บทที่1
บทนำ

1.1 ความสำคัญและที่มาของปัญหา

ผู้สูงอายุในสังคมไทยเพิ่มขึ้นอย่างรวดเร็ว คาดว่ามีผู้ป่วยด้วยโรคสมองเสื่อมอยู่ราวร้อยละ2-4 ของประชากรไทยในวัย 60 ปีขึ้นไป หรือคิดเป็นสัดส่วนร้อยละ 60-70 ของผู้ตรวจพบว่ามีอาการสมองเสื่อมในปัจจุบัน ด้วยจำนวนประชากรที่เพิ่มขึ้นและช่วงชีวิตที่ยืนยาวขึ้น จำนวนผู้ป่วยโรคอัลไซเมอร์มีแนวโน้มเพิ่มสูงขึ้นเรื่อย ๆ ในอนาคต จากยอดผู้ป่วยไทยที่มีภาวะสมองเสื่อมจำนวน 229,000 รายที่เคยสำรวจพบในปี พ.ศ. 2548 เป็นที่คาดการณ์ว่าตัวเลขจะเพิ่มสูงขึ้นเป็น 450,000 คนในปี พ.ศ. 2568 และอาจเพิ่มขึ้นเป็นจำนวนมากกว่า 1 ล้านคนในอีกราว 40 ปีข้างหน้า ผู้ป่วยอัลไซเมอร์มีความบกพร่องทางความจำที่ทำให้เกิดการหลงทางได้ง่าย การหลงทางเป็นปัญหาที่ร้ายแรงสำหรับผู้ป่วยอัลไซเมอร์ เนื่องจากผู้ป่วยอาจเสี่ยงต่อการเกิดอุบัติเหตุ เช่น การถูกรถชน หรือการตกลงในที่ไม่ปลอดภัย อีกทั้งผู้คนในสังคมยังไม่ทราบถึงปัญหานี้อย่างแพร่หลาย การขาดเครื่องมือที่สามารถช่วยให้ผู้ป่วยหาทางกลับบ้านได้อย่างมีประสิทธิภาพเป็นอีกหนึ่งปัญหาสำคัญที่ต้องได้รับการแก้ไข (Hitap, 2556)

จากนั้นจึงได้ค้นหาและศึกษาระบบแอปพลิเคชันที่มีวัตถุประสงค์หรือแนวทางคล้ายคลึงกับแอปพลิเคชันที่กำลังจะพัฒนาขึ้นเพื่อเป็นแนวทาง จึงได้พบเห็นแอปพลิเคชันแนะนำสถานที่ท่องเที่ยวและที่ พักในอำเภอจอมทอง จังหวัด เชียงใหม่บนระบบปฏิบัติการแอนดรอยด์ (เที่ยงจันตา, 2565) ที่ โดยแอปพลิเคชั่นนี้มีการใช้ระบบปฏิบัติการกูเกิล แมพ ที่เป็นระบบแบบสากลที่ทุกคนสามารถเข้าถึงได้ และมีฐานข้อมูล มายเอสคิวเอล เป็นตัวช่วยเก็บข้อมูลที่บันทึกไว้ได้หลากหลาย และยังดูแลการจัดการระบบได้อย่างสะดวก จึงทำให้นำรูปแบบของแอพลิเคชั่นดังกล่าวนี้มาเป็นมีระบบวัตถุประสงค์หรือแนวทางคล้ายคลึงกับที่ต้องการ

จากปัญหาดังกล่าวมาข้างต้นจึงได้พัฒนาแอปพลิเคชันช่วยหาเส้นทางกลับบ้านด้วยคิวอาร์โค้ด แอปพลิเคชันมีระบบสแกนจดจำใบหน้า สามารถช่วยเหลือผู้ป่วยอัลไซเมอร์โดยค้นหาเส้นทางกลับบ้านได้ แอปพลิเคชันมีระบบให้ผู้ใช้งานทำการเข้าสู่ระบบแล้วกรอกข้อมูลดังกล่าวและถ่ายรูปภาพของผู้ป่วยลงไปในแอพพลิเคชั่นที่ระบบต้องการไว้ จากนั้นระบบผู้ดูแลกระทำการตรวจสอบข้อมูลดังกล่าวและกระทำการส่งข้อมูลตำแหน่งที่อยู่ของผู้ป่วยผ่านระบบ google map ให้ผู้ใช้ได้รับรู้เพื่อดำเนินการช่วยเหลือผู้ป่วยในการนำทางกลับบ้านอย่างมีประสิทธิภาพ ซึ่งในปัจจุบันยังไม่มีเครื่องมือหรือแอปพลิเคชันที่สามารถตอบโจทย์นี้ได้อย่างมีประสิทธิภาพ

1.2 วัตถุประสงค์ของโครงการ

1.2.1 พัฒนาแอปพลิเคชันเพื่อช่วยหาเส้นทางกลับบ้านด้วยคิวอาร์โค้ด สำหรับผู้ป่วยหรือผู้สูงอายุ

1.2.2 เพื่อเป็นการทดสอบแอปพลิเคชันเพื่อช่วยหาเส้นทางกลับบ้านด้วยคิวอาร์โค้ดสำหรับผู้ป่วยหรือผู้สูงอายุ

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 ได้แอปพลิเคชันที่สามารถช่วยหาเส้นทางกลับบ้านด้วยคิวอาร์โค้ดลดภาระและความกังวลของผู้ดูแล

1.3.2 ได้โอกาสทดสอบระบบแอปพลิเคชันเพื่อช่วยหาเส้นทางกลับบ้านด้วยคิวอาร์โค้ดสำหรับผู้ป่วยหรือผู้สูงอายุ

1.4 ขั้นตอนและวิธีการดำเนินงาน

1.4.1 ศึกษาและสืบค้นข้อมูลเกี่ยวกับผู้สูงอายุและโปรแกรมที่ใช้ในการพัฒนา

1) รวบรวมข้อมูลเกี่ยวกับผู้สูงอายุ

2) ศึกษาระบบปฏิบัติการแอนดรอยด์ (Andriod)

3) ศึกษา ฟลัตเตอร์ เฟรมเวิร์ค (Flutter Flamework)

4) ศึกษาข้อมูลเกี่ยวกับโปรแกรมประยุกต์บนอุปกรณ์พกพาหรือโทรศัพท์เคลื่อนที่มีระบบปฏิบัติการแอนดรอยด์ (Andriod Operating System) ศึกษาโดยการใช้โปรแกรม Visual Studio Code

1.4.2 วิเคราะห์ความต้องการ (Requirement Analysis) โมบายแอปพลิเคชัน สำหรับค้นหาเป้าหมาย บนแผนที่ Google Map ของ แอพลิเคชันบนระบบปฏิบัติการแอนดรอยด์ส่วนของผู้ใช้งาน เข้าสู่ระบบ เพิ่ม ลบ แก้ไข ข้อมูลรายละเอียดข้อมูลของตนเองได้ส่วนของผู้ดูแลระบบ สามารถเข้าสู่ระบบได้ สามารถกำหนดสิทธิการเข้าถึงของผู้เยี่ยมชมได้สามารถดูข้อมูลการใช้งานของผู้เยี่ยมชมได้ และยังสามารถ เพิ่ม ลบ แก้ไข ผู้ใช้งานได้ สำหรับ แอปพลิเคชันช่วยหาเส้นทางกลับบ้านด้วยคิวอาร์โค้ด

1.4.3 ขั้นตอนการออกแบบระบบ (Design) ออกแบบระบบการทำงาน ข้อมูลนำเข้า การแสดงผล ข้อมูลลำดับตามขั้นตอนให้สอดคล้องกับความต้องการของผู้ใช้ วิเคราะห์ข้อมูลการออกแบบระบบโดยใช้ UML (Unified Modeling Language) ได้แก่

1.4.3.1 ยูธเคสไดอะแกรม (Use Case Diagram) เพื่อแสดงแผนภาพแสดงการทำงาน ของระบบผู้ใช้ระบบ (User)และความสัมพันธ์กับระบบย่อย (Sub systems) ได้นำเอายูสเคส ไดอะแกรมมาใช้อธิบายดังนี้ เมื่อผู้ใช้งานเข้าแอปพลิเคชัน ระบบจะแสดงภาพรวมของแอปพลิเคชัน รวมถึงรายละเอียดต่าง ๆ ทั้งหมดในแอปพลิเคชัน

1.4.3.2 ซีเควนซ์ไดอะแกรม (Sequence Diagram) เพื่อแสดงส่วนต่างๆของแอปพลิเคชันช่วยหาเส้นทางกลับบ้านด้วยคิวอาร์โค้ดแสดงลำดับกิจกรรมการใช้ของผู้ใช้งานโดยที่สามารถเห็นถึงการทำงานทั้งระบบตั้งแต่ต้นจนจบการทำงาน

1.4.3.3 แอคทิวิตี้ไดอะแกรม (Activity Diagram) ใช้ในส่วนของการแสดง Workflow การทำงานของกิจกรรมของระบบจากจุดหนึ่งไปสู่อีกจุดหนึ่งตั้งแต่ต้นจนจบโปรแกรม

1.4.3.4 คลาส ไดอะแกรม (Class Diagram) ของแอปพลิเคชันช่วยหาเส้นทางกลับบ้านด้วยคิวอาร์โค้ด จะแสดงถึงการทำงานของระบบจะแบ่งออกเป็นคลาสโดยแต่ละคลาสนั้นจะแสดงส่วนของตัวข้อมูลระบบที่จะถูกใช้งาน เพื่อให้มองเห็นภาพรวมระบบ

1.4.4 ขั้นตอนการเขียนโปรแกรม (Programming) เขียนโปรแกรมที่ออกแบบไว้เพื่อให้สอดคล้องกับ วัตถุประสงค์ สร้างระบบงานโดยโปรแกรม Visual Studio Code ที่ใช้ฟลัดเตอร์เฟรมเวิร์ก (Flutter Flamework) ในการเขียนโปรแกรม ประกอบการใช้งานแอปพลิเคชันเอกสารประกอบการใช้งานเพื่อนำเสนอเป็นรูปเล่ม

1.4.5 ขั้นตอนการทดสอบระบบ (Testing) มีการทดลองใช้ระบบและรับฟีดแบ็คเพื่อนำมาแก้ไขและบำรุงตัวแอปพลิเคชันให้ดียิ่งขึ้น

1.4.6 มีการแก้ไขเพิ่มเติมตามที่ได้ทดสอบเพื่อให้ประสิทธิภาพดีที่สุด

1.4.7 ขั้นตอนในการจัดทำเอกสารเกี่ยวกับระบบ (Document) คู่มือการติดตั้งและการใช้งานประกอบการใช้งานแอปพลิเคชัน เอกสารประกอบการใช้งานเพื่อนำเสนอในรูปแบบรูปเล่ม

1.5 ขอบเขตการจัดโครงการ

ขอบเขตการพัฒนาแอปพลิเคชันช่วยหาเส้นทางกลับบ้านด้วยคิวอาร์โค้ดดังต่อไปนี้

ด้านผู้ดูแลระบบ

1.5.1 ผู้ดูแลระบบสามารถจัดเก็บข้อมูลผู้ป่วย อนุญาตให้ผู้ใช้ทั่วไปสามารถเข้าถึงข้อมูลผู้ป่วยได้โดยการขออนุญาต

1.5.2 ผู้ดูแลระบบสามารถตรวจสอบข้อมูลผู้ใช้ทั่วไปและทำการอนุญาตให้ผู้ใช้ทั่วไปเข้าถึงข้อมูลได้

1.5.3 ผู้ดูแลระบบสามารถดูแลระบบ เพิ่ม ลบ และแก้ไขข้อมูลของผู้ป่วยและผู้ใช้ทั่วไปได้ทั้งหมด

1.5.4 ผู้ดูแลระบบเมื่อได้รับข้อมูลจากผู้ใช้ทั่วไปจะทำการตรวจสอบและส่งข้อมูลที่อยู่ด้วยการสแกนให้ผู้ใช้ทั่วไป

ด้านผู้ใช้ทั่วไป

1.5.1 ผู้ใช้จำเป็นต้องลงชื่อผู้ใช้และกรอกรหัสผ่าน

1.5.2 ผู้ใช้ต้องกรอกข้อมูลที่อยู่และเบอร์ติดต่อปัจจุบัน

1.5.3 ผู้ใช้ต้องถ่ายรูปภาพผู้ป่วย

1.5.4 ผู้ใช้เมื่อกดส่งข้อมูลแล้วรอซักครู่เพื่อรับ รหัสแกน

1.6 รายละเอียดเครื่องมือที่ใช้ในการจัดการ

1.6.1. คอมพิวเตอร์ส่วนบุคคล (PC) - รุ่น: ASUS TUF Gaming A15 FA506QM_FA506QMฟอร์มแฟกเตอร์: แล็ปท็อป Windows 11 (64 บิต)

1.6.2. ข้อมูลโปรเซสเซอร์: ผู้จำหน่าย CPU: AuthenticAMD ยี่ห้อ CPU: AMD Ryzen 7 5800H with Radeon Graphics ตระกูล CPU: 0x19 รุ่น CPU: 0x50 จำนวนการปรับปรุง CPU: 0x0 ชนิด CPU: 0x0 ความเร็ว: 3194 Mhz 16 หน่วยประมวลผลทางตรรกะ 8 หน่วยประมวลผลทางกายภาพ HyperThreading:

1.6.3. โปรแกรมที่ใช้พัฒนา:

- โปรแกรมMicrosoft Word 2019 ใช้ในการทำเอกสาร
- โปรแกรมวิชวลสตูดิโอโค้ด (Visual Studio Code) ใชในการเขียนแอปพลเคชัน
- โปรแกรมแอนดรอยด์สตูดิโอ (Android Studio) ใช้ในการพัฒนาแอปพลเคชัน
- ภาษา Dart ใชในการพัฒนาแอปพลิเคชัน

1.7 สถานที่ใช้ในการทำวิจัยและเก็บข้อมูล

ห้องเรียนคณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏเชียงใหม่ (สะลวง)

บทที่2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในการพัฒนาแอปพลิเคชันช่วยหาเส้นทางกลับบ้านด้วยคิวอาร์โค้ด ได้ทำการศึกษาได้มาแบ่งทฤษฎีและงานวิจัยที่ เกี่ยวข้องออกเป็น 6 ส่วนใหญ่ โดยมีรายละเอียดดังนี้

2.1 ความรู้ทั่วไปเกี่ยวกับผู้ป่วยอัลไซเมอร์

2.2 คิวอาร์โค้ด

2.3 กูเกิลแมพและเอพีไอ

2.4 การวิเคราะห์และออกแบบระบบเชิงวัตถุ

2.5 โปรแกรมและภาษาที่ใช้พัฒนาโปรแกรม

2.6 ระบบการจัดการฐานข้อมูล

2.7การสอบระบบ

2.8 งานวิจัยที่เกี่ยวข้อง

2.1 ความรู้ทั่วไปเกี่ยวกับผู้ป่วยอัลไซเมอร์

2.1.1 โรคอัลไซเมอร์คือความบกพร่องในการทำหน้าที่ของสมองในกลุ่มโรคสมองเสื่อม อาการที่เด่นชัดคือ ความบกพร่องในกระบวนการทางความคิด ความจำ หลงวัน เวลา สถานที่ และแม้กระทั่งลืมคนใกล้ชิดหรือผู้ดูแล ความสามารถทางสติปัญญาลดลง เรียนรู้ได้ช้าลง ในด้านการรับรู้ก็จะเปลี่ยนไป ความสนใจและสมาธิลดลง การโต้ตอบและความสามารถในการอยู่ร่วมกับผู้อื่นในสังคมก็ถดถอยลง ผู้ป่วยอัลไซเมอร์ จำเป็นต้องได้รับการดูแลเอาใจใส่อย่างถูกวิธี เนื่องจากผู้ป่วยจะมีความสามารถในการทำกิจวัตรประวันลดลงเรื่อย ๆ การสังเกตอย่างเอาใจใส่ และการเข้าใจการดำเนินโรคจะเป็นสิ่งที่จะช่วยประคับประคองให้ผู้ป่วยอยู่ได้อย่างมีความสุข และตัวผู้ดูแลเองก็จะไม่เกิดความวิตกกังวลและรู้สึกเหนื่อยมาก จากภาวะและปัญหาดังที่กล่าวมา เมื่อผู้ดูแลต้องดูแลผู้ป่วยไปนานๆ มักจะเกิดความอ่อนล้า เกิดความเครียด เบื่อหน่าย กังวล ซึมเศร้า และบางครั้งอาจรู้สึกผิดหวังที่ผู้ป่วยอาการไม่ดีขึ้น ทั้งยังแสดงออกถึงการไม่พอใจในการดูแลของตนและไม่ให้ ความร่วมมือในการดูแลตนเอง ถึงจุดนี้ผู้ทำหน้าที่ดูแลผู้ป่วยก็มักจะมีคำถามว่า จะดูแลผู้ป่วย อัลไซเมอร์อย่างไรดี เพื่อให้ทั้งตนเองและผู้ป่วยมีความสุข

2.1.2 ทำความเข้าใจโรคอัลไซเมอร์ ควรทำความเข้าใจกับโรคสมองเสื่อมและอัลไซเมอร์ให้ดี โดยหาความรู้จากสื่อต่างๆ เช่น อ่านหนังสือ ฟังวิทยุ ดูโทรทัศน์ อ่านบทความทางอินเตอร์เน็ต หรืออาจสอบถามจากแพทย์ที่ดูแลผู้ป่วย เมื่อมีความรู้ความเข้าใจ ไม่ว่าจะเป็นลักษณะอาการ ระยะเวลา วิธีการรักษาการพยากรณ์โรค ตลอดจนวิธีการช่วยเหลือดูแลผู้ป่วยแล้ว ก็จะสามารถรับมือและแก้ปัญหาต่างๆ

ที่จะเกิดขึ้นได้อย่างเหมาะสมยอมรับผู้ป่วย และอาการของโรคของผู้ป่วยว่าโรคนี้รักษาไม่หาย การดูแลที่เหมาะสมจะทำให้ผลกระทบด้านต่างๆ ลดลงได้

2.1.3 ผู้ดูแลเข้าใจผู้ป่วยอัลไซเมอร์ ควรแก้ไขอารมณ์และพฤติกรรมที่เป็นปัญหามากที่สุดของผู้ป่วยก่อน เพราะจะช่วยทำให้การดูแลผู้ป่วยง่ายขึ้นสิ่งใดก่อให้เกิดอารมณ์หรือความไม่พอใจแก่ผู้ป่วย ควรหาสาเหตุแก้ไขหรือหลีกเลี่ยง ช่วยลดความเครียดแก่ผู้ป่วย ถ้าผู้ดูแลเข้าใจถึงจุดนี้ ก็จะไม่รู้สึกว่าตนเองดูแลผู้ป่วยได้ไม่ดีพอ ความเครียดก็จะไม่เกิดขึ้นบางครั้งผู้ป่วยอาจแสดงอารมณ์ที่ทำให้ผู้ดูแลรู้สึกผิดหวัง ผู้ดูแลต้องเข้าใจว่าเป็นผลมาจากอาการของโรค ไม่ใช่ผู้ป่วยไม่พึงพอใจ โกรธ หรือตั้งใจจะต่อว่าผู้ดูแล เนื่องจากก่อนป่วยผู้ป่วยมิได้มีบุคลิกภาพเช่นนั้น

2.1.4 ให้ผู้ป่วยอัลไซเมอร์เข้าใจตนเองควรอธิบายให้ผู้ป่วยทราบถึงสุขภาพที่เปลี่ยนไปของผู้ป่วยในขณะที่ยังสามารถรับรู้และเข้าใจได้ตั้งแต่อาการยังไม่รุนแรงนัก เพื่อให้ผู้ป่วยเตรียมพร้อมและยินดีให้ความร่วมมือในการดูแลตนเอง คอยให้กำลังใจและสนับสนุนให้ผู้ป่วยเข้าใจว่า มีกิจวัตรประจำวันหลายอย่างที่ผู้ป่วยสามารถทำเองได้ เพื่อให้ผู้ป่วยไม่รู้สึกด้อยค่าหรือเป็นภาระ จะทำให้ผู้ป่วยเกิดความภาคภูมิใจ รู้สึกมีคุณค่า และมีความมั่นใจมากขึ้น

2.1.5 หลีกเลี่ยงและป้องกันก่อนเกิดปัญหาผู้ป่วยจำนวนมากเมื่อมีอาการ อาจทำให้การพูดคุยสื่อสารได้ลดลง อาจจะพูดไม่ออก หรือดูนิ่งเงียบ แต่โดยปกติจะฟังเข้าใจ ฉะนั้นการพูดแต่ไม่เห็นผู้ป่วยตอบสนองอาจจะไม่ใช่ไม่ได้ยินหรือไม่เข้าใจหลีกเลี่ยงคำพูดหรือพฤติกรรมใดๆ ที่ส่งผลกระทบทางอารมณ์กับผู้ป่วย อะไรที่ผู้ป่วยไม่ชอบ โกรธ เสียใจ ผิดหวังก็และเกิดความเครียด และเพิ่มการเก็บตัวซึมเศร้าได้ ไม่หลอกล่อให้ผู้ป่วยทำสิ่งใดสิ่งหนึ่ง แล้วไม่รักษาสัญญาที่ให้ไว้กับผู้ป่วย หากรับปากผู้ป่วยไว้อย่างไรก็ควรทำตามนั้น เพราะผู้ป่วยจะหมดความเชื่อถือและเกิดการต่อต้านได้ ไม่วิจารณ์ ต่อว่าโต้เถียง หรือตะคอกผู้ป่วยต่อหน้าผู้อื่น โดยเฉพาะเวลาที่ผู้ป่วยจำอะไรไม่ได้ เพราะจะทำให้ผู้ป่วยอับอาย และอย่าทำโทษผู้ป่วย เพราะจะทำให้ผู้ป่วยเกิดความรู้สึกผิดนำไปสู่อาการทางจิตได้ทำสร้อยหรือกำไลข้อมือให้ผู้ป่วยสวมข้อมือไว้ โดยมีข้อความระบุว่า ผู้ที่สวมใส่มีปัญหาด้านความจำ พร้อมใส่หมายเลขโทรศัพท์ของญาติหรือผู้ดูแล เพราะหากผู้ป่วยพลัดหลงหรือออกจากบ้านไปโดยไม่มีใครรู้ เมื่อมีผู้พบเห็นจะได้ติดต่อผู้ดูแลได้ ทำให้ตามหาตัวผู้ป่วยได้ง่ายขึ้น

2.1.6 ปฏิบัติตามคำแนะนำของแพทย์

1) เพื่อให้การรักษาได้ผลดียิ่งขึ้น ผู้ดูแลควรสังเกตอาการที่ผิดปกติของผู้ป่วย บันทึกพฤติกรรม และแจ้งให้แพทย์ทราบเมื่อถึงเวลานัดตรวจโรค

2) การรักษาผู้ป่วย อาจมีความจำเป็นที่ต้องใช้ยาที่ออกฤทธิ์ต่อจิตประสาท เช่น นอนไม่หลับ วิตกกังวล ซึมเศร้า ก้าวร้าว ผู้ดูแลต้องให้ผู้ป่วยรับประทานยาอย่างสม่ำเสมอและสังเกตอาการหลังจากใช้ยา เพื่อแจ้งแก่แพทย์ได้อย่างถูกต้อง

3) หากผู้ป่วยมีอาการผิดปกติควรรีบปรึกษาแพทย์ เช่น อาการนอนไม่หลับ วิตกกังวลหรือซึมเศร้ามากเกินไป พฤติกรรมเปลี่ยนแปลง ก้าวร้าว หลงผิด หูแว่ว หรืออาการทางจิตอื่นๆ

2.1.8 การดูแลตนเองของผู้ดูแลผู้ป่วยโรคอัลไซเมอร์ ผู้ดูแลผู้ป่วยควรจัดสรรเวลาพักผ่อนให้เพียงพอโดยเปลี่ยนให้ผู้อื่นดูแลผู้ป่วยแทนบ้าง เพราะการดูแลผู้ป่วยอย่างต่อเนื่องนานๆ อาจทำให้เกิดความอ่อนล้า ความเครียด หงุดหงิด ซึ่งไม่เป็นผลดีต่อทั้งผู้ดูแลและผู้ป่วยในระยะยาว ผู้ดูแลผู้ป่วยควรมีเวลาทำกิจกรรมที่ตนเองชอบ เพื่อความผ่อนคลาย หากผู้ป่วยสามารถร่วมกิจกรรมต่างๆ ไปพร้อมกันได้ก็จะเป็นการสร้างความสัมพันธ์และส่งผลดีต่อทั้งผู้ดูแลและผู้ป่วยไปพร้อมๆ กัน

2.2 คิวอาร์โค้ด

2.2.1 คิวอาร์โค้ด (QR Code) คือ สัญลักษณ์รูปสี่เหลี่ยมที่ประกอบไปด้วยโมดูลสีดำ (จุดสี่เหลี่ยม) ที่จัดอยู่ตารางสี่เหลี่ยมพื้นผ้าสีขาว เป็นสัญลักษณ์แทนข้อมูล ซึ่งอุปกรณ์ที่ใช้ในการสแกน QR Code นั้นก็จะใช้กล้องที่ติดกับโทรศัพท์มือถือในการสแกน นอกจากนี้ QR Code ยังเป็นสัญลักษณ์ที่ได้รับความนิยมเป็นอย่างมากเนื่องจากช่วยความสะดวกสบาย

2.2.2 ประเภทของ QR Code ในปัจจุบันนี้จะมีทั้งหมด 5 ประเภท ดังนี้

1) QR Code Model เป็น QR Code แบบดั้งเดิมที่มีขนาดใหญ่ที่สุด โดยจะมีขนาด 73*73 โมดูล สามารถบรรจุข้อมูลได้มากถึง 1,167 ตัว และอีกหนึ่งแบบ คือ Model 2 เป็นเวอร์ชันที่ถูกพัฒนามาจาก Model 1 สามารถบรรจุข้อมูลได้มากถึง 7,089 ตัว ซึ่งในปัจจุบันนี้ Model 2 เป็นที่นิยมใช้กันอย่างมากเลยทีเดียว

2) Micro QR Code เป็น QR Code ที่มีขนาดเล็กกว่าแบบแรกมากพอสมควร เนื่องจากจะแสดงผลบางจุดตรวจตำแหน่งเพียงตำแหน่งเดียว ซึ่งขนาดใหญ่ที่สุดของ Micro QR Code 17*17 โมดูล ที่สามารถบรรจุข้อมูลได้ทั้งหมด 35 ตัว

3) IQR Codeเป็น QR Code ที่มีขนาดเล็กและพิมพ์ออกมาแนวนอน ที่สามารถทำการเก็บข้อมูลได้มากกว่า 80% แต่หากมีการจัดเก็บข้อมูลในปริมาณที่เท่ากัน ก็จะประหยัดพื้นที่ในการแสดงผลได้มากถึง 30% ซึ่งสามารถเก็บข้อมูล 40,000 ตัวอักษร

4) SQRCเป็น QR Code ที่จะมีคุณลักษณะเหมือนกับ QR Code Model 1 และ QR Code Model 2 ทุกประการ แต่จะมีข้อแตกต่างกันเล็กน้อย คือ สามารถทำการเก็บข้อมูลลับได้นั่นเอง

5) Frame QR สามารถนำรูปภาพ กราฟิกมาติดบริเวณตรงกลางของ QR Code ได้ ส่วนใหญ่ก็มักจะนิยมใช้ QR Code ประเภท Frame QR ในงานประชาสัมพันธ์ Event นิทรรศการ เพื่อทำให้เกิดความสะดุดตามากที่สุดนั่นเอง

2.2.3 ระบบ QR Code นำมาใช้กับระบบสแกน QR Code ในปัจจุบันนี้ เป็นสิ่งที่ได้รับความนิยมเป็นอย่างมาก เนื่องจากสามารถนำมาประยุกต์ใช้งานได้หลากหลายรูปแบบมาก ซึ่งสามารถนำมาประยุกต์ใช้ได้ดังนี้

1) ใช้ในการชำระเงินได้ทันที ซึ่งในยุคนี้จะนิยมนำระบบคิวอาร์โค้ดมาใช้ในการชำระเงิน เนื่องจากสามารถทำการชำระเงินได้อย่างรวดเร็ว ทั้งยังตอบโจทย์การใช้เงินในสังคมไร้เงินสดอีกด้วย

2) ใช้โปรโมทร้านค้า เนื่องจากในยุคนี้คนจำนวนมากไม่ชอบอ่านหนังสือที่มีจำนวนมาก ดังนั้นการนำ QR Code ไปแปะไว้บนโปสเตอร์ก็จะทำให้กลุ่มลูกค้าสามารถเข้าถึงร้านได้ง่ายขึ้นและรวดเร็วมากยิ่งขึ้นนั่นเอง

3) โปรโมชั่นเสริมการขาย โดยอาจจะใช้ในการสแกนเพื่อรับสะสมแต้ม รับโปรโมชั่นพิเศษต่างๆ รวมถึงการทำแบบสอบถามเพื่อรับส่วนลด

4) อีเวนท์ สำหรับการจัดงาน Event ที่มีการสแกนเพื่อลงทะเบียนเข้างานการใช้คิวอาร์โค้ด ถือเป็นสิ่งที่ช่วยเพิ่มความรวดเร็วในการลงทะเบียนได้เป็นอย่างดี ทำให้ไม่ต้องต่อแถวลงทะเบียน เป็นการสร้างความประทับใจให้กับผู้ร่วมงานได้เป็นอย่างดี

2.2.4 QR Code เป็นบาร์โค้ดที่มีทั้งหมด 2 มิติ ที่มีต้นกำเนิดมาจากประเทศญี่ปุ่น โดยบริษัท เดนโซ เวฟ ตั้งแต่ปี ค.ศ 1994

2.2.5 ความแตกต่างระหว่างบาร์โค้ดกับคิวอาร์โค้ด คือ QR Code นั้นจะมีลักษณะเป็นบาร์โค้ด 2 มิติ ในขณะที่ Barcode นั้นมี 1 มิติ และ Barcode จะมีลักษณะเป็นเส้นสีดำวางอยู่บนพื้นสีขาว ส่วน QR Code จะมีลักษณะเป็นสี่เหลี่ยม ซึ่งทั้งบาร์โค้ดและคิวอาร์โค้ดนั้นจะสามารถเห็นได้ง่ายทั่ว ๆ ไป

2.3 กูเกิลแมพเอพีไอ

กูเกิลแมพเอพีไอ (Google Map API) เป็นชุดเอพีไอ (API) ของ Google สำหรับพัฒนา เว็บแอปพลิเคชัน และ โมบายแอปพลิเคชัน (Android, iOS) ไว้สำหรับเรียกใช้แผนที่และชุด เซอร์วิส ต่าง ๆ ของกูเกิล เพื่อพัฒนา แอปพลิเคชัน ได้เหมือนกับที่ กูเกิล โดยมีแผนที่ยังหน้าต่าง มากมายให้เรียกใช้ ประกอบด้วย

– การปรับแต่งแผนที่ (Styled Map)

– ชุดควบคุมแผนที่ (Map Control)

– ชุดเครื่องมือวาดภาพบนแผนที่ (Drawing)

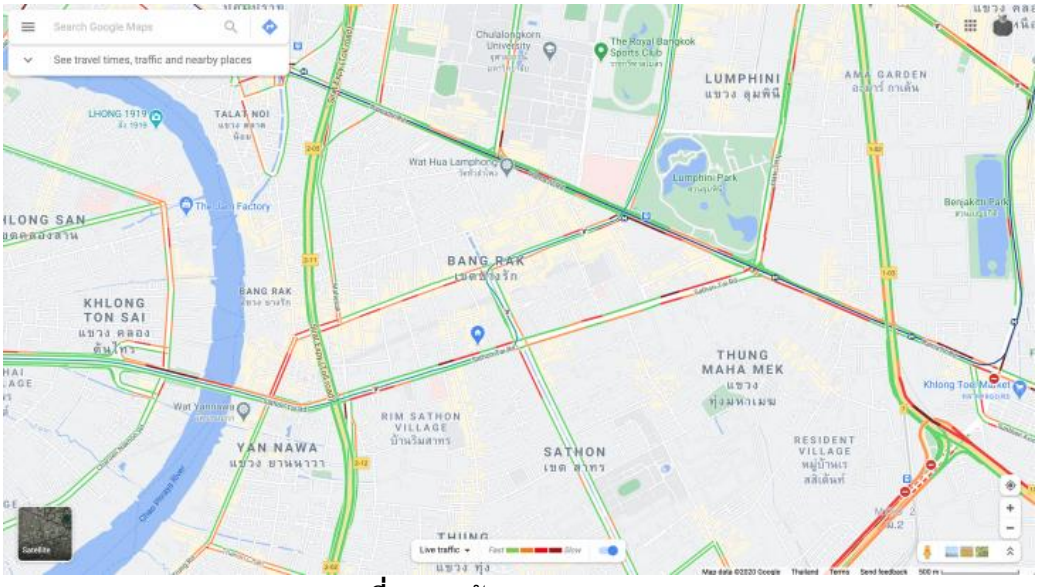
– การนำทางจากจุดหนึ่งไปยังอีกจุดหนึ่ง (Directions Service)

– การคำนวณความสูงของจุดพิกัด (Elevation Service)

– การแปลงที่อยู่เป็นพิกัด Latitude และ Longitude (GeoCoding Service)

– การดึงข้อมูล POI (Point of Interest) คือข้อมูลสถานที่ต่าง ๆ ที่ Google รวบรวมไว้ให้ เช่น โรงแรม ห้างสรรพสินค้า โรงเรียน -สถานที่ราชการต่างๆ และอื่นๆ อีกมากมาย (Places API) มาใช้งานใน Application

– Street View เป็นโปรแกรมเสริมซึ่งพัฒนามาจาก กูเกิลแมพ ที่ให้มุมมองภาพแบบเสมือนจริงจากตำแหน่งต่าง ๆ ตามถนนหลายแห่งบนโลก โปรแกรมนี้ได้มีการเปิดตัวอย่างเป็นทางการวันที่ 25 พฤษภาคม ค.ศ. 2007 (พ.ศ. 2550) ในหลายเมืองของประเทศสหรัฐอเมริกา และขยายตัวสำหรับเมืองต่าง ๆ และชนบททั่วโลก



ภาพที่ 2.1 หน้าต่าง Google Map

ที่มา: https://tips.thaiware.com/1433.html

2.3 การวิเคราะห์และออกแบบระบบเชิงวัตถุ

การวิเคราะห์และออกแบบระบบเชิงวัตถุ เป็นวิธีการที่ได้รับความนิยม โดยการดูระบบจากมุมมองของตัว Object เอง เพราะ Object ทำหน้าที่ปฏิบัติงานและเป็นตัวโต้ตอบหรือปฏิสัมพันธ์กับระบบ โดยผลผลิตสุดท้ายของการวิเคราะห์เชิงวัตถุ คือ การจำลองแบบเชิงวัตถุ (Object Model) ซึ่งจะเป็นตัวแทนของระบบสารสนเทศใน ความหมายของ Object และแนวความคิดเชิงวัตถุ ซึ่งเมื่อถึงระยะของการทำให้เกิดผลในวงจรการ พัฒนาระบบ นักวิเคราะห์ระบบและนักเขียนโปรแกรมก็จะทำการแปลง Object ให้เป็น ส่วนจำเพาะ ของรหัสชุดคำสั่ง ซึ่งการใช้วิธีการแยกเป็นส่วนจำเพาะหรือ Modular จะช่วยประหยัดเงินและเวลา เนื่องจากสามารถถูกใช้อย่างเต็มที่ สามารถถูกตรวจสอบ และสามารถนำเอากลับมาใช้ใหม่ได้อีก

2.3.1 ข้อดีของ Object-Oriented (OO)

1) ลดความซับซ้อนของการพัฒนาระบบ และยังทำให้การสร้างและการดูแลเป็นไปได้ง่าย และรวดเร็ว

2) พัฒนาความสามารถในการสร้าง และคุณภาพของโปรแกรมเมอร์ เนื่องจากมีการวางโครงร่างการนำมาใช้งาน และมีการทดสอบ สามารถที่จะนำระบบนี้ไปใช้กับระบบอื่นๆได้อีก

3) ระบบที่มีการพัฒนาด้วย Object-Oriented (OO) จะมีความยืดหยุ่น สามารถแก้ไขและเพิ่มเติมได้

4) Object-Oriented (OO) จะถูกนักวิเคราะห์ระบบมองในแง่ของระบบในโลกของ ความเป็นจริง ไม่ใช่แค่เพียงระดับของโปรแกรมทางภาษา (Programming Language) คือสามารถหาทางแก้ไขปัญหาที่เกิดขึ้นได้อย่างทันที

2.3.2 ยูเอ็มแอล (UML) ย่อมาจาก Unified Modeling Language เป็นภาษาที่ใช้อธิบายแบบจำลองต่างๆ หรือเป็นภาษาสัญลักษณ์รูปภาพมาตรฐาน สำหรับใช้ในการสร้าง แบบจำลองเชิงวัตถุ โดย ยูเอ็มแอล เป็นภาษามาตรฐานสำหรับสร้างแบบพิมพ์เขียวให้แก่ ระบบงาน สามารถใช้ยูเอ็มแอลในการสร้างมุมมอง กำหนดรายละเอียด สร้างระบบงานและ จัดทำเอกสารอ้างอิงให้แก่ระบบงานได้ เนื่องจากยูเอ็มแอล เป็นภาษาที่มีการใช้สัญลักษณ์ รูปภาพ จึงอาจมีสับสนว่า ยูเอ็มแอล เป็นการสร้างแผนภาพหรือเป็นเพียงการใช้สัญลักษณ์ เพื่ออธิบายระบบงานเท่านั้น แต่แท้จริงแล้ว ยูเอ็มแอลมีลักษณะของแบบจำลองข้อมูล คือ เป็นแบบจำลองที่เอาไว้อธิบายแบบจำลองอื่น ๆ อีกที การใช้งานภาษายูเอ็มแอล นอกจากจะต้องเข้าใจในแนวความคิดเชิงวัตถุแล้ว ยัง จำเป็นต้องมีพื้นฐานความเข้าใจเกี่ยวกับแบบจำลองภาพด้วยเช่นกัน แบบจำลอง (Modeling) เป็นวิธีการวิเคราะห์ออกแบบ (Analysis and Design) อย่างหนึ่งที่เน้นการใช้ งานแบบจำลองเป็นหลัก ซึ่งแบบจำลองที่สร้างขึ้นมาจะสามารถช่วยให้เข้าใจในปัญหาได้ง่าย ขึ้น อีกทั้งยังสามารถนำแบบจำลองมาเป็นเครื่องมือในการสื่อสารถ่ายทอดความคิดกับบุคคลอื่นๆ ที่เกี่ยวข้องในโครงการได้ เช่น ลูกค้า นักวิเคราะห์ระบบ นักออกแบบระบบ เป็นต้น ส่วนแบบจำลองภาพ คือการใช้สัญลักษณ์รูปภาพในการสร้างแบบจำลองของระบบที่จะ พัฒนาเพื่อประโยชน์ที่คล้ายคลึงกันในการทำความเข้าใจกับความต้องการของลูกค้าการออกแบบระบบที่เป็นไปได้อย่างชัดเจนขึ้นและการบำรุงรักษาที่ง่ายยิ่งขึ้นแบบจำลอง เกิดขึ้นโดยการนำเสนอส่วนต่างๆ ของระบบแต่เพียงส่วนที่สำคัญโดยไม่คำนึงถึงรายละเอียดปลีกย่อยต่างๆ ในการพัฒนาระบบซอฟต์แวร์ที่ซับซ้อนนักพัฒนาจำเป็นต้องทำความเข้าใจ กับมุมมองด้านต่างๆ ของระบบก่อนทำการพัฒนาจริง โดยการสร้างแบบจำลองอัน เปรียบเสมือนพิมพ์เขียวที่แสดงถึงภาพรวมทั้งหมดของระบบ แบบจำลองที่สร้างขึ้นจะต้องมี ความสอดคล้องกับความต้องการของผู้ใช้งานระบบเป็นสำคัญ ในส่วนของรายละเอียดต่าง ๆ จะค่อย ๆ ถูกเพิ่มเติมลงไปในตัวแบบจำลองและในที่สุดแบบจำลองจะถูกนำไปพัฒนาขึ้นเป็น ระบบจริง สรุปเป้าหมายของ UML นั้นสามารถกำหนดให้เป็นกลไกการสร้างแบบจำลองอย่าง ง่ายเพื่อจำลองระบบการปฏิบัติที่เป็นไปได้ทั้งหมดในสภาพแวดล้อมที่ซับซ้อน 2.2.2 แผนผังกรณี (Use Case Diagram) Use Case Diagram คือ แผนภาพที่แสดงการทำงานของผู้ใช้ระบบ (User) และ ความสัมพันธ์กับระบบย่อย (Sub systems) ภายในระบบใหญ่ ในการเขียน Use Case Diagram ผู้ใช้ระบบ (User) จะถูกกำหนดว่าให้เป็น Actor และ ระบบย่อย (Sub systems) คือ Use Case จุดประสงค์หลักของการเขียน Use Case Diagramก็เพื่อเล่าเรื่องราวทั้งหมด ของระบบว่ามีการทำงานอะไรบ้าง เป็นการดึง Requirement หรือเรื่องราวต่างๆ ของระบบจากผู้ใช้งาน ซึ่งถือว่าเป็นจุดเริ่มต้นในการวิเคราะห์และออกแบบระบบ สัญลักษณ์ที่ใช้ใน Use Case Diagram จะใช้สัญลักษณ์รูปคนแทน Actorใช้สัญลักษณ์วงรีแทน Use Case และใช้เส้นตรงในการเชื่อม Actor กับ Use Case เพื่อแสดงการใช้งานของ Use Case ของ Actor นอกจากนั้น Use Case ทุกๆ ตัวจะต้องอยู่ภายในสี่เหลี่ยมเดียวกันซึ่งมีชื่อของระบบระบุอยู่ด้วย

2.3.3 Use case Diagram คือแผนภาพที่ใช้แสดงปฏิสัมพันธ์ระหว่างระบบงานและสิ่งที่อยู่นอกระบบงาน และแสดงให้เห็นถึงส่วนประกอบทั้งหมด หรือ ภาพรวมของระบบ เป็นรากฐานในการเริ่มต้น

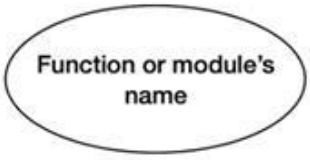
การวิเคราะห์ระบบ โดยค้นหาคำว่าระบบทำอะไร โดยไม่สนใจกลไกการทำงานหรือเทคนิคการทำงาน เปรียบเสมือน "กล่องดำ" โดย Use Case Diagram จะช่วยให้ผู้พัฒนาระบบมสามารถแยกแยะกิจกรรมที่อาจจะเกิดขึ้นในระบบ เป็น Diagram พื้นฐาน ที่สามารถอธิบายสิ่งต่าง ๆ ได้โดยใช้รูปภาพที่ไม่ซับซ้อนประโยชน์ของ Use Case Diagram

- ช่วยให้ผู้พัฒนาระบบมสามารถแยกแยะกิจกรรมที่อาจจะเกิดขึ้นในระบบ
- เป็น Diagram พื้นฐาน ที่สามารถอธิบายสิ่งต่าง ๆ ได้โดยใช้รูปภาพที่ไม่ซับซ้อน
- Use Case Diagram จะมีประสิทธิภาพหากผู้เขียนมีความเข้าใจใน Problem Domain อย่างแท้จริง

ส่วนประกอบของ Use Case Diagram ประกอบด้วย

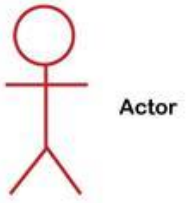
1) Actor คือผู้ที่กระทำกับระบบ อาจเป็นผู้ที่ทำการส่งข้อมูล, รับข้อมูล หรือ แลกเปลี่ยนข้อมูลกับระบบนั้น ๆ เช่นลูกค้ากับระบบสั่งซื้อสินค้าทาง โทรศัพท์

- Use Case คือ หน้าที่หรืองานต่าง ๆ ในระบบ เช่น การเช็คสต็อค การสั่งซื้อสินค้า เป็นต้น
- Relationship คือความสัมพันธ์ระหว่าง Use Case กับ Actor
- System & Use Case Diagram ในระบบใหญ่มักแบ่งระบบออกเป็นระบบย่อย เรียกว่า Subsystem
- ใน Use Case Diagram จะใช้ Use Case แทน Subsystem ผู้ใช้งานระบบย่อยจะเรียกว่า User
- ใช้ Use case Diagram จะใช้ Actor แทน User สัญลักษณ์ที่ใช้ดังภาพที่ 2.2



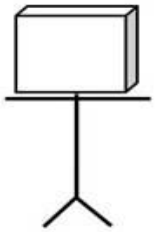
ภาพที่ 2.2 สัญลักษณ์ของ Use Case Diagram

ที่มา: https://devjourneys.com/2020/08/24/use-case-model



ภาพที่ 2.3 สัญลักษณ์ของ Actor

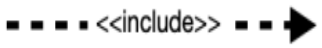
ที่มา: https://devjourneys.com/2020/08/24/use-case-model



ภาพที่ 2.4 สัญลักษณ์ของ Actor System
ที่มา: https://devjourneys.com/2020/08/24/use-case-model

2) ความสัมพันธ์ของ Use Case มี 2 แบบคือ

2.1)ความสัมพันธ์แบบ include คือ ความสัมพันธ์ที่ Use Case หนึ่งต้องพึ่งพาความสามารถหรือฟังก์ชั่นอื่นๆของ Use Case อื่น กล่าวได้ว่าการทำงานของ Use Case หนึ่งเป็นส่วนหนึ่งของการทำงานของอีก Use Case หนึ่ง โดยเรียก Use Case ที่เป็นส่วนหนึ่งของ Use Case อื่นว่า Base Use Case และ Use Case ที่ถูกดึงมาใช้ว่า Include Use Case การวาดความสัมพันธ์ คือ ลากเส้นประจาก Base Use Case หันลูกศรชี้ไปที่ Include Use Case และเขียนชื่อความสัมพันธ์ไว้ตรงกลาง



ภาพที่ 2.5 สัญลักษณ์ <<include>>
ที่มา: https://devjourneys.com/2020/08/24/use-case-model

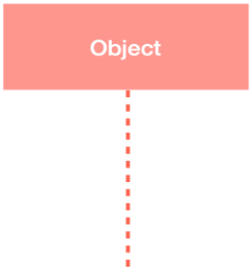
2.2)ความสัมพันธ์แบบ Extends หมายถึงการที่ Use case หนึ่งไปมีผลต่อการทำงานตามปกติของอีก Use case หนึ่ง Use Case ที่มา Extends นั้นจะมีผลให้การดำเนินงานของ Use Case ถูกรบกวนหรือมีการสะดุด หรือมีการเปลี่ยนแปลงกิจกรรมไป สัญลักษณ์แทน Extends ดังภาพที่ 2.6 เส้นปะพร้อมหัวลูกศร ชี้ไปยัง Use Case ที่ถูก Extends มีคำว่า **<<Extends>>** กำกับอยู่บนเส้นหา Use Case และ actor ของระบบ

ภาพที่ 2.6 สัญลักษณ์แทน Extends
ที่มา: https://devjourneys.com/2020/08/24/use-case-model

2.3.4 Sequence Diagram เป็นแผนผังการทำงานแบบลำดับปฏิสัมพันธ์ คือ แผนภาพที่แสดงรายละเอียดความสัมพันธ์ของการดำเนินงาน หรือการทำงานของระบบโดย Sequence Diagram จะโฟกัสที่เวลาและลำดับการโต้ตอบกันระหว่าง Object (วัตถุ) โดยแผนภาพจะอธิบายว่าลำดับเหตุการณ์นั้นเกิดขึ้นได้อย่างไรและเกิดขึ้นเมื่อไหร่ มีลำดับการทำงานอย่างไร ความสัมพันธ์ระหว่าง Use Case Diagram และ Sequence Diagram โดย Use Case Diagram จะมี Actor ที่มีปฏิสัมพันธ์กับแต่ละ Use Case โดยในแต่ละ Use Case ก็จะเป็นตัวแทนของภารกิจหรือ task ในระบบที่ Actor ต้องทำให้สำเร็จ จะสร้าง Sequence Diagram สำหรับแต่ละ Use Case โดยแต่ละ Sequence Diagram จะระบุขั้นตอนหลักสำหรับโต้ตอบเพื่อทำภารกิจนั้นให้สำเร็จ

1) Notations ใน Sequence Diagram

- Lifeline Notationคือเส้นชีวิตของวัตถุหรือ class เป็นตัวแทนของวัตถุหรือส่วนประกอบต่างๆที่มีปฏิสัมพันธ์ซึ่งกันและกันในระบบในลำดับต่างๆ Format การเขียนชื่อเส้นชีวิตคือ Instance Name:Class Name ดังภาพที่ 2.6



ภาพที่ 2.7 สัญลักษณ์แทน Object

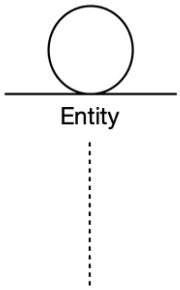
ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

- เส้นชีวิตกับ Actorเส้นชีวิตกับสัญลักษณ์ Actor จะใช้เมื่อลำดับใดลำดับหนึ่งโดยเฉพาะนั้นเป็นส่วนหนึ่งของ Use Case ดังภาพที่ 2.7

ภาพที่ 2.8 สัญลักษณ์แทน Actor

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

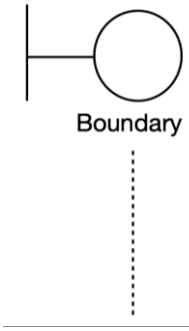
- เส้นชีวิตกับ Entity เส้นชีวิตที่มีองค์ประกอบเป็น Entity แสดงถึงข้อมูลระบบ ตัวอย่างเช่น แอพพลิเคชั่นสำหรับ customer service เอนทิตี้ลูกค้าจะจัดการข้อมูลทั้งหมดที่เกี่ยวข้องกับลูกค้า ดังภาพที่ 2.7



ภาพที่ 2.9 สัญลักษณ์แทน Entity

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

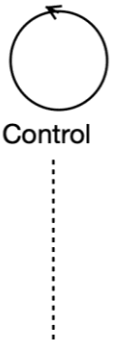
- เส้นชีวิตกับ Boundary element เส้นชีวิตที่มีองค์ประกอบเป็น Boundary element แสดงขอบเขตของระบบหรือ software element ในระบบ เช่น หน้าจอสำหรับผู้ใช้งาน, data gateways หรือ เมนูที่ผู้ใช้งานสามารถตอบโต้ได้



ภาพที่ 2.10 สัญลักษณ์ Boundary element

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

- เส้นชีวิตกับ control element เส้นชีวิตที่มีองค์ประกอบเป็น control element แสดงเอนทิตี้การควบคุมหรือผู้จัดการ มันจะจัดระเบียบและกำหนดเวลาการตอบโต้ และทำหน้าที่เป็นเป็นสื่อกลางระหว่างขอบเขตและเอนทิตี้

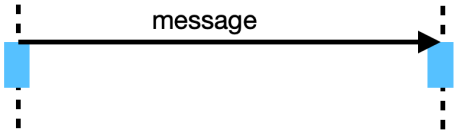


ภาพที่ 2.11 สัญลักษณ์แทน เส้นชีวิตกับ control element

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

- Activation Bars จะถูกวางอยู่บนเส้นชีวิตเพื่อแสดงการโต้ตอบระหว่างวัตถุ ฟังก์ชั่น หรือ module ความยาวของสี่เหลี่ยมจะแสดงระยะเวลาการโต้ตอบของวัตถุ หรือ จุดเริ่มต้นและจุดสิ้นสุดของแต่ละกิจกรรมของวัตถุนั้น
- การโต้ตอบระหว่าง 2 วัตถุ เกิดเมื่อวัตถุหนึ่งส่งข้อความไปให้อีกวัตถุหนึ่ง วัตถุที่ส่งข้อความเรียกว่า Message Caller ในขณะที่วัตถุที่รับข้อความเรียกว่า Message Receiver เมื่อมีแถบ Activation บนเส้นชีวิตของวัตถุ นั่นหมายความวัตถุนั้นมีการทำงานในขณะที่ส่งข้อความโต้ตอบกัน

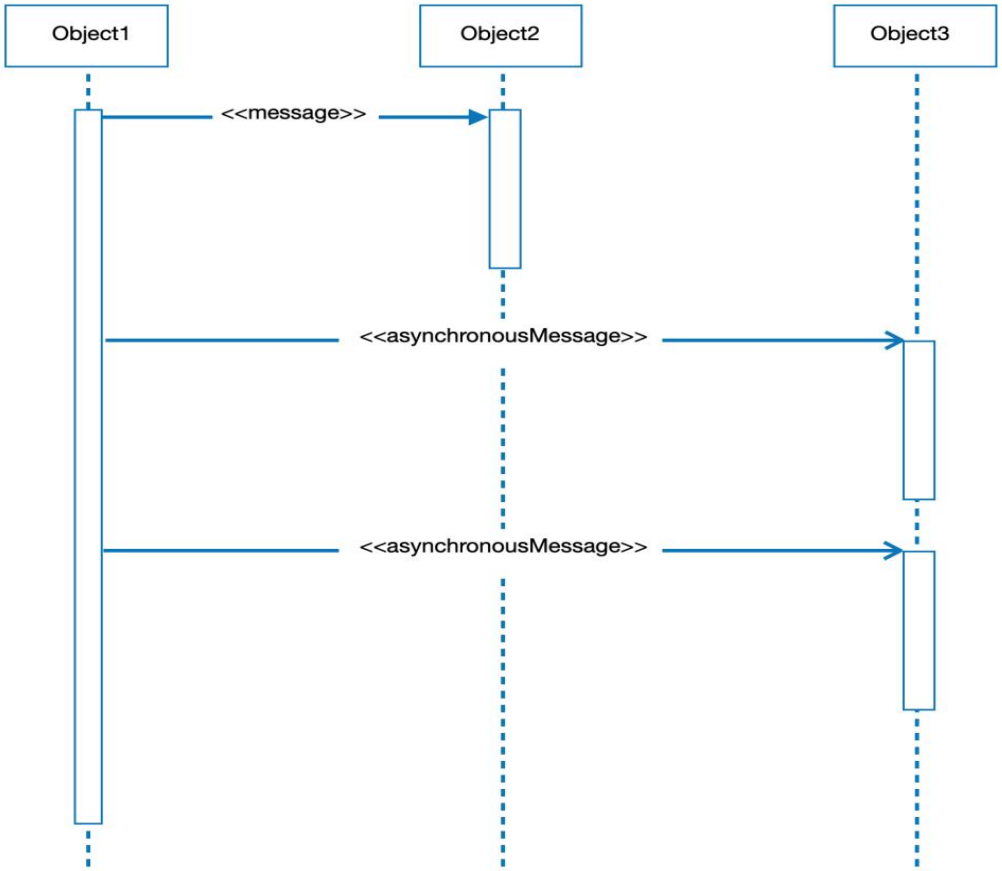
-



ภาพที่ 2.12 สัญลักษณ์แทน Message

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

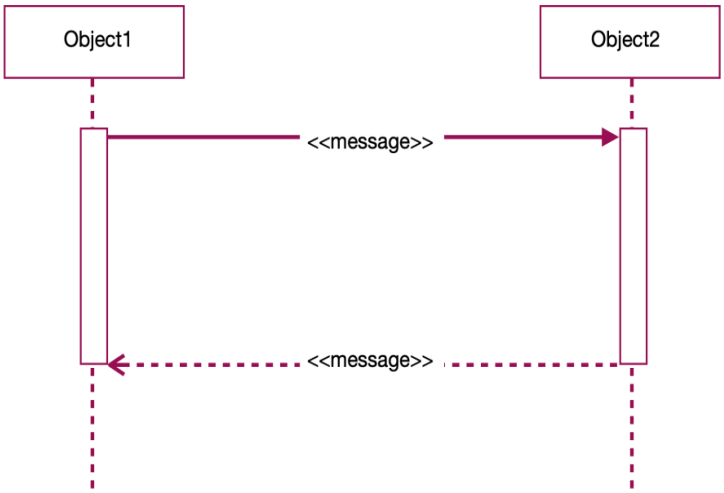
- Message Arrows ลูกศรจาก Message Caller ชี้ไปที่ Message Receiver ระบุข้อความที่เกิดขึ้นใน Sequence Model ข้อความสามารถไหลไปในทิศทางใดก็ได้ จากซ้ายไปขวา ขวาไปซ้าย หรือส่งข้อความกลับไปที่ตัวมันเองก็ได้ การอธิบายทิศทางการไหลของข้อความได้ด้วยหัวลูกศรที่ชี้ไปในทิศทางที่ต้องการ และพิจารณาว่าวัตถุใดเป็นตัวส่ง วัตถุใดเป็นตัวรับข้อความ
- รูปแบบของ Message
  - Synchronous message จะถูกใช้เมื่อวัตถุที่ส่งข้อความรอให้วัตถุที่รับปข้อความระมวลผลและส่ง return กลับมาก่อนที่จะส่งข้อความอันต่อไป หัวลูกศรที่ใช้จะเป็นลูกศรแบบทึบ
  - Asynchronous messageจะถูกใช้เมื่อวัตถุที่ส่งข้อความไม่รอให้วัตถุรับข้อความประมวลผลข้อความและส่งค่า return กลับมา แต่จะส่งข้อความต่อไปให้แก่วัตถุอื่นในระบบเลย หัวลูกศรที่แสดงในข้อความประเภทนี้เป็นหัวลูกศรเส้น



ภาพที่ 2.14 Asynchronous message

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

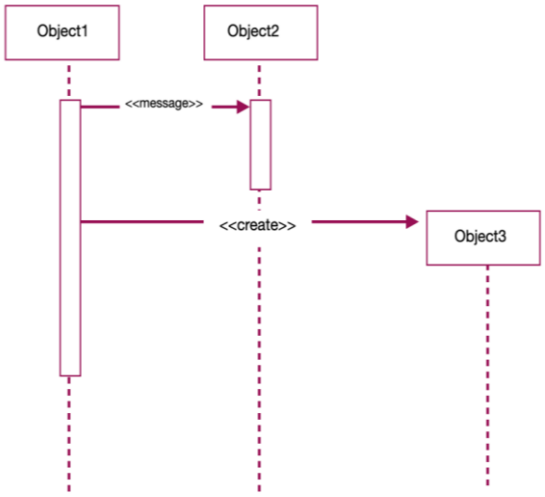
- return message ใช้เพื่อระบุว่าวัตถุรับข้อความประมวลผลข้อความเสร็จสิ้นแล้ว และกำลังส่งคืนการควบคุมไปยังวัตถุที่ทำหน้าที่ส่งข้อความ return message เป็นตัวเลือกที่จะเลือกให้มีหรือไม่มีก็ได้ สำหรับการส่งข้อความบนแถบ Activation ด้วย Synchronous message จะให้ความหมายโดยนัยว่ามี return message ด้วยแม้จะไม่ได้มีเส้น return message แสดงก็ตาม สามารถหลีกเลี่ยงการทำให้แผนภาพดูยุ่งเหยิงโดยการไม่ใช้ return message เมื่อไม่จำเป็น



ภาพที่ 2.15 return message

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

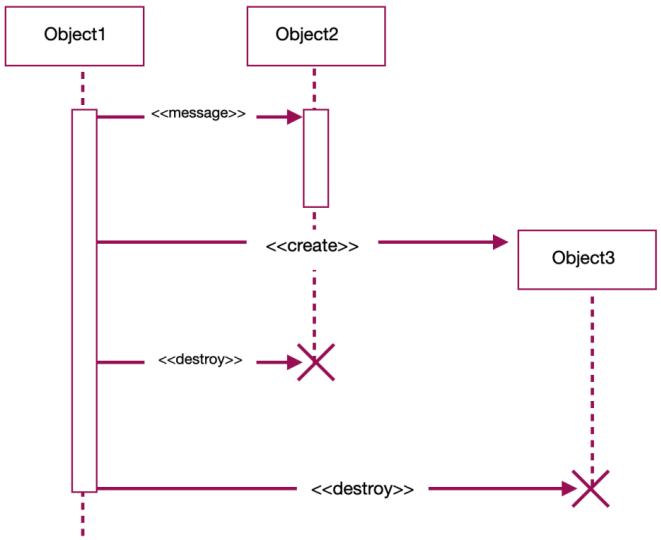
- Participant creation message วัตถุไม่จำเป็นต้องปรากฏอยู่ทุกช่วงเวลาในแผนภาพ วัตถุหรือผู้มีส่วนร่วมสามารถถูกสร้างขึ้นได้เมื่อข้อความถูกส่งออกไปหาวัตถุนั้น เครื่องหมายสำหรับวัตถุใหม่ที่ถูกสร้างคือกล่องสี่เหลี่ยมผืนผ้าที่มีชื่อของวัตถุอยู่ด้านใน เมื่อวัตถุใหม่ถูกสร้างขึ้น จะแสดงให้เห็นว่าวัตถุนั้นไม่ได้มีตัวตนมาก่อนจนกระทั่งมีการส่งข้อความจากวัตถุอื่นที่ปรากฏอยู่แล้วในแผนภาพ เมื่อสร้างวัตถุใหม่ขึ้นมาแล้ว หากวัตถุนั้นมีการทำงานใดทันทีที่ถูกสร้าง ก็ควรจะใส่แถบ Activation ด้านล่างวัตถุนั้นด้วย ดังภาพที่ 2.16



ภาพที่ 2.16 Participant creation message

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

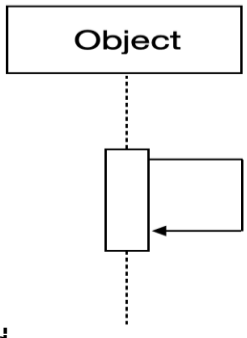
- Participant destruction message เมื่อวัตถุนั้นดำเนินงานมาจนถึงลำดับที่ไม่จำเป็นอีกต่อไป สามารถลบวัตถุนั้นออกจากแผนภาพได้ วิธีทำคือ การเพิ่มเครื่องหมาย X ที่ปลายเส้นชีวิตของวัตถุนั้น ดังภาพที่ 2.17



ภาพที่ 2.17 สัญลักษณ์ Participant destruction message

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

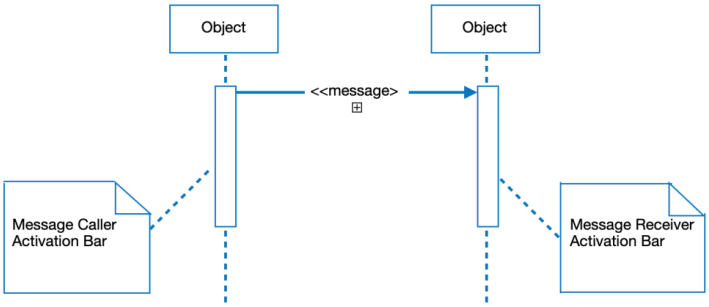
- Reflexive message เมื่อวัตถุส่งข้อความหาตัวเอง จะเรียกว่า reflexive message แสดงข้อความประเภทนี้โดยการใช้ message arrow ที่เริ่มจากจบที่เส้นชีวิตเดียวกันอย่างตัวอย่างด้านล่างนี้



ภาพที่ 2.18 Reflexive message

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

- Comment ในแผนภาพ UML ปกติแล้วจะอนุญาตให้มีเครื่องหมาย comment เพื่อแสดงความคิดเห็นได้ การแสดงความคิดเห็นจะอยู่ในกล่องสี่เหลี่ยมผืนผ้าที่พับมุมบนด้านนึง ความคิดเห็นสามารถถูกเชื่อมโยงกับวัตถุที่เกี่ยวข้องด้วยเส้นประได้ ดังภาพที่ 2.19



ภาพที่ 2.19 Comment

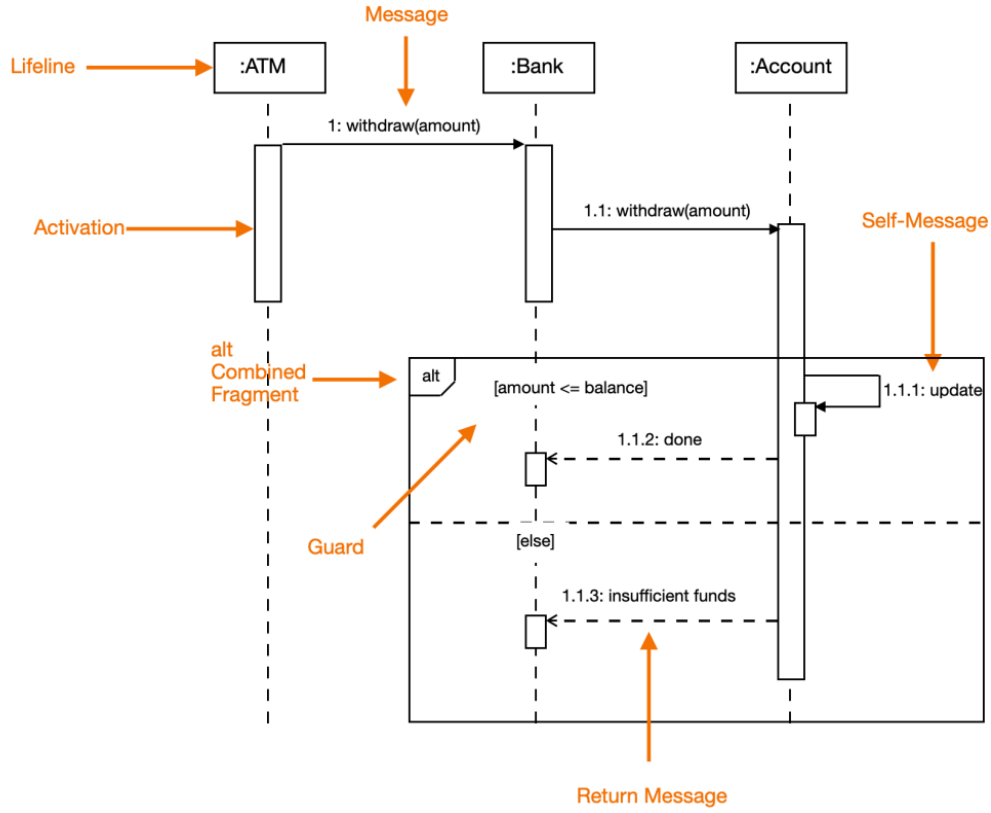
ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

- Sequence Fragment คือกล่องที่มีเครื่องหมายแสดง section การโต้ตอบระหว่างวัตถุ sequence diagram Fragment จะใช้ในแผนภาพที่มีการโต้ตอบของวัตถุที่ซับซ้อน เช่น การไหลแบบ alternative หรือ loop ที่มีวิธีที่เป็นโครงสร้างมากขึ้น ด้านบนซ้ายของกล่องจะแสดง fragment operator ระบุว่า fragment นี้เป็น fragment ประเภทใด

- Alternatives

  - Alternative combination fragment ถูกใช้เมื่อมีตัวเลือกให้เลือกตั้งแต่ 2 ตัวเลือกขึ้นไป ใช้ตรรกะแบบ “if then else”

  - Alternative fragment แสดงโดยเฟรมหรือกล่องสี่เหลี่ยมผืนผ้าขนาดใหญ่ มี ‘alt’ ซึ่งเป็น fragment operator ระบุเป็นชื่อกล่องในมุมซ้ายด้านบนของกล่อง ในการเลือกตัวเลือกที่มากกว่าหนึ่ง ข้างในกล่องสี่เหลี่ยมผืนผ้าจะถูกแบ่งออกเป็นส่วนๆ เรียกว่า interaction operand โดยใช้เส้นประเป็นตัวแบ่ง แต่ละ operand จะมี Guard ระบุไว้ที่ด้านบนของแต่ละ operand ดังภาพที่ 2.20



ภาพที่ 2.20 Alternatives

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

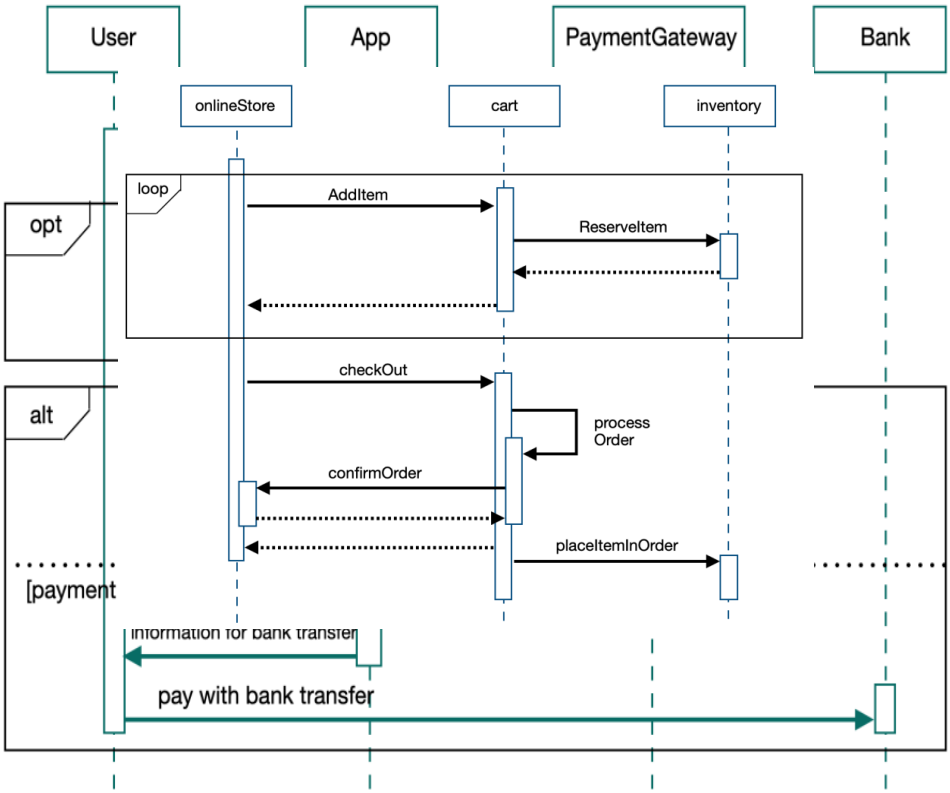
- Options

Option Combination fragment ถูกใช้เพื่อแสดงถึงลำดับเหตุการณ์ที่เกิดขึ้นภายใต้เงื่อนไขใดเงื่อนไขหนึ่งเท่านั้น ไม่เช่นนั้นเหตุการณ์นั้นจะไม่สามารถเกิดขึ้นได้ ใช้ตรรกะแบบ ‘if then’Option fragment สามารถแสดงได้โดยกล่องสี่เหลี่ยมผืนผ้าเช่นเดียวกับ Alternative fragment ส่วน fragment operator จะใช้ ‘opt’ ความแตกต่างระหว่าง Option fragment และ Alternative fragment คือ Option fragment จะไม่แบ่งออกเป็นแต่ละ Operand ส่วน Option Guard จะวางอยู่ด้านบนซ้ายในกล่อง ความแตกต่างระหว่าง alt และ opt fragment ใน Sequence diagram

alt ถูกใช้เพื่ออธิบายทางเลือกที่สามารถเลือกได้ในการไหลของเหตุการณ์ ทางเลือกเดียวเท่านั้นที่จะถูก executed

opt ถูกใช้เพื่อแสดงถึงขั้นตอนทางเลือกในการไหลของเหตุการณ์ หรือ workflow

เช่น ในร้านค้าออนไลน์ อาจใช้ opt เพื่อกำหนดว่าลูกค้าสามารถเลือกที่จะห่อของขวัญก็ได้หากต้องการและใช้ alt เพื่อกำหนดว่ามีวิธีจ่ายเงิน 2 วิธี คือ จ่ายผ่านบัตรเครดิตหรือโอนเงินผ่านธนาคารดังภาพที่ 2.21



ภาพที่ 2.21 Options

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

- Loops

Loop fragment ถูกใช้เพื่อแสดงลำดับเหตุการณ์ที่เกิดขึ้นซ้ำ โดยมี ‘loop’ เป็น fragment operation และ guard condition ระบุที่มุมด้านซ้ายของกล่อง

guard ใน loop fragment สามารถมีเงื่อนไขพิเศษเพิ่มได้อีกสองเงื่อนไข คือ

minimum iterations (minint = [the number])

maximum iterations (maxint = [the number])

หากใช้เงื่อนไข minimum iterations guard ลูปต้องแสดงผลไม่น้อยกว่าจำนวนที่ระบุใน minimum iteration แต่หากใช้เงื่อนไข maximum iteration guard ลูปก็ต้องแสดงผลไม่เกินจำนวนที่ระบุใน

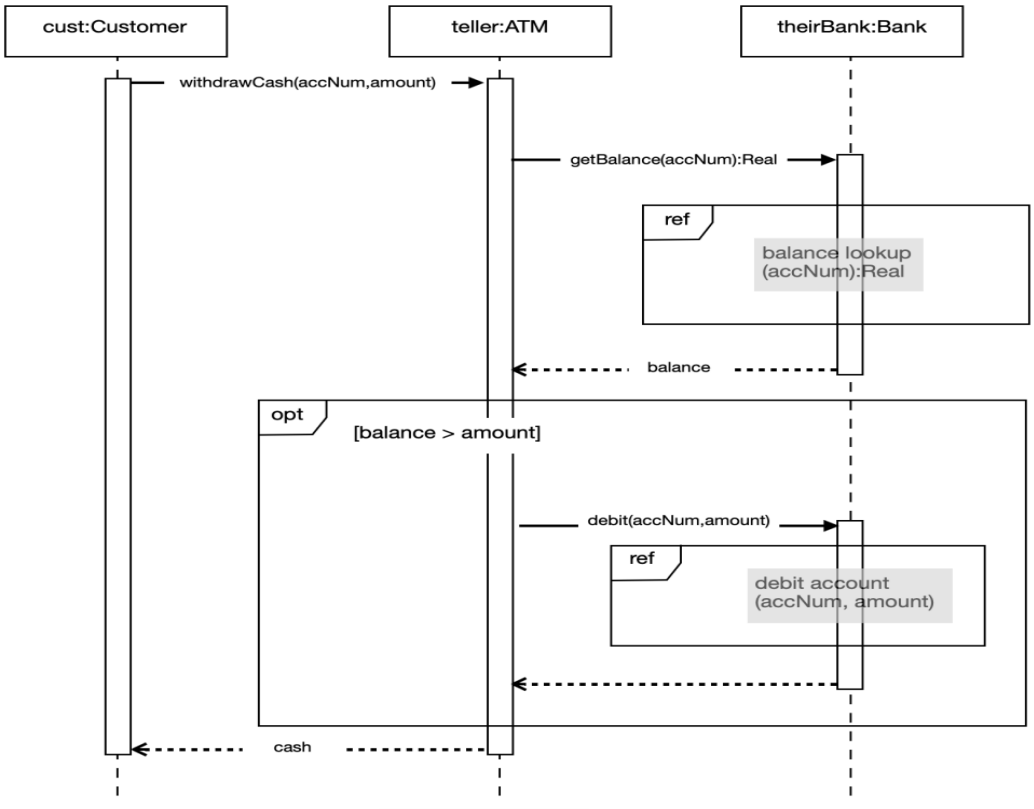
maximum iteration เช่นกัน loop fragment จะทำงานซ้ำจนกว่าค่าของ guard จะเป็น false ดังภาพที่ 2.22

ภาพที่ 2.22 Loops

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

-Reference fragment

สามารถใช้ Reference fragment เพื่อจัดการกับขนาดของ sequence diagram ที่มีขนาดใหญ่ ref fragment จะช่วยให้ทำบางส่วนของ sequence diagram มาใช้กับส่วนอื่นๆ ได้ หรือเรียกว่า สามารถอ้างถึงส่วนอื่นๆของ sequence diagram ได้โดยใช้ ref fragment การแสดงถึง ref fragment จะต้องกำหนด 'ref' เป็นชื่อกล่อง หรือ fragment operation และชื่อของ sequence diagram ที่ถูกอ้างถึงข้างในกล่อง ดังภาพที่ 2.23

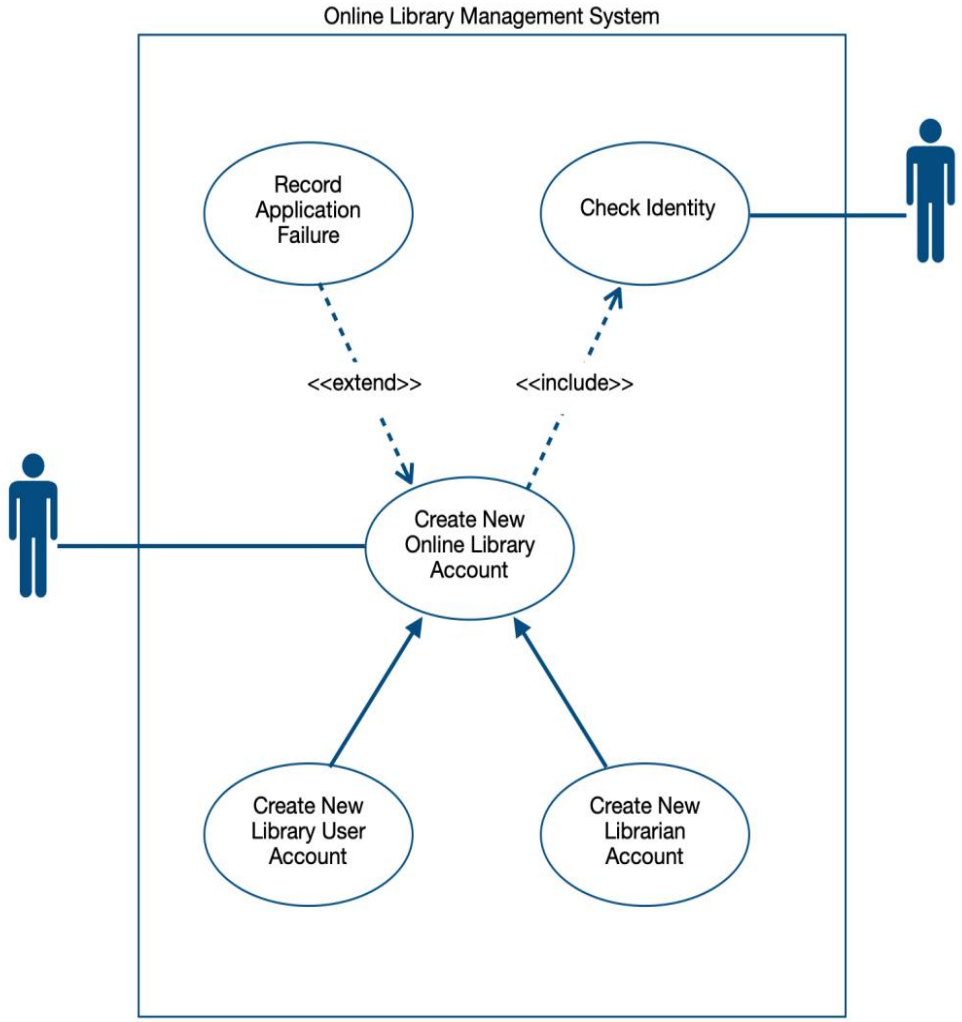


ภาพที่ 2.23 Reference fragment

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

วิธีวาด Sequence Diagram

เนื่องจาก sequence diagram แสดงลำดับการไหลของเหตุการณ์ในหนึ่ง UseCase การไหลไปของข้อความหรือ Message ใน Sequence Diagram จะเป็นไปตามทิศทางของ Use Case นั้นๆดังนั้นก่อนวาด Sequence Diagram จะต้องวาด Use Case ก่อน หรือ ต้องมีการตัดสินใจก่อนว่าการโต้ตอบหรือเหตุการณ์ใดบ้างที่จะระบุถึง ดังในตัวอย่างภาพที่ 2.24 Use Case ของ Online Library Management System เลือกว่าจะวาด Sequence Diagram ที่ขั้นตอนย่อยใด ในตัวอย่างจะเลือกขั้นตอน ‘Create New User Account’ ของ ‘Create New Online Library Account’



ภาพที่ 2.24 ตัวอย่าง Online Library Management system

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

ก่อนที่จะวาดไดอะแกรม สิ่งสำคัญคือต้องระบุวัตถุ (Object) หรือ Actor ที่มีเกี่ยวข้องสำหรับการสร้าง new user account ซึ่งจะลิสออกมาได้ดังนี้

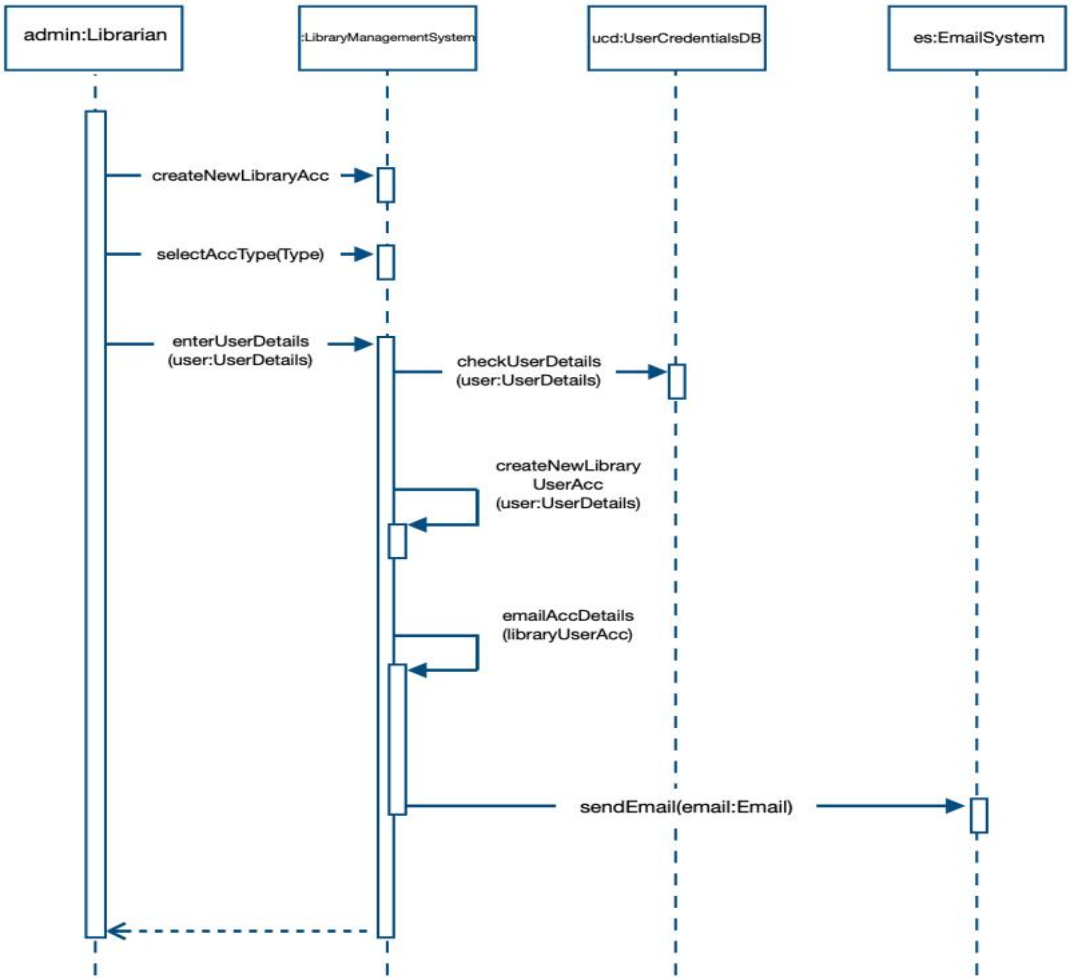
- Librarian
- Online Library Management system
- User credential database
- Email system

เมื่อระบุวัตถุหรือ Actor ที่เกี่ยวข้องได้แล้ว สิ่งต่อมาคือต้องเขียนรายละเอียดอธิบายว่า Use Case นี้มีกระบวนการใดบ้าง หากสามารถอธิบายได้ ก็จะสามารถทราบได้ว่าแต่ละส่วนมีการตอบโต้หรือ Interaction อย่างไร จากนั้นการตอบโต้เหล่านี้จะถูกจับเอาไปใส่ไว้ใน Sequence Diagram

ตัวอย่างเหตุการณ์ที่เกิดขึ้นใน Use Case ‘Create New Library User Account’

- Librarian ร้องขอระบบให้สร้าง online library account ใหม่
- Librarian เลือกประเภทของ user account
- Librarian ใส่รายละเอียดของผู้ใช้งาน
- รายละเอียดของผู้ใช้งานจะถูกตรวจสอบโดย User Credential Database (หรือฐานข้อมูลนั่นแหละ)
- บัญชีผู้ใช้ใหม่ถูกสร้าง
- รวมข้อมูลทั้งหมดสำหรับบัญชีใหม่ และส่งอีเมลให้เจ้าของบัญชี

แต่ละลำดับเหตุการณ์ดังกล่าว ทำให้สามารถระบุได้ว่าข้อความใดจะเป็นข้อความที่ใช้ในการโต้ตอบระหว่างวัตถุใน Sequence Diagram



ภาพที่ 2.25 ตัวอย่าง Sequence Diagram

ที่มา: https://devjourneys.com/2020/09/13/sequence-diagram/

2.3.5 Class Diagram คือแผนภาพที่ใช้แสดง Class และความสัมพันธ์ในแง่ต่างๆ (Relation) ระหว่าง Class เหล่านั้น ซึ่งความสัมพันธ์ที่กล่าวถึงใน Class Diagram นี่ถือเป็นความสัมพันธ์เชิงสถิตย์ (Static Relationship) หมายถึงความสัมพันธ์ที่มีอยู่แล้วเป็นปกติในระหว่าง Class ต่างๆ ไม่ใช่ความสัมพันธ์ที่เกิดขึ้นเนื่องจากกิจกรรมต่างๆ ซึ่งเรียกว่าความสัมพันธ์เชิงกิจกรรม(Dynamic Relationship) lสิ่งที่ปรากฏใน Class Diagram นั้นประกอบด้วยกลุ่มของ Class และกลุ่มของ Relationship โดยสัญลักษณ์ที่ใช้ในการแสดง Class นั้นจะแทนด้วยสี่เหลี่ยมแบ่งออกเป็น 3 ส่วน โดยแต่ละส่วนนั้น (จากบนลงล่าง) จะใช้ในการแสดง ชื่อของ Class,Attribute และฟังค์ชั่นต่างๆ ตามลำดับ

1)ความสัมพันธ์ระหว่าง Class

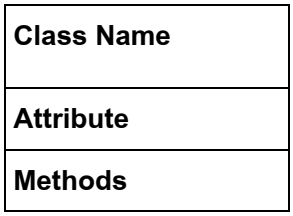
ความสัมพันธ์ระหว่าง Class (Relationship) คือ ความสัมพันธ์ระหว่าง Class ที่ทำงานร่วมกัน สามารถจำแนกได้ดังนี้

- ความสัมพันธ์แบบพึ่งพา (Dependency) เช่น “Class ลูกค้า” กับ “Class ขายสินค้า” กล่าวได้ว่า “Class ขายสินค้า” ขึ้นอยู่กับ “Class ลูกค้า” เพราะเมื่อลูกค้ามีการเปลี่ยนแปลงคำสั่งซื้อหรือคำสั่งผลิตรายการขายก็จะต้องถูกเปลี่ยนแปลง (Update) ตามลูกค้า

- ความสัมพันธ์แบบสืบทอดคุณสมบัติ (Inheritance) เช่น “Class แม่” (super class) สืบทอดคุณลักษณะเฉพาะที่ตนมีอยู่ไปยัง “Class ลูก” (sub class)

- ความสัมพันธ์แบบร่วมกัน (Association) คือความสัมพันธ์ที่เกี่ยวเนื่องมีความสัมพันธ์ซึ่งกันและกันเช่น “Class นักเรียน” สัมพันธ์กับ “Class รายวิชา” ในเรื่องของการลงทะเบียนเรียน

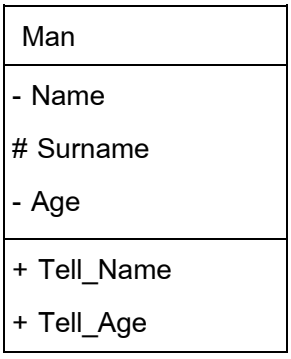
2)องค์ประกอบของ Class diagram



ภาพที่ 2.26 องค์ประกอบของ Class diagram

ที่มา: https://bloger-classdiagram.blogspot.com/p/class-diagram.html

3)สัญลักษณ์ของ Class diagram



ภาพที่ 2.27 สัญลักษณ์ของ Class diagram

ที่มา: https://bloger-classdiagram.blogspot.com/p/class-diagram.html

ในการเขียนสัญลักษณ์แทน Class สิ่งที่ต้องคำนึงถึงอีกสิ่งหนึ่งคือระดับการเข้าถึงเรียกสัญลักษณ์ที่ใช้แทนการเข้าถึงนี้ว่า Visibility แบ่งออกได้เป็น 3 ประเภท ได้แก่

3.1 Private เขียนแทนด้วยสัญลักษณ์ - หมายถึง Attribute หรือ ฟังก์ชันที่ไม่สามารถมองเห็นได้จากภายนอก แต่สามารถมองเห็นได้จากภายในตัวของ Class เองเท่านั้น

3.2 Protect เขียนแทนด้วยสัญลักษณ์ # หมายถึง Attribute หรือ ฟังก์ชันที่สงวนไว้สำหรับการทำ Inheritance โดยเฉพาะ Attribute หรือ ฟังก์ชันเหล่านี้ จะเป็นของ Super class เมื่อทำการ Inheritance แล้ว Attribute หรือ ฟังก์ชัน ที่มี Visibility แบบ Protect จะกลายไปเป็น Private Attribute/ฟังก์ชัน หรือ Protected ขึ้นอยู่กับภาษา Programming ที่นำไปใช้

3.3 Public เขียนแทนด้วยสัญลักษณ์ + หมายถึง Attribute หรือ ฟังก์ชัน ที่สามารถมองเห็นได้จากภายนอก และสามารถเข้าไปเปลี่ยนค่า อ่านค่าหรือเรียกใช้งาน Attribute หรือฟังก์ชัน นั้นได้ทันทีโดยอิสระจากภายนอก (โดยทั่วไปแล้ว Visibility แบบ Public มักจะใช้กับฟังก์ชันมากกว่า Attribute)

4)ความสัมพันธ์ระหว่าง Object ประกอบด้วย

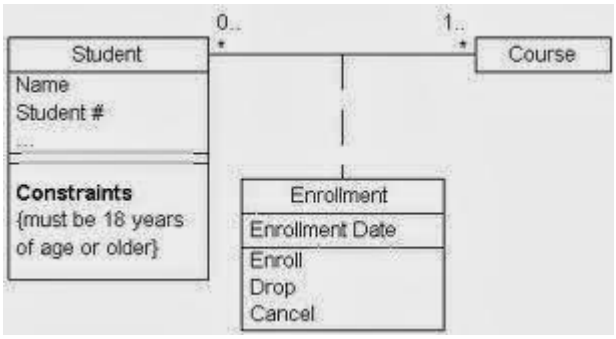
4.1 Association

4.2 Aggregation

4.3 Composition

4.4 Generalization

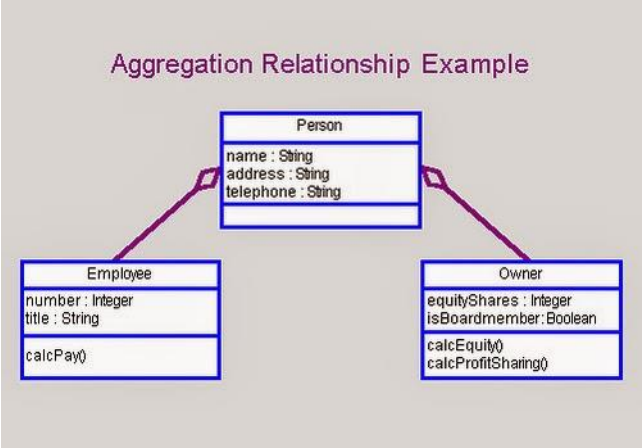
4.1Association เป็นความสัมพันธ์ระหว่าง Object หรือ Class แบบ 2 ทิศทาง



ภาพที่ 2.28 Aggregation

ที่มา: https://bloger-classdiagram.blogspot.com/p/class-diagram.html

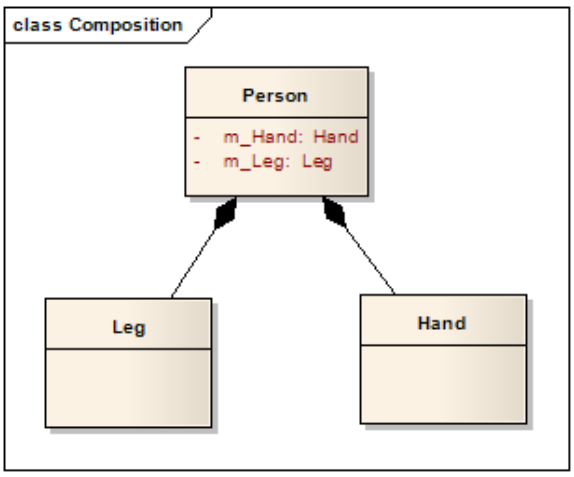
4.2 Aggregation เป็นความสัมพันธ์ระหว่าง Object หรือ Class แบบ “Whole-Part” หรือ “is part of” โดยจะมี Class ที่ใหญ่ที่สุดที่เป็น Object หลัก และมี Class อื่นเป็นส่วนประกอบ



ภาพที่ 2.29 Aggregation Relationship Example

ที่มา: https://bloger-classdiagram.blogspot.com/p/class-diagram.html

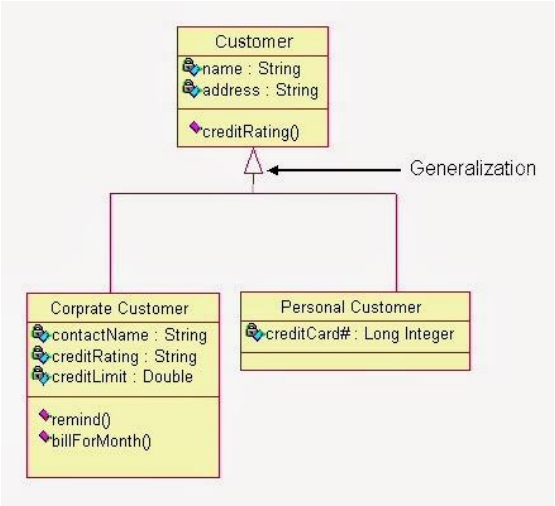
4.3 Composition เป็นความสัมพันธ์ระหว่าง Object หรือ Class แบบขึ้นต่อกันและมีความเกี่ยวข้องกันเสมอ โดยจะมี Class ซึ่งเป็นองค์ประกอบของ Class อื่นที่ใหญ่กว่า เมื่อ Class ที่ใหญ่กว่าถูกทำลาย Class ที่เป็นองค์ประกอบก็จะถูกทำลายไปด้วย



ภาพที่ 2.30 Composition

ที่มา: https://bloger-classdiagram.blogspot.com/p/class-diagram.html

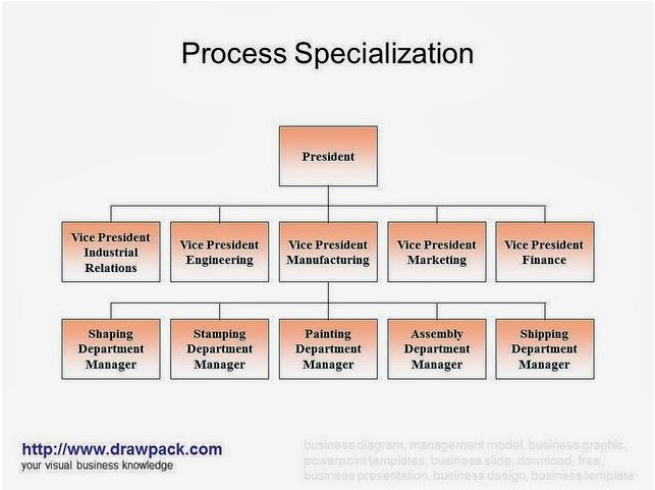
4.4 Generalization เป็นความสัมพันธ์ระหว่าง Object หรือ Class ในลักษณะของการสืบทอดคุณสมบัติจาก Class หนึ่ง (Super class) ไปยังอีก Class หนึ่ง (Subclass)



ภาพที่ 2.31 Generalization

ที่มา: https://bloger-classdiagram.blogspot.com/p/class-diagram.html

4.5 Specialization คือกระบวนการที่ตรงกันข้ามกับ กระบวนการ Generalization Abstraction กล่าวคือ ถ้าต้องการสร้าง Class ใหม่ โดยอาศัย Concept ของ Class เก่าบางส่วน



ภาพที่ 2.32 Specialization

ที่มา: https://bloger-classdiagram.blogspot.com/p/class-diagram.html

5. หลักในการสร้าง Class Diagram

สิ่งที่ต้องคำนึงถึงสำหรับการจำลอง Class และ Relationship ต่างๆใน Class Diagram ใน OOA คือ

- Class ทั้งหมดที่ต้องมีอยู่ในระบบหรือใน Real World
- ต้องมีอยู่ครบ ไม่ขาดหาย
- ไม่มากเกินความจำเป็น

5.1) กำหนดกรอบของ Problem Domain ให้ชัดเจน

- ให้ยึดถือ Problem Domain นี้เป็นบรรทัดฐานในการวิเคราะห์ระบบ
- เขียน Use Case Diagram ของ Problem Domain
- พิจารณาว่า ในแต่ละ Use Case จะมี Objects อะไรอยู่บ้าง

5.2) พิจารณาหา Objects ที่สามารถจับต้องได้เห็นได้สัมผัสได้ ซึ่งเรียกว่า Tangible Objects

5.3) พิจารณาหา Objects ที่ไม่สามารถจับต้องได้ซึ่งเรียกว่า Intangible Objects

5.4) ใช้ Classification Abstraction เพื่อแยกแยะและสร้าง Class จาก Objects ที่มีอยู่

- พยายามหา Attributes และ Functions ของ Class เท่าที่จะหาได้
- วาด Class ทั้งหมดที่ได้ ลงใน Class Diagram

5.5) หา Aggregation Abstraction (โดยพิจารณาการเป็นส่วนประกอบ)

- เพิ่มเติมสัญลักษณ์
- ใส่ Cardinality ให้ถูกต้อง

5.6) ใช้ Generalization มาพิจารณา

- เพิ่มเติมสัญลักษณ์
- อาจเกิด Class ใหม่เพื่อเป็น Generalized Class ได้

5.7) ใช้ Association มาพิจารณา

- เพิ่มเติมสัญลักษณ์
- พิจารณาประเภทของความสัมพันธ์และ Cardinality ให้ถูกต้อง

5.8) พิจารณา Class Diagram ว่ามี Class หรือ กลุ่มของ Class ที่ไม่มีความสัมพันธ์กับ Class อื่นๆ หรือไม่

- อาจจะพบ Class ที่ไม่จำเป็นสำหรับระบบ
- อาจจะขาด Class อื่นๆที่จำเป็นในระบบ

2.2.6 Activity Diagram คือแผนภาพกิจกรรมใช้อธิบายกิจกรรมที่เกิดขึ้นในลักษณะกระแสการไหลของการทำงาน(Workflow) Activity Diagram จะมีลักษณะเดียวกับ Flowchart (แสดงขั้นตอนการทำงานของระบบ) โดยขั้นตอนในการทำงานแต่ละขั้นจะเรียกว่า Activity

1)การใช้Activity Diagram

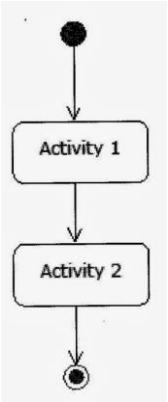
- อธิบาย กระแสการไหลของการทำงาน (Workflow)
- แสดงขั้นตอนการทำงานของระบบ

Activity อาจเป็นการทำงานต่างๆ ได้แก่

- การคำนวณผลลัพธ์บางอย่าง
- การเปลี่ยนแปลงสถานะ (State) ของระบบ
- การส่งค่ากลับคืน
- การส่งสัญญาณ
- การเรียกใช้ Operation (Method) อื่นๆ เพื่อทำงาน
- การสร้าง หรือ ทำลายวัตถุ

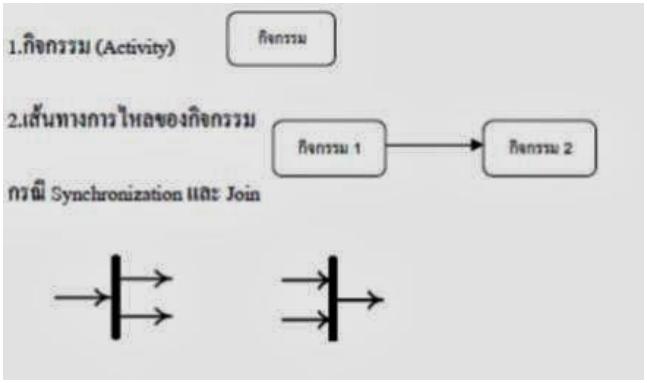
2)ลักษณะของ Activity Diagram

- Activity Diagram จะต้องมีจุดเริ่มต้นกับจุดสิ้นสุด และในระหว่างจุดเริ่มต้นกับจุดสิ้นสุดจะมีขั้นตอนหรือ Activity ต่างๆ ของระบบ
- ปกติแล้วจะเขียน Activity Diagram โดยอ่านจากด้านบนลงล่าง (ดังภาพที่ 2.33)



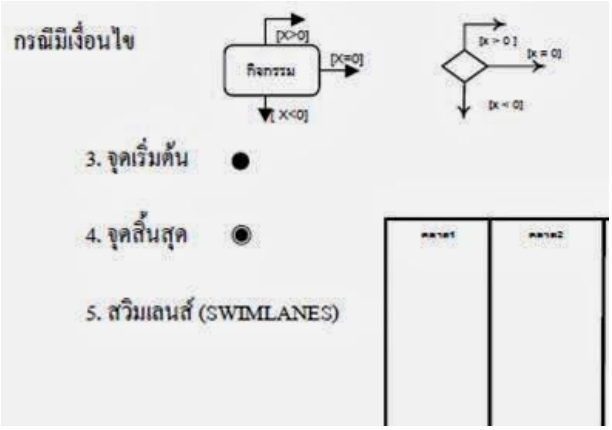
ภาพที่ 2.33 Activity Diagram แผนภาพแสดงการไหลของข้อมูล

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

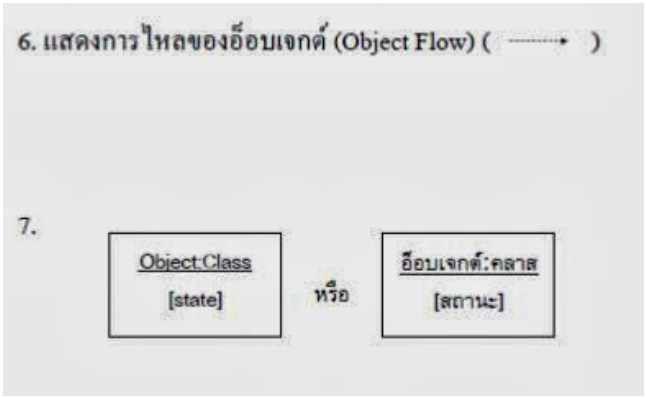


ภาพที่ 2.34 สัญลักษณ์ที่ใช้ใน Activity Diagram

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html



ภาพที่ 2.35 สัญลักษณ์ที่ใช้ใน Activity Diagram



ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

ภาพที่ 2.36 สัญลักษณ์ที่ใช้ใน Activity Diagram

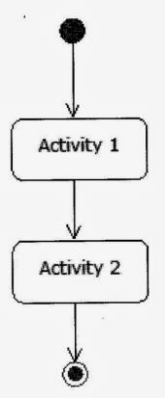
ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

3)ขั้นตอนในการเขียน Activity Diagram

- พิจารณากิจกรรมต่างๆ ที่ที่ได้จากผลการวิเคราะห์ที่ควรอธิบาย
- พิจารณากิจกรรมย่อยที่เกิดขึ้น เงื่อนไขหรือกรณีต่างๆ ที่เกิดขึ้นเมื่อเป็นไปตามเงื่อนไข
- เรียงลำดับกิจกรรมที่เกิดก่อนหลัง
- เขียนกิจกรรมย่อยด้วยสัญลักษณ์แสดงกิจกรรม
- เขียนจุดเริ่มต้น
-เขียนจุดสิ้นสุด

4)รูปแบบการใช้ Activity Diagram

4.1. แบบทั่วไป

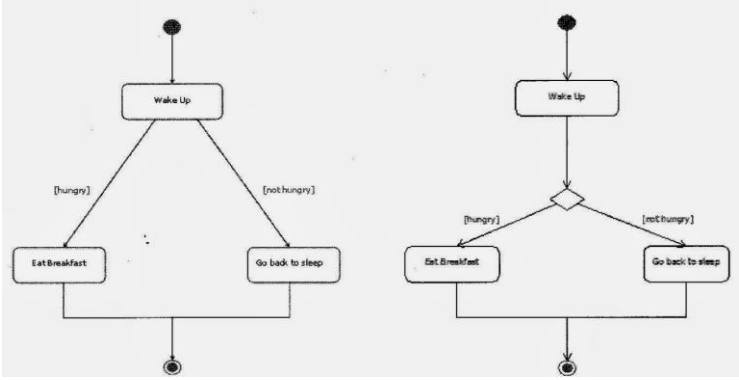


ภาพที่ 2.37 Activity Diagram แบบทั่วไป

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

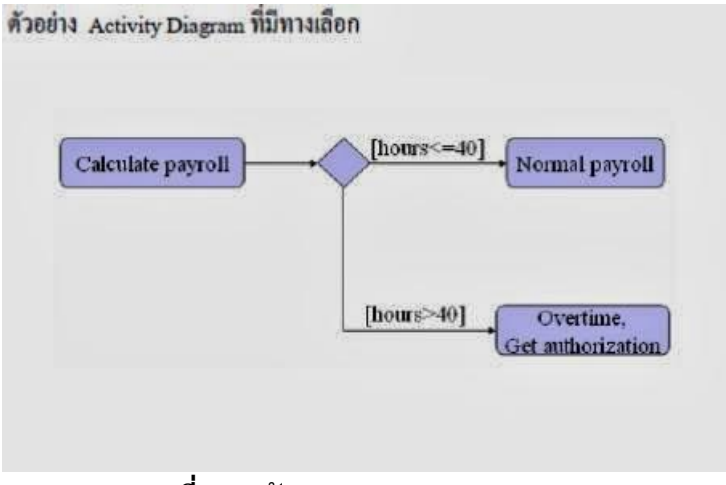
4.2) แบบมีทางเลือกให้ตัดสินใจ การกำหนดทางเลือกให้แก่ Activity Diagram ทำได้ 2 วิธี

- ลากลูกศรของแต่ละทางเลือกไปยัง Activity ผลลัพธ์ของทางเลือกโดยตรง
- ลากลูกศรของแต่ละทางเลือกผ่านรูปสี่เหลี่ยมขนมเปียกปูนก่อน



ภาพที่ 2.38 Activity Diagram แบบมีทางเลือกให้ตัดสินใจ

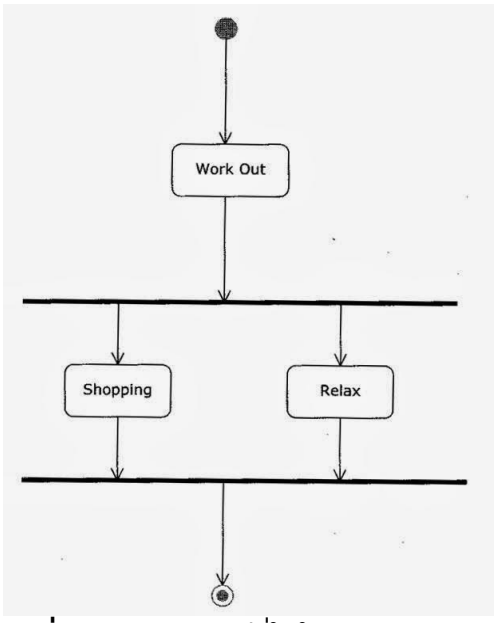
ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html



ภาพที่ 2.39ตัวอย่างActivity Diagram

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

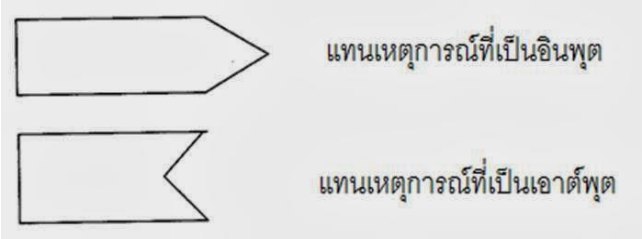
4.3) แบบมีการทำงานพร้อมๆกันหลายงาน ให้ใช้เส้นตรงแนวนอนเส้นหนาที่เรียกว่า Swim Lanes มาเป็นสัญลักษณ์ที่ใช้จัดกลุ่มงานที่มีการทำงานพร้อมๆกันหรือการทำกิจกรรมในลักษณะคู่ขนาน



ภาพที่ 2.40 สัญลักษณ์ที่ใช้ใน Activity Diagram

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

4.4) แบบการส่งสัญญาณ ในกระบวนการทำงาน อาจเป็นไปได้ว่าจะมีการส่งสัญญาณบางอย่างในระหว่างการทำงาน เมื่อเกิดการส่ง - รับ สัญญาณ เรียกว่าเกิด Activity ได้เช่นกัน

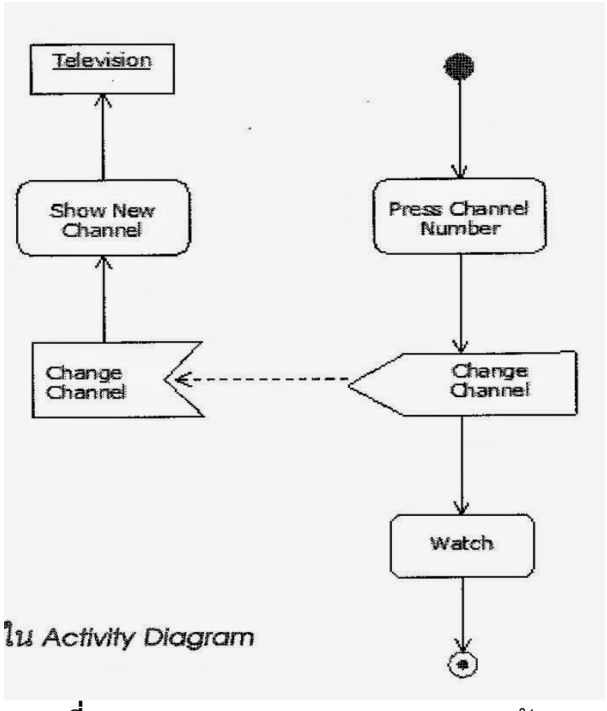


ภาพที่ 2.41 Activity Diagram แบบการส่งสัญญาณ

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

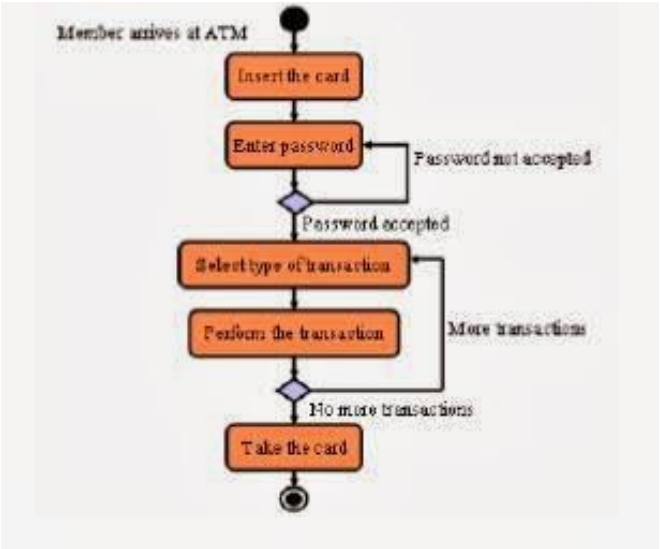
5)การใช้ Activity Diagram แสดงการส่งสัญญาณ

ตัวอย่างการใช้ Activity Diagram แสดงการส่งสัญญาณที่เป็นการแสดงความสัมพันธ์ระหว่าง Activity ทั้งสอง ภายใต้เหตุการณ์เดียวกันโดยระบบที่สนใจ คือ การกดปุ่มรีโมทคอนโทรลเพื่อเปลี่ยนช่องโทรทัศน์



ภาพที่ 2.42 Activity Diagram แสดงการส่งสัญญาณ

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html



ภาพที่ 2.43 ตัวอย่าง Activity Diagram ของ ATM

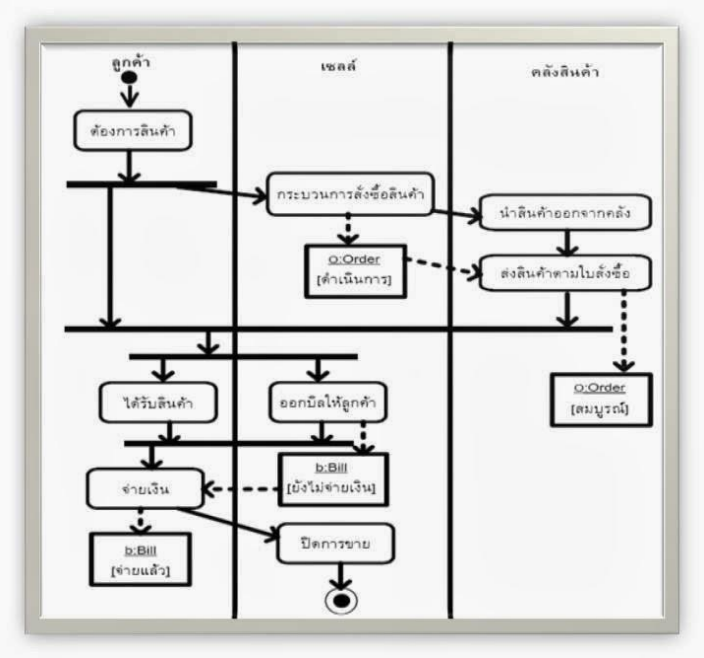
ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

โดยแบ่งการทำงานให้เป็นสัดส่วนด้วย Swim lanes

- คุณลักษณะอีกอย่างหนึ่ง คือ ความสามารถแสดงให้เห็นได้ว่าใครเป็นผู้มีหน้าที่รับผิดชอบในแต่ละ Activity ในกระบวนการทำงานหนึ่งๆ

- หลักการของการแสดงหน้าที่ จำทำโดยการแบ่งกลุ่มของการรับผิดชอบเป็นกลุ่มๆ ซึ่งเปรียบเหมือนการแข่งว่ายน้ำ เรียกกลไกนี้ว่า Swim lanes

- ในแต่ละ Swim lanes จะมีการกำหนดชื่อกำกับเอาไว้ เช่นกระบวนการของการสั่งซื้อสินค้า อาจแบ่งกลุ่มของคนที่มีส่วนเกี่ยวข้องเป็น 3 ส่วน ได้แก่ ลูกค้า, ฝ่ายขาย, คลังสินค้า



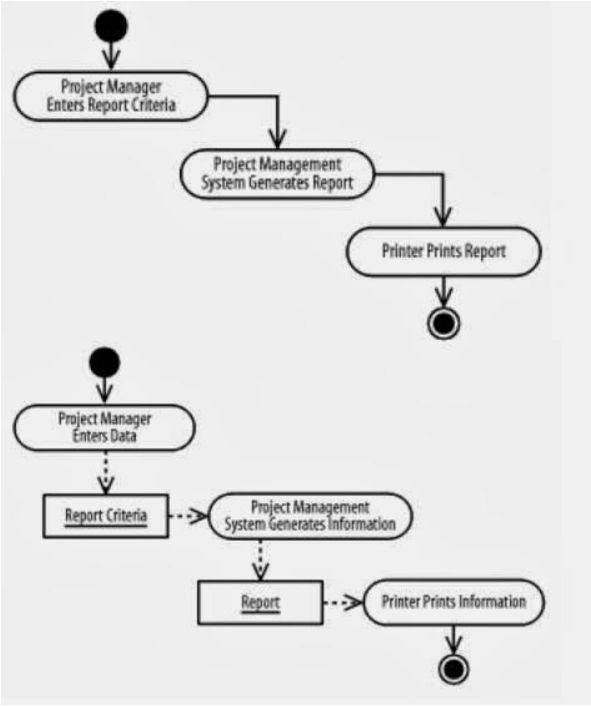
ภาพที่ 2.44 แบ่งการทำงานให้เป็ยสัดส่วนด้วย Swim lanes

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

- Activity Diagram : Transitions

(ภาพบน) Control-flow transitions ใช้เพื่อเรียงลำดับของการเกิด Activity โดยจะเริ่มทำ Action ถัดไปก็ต่อเมื่อ Action ก่อนหน้าทำงานเสร็จเรียบร้อย

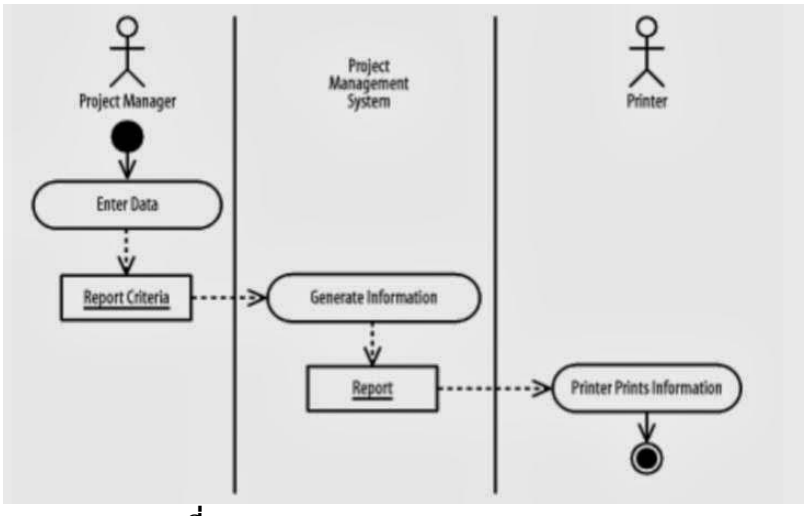
(ภาพล่าง) Object-flow transitions ใช้เพื่อระบุ Input หรือ Output ที่เกิดขึ้นจากการทำงานใน Action นั้นโดย Input /Output จะแสดงเป็น Object



ภาพที่ 2.45 Activity Diagram : Transitions

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

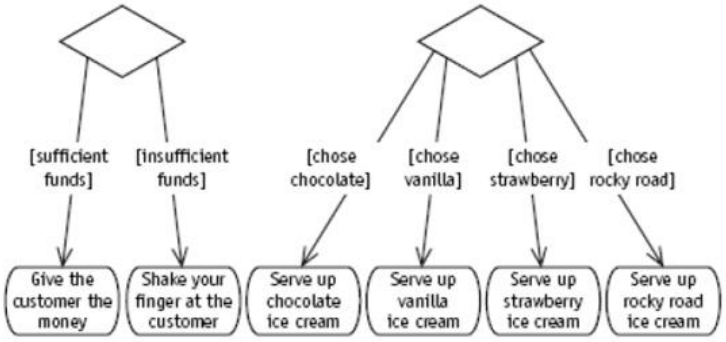
- Activity Diagram : Swim lanes กิจกรรมในการทำงาน สามารถแบ่งหน่วยงานที่รับผิดชอบได้ด้วย Swim lanes



ภาพที่ 2.46 Activity Diagram : Swim lanes

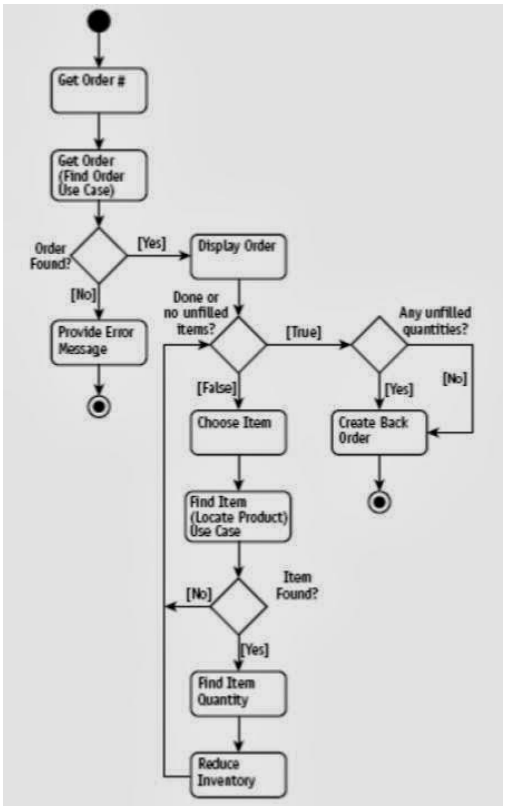
ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

- Activity Diagram : Decision แทนด้วยสัญลักษณ์สี่เหลี่ยมข้าวหลามตัด พร้อมระบุเงื่อนไขของแต่ละกรณีเอาไว้



ภาพที่ 2.47 Activity Diagram : Decision

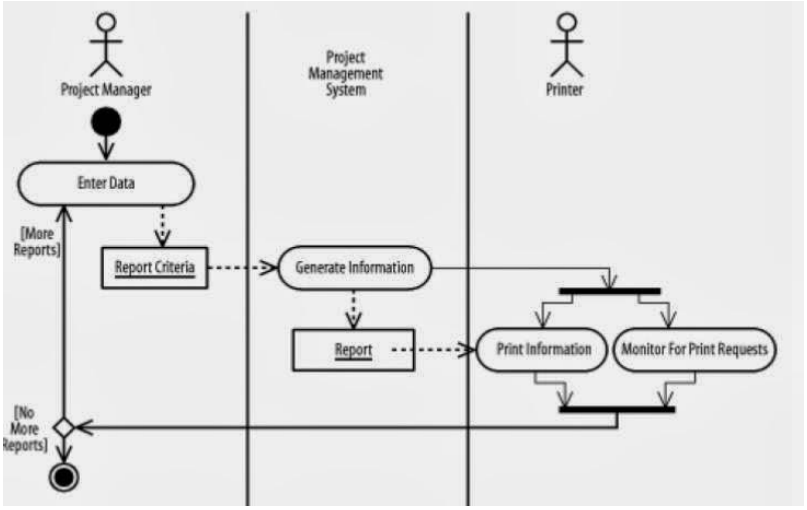
ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html



ภาพที่ 2.48 ตัวอย่างActivity Diagram : Decision

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

- Activity Diagram : Concurrency เป็นการแสดงการทำงานที่สามารถทำกิจกรรมใด พร้อมๆกันได้



ภาพที่ 2.47 Activity Diagram : Concurrency

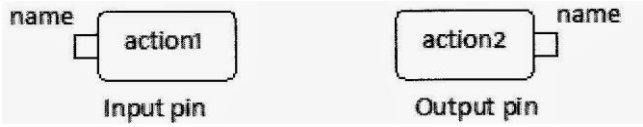
ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

6)การระบุส่วนของข้อมูลให้แก่กิจกรรม

- โดยปกติกิจกรรมการทำงานมักเกี่ยวข้องกับข้อมูล เช่น การสร้าง ลบ หรือ การโยกย้ายข้อมูลเป็นต้น

- Activity Diagram จึงมีส่วนที่เรียกว่า Input Pin และ Output Pin สำหรับการแสดงส่วนที่เป็นข้อมูล Input และ Output

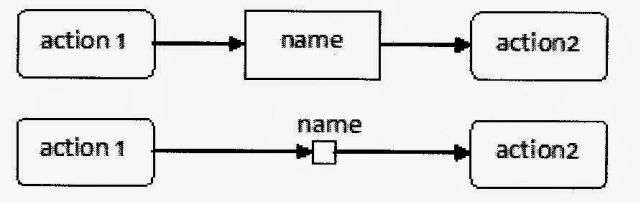
- Pin จะมีลักษณะเป็นรูปสี่เหลี่ยมเล็กๆ ที่วางไว่ก่อนหรือหลังรูปสี่เหลี่ยม มุมโค้งของ Activity เพื่อแสดง Input และ Output ของกิจกรรม



ภาพที่ 2.48 การระบุส่วนของข้อมูลให้แก่กิจกรรม

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

- ในกรณีที่ข้อมูล Input จากกิจกรรมหนึ่งเป็น Output ของอีกกิจกรรมหนึ่ง การแสดงความสัมพันธ์ระหว่างกิจกรรมดังกล่าว สามารถเขียนได้ 2 แบบ ดังนี้

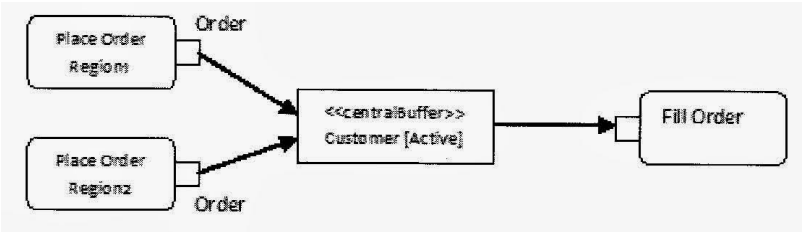


ภาพที่ 2.49 การแสดงความสัมพันธ์ระหว่างกิจกรรม 2 แบบ

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

7) การจัดระเบียบข้อมูล

- ในบางครั้งข้อมูลอาจมาจากต้นทางหลายแห่ง หรือมาจากต้นทางเดียวกัน แต่มีการส่งข้อมูลมาเรื่อยๆ อย่างต่อเนื่อง ดังนั้นจึงต้องจัดเรียงข้อมูลเหล่านั้นในระหว่างกระบวนการทำงานหนึ่งๆ โดยใช้สัญลักษณ์ <<centralBuffer>>

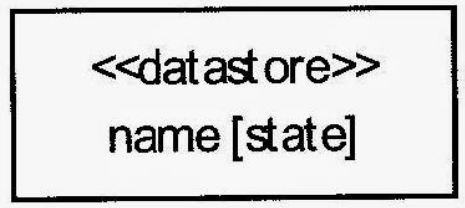


ภาพที่ 2.50 การจัดระเบียบข้อมูล

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

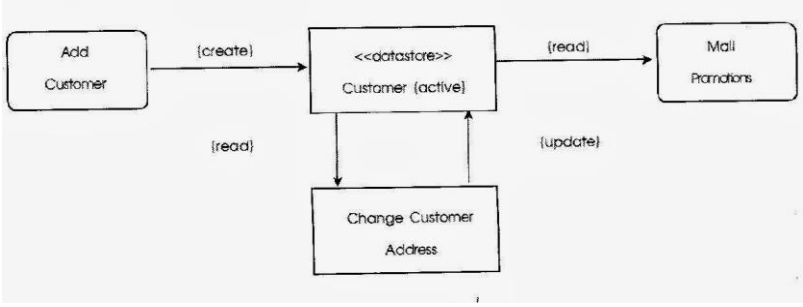
8)การสร้างที่พัก/เก็บข้อมูล

- ระหว่างกิจกรรมหนึ่งๆ อาจมีการสร้าง ลบ โยกย้ายข้อมูล รวมถึงมีการเรียกใช้ข้อมูลเพื่อ ประมวลผลบางอย่าง บางครั้งข้อมูลเหล่านี้จะเกิดขึ้นชั่วคราวในระหว่างการทางาน เมื่อการประมวลผลเสร็จสิ้น ข้อมูลนั้นจะหายไป หรือในบางครั้งอาจต้องเก็บข้อมูลดังกล่าวไว้เพื่อการทากิจกรรมอื่นๆ ต่อไป ไม่ว่ากรณีใดก็ตามต้องมีที่สาหรับพัก/เก็บข้อมูลนั้นเอาไว้ที่ Data Store โดยใน Activity Diagram จะใช้สัญลักษณ์ <<datastore>>



ภาพที่ 2.51 จะใช้สัญลักษณ์ <<datastore>>

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html



ภาพที่ 2.52 ตัวอย่างการนา Data Store มาใช้งาน

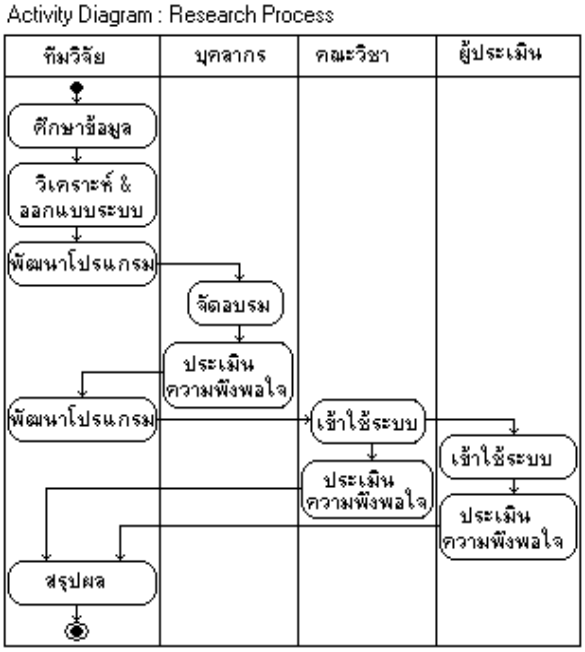
ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

8)คุณสมบัติของ Activity Diagram ที่ดี ได้แก่

- มุ่งเน้นการติดต่อสื่อสารของระบบในเชิงไดนามิก
- เฉพาะอีลิเมนต์ที่มีความสาคัญต่อกระบวนการทางานเท่านั้น
- แสดงรายละเอียดในแต่ละระดับการทางาน โดยเลือกแสดง เฉพาะที่มีความสาคัญต่อการเข้าใจการทางานของระบบเท่านั้น
- ถ้าการทางานส่วนใดมีความสาคัญ ก็ควรเขียน Activity Diagram ไม่ควรละเอาไว้หรือแสดงเพียงอย่างย่อๆ

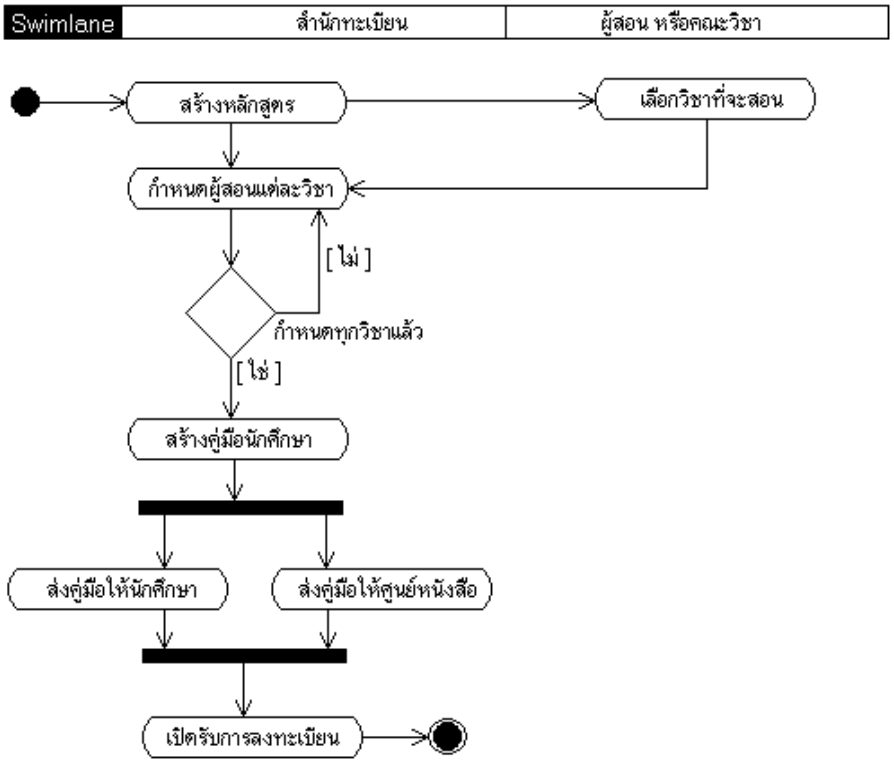
ตัวอย่าง Activity Diagram

- เรื่อง Research Process (กระบวนการวิจัย)
- ตัวอย่างการลงทะเบียนเรียน



ภาพที่ 2.53 ตัวอย่าง Research Process (กระบวนการวิจัย)

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html



ภาพที่ 2.54 ตัวอย่างการลงทะเบียนเรียน

ที่มา:https://infomation54.blogspot.com/2014/01/activity-diagram-workflow-activity.html

2.4 โปรแกรมและภาษาที่ใช้พัฒนาโปรแกรม

2.4.1Visual Studio Code

VS Code หรือ Visual Studio Code จากบริษัทไมโครซอฟต์ เป็นโปรแกรมประเภท Editor ใช้ในการแก้ไขโค้ดที่มีขนาดเล็ก แต่มีประสิทธิภาพสูง เป็น OpenSource โปรแกรมจึงสามารถนำมาใช้งานได้โดยไม่มีค่าใช้จ่าย เหมาะสำหรับนักพัฒนาโปรแกรมที่ต้องการใช้งานหลายแพลตฟอร์ม รองรับการใช้งานทั้งบน Windows , macOS และ Linux รองรับหลายภาษาทั้ง JavaScript, TypeScript และ Node.js ในตัว และสามารถเชื่อมต่อกับ Git ได้ง่าย สามารถนำมาใช้งานได้ง่ายไม่ซับซ้อน มีเครื่องมือและส่วนขยายต่าง ๆ ให้เลือกใช้มากมาย รองรับการเปิดใช้งานภาษาอื่น ๆ ทั้ง ภาษา C++ , C# , Java , Python , PHP หรือ Go สามารถปรับเปลี่ยน Themes ได้ มีส่วน Debugger และ Commands เป็นต้น ซึ่งบทความนี้จะเป็นการสอน วิธีการใช้งาน Visual Studio Code เบื้องต้น มาเริ่มกันเลย

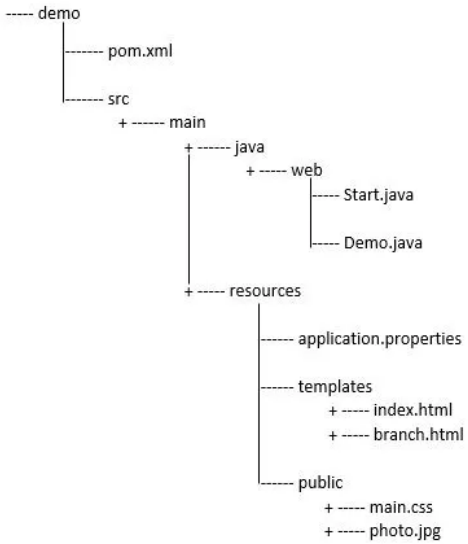
2.4.2 Flutter

Flutter เป็น framework ที่จะบอกว่าใหม่ก็ไม่ใหม่ขนาดนั้น มีมาประมาณ 4-5 ปีแล้ว ซึ่งเขามองว่าเป็น UI framework คือมันไม่ได้เป็นเหมือน full solution ที่จะเอามาทำ app หรือ web ทั้งหมด แต่เป็น UI layer ที่สมมติว่าต้องการทำ app ขึ้นมาง่ายๆ อยากให้มันมีหน้าตาสวยงาม หรือหน้าตาที่อยากให้เป็น Flutter ก็จะเป็น toolkit ที่ทำขึ้นมาเพื่อให้สามารถเขียน code ง่ายๆ แล้วก็ได้ app ที่หน้าตาตรงกับ code ที่เขียนเป๊ะๆ Flutter ใช้ภาษา Dart ในการเขียน ซึ่งหลายๆ คนอาจจะไม่คุ้นเคยกับภาษา Dart มาก เพราะมันฟังดูเป็นภาษาใหม่ แต่ว่าจริงๆ Dart ก็เป็นภาษาที่ค่อนข้างเก่าแก่เหมือนกัน แต่ก่อนหน้านี้มักถูกใช้ในฝั่ง back-end มากกว่า ซึ่ง Google ก็หยิบภาษา Dart มาทำ Flutter เพราะว่ามันมีความสามารถบางอย่าง เช่น เรื่อง hot reload คือการที่เขียน code เข้าไป ไม่จำเป็นต้อง compile ใหม่ทั้ง app เพราะปกติมักจะมีปัญหาว่า app มัน build นาน แก้ code 1 บรรทัด ก็นั่งรอไป 3 นาทีให้มัน build ใหม่ แต่พอมีความสามารถของภาษา Dart ก็ทำพัฒนา app ได้เร็วขึ้น พอแก้ code แล้วกด save ก็ได้เห็นผลลัพธ์ที่ต้องการเลย ก็จะช่วยประหยัดเวลาให้นักพัฒนาได้พอสมควร

2.4.3 Spring Boot

Spring Boot คือ Framework ใน Spring อันหนึ่ง สามารถช่วยทำให้สร้าง Web application หรือ Web serviceได้ง่ายขึ้น เพราะ Spring Boot มี Auto Configuration ซึ่งช่วยลดความยุ่งยากในการกำหนดค่าต่างๆ และสามารถใช้งานได้ทันที เนื่องจาก Spring Boot มี Java Web Server ที่ built-in มาให้แล้ว ก็คือ Tomcat ทำให้ง่ายต่อการใช้งาน โดยมี Port default คือ 8080 ซึ่งสามารถแก้ไขเปลี่ยน Port ได้ที่ไฟล์ application.properties

ซึ่งโครงสร้างการเก็บไฟล์ของ Spring Boot มี ดังนี้



ภาพที่2.55 application.properties

ที่มา: https://medium.com/@Teerawat.amo/spring-boot

Spring Boot จะเริ่มต้นทำงานที่ Start.java ซึ่งมีรายละเอียดดังนี้

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

	public static void main(String[] args) {
		SpringApplication.run(DemoApplication.class, args);
	}
}
```

ภาพที่2.56 Spring Boot Start.java

ที่มา: https://medium.com/@Teerawat.amo/spring-boot

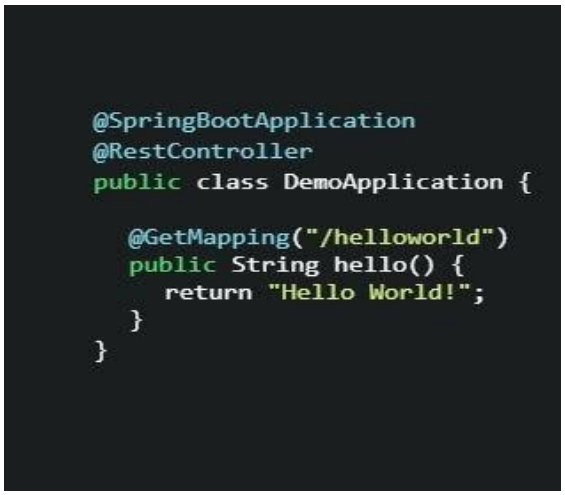
การระบุ *@SpringBootApplication* คือ การกำหนดที่เท่ากับการใช้ 3 Annotation ต่อไปนี้

- @Configuration ใช้การกำหนดค่าด้วยตนเอง
- @EnableAutoConfiguration ใช้การกำหนดค่าอัตโนมัติ
- @ComponentScan ค้นหา class ที่เป็น component อัตโนมัติ เช่น *@Controller, @Service, @Component, @Repository* เป็นต้น

ต่อมาจะเป็นส่วนของ Controller คือ ไฟล์ Demo.java โดยจะขอยกตัวอย่างเป็นการทำ Web service แบบ RESTful โดยทำงานผ่าน verb ของ HTTP เช่น Get , Post , Put , Delete อย่างเช่นในรูปนี้ เมื่อมี

request ไปที่ http://localhost:8080/helloworld เมื่อ web browser จะแสดงข้อความว่า “ Hello World!” ขึ้นมา

ต่อมาจะเป็นส่วนของ Controller คือ ไฟล์ Demo.java โดยจะขอยกตัวอย่างเป็นการทำ Web service แบบ RESTful โดยทำงานผ่าน verb ของ HTTP เช่น Get , Post , Put , Delete อย่างเช่นในรูปนี้ เมื่อมี request ไปที่ http://localhost:8080/helloworld เมื่อ web browser จะแสดงข้อความว่า “ Hello World!” ขึ้นมา



ภาพที่2.57 http://localhost:8080/helloworld

ที่มา: https://medium.com/@Teerawat.amo/spring-boot

2.4.4 ภาษาdart

Dart นั้นเป็นภาษาโปรแกรมที่เอาไว้สำหรับสร้างแอพพลิเคชันบนแพลตฟอร์มที่หลากหลายโดยได้ทั้ง mobile, desktop, server และก็ web สิ่งที่เป็นที่นิยมที่สุดที่ทำให้คนสนใจมาเรียนภาษา Dart กันก็คือเพื่อที่จะเอาไปใช้ร่วมกับ Flutter ที่เป็นเครื่องมือช่วยสร้าง UI ของ Google ซึ่งใช้ได้ทั้งกับ Android และ iOS หรือจะเป็นใน Desktop กับ Web ก็ยังได้ ภาษา Dart นี้ถูกสร้างโดย Google และปล่อยให้ใช้งานแบบ open source ทำให้ทุกคนสามารถนำไปใช้งานได้ฟรีๆ และการที่ Dart ถูกออกแบบมาให้ใช้งานได้ง่ายและมีประสิทธิภาพแบบภาษาเชิงวัตถุอื่นๆอย่าง Java C# C++ จึงเป็นตัวเลือกภาษาที่น่าสนใจในการศึกษาเป็นภาษาแรกอีกด้วย

2.6 ระบบจัดการฐานข้อมูล

2.6.1) ER-Diagram

คือ Entity–relationship model (ER model) หรือที่นิยมเรียกสั้น ๆว่า E-R Model เป็น Diagram ที่จะช่วยอธิบายโครงสร้าง Database ของระบบต่างๆที่ออกแบบมา อธิบายความสัมพันธ์ (Relationship) ของแต่ละ Entity รวมถึง attributes ของ Entity นั้นๆ ถ้าอธิบายในมุมของ DBMS Entity คือ table และ attributes คือ field ที่อยู่ใน table นั่นเองครับ ผลการออกแบบโดยใช้ E-R Model สามารถแสดงได้ด้วยการเขียนแผนภาพที่เรียกว่า Entity Relationship Diagram(ERD) ซึ่งถือว่าเป็นเครื่องมือที่ใช้อธิบายองค์ประกอบและข้อกำหนดของฐานข้อมูล ที่นักวิเคราะห์และออกแบบระบบใช้เป็นสื่อกลางในการสื่อสารระหว่างผู้ใช้และนักพัฒนาโปรแกรม เนื่องจากมีสัญลักษณ์ที่สื่อความหมายให้เข้าใจได้ง่าย ซึ่งในปัจจุบันมี Tool ที่สามารถแปลงจาก ER-Diagram กลายเป็น Database ได้ในภายหลังด้วย เป็นอะไรที่สะดวกมากเลยใช่ไหมล่ะ โดยจะมีองค์ประกอบหลักๆอยู่ 3 ส่วนคือ

- Entity
- Attribute
- Relationship

Simple ER-Diagram

ภาพที่ 2.58 ตัวอย่าง Simple ER-Diagram

ที่มา: https://blog.clicknext.com/what-is-er-diagram/

จากภาพที่ 2.55 Diagram ด้านบนเป็นตัวอย่าง ER-Diagram แบบง่ายๆที่ยกมาให้เห็นภาพกันก่อน เป็นความสัมพันธ์ระหว่าง 2 Entity Student กับ Course โดยความสัมพันธ์ที่มีคือ นักเรียน 1 คนสามารถลงเรียนได้ตั้งแต่ 1 ถึงหลายรายวิชา และใน 1 วิชารองรับนักเรียนได้หลายคนเป็นความสัมพันธ์แบบ Many-to-Many

1)Component of an ER-Diagramที่นี้มาทำความรู้จักองค์ประกอบต่างๆของ ER Diagram กัน

1.1) Entity เอนติตี้(Entity) หมายถึงกลุ่มของสิ่งต่างๆที่สนใจจะเก็บข้อมูลไว้ในฐานข้อมูล ซึ่งอาจจะเป็น บุคคล สถานที่ การกระทำ หรือกิจกรรมต่าง ๆ ซึ่งสัญลักษณ์ที่ใช้ใน ERD คือสี่เหลี่ยมผืนผ้า ตัวอย่างของเอนติตี้ ได้แก่

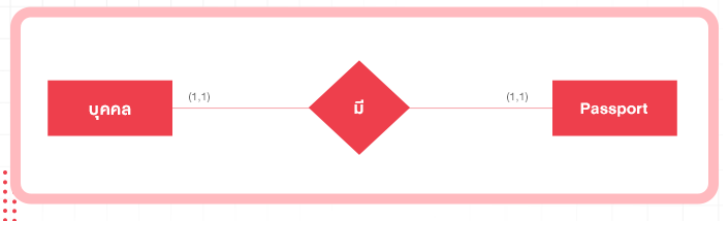
- เอนติตี้ที่เป็น บุคคล เช่น พนักงาน , นักศึกษา , อาจารย์ , แพทย์ , พยาบาล , ผู้ป่วย , นักบิน , พนักงานขับรถ เป็นต้น
- เอนติตี้ที่เป็น สถานที่ เช่น ประเทศ , จังหวัด , อำเภอ , น้ำตก , ภูเขา , โรงแรม , ห้องพัก , ห้องเช่า , ห้องเรียน เป็นต้น

1.2) Attribute แอททริบิวต์(Attribute) หมายถึง ลักษณะหรือคุณสมบัติที่นำมาอธิบาย Entity และ ความสัมพันธ์ ตัวอย่างของแอททริบิวต์ของ Entity ซึ่งสัญลักษณ์ที่ใช้ใน ERD คือวงรี สำหรับแอททริบิวต์ที่ถูกกำหนดให้ทำหน้าที่เป็นคีย์หลักมีค่าได้เพียงค่าเดียวห้ามซ้ำกัน(primary key) ของ Entity ก็จะขีดเส้นทึบใต้ชื่อของแอททริบิวต์ เพื่อแสดงให้รู้ว่าเป็นคีย์หลัก เช่น

- แอททริบิวต์ของ Entity “นักศึกษา” ได้แก่ รหัสนักศึกษา , คำนำหน้าชื่อ , ชื่อ , นามสกุล , วันเกิด , โปรแกรมวิชาที่สังกัด , เกรดเฉลี่ยสะสม
- แอททริบิวต์ของ Entity “ผู้ป่วย” ได้แก่ รหัสผู้ป่วย , ชื่อ , นามสกุล , สถานภาพ , วันที่เข้ารักษาครั้งแรก , ที่อยู่ , โทรศัพท์

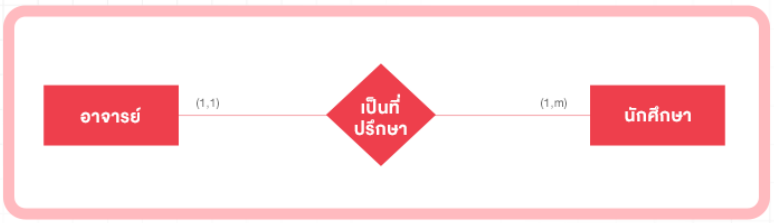
1.3) Relationship ความสัมพันธ์ (Relationship) หมายถึง ความสัมพันธ์ระหว่าง Entity ต่าง ๆ ซึ่งสัญลักษณ์ที่ใช้ใน ERD คือสี่เหลี่ยมผืนผ้า มีอยู่ด้วยกัน 4 แบบ ดังนี้

1.4) One-to-One Relationship หรือ 1 : 1 จากภาพที่2.59เป็นการแสดงความสัมพันธ์ของจำนวนข้อมูลของ Entity A ว่า ข้อมูล 1 รายการ มีความสัมพันธ์กับข้อมูล Entity B ได้ไม่เกิน 1 รายการ ตัวอย่าง เช่น บุคคล 1 คน จะสามารถมี passport ได้ 1 ใบ และในขณะเดียวกัน passport 1 ใบมีข้อมูลได้แค่ 1 คนเท่านั้น



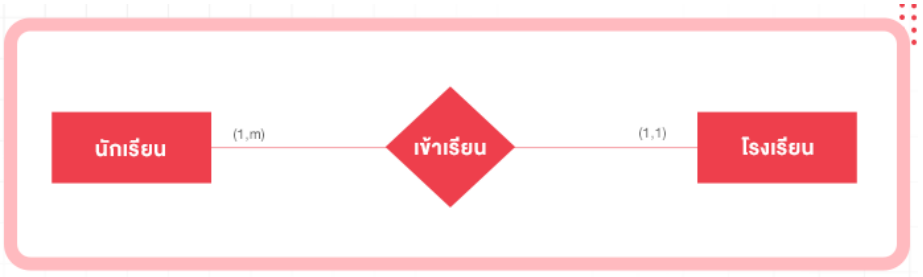
ภาพที่ 2.59 ตัวอย่าง One-to-One Relationship
ที่มา: https://blog.clicknext.com/what-is-er-diagram/

1.5) One-to-Many Relationship หรือ 1 : N จากภาพที่ 2.60เป็นการแสดงความสัมพันธ์ของจำนวนข้อมูลของ Entity A ว่า ข้อมูล 1 รายการ มีความสัมพันธ์กับข้อมูล Entity B ได้มากกว่า 1 รายการ ตัวอย่างเช่น อาจารย์ 1 คน จะสามารถมีนักศึกษาที่ปรึกษาได้มากกว่า 1 คน และในขณะเดียวกัน นักศึกษาแต่ละคนต้องมีอาจารย์ที่ปรึกษาคนใดคนหนึ่งเท่านั้น ดังภาพที่ 2.60



ภาพที่ 2.60 ตัวอย่าง One-to-Many Relationship
ที่มา: https://blog.clicknext.com/what-is-er-diagram/

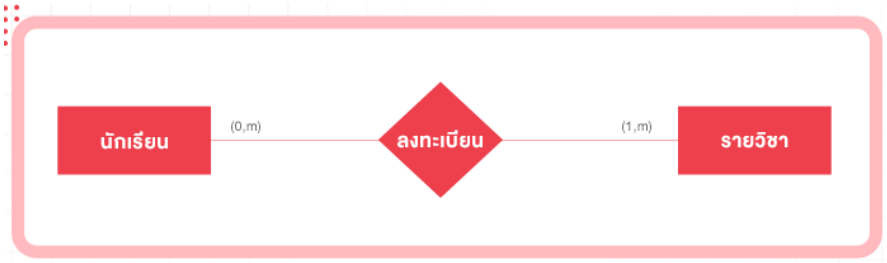
1.6) Many-to-One Relationship หรือ N : 1 จากภาพที่ 2.61เป็นการแสดงความสัมพันธ์ของจำนวนข้อมูลของ Entity A ว่า ข้อมูล 1 รายการ มีความสัมพันธ์กับข้อมูล Entity B ได้แค่ 1 รายการ ในขณะที่ข้อมูล Entity B มีความสัมพันธ์กับ Entity A ได้มากกว่า 1 รายการ ตัวอย่างเช่น นักเรียน 1 คน จะสามารถเข้าเรียนที่โรงเรียนได้แค่ 1 โรงเรียนเท่านั้น แต่ในขณะเดียวกันโรงเรียน 1 โรงเรียนมีจำนวนนักเรียนได้หลายคน



ภาพที่ 2.61 ตัวอย่าง Many-to-One Relationship

ที่มา: https://blog.clicknext.com/what-is-er-diagram/

1.7)Many-to-Many Relationship หรือ M : N จากภาพที่ 2.62เป็นการแสดงความสัมพันธ์ของจำนวนข้อมูลของ Entity A ว่า ข้อมูล 1 รายการ มีความสัมพันธ์กับข้อมูลเอนติตี้ B ได้แค่หลายรายการ ในขณะที่ข้อมูล Entity B มีความสัมพันธ์กับ Entity A ได้มากกว่า 1 รายการเช่นเดียว ตัวอย่างเช่น นักเรียน 1 คนสามารถลงเรียนได้หลายรายวิชา และใน 1 วิชารองรับนักเรียนได้หลายคน



ภาพที่ 2.62 ตัวอย่าง Many-to-Many Relationship

ที่มา: https://blog.clicknext.com/what-is-er-diagram/

2.6.2) ดาต้าดิคท์

คือ พจนานุกรมข้อมูล ที่แสดงรายละเอียดตารางข้อมูลต่างๆ ในฐานข้อมูล (Database) ซึ่งประกอบด้วยรีเลชั่น (Relation Name), แอตทริบิวต์ (Attribute), ชื่อแทน (Aliases Name), รายละเอียดข้อมูล (Data Description), แอตทริบิวโดเมน (Attribute Domain), ฯลฯ ทำให้สามารถค้นหารายละเอียดที่ต้องการได้สะดวกมากยิ่งขึ้น พจนานุกรมข้อมูลเป็นการผสมผสานระหว่างรูปแบบของพจนานุกรมโดยทั่วไปและรูปแบบของข้อมูลในระบบงานคอมพิวเตอร์ เพื่ออธิบายชนิดของข้อมูลแต่ละตัวว่าเป็น

ตัวเลข อักขระ ข้อความ หรือวันที่ เป็นต้น เพื่อช่วยในการอธิบายรายละเอียดต่างๆ ในการอ้างอิงหรือค้นหาที่เกี่ยวกับข้อมูล หรือจะเรียกง่ายๆ ว่า Data Dictionary คือ เอกสารที่ใช้อธิบายฐานข้อมูลหรือการจัดเก็บฐานข้อมูล ซึ่ง Data Dictionary มีประโยชน์ ดังนี้

- จัดเก็บรายละเอียดข้อมูล
- แสดงความหมายที่เกี่ยวกับระบบ
- ทำเอกสารที่บอกคุณลักษณะของระบบ
- หาข้อบกพร่องและสิ่งที่หายไปจากระบบ

ส่วนประกอบของ Data Dictionary

1. ข้อมูลย่อย (Data Element) : ส่วนประกอบพื้นที่ ที่ไม่สามารถแบ่งให้เล็กลงได้อีก
2. โครงสร้างข้อมูล (Data Structure) : สร้างขึ้นโดยการนำส่วนย่อยของข้อมูล ตั้งแต่ตัวขึ้นไป ที่สัมพันธ์กันมารวมเข้าด้วยกัน

สัญลักษณ์ที่ใช้ในพจนานุกรมข้อมูล ได้แก่

= หมายถึง เท่ากับ

+ หมายถึง และ

{} หมายถึง มีการซ้ำของส่วนย่อยข้อมูล

[I] หมายถึง ทางเลือกให้เลือกส่วนย่อยของข้อมูลตัวใดตัวหนึ่ง

() หมายถึง การเกิดขึ้นเป็นกรณีพิเศษ จะปรากฏหรือไม่ปรากฏก็ได้

2.6.3)มายเอสคิวเอล

คือ ระบบจัดการฐานข้อมูล หรือ Database Management System (DBMS) แบบข้อมูลเชิงสัมพันธ์ หรือ Relational Database Management System (RDBMS) ซึ่งเป็นระบบฐานข้อมูลที่จัดเก็บรวบรวมข้อมูลในรูปแบบตาราง โดยมีการแบ่งข้อมูลออกเป็นแถว (Row) และในแต่ละแถวแบ่งออกเป็นคอลัมน์ (Column) เพื่อเชื่อมโยงระหว่างข้อมูลในตารางกับข้อมูลในคอลัมน์ที่กำหนด แทนการเก็บข้อมูลที่แยกออกจากกัน โดยไม่มีความเชื่อมโยงกัน ซึ่งประกอบด้วยข้อมูล (Attribute) ที่มีความสัมพันธ์เชื่อมโยงกัน (Relation) โดยใช้ RDBMS Tools สำหรับการควบคุมและจัดเก็บฐานข้อมูลที่จำเป็น ทำให้นำไปประยุกต์ใช้งานได้ง่าย ช่วยเพิ่มประสิทธิภาพในการทำงานให้มีความยืดหยุ่นและรวดเร็วได้มากยิ่งขึ้น รวมถึงเชื่อมโยงข้อมูล ที่จัดแบ่งกลุ่มข้อมูลแต่ละประเภทได้ตามต้องการ จึงทำให้ MySQL เป็นโปรแกรมระบบจัดฐานข้อมูลที่ได้รับความนิยมสูง

โปรแกรมนี้เป็น Open Source ที่ถูกพัฒนาขึ้นจาก MySQL AB ในประเทศสวีเดน โดยชาวสวีเดน 2 คน คือ David Axmark และ Allan Larsson ร่วมกับชาวฟินแลนด์ Michael Monty Widenius ซึ่งต่อมาในปี ค.ศ. 2008 ถูกซื้อกิจการโดย Sun Microsystems และภายหลัง Oracle Corporation ได้เข้าซื้อกิจการในปี ค.ศ. 2010

มีหน้าที่จัดเก็บข้อมูลอย่างเป็นระบบ รองรับคำสั่งภาษา Structured Query Language หรือ SQL เพื่อจัดการกับฐานข้อมูลโดยเฉพาะ เป็นภาษามาตรฐานบนระบบฐานข้อมูลเชิงสัมพันธ์และเป็นระบบเปิด (Open System) ที่มีโครงสร้างของภาษาที่เข้าใจง่าย ไม่ซับซ้อน และนิยมใช้งานร่วมกับภาษาโปรแกรม PHP รวมถึงภาษาอื่น ๆ ที่สามารถทำงานร่วมกันกับฐานข้อมูล MySQL ได้หลากหลาย เช่น C, C++, Python, Java เป็นต้น อีกทั้ง MySQL ยังได้รับการออกแบบและปรับให้มีความเหมาะสมสำหรับการพัฒนา Website และ Web Application ทำให้สามารถรองรับการทำงานได้ทุกแพลตฟอร์ม รวมถึงการอนุญาตให้ผู้ใช้งานหลายคนสามารถใช้งานพร้อมกันได้ (Multi-user) นอกจากนั้นยังสามารถจัดการและสร้างฐานข้อมูลจำนวนมากรวมถึงประมวลผลหลาย ๆ งานได้พร้อมกัน (Multi-threaded) อย่างสมบูรณ์ จึงทำให้ MySQL เป็นตัวเลือกยอดนิยมสำหรับธุรกิจการพาณิชย์อิเล็กทรอนิกส์ หรือ Electronic Commerce (E-Commerce) และเหมาะสำหรับการนำไปใช้งานสร้างเว็บไซต์ทั่วไป เพราะมีความแม่นยำ ครบครัน ช่วยให้เข้าถึงข้อมูลได้อย่างรวดเร็ว อีกทั้งยังมีความน่าเชื่อถือสูง และยังมีโปรแกรมเสริมช่วยจัดฐานข้อมูลที่ใช้งานง่าย เช่น Mysql Admin, phpMyAdmin เป็นต้น

1.1)ใช้ทำงานอะไรบ้าง

มีให้เลือกใช้งาน 2 รุ่น ได้แก่ MySQL Community Edition ที่เป็นเวอร์ชันฟรี ซึ่งเป็น Open Source และ MySQL Enterprise Edition ที่มีคุณสมบัติมากกว่าและการสนับสนุนด้านเทคนิคที่ครอบคลุม รวมถึงยังได้รับอนุญาตให้ใช้ในเชิงพาณิชย์ โดย MySQL เป็นตัวเลือกยอดนิยมสำหรับเว็บไซต์ขนาดใหญ่และ Web Application เนื่องจากสามารถรองรับการรับส่งข้อมูลในระดับสูง รวมถึงยังมีคุณสมบัติที่ช่วยเพิ่มประสิทธิภาพการทำงาน เช่น กระบวนการจัดเก็บข้อมูล (Store Procedure), กระบวนการทำงานแบบอัตโนมัติ (Database Trigger), มุมมองฐานข้อมูล (Database View) และภาพรวมระบบฐานข้อมูล (Database Schema) เป็นต้น

โดย MySQL ถูกนำไปใช้ในองค์กรหรือกลุ่มธุรกิจชั้นนำต่าง ๆ มากมาย เพราะสามารถปรับใช้ให้เข้ากับความต้องการของแต่ละองค์กรได้อย่างมีประสิทธิภาพ เช่น

- การจัดเก็บข้อมูลสำหรับ Website
- การจัดเก็บข้อมูลสำหรับ Mobile Application
- การจัดเก็บข้อมูลสำหรับ Application สำหรับองค์กร
- การจัดเก็บข้อมูลทางการแพทย์
- การจัดเก็บข้อมูลทางการเงิน

▪ การจัดเก็บและสร้างฐานข้อมูลของลูกค้า

2.7การทดสอบระบบ

คือขั้นตอนการตรวจสอบให้พบข้อผิดพลาดเมื่อพบข้อผิดพลาดจะได้ทำการแก้ไขและป้องกันข้อผิดพลาดที่เกิดขึ้นของโปรแกรมหรือระบบนั้นๆ เพื่อเป็นการทดสอบความสมบูรณ์ของโปรแกรม

การทดสอบโปรแกรมจะทำหลังจากที่โปรแกรมเมอร์ได้เขียนโปรแกรมเสร็จสิ้น จึงต้องทดสอบว่าโปรแกรมนั้นให้ผลลัพธ์ที่ถูกต้องหรือไม่

2.7.1 วัตถุประสงค์ของการทดสอบ

เพื่อระบุและแก้ไขข้อผิดพลาดระหว่างการพัฒนาโปรแกรม โดยใช้หลักการการทวนสอบและทดสอบเพื่อรับรองผล (Verification and Validation) ในการประเมินโปรแกรมที่ทำการทดสอบมีรายละเอียดดังนี้

-การทวนสอบ (Verification) เป็นกระบวนการวิเคราะห์แบบ Static คือการประเมินว่าโปรแกรมทำงานได้ถูกต้องตามขั้นตอน กระบวนการทำงานหรือไม่ ซึ่งตรงกับการทดสอบแบบเป็นระดับโดยมีการทดสอบระดับ Unit Test การทดสอบระดับ Integration Test และการทดสอบระดับ System Test

-การทดสอบเพื่อรับรองผล (Validation) เป็นกระบวนการวิเคราะห์แบบ Dynamic คือการประเมินว่าโปรแกรม (Software) ตรงตามความต้องการของผู้ใช้ (User Requirement) ในกระบวนการช่วงหลังของการพัฒนา ซึ่งตรงกับการทดสอบระดับการยอมรับ (Acceptance Test)

2.7.2 ระดับการทดสอบโปรแกรม

1) Unit Testing เป็นการทดสอบการทำงานของแต่ละ Module ซึ่งกระทำโดยโปรแกรมเมอร์

2)Integration Testing คือการนำเอา Module ต่าง ๆ มาประกอบรวมกัน ซึ่งการทดสอบในขั้นตอนนี้จะเป็นการทดสอบการทำงานร่วมกันของ Module หรือ Function อื่น ๆ ที่ผ่านการทำ Unit Test มาแล้ว ผู้ทดสอบจะต้องทำการทดสอบตาม Test script ที่ได้ทำไว้ ซึ่งถ้าหากเกิดข้อผิดพลาดก็จะส่งผลกลับไปให้โปรแกรมเมอร์ทำการแก้ไขต่อไป

3)System Testing เป็นการทดสอบระบบหรือโปรแกรม โดยดูภาพรวมของการทำงานว่ามีการตอบสนองความต้องการทั้งในส่วนของฟังก์ชันการทำงานและประสิทธิภาพการทำงาน และสอดคล้องกับลักษณะของความต้องการของซอฟต์แวร์ (Requirement Specification) หรือไม่ โดยใช้การทดสอบแบบ Functional Testing

4)User Acceptance Testing (UAT) เป็นกระบวนการทดสอบการยอมรับ เพื่อตรวจสอบว่าโปรแกรมสามารถตอบสนองตามความต้องการของลูกค้าในระดับที่ยอมรับได้ และทดสอบกับสภาพแวดล้อมที่ใกล้เคียงสถานการณ์จริงมากที่สุด

2.7.3 ความสำคัญของการทดสอบระบบ

1)การทดสอบการทำงานสูงสุด (Peak Load Testing)

2)การทดสอบประสิทธิภาพของเวลา (Performance Testing)

3)การทดสอบการกู้ระบบ (Recovery Testing)

4)การทดสอบการเก็บข้อมูล (Storage Testing)

5)การทดสอบกระบวนการ (Procedure Testing)

6)การทดสอบผู้ใช้ (User Testing) เป็นการทดสอบการใช้งานจริงของระบบ

2.7.4 การทดสอบการยอมรับระบบโดยผู้ใช้

1) Alpha Testing คือ การทดสอบความสมบูรณ์ของระบบโดยผู้ใช้ และใช้ข้อมูลสมมติ ในการทดสอบ โดยการทดสอบจะมี 4 แบบดังนี้

- Recovery Testing เป็นการทดสอบการกู้ระบบ
- Security Testing เป็นการทดสอบความปลอดภัยของระบบ
- Stress Testing เป็นการทดสอบประสิทธิภาพการทำงานของระบบภายใต้ความกดดัน
- Performance Testing เป็นการทดสอบประสิทธิภาพการทำงานของระบบภายใต้สภาพแวดล้อมของคอมพิวเตอร์

2) Bata Testing คือการทดสอบความสมบูรณ์ของระบบโดยผู้ใช้ และใช้ข้อมูลจริงในการ ทดสอบภายใต้สถานการณ์ที่เกิดขึ้นจริง

2.7.5 เทคนิคพื้นฐานในการทดสอบซอฟต์แวร์

1) การทดสอบแบบ White Box

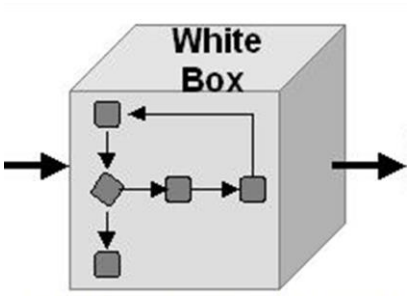
การทดสอบแบบนี้เป็นการทดสอบกลไกภายในของโปรแกรมที่ถูกพัฒนาขึ้น โดยนักทดสอบใช้แนวทางในการพัฒนาโปรแกรมมาทำการวิเคราะห์ซอร์สโค้ด เพื่อนำมาใช้ในการกำหนดข้อมูลที่ใช้สำหรับการทดสอบ เพื่อให้ครอบคลุมการทำงานต่าง ๆ ของโปรแกรม เช่น เส้นทางการทำงาน (Paths) เงื่อนไขการทำงาน (Branches) และชุดคำสั่งต่าง ๆ (Statements) เป็นต้น ดังนั้นอาจสรุปได้ว่าการทดสอบแบบนี้เกี่ยวข้องกับการตรวจสอบโค้ดและค้นหาว่าส่วนใดของโค้ดที่มีการทำงานผิดพลาด การทดสอบแบบนี้นักทดสอบจำเป็นต้องมีความรู้ทางด้านการทำงานทางลอจิกและการพัฒนาโปรแกรม และการออกแบบข้อมูลที่ใช้สำหรับการทดสอบ

รูปแบบของการทดสอบนี้จะเน้นไปที่รายละเอียดของการทำงานทางลอจิกภายในโปรแกรมเป็นหลัก แต่อย่างไรเสียยังคงไม่มีซอฟต์แวร์หรือวิธีการใด ๆ ที่สามารถใช้กับการทดสอบแบบนี้ได้โดยตรง และได้ผลลัพธ์ที่สมบูรณ์จากการทดสอบ

แบ่งออกได้เป็น 2 วิธีหลัก ๆ ได้แก่

- การทดสอบกระแสควบคุม (Control Flow Testing) เป็นการทดสอบโดนเน้นไปที่กลไกควบคุมการทำงานหลัก

- การทดสอบกระแสข้อมูล (Data Flow Testing) เป็นการทดสอบโดยเน้นไปที่การนิยาม (Definition) ข้อมูลและการใช้งาน (Uses) ของตัวแปร ตลอดจนเส้นทางการประมวลผลที่มีความสัมพันธ์ระหว่างกัน



ภาพที่ 2.63 ตัวอย่าง White Box

ที่มา: COM 4501 การทดสอบซอฟต์แวร์ (Software Testing)

2) การทดสอบแบบ Black Box

เป็นเทคนิคในการทดสอบที่นักทดสอบไม่จำเป็นต้องมีความรู้เกี่ยวกลไกภายในของโปรแกรม การทดสอบภาตใต้วิธีนี้จะเน้นไปที่ความต้องการของระบบและฟังก์ชันการทำงานของระบบหลัก การทดสอบแบบ Black Box บางครั้งอาจถูกเรียกว่า Opaque Testing, Behavioral testing หรือ Closed Box Testing เป็นต้น

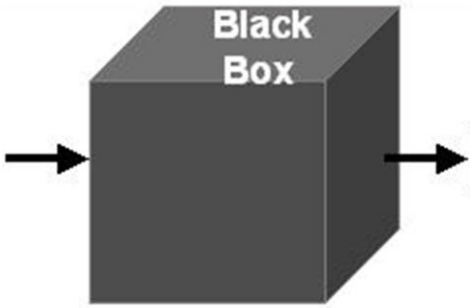
วิธีนี้นิยมให้การทดสอบฟังก์ชันการทำงานเพื่อค้นหาความผิดพลาดของฟังก์ชันการทำงานที่ไม่ถูกต้องหรือขาดหายไป รวมไปถึงความผิดพลาดที่เกิดขึ้นในระหว่างการติดต่อกันภายในระบบ ไม่ว่าจะเป็นความผิดพลาดที่เกิดขึ้นภายในโครงสร้างข้อมูล หรือระบบฐานข้อมูล ความผิดพลาดจากพฤติกรรมการทำงานหรือสมรรถนะของระบบ รวมไปถึงความผิดพลาดที่เกิดจากการกำหนดค่าเริ่มต้นและสิ้นสุดการทำงาน เป็นต้น

แบ่งออกได้เป็น 3 วิธี ได้แก่

- การแบ่งชั้นสมมูล (Equivalence Partition) เป็นการจัดแบ่งข้อมูลเข้ามาเป็นกลุ่มขนาดเท่า ๆ กัน และเป็นเงื่อนไขในการกำหนดข้อมูลเข้าสู่ระบบ

- การวิเคราะห์ค่าขอบเขต (Boundary Value Analysis) วิธีการทดสอบแบบนี้ตั้งสมมุติฐานไว้ว่าความผิดพลาดมักจะเกิดขึ้นได้ที่บริเวณของเขตของข้อมูลเข้า ดังนั้นการทดสอบจะเน้นไปที่การออกแบบข้อมูลทดสอบที่ใช้ตรวจสอบขีดจำกัดบนและขีดจำกัดล่างของชั้นสมมูลที่ได้จากวิธีการแรก

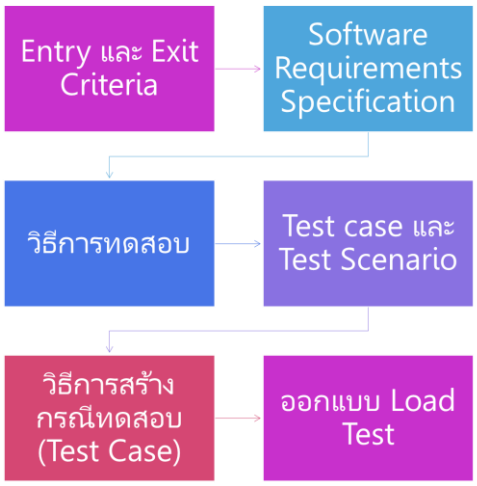
- กราฟสาเหตุและผล (Cause-effect graphing) เป็นวิธีการออกแบบข้อมูลทดสอบที่สามารถมองเห็นถึงเงื่อนไขทางลอจิกและการกระทำที่เกี่ยวข้อง ซึ่งถูกนำเสนอผ่านโหนด (node) และแสดงการเชื่อมต่อระหว่างโหนดเพื่อแสดงให้เห็นถึงความสัมพันธ์ระหว่างกันแต่ละ โหนดและลิงค์ (link) จะถูกกำหนดน้ำหนักขึ้นตามลำดับ เพื่อนำไปใช้การสร้างตารางสำหรับตัดสินใจและข้อมูลทดสอบต่อไป



ภาพที่ 2.64 ตัวอย่าง Black Box

ที่มา: COM 4501 การทดสอบซอฟต์แวร์ (Software Testing)

2.7.6 แนวทางการทดสอบโปรแกรม



ภาพที่ 2.65 ตัวอย่าง แนวทางการทดสอบโปรแกรม

ที่มา: COM 4501 การทดสอบซอฟต์แวร์ (Software Testing)

เพื่อให้สามารถควบคุมคุณภาพของซอฟต์แวร์และการทดสอบได้อย่างมีประสิทธิภาพขอแนะนำให้มีเกณฑ์ที่กำหนดว่ากิจกรรมการทดสอบที่กำหนดควรเริ่มต้นเมื่อใดและเมื่อกิจกรรมเสร็จสมบูรณ์ Entry Criteria (โดยทั่วไปเรียกว่าคำจำกัดความพร้อมในการพัฒนาแบบ Agile) กำหนดเงื่อนไขเบื้องต้นสำหรับการทำกิจกรรมการทดสอบที่กำหนด หากไม่ผ่านเกณฑ์การเข้าร่วมอาจเป็นไปได้ว่ากิจกรรมจะพิสูจน์ได้ยากขึ้นใช้เวลามากขึ้นมีค่าใช้จ่ายมากขึ้นและมีความเสี่ยงมากขึ้น Exit Criteria (โดยทั่วไปเรียกว่าคำจำกัดความของการทำในการพัฒนาแบบ Agile) กำหนดเงื่อนไขที่จะต้องบรรลุเพื่อประกาศระดับการทดสอบหรือชุดการทดสอบที่เสร็จสมบูรณ์ ควรกำหนดเกณฑ์การเข้าและออกสำหรับแต่ละระดับการทดสอบและประเภทการทดสอบและจะแตกต่างกันไปตามวัตถุประสงค์ของการทดสอบ

การกำหนดเงื่อนไข/เกณฑ์

- ความพร้อมของสภาพแวดล้อมในการทดสอบ ฮาร์ดแวร์และซอฟต์แวร์ที่รับรอง
- ความเหมาะสมและความพร้อมของข้อมูลที่จะใช้ในการทดสอบ
- มีข้อมูลในการทดสอบที่เหมาะสม ผู้ทดสอบผ่านการฝึกอบรมมาอย่างดี รวมถึงมีทรัพยากรอื่น ๆ ที่จำเป็นครบถ้วน
- ข้อกำหนด เป้าหมายที่ต้องการควรมีการระบุให้ชัดเจน และมีการอนุมัติ รับรอง
- จัดทำ Test design & document plan ให้พร้อม
- ใช้งบประมาณตามแผน
- เสร็จทันกำหนด
- ทดสอบครบตามรายการที่วางไว้ โดยไม่มีอุปสรรคใด ๆ
- ทดสอบครอบคลุมตามเป้าหมายที่กำหนดและใช้งานได้จริง
- ข้อบกพร่อง (bugs fixed) ต่าง ๆ ได้รับการแก้ไข
- พื้นที่ที่ระบุให้เป็นบริเวณเสี่ยงได้รับการปรับปรุงและทดสอบอีกครั้ง

ภาพที่ 2.66 การกำหนดเงื่อนไข

ที่มา: COM 4501 การทดสอบซอฟต์แวร์ (Software Testing)

โครงสร้าง Specific Requirements

มาตรฐาน SRS ที่เป็นที่รู้จักกัน คือ IEEE/ANSI 830-1998 ซึ่งมีโครงสร้างของเอกสาร ที่สามารถนำไปปรับใช้ในองค์กร ได้ดังนี้

1. บทนำ (Introduction)

เป้าหมายของเอกสารความต้องการ (Purpose of the Requirements document)

ขอบเขตของผลิตภัณฑ์ (Scope of the product)

นิยาม (Definitions), (Acronyms), (Abbreviations)

บทอ้างอิง (References)

ภาพรวมของส่วนที่เหลือของเอกสาร (Overview of the remainder of the document)

2. คำอธิบายทั่วไป (General Description)

มุมมองของผลิตภัณฑ์ (Product perspective)

ฟังก์ชันการทำงานของผลิตภัณฑ์ (Product functions)

คุณลักษณะของผู้ใช้งาน (User characteristics)

ข้อจำกัดโดยทั่วไป (General constraints)

ข้อสมมติฐาน (Assumptions) และการขึ้นอยู่ต่อกัน (Dependencies)

3. ความต้องการเฉพาะ (Specific Requirements)

ซึ่งส่วนนี้เป็นส่วนที่สำคัญมากที่สุดของเอกสาร แต่เนื่องจากแต่ละองค์กรมีแนวทางการปฏิบัติแตกต่างกัน ดังนั้นในส่วนนี้จะไม่มีโครงสร้างที่เป็นมาตรฐาน ความต้องการในส่วนนี้เป็นการทำเอกสารส่วนติดต่อกับภายนอก, บรรยายหน้าที่และการทำงานของระบบ, ข้อจำกัดในส่วนของการออกแบบ, คุณสมบัติที่เกิดขึ้นในระบบ และคุณลักษณะเชิงคุณภาพ

4. ภาคผนวก (Appendices)

5. ดัชนี (Index)

วิธีการทดสอบโปรแกรม

1) การทดสอบโดยไม่ใช้เครื่องคอมพิวเตอร์ (Manual Testing)

เป็นการทดสอบ โปรแกรมการตรวจสอบจากโปรแกรมเมอร์เอง ซึ่งการทดสอบแบบโดยไม่ใช้เครื่องคอมพิวเตอร์ แบ่งออกเป็น 2 แบบ ได้แก่

1.1 การทดสอบแบบตรวจการณ์ (Inspection) เป็นเทคนิคการทดสอบโปรแกรม โดยโปรแกรมเมอร์ตรวจสอบเอง ด้วยการเปรียบเทียบโค้ดของโปรแกรมที่เขียนขึ้น กับข้อผิดพลาด (Error) ที่โปรแกรมเมอร์ทราบแล้วว่าจะต้องเกิดขึ้นจากโปรแกรมภาษาที่ใช้ในการพัฒนาโปรแกรม โดยการตรวจสอบว่าโค้ดที่เขียนขึ้นนั้นมีข้อผิดพลาดเกิดขึ้นตามรายการหรือไม่ เช่น ข้อผิดพลาด ที่สามารถเกิดขึ้นได้ในภาษาโคบอล (COBOL) ซึ่งจะมีรายการข้อผิดพลาดในคู่มือ ดังนั้นสามารถ นำคู่มือนั้นมาทำการเปรียบเทียบกับโปรแกรมที่พัฒนาขึ้นได้ บางกรณีโปรแกรมเมอร์สามารถ ทดสอบโปรแกรมแบบตรวจการณ์ในขณะเขียนโปรแกรมได้ ซึ่งจะทำให้ลดเวลาในการทดสอบ โปรแกรมแบบนี้ได้ภายหลัง แต่การทดสอบแบบนี้ไม่ได้ทำให้โปรแกรมเมอร์ทราบว่ามีผลลัพธ์ที่ ถูกต้องหรือไม่ เนื่องจากไม่ได้ทดสอบการทำงานของโปรแกรม เป็นเพียงการทดสอบความผิด พลาดของโค้ด (Code) เท่านั้น

1.2 การทดสอบตามลำดับคำสั่งในโปรแกรม (Desk Checking) เทคนิคการ ทดสอบโปรแกรมแบบนี้ กระทำโดยผู้ที่ได้รับการแต่งตั้งเป็นผู้ทดสอบโปรแกรม ซึ่งอาจจะเป็น

โปรแกรมเมอร์หรือไม่ก็ได้ แต่จะต้องมีความเข้าใจในการทำงานทางตรรกะของโปรแกรม ด้วยการ ตรวจสอบโค้ดของโปรแกรมตามลำดับคำสั่งในโปรแกรมว่า มีตรรกะที่ผิดปกติหรือไม่ แต่วิธีการ นี้จะทำให้เสียเวลาในการทดสอบโปรแกรมค่อนข้างมาก ถ้าระบบงานมีความซับซ้อนสูง

2) การทดสอบด้วยเครื่องคอมพิวเตอร์ (Automated Testing)

เป็นการทดสอบ โปรแกรมด้วยเครื่องคอมพิวเตอร์ ซึ่งจะทำให้ไม่เสียเวลาในการทดสอบ แบ่งออกเป็น 5 แบบ ได้แก่

2.1 การทดสอบด้วยการตรวจสอบไวยากรณ์ (Syntax Checking) เป็นการทดสอบโปรแกรมด้วยการตรวจสอบไวยากรณ์ (Syntax) ที่เขียนขึ้น โดยปกติแล้วจะได้รับการตรวจสอบ ด้วยผู้รวบรวมข้อมูล ซึ่งจะใช้เวลาไม่นานก็สามารถทราบผลได้ แต่วิธีการนี้ไม่สามารถทำให้ทราบ ได้ว่าผลลัพธ์จากการทำงานของโปรแกรมนั้นถูกต้องหรือไม่ เนื่องจากเป็นการทดสอบเพียง ไวยากรณ์ของโปรแกรมเท่านั้น

2.2 การทดสอบทีละโมดูล (Unit Testing) หรือบางครั้งเรียกอีกอย่างหนึ่งว่า “Module Testing” เป็นการทดสอบโปรแกรมที่ละโมดูล เพื่อหาข้อผิดพลาดที่จะเกิดขึ้นภายในการทำงาน ของแต่ละโมดูล

2.3 การทดสอบแบบเพิ่มโมดูล (Integration Testing) เป็นการทดสอบโปรแกรม โดยการเพิ่มจำนวนโมดูล เพื่อการทดสอบหาข้อผิดพลาดของโปรแกรม ซึ่งวิธีการในการทดสอบแบบเพิ่มโมดูลนี้แบ่งออกเป็น 2 ลักษณะได้แก่

- การทดสอบจากบนลงล่าง (Top-down Approach) เป็นการทดสอบโปรแกรมโดยทดสอบโมดูลจากบนลงล่าง จะเริ่มทดสอบที่โมดูล A ก่อนอันดับแรก หลังจากนั้นให้เพิ่มโมดูล B, C และ D แล้วทดสอบรวมกับโมดูล A ลำดับต่อไปคือ เพิ่มโมดูล E, F แล้วทดสอบรวมกับโมดูล B จากนั้นจึงทดสอบรวมกับโมดูลทั้งหมด

- การทดสอบจากล่างขึ้นบน (Bottom-up Approach) เป็นการทดสอบโปรแกรมโดยทดสอบโมดูลจากล่างขึ้นบน จะเริ่มจากการทดสอบโมดูล E และ F ก่อน จากนั้นทดสอบรวมกับ โมดูล B ลำดับต่อไปคือ เพิ่มโมดูล C, D ลำดับสุดท้ายคือ เพิ่มโมดูล A แล้วจึงทดสอบรวมกับ โมดูลทั้งหมด

2.4 การทดสอบด้วยโมดูลตัวแทน (Stub Testing) โดยทั่วไปแล้วโมดูลที่อยู่ในระดับ บนจะเรียกใช้ข้อมูลจากโมดูลระดับล่าง แต่การทดสอบโปรแกรมแบบเพิ่มโมดูลนั้นจะทดสอบโมดูลที่ละระดับ สมมติว่ากำลังจะทดสอบโมดูล A, B, C, D แต่ความจริงแล้วโมดูล B ต้อง

เรียก ใช้ข้อมูลจากโมดูล E, F ที่อยู่ในระดับล่าง ซึ่งยังไม่ถึงรอบการทดสอบ ปัญหาคือ การทดสอบ อาจจะไม่สามารถเห็นผลลัพธ์ได้ ดังนั้นจึงต้องสร้างโมดูลตัวแทนเสมือนเป็นตัวแทนของโมดูล E, F เพื่อการทดสอบชั่วคราว ส่วน Stub Testing คือกลุ่มคำสั่งสั้นๆ ที่เขียนขึ้นมาเพื่อเป็นโมดูล ตัวแทนในการทดสอบโปรแกรม

2.5 การทดสอบรวม (System Testing) เป็นการทดสอบโปรแกรมที่มีวิธีการคล้าย กับการทดสอบแบบเพิ่มโมดูล แตกต่างกันตรงที่การทดสอบแบบเพิ่มโมดูลจะทดสอบโดยใช้โมดูล เพิ่มไปเรื่อย ๆ จนกระทั่งครบทุกโมดูลของโปรแกรมของระบบงานว่า โปรแกรมทุกโปรแกรมเมื่อ ทำงานร่วมกันแล้วจะให้ผลลัพธ์ที่ถูกต้องหรือไม่ นอกจากนี้แล้วการทดสอบรวมยังเป็นการทดสอบ ระบบงานว่า สามารถทำงานให้ผลลัพธ์ที่มีประสิทธิภาพเป็นที่ยอมรับได้หรือไม่ และเพื่อทำให้ มั่นใจได้ว่าระบบงานนั้นสามารถตอบสนองความต้องการของผู้ใช้ได้อย่างตรงจุดมากที่สุด

กรณีทดสอบและสถานการณ์ทดสอบ (Test case and Test scenario)

Test case คือ กรณีที่ใช้ในการทดสอบ ซึ่ง Test case จะอิงกับ Business requirement ที่ได้จากการเริ่มเก็บ Requirement จากลูกค้าภายในหน่วยงานหรือองค์กรที่เราต้องการเข้าไปทำการ Implement ระบบ การเขียน Test case จะต้องมีการกำหนดสถานการณ์จำลองที่จะต้องเกิด ซึ่งต้องครอบคลุมถึงกรณีที่เป็น Negative หรือกรณีที่ทำรายการผิด กรอกข้อมูลผิดพลาดเพื่อทดสอบ Response ที่ระบบจะแจ้งเตือนกับ User สถานการณ์ดังกล่าวเรียกว่า Test scenario

Test script คือ ขั้นตอนในการทดสอบ หรือการบอก Step ของ Activity ต่าง ๆ ที่จะกระทำกับระบบแล้วเกิดผลลัพธ์ เช่น User ต้องกรอกข้อมูลลงในฟิลด์ก่อน แล้วคลิกปุ่ม submit ซึ่งเอกสาร Test script จะเป็นตัวที่บ่งบอกผลการทดสอบระบบทั้งหมดนั่นเอง

2.8 งานวิจัยที่เกี่ยวข้อง

นายณัฐวุฒิ แย้มมาค (2564) ได้พัฒนาระบบแนะนำแหล่งท่องเที่ยวในเขตทหารของประเทศไทย ของ ซึ่งเป็นแอพพลิเคชันบนระบบปฏิบัติการ แอนดรอยด์ (Android OS) และมีจุดประสงค์ในการเผยแพร่ข้อมูลเกี่ยวกับแหล่งท่องเที่ยวภายในเขต ทหารที่น่าสนใจในในแต่ละทุกภาคส่วนในประเทศ แอพพลิเคชั่นนี้มีการใช้ในรูปแบบ ได้แก่ภาษาไทย โดยการทำงานของแอพพลิเคชันจะมีเนื้อหาต่างๆ เกี่ยวกับแหล่งท่องเที่ยวที่มีการแบ่งภาคส่วน มี ทั้งหมด 3 ส่วนได้แก่ กรมทหารบก กรมทหารเรือ และกรมทหารอากาศ แต่ละประเภทแสดงแหล่ง ท่องเที่ยวหลักและจักน่าสนใจโดยรอบ โดยแสดงข้อมูลรายละเอียดของพื้นที่ เช่น ประวัติของพื้นที่ แหล่งท่องเที่ยวที่น่าสนใจ รวมทั้งรูปภาพ วิดีโอตัวอย่าง และตำแหน่งของพื้นที่ที่นำทางได้ ในแอพ พลิเคชันจะมีการเรียกใช้งาน Google Map เพื่อนำทางไปยังตำแหน่งของแหล่งท่องเที่ยวนั้นๆ อีกทั้ง ผู้ใช้สามารถโหวตให้คะแนนความพึงพอใจของแหล่งท่องเที่ยวที่ได้ชมอีกด้วย การพัฒนาระบบแนะนำ แหล่งท่องเที่ยวในเขตทหารของประเทศไทย บนสมาร์ทโฟนระบบปฏิบัติกำรแอนดรอยด์ ผู้พัฒนาได้ ใช้ภาษา Java ในการสั่งการทำงานของแอพพลิเคชั่น และใช้ภาษา Extensible Markup Language (XML) ในการออกแบบและจัดรูปแบบหน้าจอแอพพลิเคชั่นบนโปรแกรมแอนดรอยด์สตูดิโอ (Android Studio) และใช้ระบบ GPS และ Google Map เป็นส่วนหนึ่งในการพัฒนา Copyright ©

นายอภิวัฒน์ เที่ยงจันตา (2565) ได้พัฒนาแอพพลิเคชันแนะนำสถานที่ท่องเที่ยวและที่ พักในอำเภอจอมทอง จังหวัด เชียงใหม่บนระบบปฏิบัติการแอนดรอยด์ มีวัตถุประสงค์เพื่อ พัฒนาแอพพลิเคชันแนะนำสถานที่ ท่องเที่ยวและที่พัก เครื่องมือที่ใช้ในการศึกษา ประกอบด้วย ภาษา ไอออนิก เฟรมเวิร์ค (Ionic Framework) ภาษา แองกูล่า (Angular) ภาษา โหนดเจเอส (Node.js) ระบบปฏิบัติการ Windows และ ระบบปฏิบัติการ แอนดรอยด์ (Android) ระบบฐานข้อมูล MySQL ระบบเครือข่ายอินเทอร์เน็ต JavaScript และ จากการศึกษาพบว่า ระบบที่พัฒนาขึ้นสามารถอำนวยความสะดวกให้กับนักท่องเที่ยวได้อย่างถูกต้องและรวดเร็วขึ้น ทั้งยังลดปัญหาการใช้เอกสารแนะนำสถานที่ต่าง ๆ ซึ่งสามารถลดการใช้กระดาษได้มากขึ้น ในส่วน ของข้อมูลและรายละเอียดของสถานที่ท่องเที่ยวต่าง ๆ นักท่องเที่ยวสามารถตรวจเช็คเวลาเปิดปิดของสถานที่นั้น ๆได้ตลอด 24 ชั่วโมง ทำให้เกิดความ สะดวกรวดเร็ว รวมทั้งสามารถลดการสิ้นเปลืองทรัพยากรต่าง ๆ ได้เป็นอย่างดี แอพพลิเคชั่นแนะนำ สถานที่ท่องเที่ยวและที่พักในอำเภอจอมทอง จังหวัดเชียงใหม่บนระบบปฏิบัติการแอนดรอยด์ ที่ พัฒนาขึ้นเน้นเกี่ยวกับการแสดงรายละเอียดและสามารถนำทางไปยังสถานที่ต่าง ๆ สามารถนำไปใช้ ได้จริงและคาดว่าจะช่วยอำนวยความสะดวกให้แก่นักท่องเที่ยวสามารถได้ใช้งานแอพพลิเคชันนี้และ เพื่อเป็นต้นแบบในการพัฒนาแอพพลิเคชันแนะนำสถานที่ท่องเที่ยวและที่พักในอำเภอจอมทอง จังหวัดเชียงใหม่บนระบบปฏิบัติการแอนดรอยด์ ต่อไปในอนาคต

นายภราดร ชมใจ (2565) ได้พัฒนาแอพพลิเคชันแนะนำโรงพยาบาลสัตว์ในเขตตัวเมืองเชียงใหม่ ซึ่งเป็นแอปพลิเคชันบนระบบปฏิบัติการ แอนดรอยด์ (Android OS) มีวัตถุประสงค์เพื่ออำนวยความสะดวกให้แก่เจ้าของสัตว์เลี้ยงที่ต้องการนำสัตว์เขาใช้ บริการทโรงพยาบาลสัตว์ โดยการทำงานของแอปพลิเคชันจะมีเนื้อหาเกี่ยวกับ โรงพยาบาลสตัวทั้งหมด 15 แห่ง ในเขตตัวเมืองจังหวัดเชียงใหม่ ซึ่งจะแสดงข้อมูลของโรงพยาบาลสัตว์ บริการ เวลาเปิด เวลาปิด พิกัดตำแหน่ง ช่องทางการติดต่อ แอปพลิเคชันจะมการเรียกใช้งาน Google Map เพื่อใช้ดตำแหน่งที่ตั้งโดยรวมของ โรงพยาบาลสัตว์และใช้นำทางไปยังโรงพยาบาลสัตว์ได้ และการพัฒนาแอปพลิเคชันแนะนำโรงพยาบาลสัตว์ ในเขตตัวเมืองเชียงใหม่ บนสมาร์ทโฟนแอนดรอยด์ ผู้พัฒนาได้ใช้โปรแกรมวิชวล สตูดิโอ (Visual Studio) ซึ่งใช้ภาษา Kotlinในการออกแบบและพัฒนา

นายภานุวัฒน์ ฟองเมฆ (2564) ได้พัฒนาแอพพลิเคชั่นแนะนำสถานที่ท่องเที่ยวใน อำเภอแม่สะเรียง จังหวัดแม่ฮ่องสอน บนปฏิบัติการแอนดรอยด์ ผู้วิจัยได้พัฒนาขึ้นเพื่ออำนวยความสะดวกต่อผู้ใช้ โดยพัฒนาให้มีการเข้าถึงข้อมูลและรายละเอียดของสถนที่ท่องเที่ยวได้อย่างรวดเร็ว และเพื่อให้ผู้ใช้สะดวกในการเดินทางมาเที่ยวได้ด้วยต้นเอง โดยการทำงานของระบบจะทำงานผ่านเครือข่ายอินเทอร์เน็ต และแสดงผลในอุปกรณ์เคลื่อนที่ โดยใช้โปรแกรมมอนโกดีบี (MongoDB) ในการจัดเก็บฐานข้อมูลในรูปแบบฐานข้อมูลเชิงสัมพันธ์ ใช้ (API) Nest JS ในการประมวลผลการทำงานของแอพพลิเคชันในส่วนต่างๆ เช่น การดึงข้อมูลจาก เพิ่ม ลบ แก้ ไข ฐานข้อมูล ในตัวแอพพลิเคชันจะใช้ ภาษา Dart ในการพัฒนา ภาษา Dart ยังสามารถทำงานได้ทั้ง Android และ IOS การทำงาน ในตัวแอพพลิเคชันสามารถบอกเส้นทางยังสถานที่ท่องเที่ยว ที่ผู้ใช้งานสนใจได้จากการทดสอบระบบพบว่า ระบบสามารถใช้งานได้ดีและสามารถนำ ไปใช้งานได้จริง ทำ ให้มีการประชาสัมพันธ์ข่าวสารรายละเอียดต่างๆเปิดกว้างยิ่งขึ้น

นางสาวอธิชา จิตตางกูร (2563) ได้พัฒนาแอปพลิเคชันแนะนำสถานที่ท่องเที่ยวอำเภอเชียงของ จังหวัดเชียงราย บนระบบปฏิบัติการแอนดรอยด์ มีวัตถุประสงค์เพื่ออำนวยความสะดวกแก่นักท่องเที่ยวที่เข้ามาเยี่ยมชม และส่งเสริมการท่องเที่ยวให้แก่อำเภอเชียงของ จังหวัดเชียงราย โดยจะมีวิธีดำเนินการคือ ในขั้นตอนการเก็บรวบรวมข้อมูลจะค้นคว้าข้อมูลผ่านเว็บไซต์ และเอกสารที่มีเนื้อหาเกี่ยวกับระบบปฏิบัติการแอนดรอยด์ และรวบรวมข้อมูลเกี่ยวกับสถานที่ท่องเที่ยว ณ เทศบาลเชียงของ และใช้โปรแกรมแอนดรอยด์สตูดิโอ (Android Studio) ซึ่งใช้ภาษาจาวา (Java) ในการใช้งาน โดยมีเทคนิคการเรียกใช้ ไลบรารี (Library) ต่าง ๆ มาช่วยในการออกแบบแอปพลิเคชันตัวแอปพลิเคชันจะทำการเชื่อมต่อกับกูเกิล แมพ เอพีไอ (Google Maps API) เป็นตัวแสดงแผนที่ โดยผ่านทำงานของกูเกิลแมพ (Google Maps) และสามารถนำทางไปยังสถานที่ต่าง ๆ ได้อย่างถูกต้องสามารถใช้บนแอนดรอยด์เวอร์ชั่น 6.0 ขึ้นไป เมื่อพัฒนาเสร็จแล้วนำไปใช้กับกลุ่มตัวอย่างคือ นักศึกษา บุคลากรและผู้ใช้งานทั่วไปจำนวน 80 คน โดยใช้แบบสอบถามความพอใจในการใช้งานแอปพลิเคชันในการประเมินมีผลตอบรับในด้านของผู้ใช้งาน โดยรวมมีความพึงพอใจมากในการใช้งานแอปพลิเคชัน