข้าใจเกี่ยวกับเหตุการณ์นั้น ๆ รูปภาพที่อัปโหลดจะแสดงบนหน้าจอเพื่อให้ผู้ใช้งานดูรายละเอียดเกี่ยวกับจุดเสี่ยงและอุบัติเหตุได้อย่างชัดเจน ยังช่วยในการจัดการข้อมูล เช่น วันที่ เวลา และสถานที่เกิดเหตุ

รูปภาพอาจถูกใช้ในการรายงานเหตุการณ์ให้กับหน่วยงานที่เกี่ยวข้อง การเชื่อมต่อกับฟังก์ชันอื่น ๆ ในแอป เช่น การแจ้งเตือนและการแสดงแผนที่ ช่วยให้การใช้งานเสริมสร้างความปลอดภัยในการขับขี่และให้ข้อมูลที่เป็นประโยชน์เกี่ยวกับสถานการณ์อุบัติเหตุบนท้องถนน



รูปที่ 34 หน้าข้อมูลเพิ่มเติมการเข้าถึงข้อมูลฉุกเฉินและการใช้งานของแอปพลิเคชัน

หน้าข้อมูลเพิ่มเติม จากรูปที่ 34 สำหรับการใช้งานแล้วของแอปพลิเคชัน RoadSafe Alert ในหน้านี้ถูกออกแบบมาเพื่อให้ผู้ใช้เข้าถึงข้อมูลฉุกเฉินได้อย่างรวดเร็วและสะดวกที่สุด โดยสามารถกด

ไอคอนโทรฉุกเฉิน ซึ่งจะลิงก์ไปยังหน้าการโทรออกได้ทันที ช่วยให้สามารถติดต่อหน่วยงานที่จำเป็นในสถานการณ์ฉุกเฉินได้อย่างรวดเร็ว นอกจากนี้ หน้านี้ยังมีรายละเอียดเบื้องต้นเกี่ยวกับหลักการใช้งานแอปพลิเคชัน รวมถึงข้อมูลผู้พัฒนาแอป เพื่อช่วยให้เข้าใจการทำงานและประโยชน์ของแอปได้ดี

การจัดการสิทธิ์การเข้าถึงและการแสดงข้อมูลในหน้านี้ ถูกออกแบบมาเพื่อให้ใช้งานได้ง่ายและปลอดภัยยิ่งขึ้นในทุกสถานการณ์ฉุกเฉิน ช่วยให้ผู้ใช้มีความมั่นใจในการใช้งาน RoadSafe Alert ในการป้องกันและเตรียมพร้อมสำหรับอุบัติเหตุที่อาจเกิดขึ้นบนท้องถนน

อภิปรายผล

จากการติดตั้งและทดลองใช้แอป RoadSafe Alert พบว่ามีศักยภาพในการช่วยผู้ขับขี่ให้รับรู้จุดเสี่ยงบนท้องถนนและเพิ่มความปลอดภัยขณะขับขี่ได้จริง ฟีเจอร์ที่ใช้งานง่าย เช่น ระบบการแจ้งเตือนที่แสดงจำนวนจุดเสี่ยงเมื่อผู้ใช้เข้าใกล้จุดที่มีประวัติการเกิดอุบัติเหตุสูง ทำให้ผู้ใช้สามารถทราบข้อมูลทันทีเมื่อมีจุดเสี่ยงมากกว่า 5 จุด นอกจากนี้ ผู้ใช้สามารถเข้าไปดูจำนวนจุดการเกิดอุบัติเหตุย้อนหลังได้ในหน้าแผนที่ พร้อมทั้งเลือกกำหนดรัศมีการแจ้งเตือนตามความต้องการ เพื่อเตรียมตัวล่วงหน้าในขณะขับขี่ ฟังก์ชันการใช้งาน Tracker ช่วยให้ผู้ใช้สามารถอัปเดตข้อมูลอุบัติเหตุในช่วงเวลาปัจจุบัน โดยสามารถบันทึกตำแหน่งที่ตั้ง เวลา วันที่ บอกสาเหตุ และถ่ายภาพ เพื่อแสดงผลบนแผนที่ ในส่วนของหน้า More ยังช่วยให้ผู้ใช้งานสามารถโทรออกได้อย่างสะดวก การแจ้งเตือนแบบ Push Notification ทำให้ผู้ใช้ได้รับข้อมูลทันทีเมื่อเข้าใกล้จุดเสี่ยง โดยรวมแล้ว แอป RoadSafe Alert มีการพัฒนาที่เหมาะสมกับสถานการณ์ใช้งานจริง และฟีเจอร์ที่หลากหลายสามารถเพิ่มความปลอดภัยให้กับผู้ขับขี่ได้อย่างมีประสิทธิภาพในสภาพแวดล้อมที่ต้องการความปลอดภัยได้

ข้อเสนอแนะ

1. พัฒนาแอปในระยะยาว ควรมีการวางแผนพัฒนาแอปในระยะยาว เช่น การเพิ่มฟีเจอร์ใหม่ๆ การปรับปรุงประสิทธิภาพของแอป รวมถึงการติดตามและวิเคราะห์ความคิดเห็นของผู้ใช้เพื่อการพัฒนาอย่างต่อเนื่อง

2. เสริมการวิเคราะห์ข้อมูล การนำเสนอข้อมูลเชิงสถิติ เช่น แนวโน้มการเกิดอุบัติเหตุในแต่ละพื้นที่หรือช่วงเวลาสามารถช่วยให้ผู้ใช้ประเมินความเสี่ยงในการเดินทางได้ดีขึ้น

บรรณานุกรม

Angsirikul, S., Kaewthawee, N., Ploensil, P., Netsawang, J., Phurijaruyangkun, S., & Kasemsawasdi, S. (2023). ***Development of notification system for traffic accidents in Thailand***. Journal of Intelligent Informatics and Smart Technology, 9(October), 1-10. August 13, 2024. Available from https://ph05.tcithaijo.org/index.php/JIIST/article/view/141

Deacon, J. A., Zegeer, C. V., & Deen, R. C. (1975). ***Identification of hazardousrural highway locations. Transportation Research Record,*** 543(543), 16-33. September 4, 2024.

Elvik, R. (2007). ***State-of-the-Art approaches to road accident black spot management and safety analysis of road networks.*** Institute of Transport Economics. Oslo. September 25, 2024.

Ignaco, M. A. E. (2021). Mobile application for incident reporting. ***Journal of Information and Visualization***, August 13, 2024. Available from https://joiv.org/index.php/joiv/article/view/741

Injury Data Collaboration Center Division of Injury Prevention. (2022). ***Situation of fatal road traffic accidents in Thailand.*** October 10, 2024, Available from https://dip.ddc.moph.go.th/new/%E0%B8%9A%E0%B8 (In Thai).

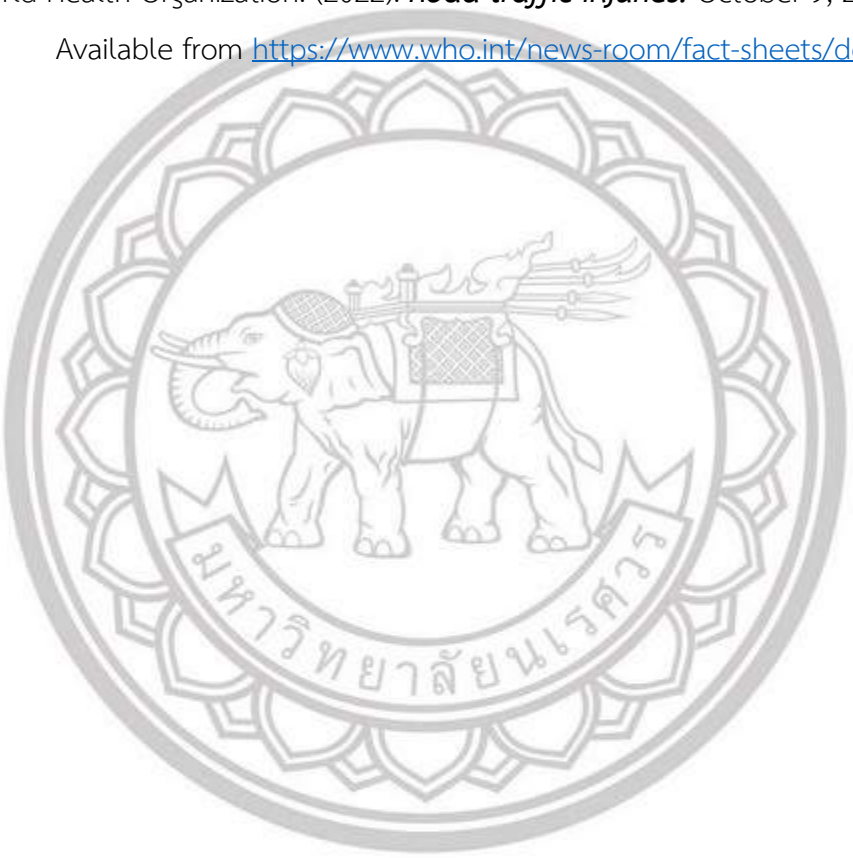
Liñan Espinoza, E., Echevarria Carpio, E. J., & Andrade-Arenas, L. (2021). ***Immediate notification of traffic accidents through a mobile application***. Preprints. August 13, 2024. Available from https://doi.org/10.20944/preprints202112.0490.v1

Maulid, H., Nurhidayat, W., & Priyono, S. J. (2020). ***SafeDri: A mobile-based.*** IOP Conference Series: Materials Science and Engineering, 850(1), 012003. August 12, 2024.Available from https://doi.org/10.1088/1757-899X/850/1/012003

Ocharoen, N. (2017). ***Road traffic accident and Thai economy***. October 9, 2024, Available from https://tdri.or.th/2017/08/econ_traffic_accidents/. (In Thai).

Sharma, S., & Singh, R. (2023). ***Android accident detection and alert system*.** August 13, 2024.Available from https://www.researchgate.net/publication/379049813_Android_accident_detection_and_alert_system

World Health Organization. (2022). ***Road traffic injuries.*** October 9, 2024, Available from https://www.who.int/news-room/fact-sheets/detail/.



ประวัติผู้วิจัย

ชื่อ - ชื่อสกุล สุชานาฏ มโนทัย

วัน เดือน ปี เกิด 25 พฤศจิกายน 2545

ที่อยู่ปัจจุบัน 30 หมู่ที่ 5 ตำบลจอมพระ อำเภอท่าวังผา จังหวัดน่าน 55140

ประวัติการศึกษา

ปี พ.ศ. 2564 – ปัจจุบัน ระดับปริญญาตรี หลักสูตรวิทยาศาสตร์บัณฑิต สาขาภูมิศาสตร์
มหาวิทยาลัยนเรศวร ตำบลท่าโพธิ์ อำเภอเมืองพิษณุโลก จังหวัดพิษณุโลก 65000 เกรดเฉลี่ย 3.28

ปี พ.ศ. 2561 – 2563 ระดับมัธยมศึกษาตอนปลาย (วิทย์ – คณิต) โรงเรียนท่าวังผาพิทยาคม ตำบลท่าวังผา อำเภอท่าวังผา จังหวัดน่าน 55140 เกรดเฉลี่ย 3.55

ปี พ.ศ. 2558 – 2560 ระดับมัธยมศึกษาตอนต้น โรงเรียนท่าวังผาพิทยาคม ตำบลท่าวังผา อำเภอท่าวังผา จังหวัดน่าน 55140

กิจกรรมที่เข้าร่วม

แข่งขัน TESA Top Gun Rally ครั้งที่ 17 ณ มหาวิทยาลัยอุบลราชธานี

ผู้ช่วยและเข้าร่วมอบรมเชิงปฏิบัติการออกแบบแผนที่ภูมิศาสตร์ ครั้งที่ 1

ผู้ช่วยอบรมเชิงปฏิบัติการออนไลน์ เรื่องการออกแบบแผนที่สุนทรียภาพและนวัตกรรมการทำแผนที่ภูมิศาสตร์ในโครงการงานวิทยาศาสตร์

ศึกษานอกสถานที่ ณ สถานีอุตุนิยมวิทยา จังหวัดพิษณุโลก

ได้รับรางวัล “ยอดเยี่ยม” ประเภทการจัด figure-ground orientation โครงการประกวดออกแบบแผนที่ภูมิศาสตร์แห่งประเทศไทย ครั้งที่ 1

ได้รับรางวัล “ดี” ประเภทแผนที่ สวยงาม โดย นิตยสาร เนชั่นแนล จีโอกราฟฟิก ประเทศไทย โครงการประกวดออกแบบแผนที่ภูมิศาสตร์แห่งประเทศไทย ครั้งที่ 1

รางวัลที่ได้รับ

ได้รับเกียรติบัตรด้านการเรียนผลการเรียนดี ประจำปีการศึกษา 2564 สาขาภูมิศาสตร์ ภาควิชาทรัพยากรธรรมชาติและสิ่งแวดล้อม มหาวิทยาลัยนเรศวร

ได้รับเกียรติบัตรด้านการเรียนผลการเรียนดี ประจำปีการศึกษา 2565 สาขาภูมิศาสตร์ ภาควิชาทรัพยากรธรรมชาติและสิ่งแวดล้อม มหาวิทยาลัยนเรศวร

ได้รับเกียรติบัตรด้านการเรียนผลการเรียนดี ประจำปีการศึกษา 2566 สาขาภูมิศาสตร์ ภาควิชาทรัพยากรธรรมชาติและสิ่งแวดล้อม มหาวิทยาลัยนเรศวร

ได้รับเกียรติบัตรรางวัลนิสิตดีเด่น ด้านความประพฤติดี ประจำปีการศึกษา 2566 สาขาภูมิศาสตร์ ภาควิชาทรัพยากรธรรมชาติและสิ่งแวดล้อม มหาวิทยาลัยนเรศวร