เว็บแอปพลิเคชันสำหรับจัดการแผนภาพเพื่อการพัฒนาเครื่องมือสร้างซอร์สโค้ด

DIAGRAM MANAGEMENT WEB APPLICATION FOR SOURCE CODE GENERATION TOOL

กัมปนาท เก่งพรสกุล

สหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2558

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เว็บแอปพลิเคชันสำหรับจัดการแผนภาพเพื่อการพัฒนาเครื่องมือสร้างซอร์สโค้ด

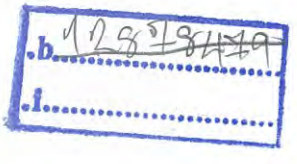
DIAGRAM MANAGEMENT WEB APPLICATION FOR SOURCE CODE GENERATION TOOL


T149252

กัมปนาท เก่งพรสกุล

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

2ฟน
ก393ว
2558
เลขหมู่
เลขทะเบียน 149252
วัน,เดือน,ปี 30 ส.ค. 2561


.b 12818479
.i

สหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2558

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

DIAGRAM MANAGEMENT WEB APPLICATION FOR SOURCE CODE GENERATION TOOL

KAMPANART KENGPOHNSAKUL

COOPERATIVE EDUCATION SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF BACHELOR OF SCIENCE (COMPUTER SCIENCE)
DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2015

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สหกิจศึกษา	เว็บแอปพลิเคชันสำหรับจัดการแผนภาพเพื่อการพัฒนาเครื่องมือสร้างซอร์สโค้ด Diagram Management Web Application for Source Code Generation Tool
ชื่อนักศึกษา	นายกัมปนาท เก่งพรสกุล รหัสนักศึกษา 55050216
ปริญญา	วิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชา	วิทยาการคอมพิวเตอร์
ปีการศึกษา	2558
อาจารย์ที่ปรึกษา	ดร.รุ่งรัตน์ เวียงศรีพนาวัลย์

คณะวิทยาศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติสหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์) ประจำปีการศึกษา 2558

คณะกรรมการสอบ	ลายมือชื่อ
ผศ.กฤษฎา บุศรา ประธานกรรมการ	
ดร.รุ่งรัตน์ เวียงศรีพนาวัลย์ กรรมการและอาจารย์ที่ปรึกษา	

ลิขสิทธิ์ของคณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อสหกิจศึกษา เว็บแอปพลิเคชันสำหรับจัดการแผนภาพเพื่อการพัฒนาเครื่องมือสร้างซอร์สโค้ด

ชื่อนักศึกษา นายกัมปนาท เก่งพรสกุล รหัสนักศึกษา 55050216

ปริญญา วิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)

ภาควิชา วิทยาการคอมพิวเตอร์

คณะ วิทยาศาสตร์

มหาวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (สจล.)

ปีการศึกษา 2558

อาจารย์ที่ปรึกษา ดร.รุ่งรัตน์ เวียงศรีพนาวัลย์

บทคัดย่อ

ในการพัฒนาแอปพลิเคชัน การออกแบบขั้นตอนการทำงานเป็นสิ่งสำคัญ และต้องพัฒนาให้ตรงตามที่ได้ออกแบบไว้ หากการขั้นตอนการออกแบบมีความล่าช้า หรือเกิดข้อผิดพลาดในการออกแบบ อาจส่งผลให้การพัฒนาล่าช้าด้วยเช่นกัน เว็บแอปพลิเคชันสำหรับจัดการแผนภาพเพื่อการพัฒนาเครื่องมือสร้างซอร์สโค้ด จะทำหน้าที่ในการช่วยลดระยะเวลาในขั้นตอนของการออกแบบ และการพัฒนาโปรแกรม โดยนักออกแบบสามารถใช้โปรแกรมที่พัฒนานี้ในการออกแบบแผนภาพตามขั้นตอนของการออกแบบ ตัวโปรแกรมสามารถทำการแปลงแผนภาพให้อยู่ในรูปแบบของข้อมูลในลักษณะเจสัน (JSON) เพื่อที่จะ ไปทำการแปลงเป็นซอร์สโค้ดในขั้นตอนต่อไป อีกทั้งแอปพลิเคชันสามารถบันทึกข้อมูลของแผนภาพที่ทำการสร้างลงฐานข้อมูล เพื่อที่สามารถนำแผนภาพที่สร้างกลับมาแก้ไขได้ ซึ่งสามารถช่วยลดระยะเวลาในการพัฒนาในอนาคตได้ แอปพลิเคชันจะอยู่ในรูปแบบของเว็บ พัฒนาบนแพลตฟอร์มที่ชื่อว่า Node.js ซึ่งเป็นแพลตฟอร์มที่ใช้ภาษาจาวาสคริปต์ (JavaScript) โดยใช้เฟรมเวิร์กที่ชื่อว่า StrongLoop ในการสร้างเซอร์เวอร์เพื่อเรียกใช้งานเซอร์วิสแบบ REST ในส่วนของการจัดเก็บข้อมูลจะใช้ฐานข้อมูล (Database) แบบ NoSQL ที่ชื่อว่า MongoDB ส่วนของการแสดงผลทางฝั่งผู้ใช้งานจะใช้ไลบรารีของภาษา JavaScript ที่ชื่อว่า AngularJS และใช้ไลบรารี GoJS ในการออกแบบและทำการแปลงรูปทรงของแผนภาพให้อยู่ในรูปแบบของ JSON

คำสำคัญ: การออกแบบโปรแกรม, เครื่องมือ, จาวาสคริปต์, ซอร์สโค้ด, เว็บแอปพลิเคชัน, แผนภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Title	Diagram Management Web Application for Source Code Generation Tool
Students	Mr. Kampanart Kengpohnsakul Student ID 55050216
Degree	Bachelor of Science (Computer Science)
Department	Computer Science
Faculty	Science
University	King Mongkut's Institute of Technology Ladkrabang (KMITL)
Academic Year	2015
Advisor	Dr.Rungrat Wiangsripanawan

Abstract

When a programmer develops an application, the design process is crucial. If the design process is delayed or errors in the design, it will delay a developing time. Diagram Management Web Application for Source Code Generation Tool will serve to reduce the design process time. A system analyst (SA) can use this application to design the diagram of the design process. The application will convert a diagram to JSON data, so the programmer can convert this data to generate source code. This JSON data can be stored in the database. The application on the server-side is developed using Node.js and StrongLoop framework to create servers for RESTful web service. In terms of storage to the database, NoSQL MongoDB has been used. The client-side uses JavaScript library called AngularJS to manage the information from the server-side. The Library called GoJS is used in the design to transform the shape of the diagram in to the JSON's form. The entire development uses the JavaScript language to develop application.

Keywords: Diagram, Generation Tool, JavaScript, RESTful, StrongLoop, Web Application

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

การดำเนินสหกิจศึกษาเกี่ยวกับการวิจัยและพัฒนาแอปพลิเคชันด้วยภาษาจาวาสคริปต์กับทางบริษัท ซีดีจี ซิสเท็ม จำกัด (CDG System ltd.) นี้ จะไม่มีทางประสบความสำเร็จได้เลย หากปราศจากบุคคลเหล่านี้ที่ให้โอกาส และคำแนะนำต่างๆ ที่เป็นแนวทางในการดำเนินงานเกี่ยวกับการวิจัยในครั้งนี้

ขอขอบพระคุณ ผศ. กฤษฎา บุศรา อาจารย์ที่ปรึกษา ที่ได้ให้คำปรึกษา คำแนะนำต่างๆในการทำสหกิจศึกษา รวมถึงการเป็นธุระในการติดต่อกับบริษัท ซีดีจี ซิสเท็ม จำกัด ทำให้มีโอกาสได้มาทำสหกิจศึกษาที่บริษัทแห่งนี้

ขอขอบพระคุณ คุณพัชรวรรณ ทันอินทรอาจ คุณอนุชา โชติวัฒนดิลก คุณวสันต์ จารุเดชารัตน์ และคุณธีรยุทธ์ ฟุ้งเกียรติไพบูลย์ ที่ได้ให้โอกาสในการเข้ามาเป็นส่วนหนึ่งของบริษัทแห่งนี้ อีกทั้งยังให้คำแนะนำในการดำเนินการวิจัย รวมถึงคำแนะนำในการใช้ชีวิตในรูปแบบของคนวัยทำงานที่จะต้องพบเจอหลังจากจบการศึกษา ทำให้การดำเนินสหกิจศึกษาสำเร็จลุล่วงไปด้วยดี และขอขอบพระคุณ คุณบุปผา จันละบุตร คุณอุไรลักษณ์ ตั้งใจกุศล และคุณสุจิตรา จินดาศรี ที่ช่วยติดต่อประสานงานในการทำสหกิจครั้งนี้

สุดท้ายนี้ขอขอบพระคุณบิดา มารดา ที่คอยให้กำลังใจ ให้คำปรึกษา และการสนับสนุนมาโดยตลอด รวมถึงเพื่อนๆ พี่ๆ ทุกท่านที่ให้คำปรึกษา ให้กำลังใจ และให้ความช่วยเหลือในกำทำสหกิจศึกษาครั้งนี้

กัมปนาท เก่งพรสกุล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความเป็นมา และความสำคัญของงาน

ในการพัฒนาโปรแกรมหรือแอปพลิเคชันขึ้นมาใช้งาน จะประกอบไปด้วยกระบวนการต่างๆ ตั้งแต่การเก็บข้อมูลตามความต้องการของผู้ใช้งาน (Requirement) การนำข้อมูลความต้องการของผู้ใช้งานมาแปลความหมายเป็นแผนภาพ (Diagram) เพื่อใช้ในการสื่อความหมาย ในการทำความเข้าใจให้ตรงกัน จากนั้น จะเป็นขั้นตอนของการลงมือเขียนโค้ด ให้สอดคล้องกับแผนภาพที่ได้ออกแบบไว้ เพื่อให้ได้โปรแกรมหรือแอปพลิเคชันที่ตรงตามความต้องการของผู้ใช้งาน หลังจากเสร็จสิ้นขั้นตอนการเขียนโปรแกรมแล้ว ต้องมีการบำรุงรักษา (Maintain) ให้มีประสิทธิภาพในการทำงาน ซึ่งกระบวนการพัฒนานี้ เรียกว่า Software Development Life Circle หรือ SDLC

ในส่วนการดำเนินงานต่างๆ จะแบ่งผู้ที่ทำงานตามหน้าที่ แต่ละคนจะมีหน้าที่ และมีความสามารถที่แตกต่างกัน การออกแบบแผนภาพเป็นส่วนที่สำคัญ เนื่องจากผู้ออกแบบ จะต้องออกแบบให้ตอบรับกับความต้องการที่รวบรวมได้ จากนั้นนักพัฒนาโปรแกรมถึงสามารถเขียนโค้ด ให้สอดคล้องกับแผนภาพนั้นๆ เนื่องจากนักพัฒนาโปรแกรมแต่ละคนอาจจะมีประสบการ์ในการเขียนโปรแกรมที่แตกต่างกัน แต่สิ่งที่สาคัญที่ทุกคนควรมีความเข้าใจที่ตรงกันคือ ตรรกะ (Logic) ต่างๆ ที่แสดงในแผนภาพจากขั้นตอนการออกแบบ เพื่อที่จะทำการเขียนโปรแกรมให้มีการทำงานในรูปแบบเดียวกัน และในการพัฒนาโปรแกรมขึ้นมา เวลาคือสิ่งสำคัญเช่นเดียวกัน ในการพัฒนาโปรแกรมจะต้องอยู่ในกรอบเวลาตามที่กำหนด เพราะฉะนั้น การทำความเข้าใจถึงแผนภาพ และตรรกะต่างๆ ให้ตรงกัน เป็นสิ่งที่สำคัญที่จะช่วยให้สามารถพัฒนาโปรแกรมได้ทันตามกรอบเวลาที่กาหนดไว้ได้

ด้วยทางบริษัท CDG System จำกัด ได้เล็งเห็นถึงความสำคัญในการพัฒนาโปรแกรมให้มีประสิทธิภาพ และตรงตามกรอบเวลาที่กาหนด จึงได้เกิดแนวคิดในการ ใช้เครื่องมือที่ช่วยอำนวยความสะดวกในการพัฒนาโปรแกรมขึ้นมา โดยเป็นเครื่องมือที่มีลักษณะในการออกแบบแผนภาพต่างๆ และสามารถสร้างซอร์สโค้ดจากแผนภาพที่ออกแบบเอาไว้ได้ โดยเครื่องมือดังกล่าว เป็นเครื่องมือที่เคยมีคนพัฒนามาอยู่ก่อนแล้ว แต่มีราคาที่มาก และรูปแบบของข้อมูลที่ได้มา ไม่ตรงกับรูปแบบของบริษัท จึงได้เกิดแนวคิดที่จะพัฒนาเครื่องมือออกแบบแผนภาพ เพื่อใช้ในการสร้างซอร์สโค้ดเป็นของตัวบริษัทเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการพัฒนาเครื่องมือดังกล่าวจะแบ่งเป็นขั้นตอนของการพัฒนาออกเป็นส่วนต่างๆ ซึ่งในขั้นตอนแรกยังอยู่ในช่วงวิจัยและทดลองนำเทคโนโลยีที่ใช้ในการสร้างมาพัฒนาให้แอปพลิเคชันเกิดขึ้น โดยทางบริษัท CDG System จำกัด ได้เล็งเห็นถึงการเติบโตของภาษา JavaScript ที่ได้มีการพัฒนาให้สามารถทำงานต่างๆได้มากกว่าการทำงานทางฝั่ง front-end และยังมีเครื่องมือ หรือ เฟรมเวิร์ก ต่างๆ ให้เลือกใช้ โดยแอปพลิเคชันที่พัฒนาในช่วงแรก จะสามารถ export ข้อมูลออกมาในรูปแบบ json โดยข้อมูลที่ได้มานั้นจะถูกนำไปใช้งานในการสร้างส่วนของภาษาที่ใช้ในการพัฒนาโปรแกรม ซึ่งเป็นขั้นต่อไปของกระบวนการพัฒนาเครื่องมือที่ใช้ในการสร้างภาษาที่ใช้ในการพัฒนาโปรแกรม

เกี่ยวกับ CDG System

บริษัท ซีดีจี ซิสเต็มส์ จำกัด ดำเนินธุรกิจเทคโนโลยีสารสนเทศในรูปแบบ System Integrator โดยเน้นการผสานเทคโนโลยีสมัยใหม่ให้เข้ากับระบบการทำงานเดิมเพื่อเสริมสร้างความสามารถและการให้บริการขององค์กร เราให้มีความเชี่ยวชาญในการพัฒนาระบบสารสนเทศสำหรับองค์กรทั้งการออกแบบเพื่อรองรับปัญหาเฉพาะด้าน เช่น ระบบจองตั๋วออนไลน์ ระบบตรวจสอบลายนิ้วมือ และการออกแบบระบบเพื่อใช้งานทั่วไป เช่น ระบบการบริหารองค์กรและหน่วยงาน ระบบงานด้านบุคลากร ระบบธุรการ เป็นต้น

บริษัท ซีดีจี ซิสเต็มส์ จำกัด ก่อตั้งขึ้นเมื่อวันที่ 7 กุมภาพันธ์ พ.ศ. 2511 ให้บริการดูแลทั้งกระบวนการทำงาน ตั้งแต่ร่วมวิเคราะห์ปัญหา ออกแบบ พัฒนา จัดหาอุปกรณ์ฮาร์ดแวร์ ซอฟต์แวร์ และเน็ตเวิร์คที่จำเป็น พร้อมจัดการอบรมให้แก่ผู้ใช้งาน ทำให้องค์กรของลูกค้าสามารถทำงานได้อย่างมีประสิทธิภาพ รวดเร็วและช่วยประหยัดค่าใช้จ่าย

สินค้าและบริการ

- General Business and Government Solutions ให้บริการพัฒนาระบบเทคโนโลยีสารสนเทศสำหรับการบริหารงานทั่วไปเพื่อใช้งานภายในองค์กรขนาดใหญ่
- Specialized Solutions บริการพัฒนาระบบเทคโนโลยีสารสนเทศแบบครบวงจรให้แก่องค์กรขนาดใหญ่สำหรับการใช้งานในรูปแบบเฉพาะทาง โดยเป็นการดูแลแบบครบวงจรในทุกขั้นตอนของการทำงานเพื่อตอบสนองความต้องการของลูกค้า (Tailor-Made Solution)
- Large and Extra Large Content Management Solutions บริการการจัดการและบริหารเนื้อหาสำหรับระบบข้อมูลขนาดใหญ่ที่มากกว่าการจัดเก็บตัวหนังสือและมีความซับซ้อนของข้อมูลสูง โดยสามารถประมวลผลได้ทั้งตัวอักษร รูปภาพ วีดีโอ ไฟล์เสียง เป็นต้น
- Document Management บริการระบบบริหารจัดการเอกสารภายในองค์กรแบบดิจิตอล
- Business Intelligence บริการออกแบบและพัฒนาระบบวิเคราะห์ข้อมูลเพื่อช่วยในการตัดสินใจของผู้บริหารทุกระดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **IT Outsource** บริการเอาท์ซอสบุคลากรด้านไอที ที่เหมาะสำหรับองค์กรที่ต้องการให้ผู้เชี่ยวชาญด้านเทคโนโลยีสารสนเทศดูแลในจุดที่ตนเองไม่มีความถนัดในลักษณะการว่าจ้างผู้เชี่ยวชาญเป็นผู้วางระบบให้พร้อมมีบุคลากรประจำอยู่ในองค์กร
- **Private Cloud** ระบบคลาวด์ภายในองค์กร สามารถใช้โปรแกรมหรือแอพพลิเคชั่นร่วมกันภายในองค์กรด้วยรูปแบบ Web-Based ซึ่งผู้ใช้งานสามารถทำงานได้โดยไม่ต้องอาศัยผู้ให้บริการภายนอก
- **Mobile Application** ให้บริการพัฒนาแอปพลิเคชั่นบนมือถือสำหรับองค์กรและภาคอุตสาหกรรมเพื่อให้การทำงานสามารถเกิดขึ้นได้ทุกที่แม้ในขณะเดินทาง

ผลงานของบริษัท CDG System กับการพัฒนาแอปพลิเคชันต่างๆ

- โครงการระบบและเชื่อมโยงข้อมูลใบอนุญาตขับรถเพื่อบันทึกประวัติผู้ได้รับใบอนุญาตขับรถและบันทึกคะแนนผู้กระทำผิดกฎหมายจราจร สำนักงานตำรวจแห่งชาติ
- โครงการจัดทำระบบคอมพิวเตอร์ตรวจสอบลายนิ้วมืออัตโนมัติ (AFIS) ซึ่งช่วยให้การทำงานของตำรวจในส่วนของการพิสูจน์ลายนิ้วมืออาชญากรมีความรวดเร็วและแม่นยำ สำนักงานตำรวจแห่งชาติ
- โครงการซื้อขายระบบสารสนเทศ สำนักงานตรวจคนเข้าเมือง สำนักงานตำรวจแห่งชาติ
- โครงการสำรองที่นั่งและจำหน่ายตั๋วโดยสารอัตโนมัติด้วยระบบคอมพิวเตอร์ การรถไฟแห่งประเทศไทย
- โครงการระบบสารสนเทศการบริหารระบบงานสารบรรณ กระทรวงคมนาคม
- โครงการซื้อขายระบบคอมพิวเตอร์และอุปกรณ์พร้อมการติดตั้งสำหรับโครงการพัฒนาระบบคลังข้อมูลและเครือข่ายของศูนย์ปฏิบัติการระดับกระทรวง (MOC) กระทรวงคมนาคม
- โครงการซื้อขายระบบคอมพิวเตอร์และอุปกรณ์สำหรับโครงการจัดหาระบบคอมพิวเตอร์เพื่อเพิ่มประสิทธิภาพงานบริการประชาชน กรมขนส่งทางบก
- โครงการจ้างพัฒนาปรับปรุงและขยายผลระบบการจัดการงบประมาณอิเล็กทรอนิกส์ (e-Budgeting) สำนักงบประมาณ
- โครงการจ้างที่ปรึกษาโครงการนำร่องระบบการตรวจสอบย้อนกลับ (Traceability) สำนักงานมาตรฐานสินค้าเกษตรและอาหารแห่งชาติ
- โครงการซื้อขายและติดตั้งระบบสารสนเทศการประชุมคณะรัฐมนตรีแบบอิเล็กทรอนิกส์ (CABNET) สำนักเลขาธิการคณะรัฐมนตรี
- โครงการพัฒนาห้องปฏิบัติการทางการบริหาร สำนักงานศาลปกครอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2 วัตถุประสงค์ของงาน

1) ทดแทนการซื้อโปรแกรมการสร้างซอร์สโค้ดที่มีราคาแพง และรูปแบบที่ไม่ตรงตามที่บริษัทต้องการ
2) เป็นการศึกษาเทคโนโลยีต่างๆ ที่สามารถนำมาพัฒนาเครื่องมือที่อำนวยความสะดวกในการพัฒนาโปรแกรม
3) ได้เรียนรู้ระบบ และขั้นตอนต่างๆ ในการพัฒนาโปรแกรมมากขึ้น
4) เป็นต้นแบบของการนำไปพัฒนาเป็นแอปพลิเคชันที่สามารถออกแบบแผนภาพ และสามารถสร้างภาษาของการพัฒนาโปรแกรมขึ้นมาได้
5) ผลลัพธ์ที่ได้จากการพัฒนาแอปพลิเคชันนี้ จะถูกนำไปใช้ในการออกแบบแอปพลิเคชันที่ใช้ในการสร้างภาษาการเขียนโปรแกรมในภายภาคหน้

1.3 ขอบเขตของงาน

1) ทำการศึกษาเทคโนโลยีของ JavaScript ที่ใช้ในการพัฒนาเว็บแอปพลิเคชัน
2) ทำการพัฒนาแอปพลิเคชันที่มีการทำงานบน web ที่สามารถสร้างแผนภาพ flowchart ได้
3) มีระบบสมัครสมาชิกในการเข้าใช้งาน
4) สามารถสร้างแผนภาพ flowchart ได้
5) สามารถจัดการข้อมูลแผนภาพที่สร้างได้
6) แผนภาพที่สร้าง สามารถบันทึกลงฐานข้อมูลได้ในรูปแบบของ JSON
7) แผนภาพที่สร้างสามารถ export เป็นไฟล์รูปภาพนามสกุล PNG ได้
8) แผนภาพที่สร้างสามารถ export เป็นไฟล์ JSON ที่มีข้อมูลเป็นรายละเอียดของแผนภาพที่สร้างได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้รู้ถึงพื้นฐานของการพัฒนา ตลอดจนโครงสร้างต่างๆ
2) สามารถนำความรู้ที่ได้ไปประยุกต์ใช้กับการพัฒนาแอปพลิเคชันในการใช้งานต่างๆ
3) สามารถนำความรู้ไปต่อยอดในการศึกษาในระดับที่สูงขึ้นไปได้
4) ผลลัพธ์ของแอปพลิเคชันสามารถนำไปพัฒนาเครื่องมือสร้างซอร์สโค้ด
5) นำความรู้ที่ได้ศึกษา ไปทำการสอนให้ผู้ที่สนใจสามารถสร้างแอปพลิเคชันเป็นของ ตัวเองได้
6) เป็นตัวเลือกเพื่อทดแทนแอปพลิเคชันที่มีผู้พัฒนาอยู่ก่อนแล้ว แต่ต้องเสียเงินจานวนมากในการใช้งาน
7) ตัวแอปพลิเคชันไม่สงวนลิขสิทธ์ ผู้ที่สนใจศึกษาในกระบวนการต่างๆของการพัฒนา สามารถนาไปใช้ หรือต่อยอดให้มีประสิทธิภาพที่ดีขึ้นได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 JSON (JavaScript Object Notation)

JSON [1] (JavaScript Object Notation) คือรูปแบบของข้อมูลขนาดเบาที่มนุษย์สามารถทำความเข้าใจ และสร้างขึ้นมาได้ง่าย อุปกรณ์อิเลกทรอนิกส์ทั้งหลายสามารถนำข้อมูลที่อยู่ในรูปแบบ JSON ไปทำการวิเคราะห์ และประมวลผลได้เช่นกัน พื้นฐานของตัว JSON นั้นมาจากภาษาที่ใช้ในการเขียนโปรแกรมที่ชื่อว่า JavaScript ซึ่งเวอร์ชันของภาษา JavaScript ที่เป็นพื้นฐานของ JSON นั้นถูกกำหนดให้เป็นมาตรฐานที่ชื่อว่า Standard ECMA-262 3rd. Edition JSON มีรูปแบบที่เป็นข้อความที่มนุษย์สามารถเข้าใจได้ง่าย (ตัวอย่างเช่น ภาษาอังกฤษ) แต่สามารถนำข้อมูลที่มีรูปแบบ JSON นี้ไปใช้ได้กับการเขียนโปรแกรมด้วยภาษาต่างๆ เช่น กลุ่มภาษา C ที่ประกอบไปด้วย C, C++, และ C#, Java, JavaScript, Perl, Python, และอีกหลายๆภาษา ด้วยคุณสมบัติที่กล่าวมานั้น ทำให้ JSON เป็นรูปแบบของข้อมูลที่นิยมในการส่งข้อมูลในแพลตฟอร์มที่แตกต่างกัน

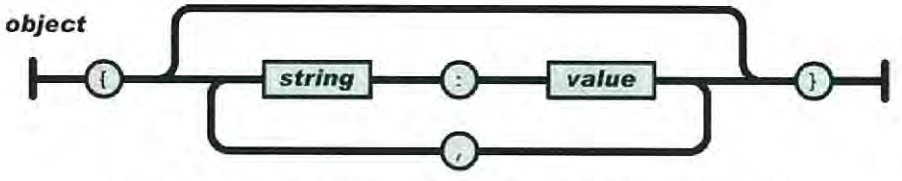
JSON สามารถสร้างได้ด้วยโครงสร้าง 2 แบบ คือ

1) กลุ่มของคุณสมบัติของข้อมูล (properties) ที่ประกอบไปด้วย ชื่อ/ค่าของคุณสมบัตินั้นๆ ซึ่งมีลักษณะเหมือนกับโครงสร้างข้อมูลแบบ Object, record, struct, dictionary, hash table หรือ keyed list ในหลายๆภาษา
2) รายการของข้อมูล มีลักษณะการเก็บข้อมูลคล้าย array, list หรือ sequence

รูปแบบของ JSON

ออบเจ็ค (Object)

JSON จะประกอบไปด้วยกลุ่มของข้อมูลในรูปแบบ ชื่อ/ค่าของคุณสมบัตินั้นๆ โดยไม่มีการเรียงลำดับ ออบเจ็คจะเริ่มต้นด้วยเครื่องหมาย “{” (ปีกกาเปิด) และสิ้นสุดด้วยเครื่องหมาย “}” (ปีกกาปิด) แต่ละชื่อของคุณสมบัติของออบเจ็คจะตามด้วยเครื่องหมาย “:” (มหัพภาคคู่) และค่าของคุณสมบัตินั้นๆ แต่ละคุณสมบัติจะแยกกันด้วยเครื่องหมาย “,” (จุลภาค)

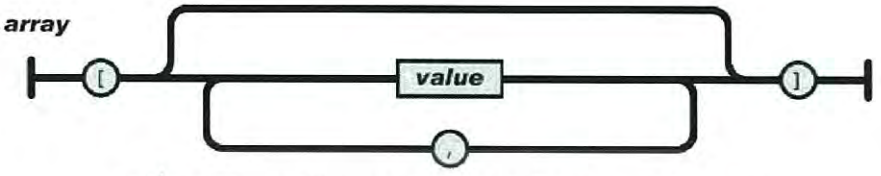


รูปที่ 2.1 รายละเอียดของ JSON file ในรูปแบบ object [1]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อาร์เรย์ (Array)

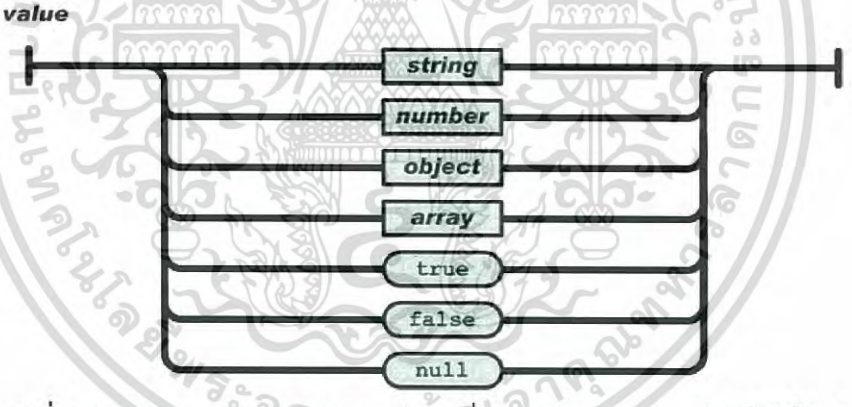
Array จะมีลักษณะเป็นกลุ่มของข้อมูลที่เริ่มต้นด้วยเครื่องหมาย “[” (นขลิขิตเปิด) และจบด้วยเครื่องหมาย “]” (นขลิขิตปิด) ข้อมูลภายใน Array จะถูกแบ่งคั่นไว้ด้วยเครื่องหมาย “,” (จุลภาค)



รูปที่ 2.2 รายละเอียดของ JSON file ในรูปแบบ Array [1]

ข้อมูล (Value)

ข้อมูลของ JSON สามารถเป็นได้หลายรูปแบบ ไม่ว่าจะเป็น สตริง (String) ที่อยู่ในเครื่องหมายคำพูด, ตัวเลข, ข้อมูลที่เป็นค่าความจริงหรือเท็จ (Boolean) , แม้กระทั่งเป็น Array หรือ Object เนื่องจากโครงสร้างดังกล่าวสามารถซ้อนทับกันได้

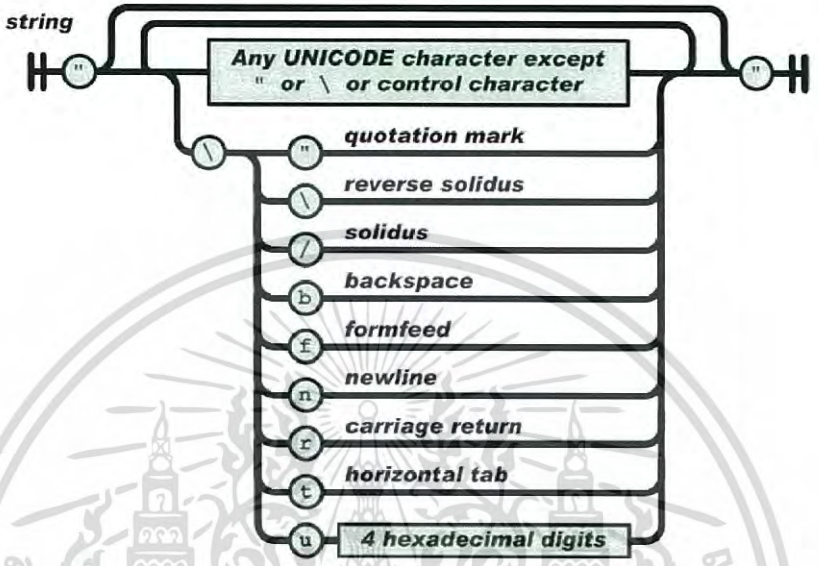


รูปที่ 2.3 รูปแบบ และประเภทของข้อมูลที่ JSON file สามารถจัดเก็บได้ [1]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สตริง (String)

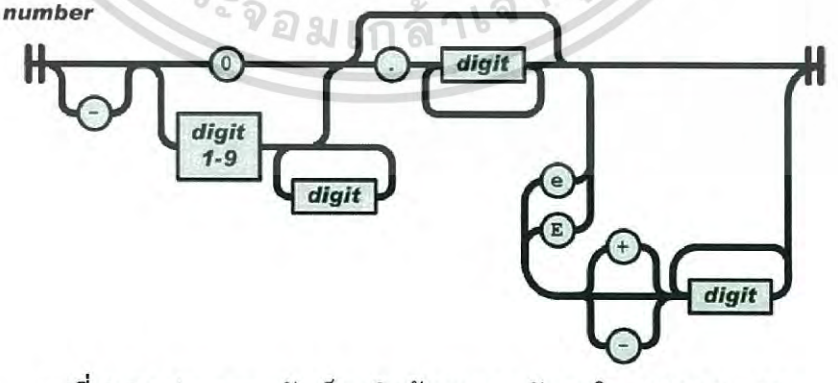
สตริง (String) จะประกอบไปด้วยกลุ่มของอักขระ Unicode ตั้งแต่ 0 ตัวขึ้นไปที่อยู่ภายในเครื่องหมายคำพูด สามารถใช้เครื่องหมาย \ (backslash) ในกรณีที่ไม่สนใจอักขระนั้นๆ



รูปที่ 2.4 รูปแบบการจัดเก็บชนิดข้อมูลแบบสตริงใน JSON file [1]

ตัวเลข (Number)

ตัวเลข (Number) ของข้อมูลในรูปแบบ JSON จะมีความคล้ายคลึงกับตัวเลขในภาษา C กับ Java ยกเว้นเพียงเลขฐาน 8 (octal) และเลขฐาน 16 (hexadecimal) จะไม่ถูกนำมาใช้



รูปที่ 2.5 รูปแบบการจัดเก็บชนิดข้อมูลแบบตัวเลขใน JSON file [1]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 เว็บเซอร์วิสแบบ REST

REST [2] (Representational State Transfer) นั้นถูกพูดถึงครั้งแรกในปี 2000 โดย Roy Thomas Fielding ซึ่ง REST (Representational State Transfer) นั้นเป็น Architecture (สถาปัตยกรรมการสื่อสารข้อมูล) รูปแบบหนึ่งที่ใช้แผร่กระจายสื่อ เป็นแนวทางใหม่ในการสร้าง เว็บเซอร์วิสแบบเรียบง่าย ซึ่งเว็บเซอร์วิสที่สร้างด้วยสถาปัตยกรรมแบบ REST นั้นเรียกว่า RESTful เว็บเซอร์วิส โดยเรียกใช้ผ่านทาง HTTP Method GET / POST / PUT / DELETE และส่งข้อมูลออกมาได้หลายรูปแบบ ไม่ว่าจะเป็น XML, JSON ทำให้ปริมาณข้อมูลที่รับส่ง น้อยกว่าการใช้ Protocol SOAP อยู่มาก ซึ่งข้อดีข้อนี้ของ REST ทำให้ Developer หลายๆ คนหันมาสนใจการเขียนโปรแกรมแบบใช้ RESTful กันมากขึ้น เพราะมีผลกับเรื่องประสิทธิภาพของการใช้งานโปรแกรมเป็นอย่างมาก แต่เนื่องจากเรื่อง REST นี้เพิ่งเกิดขึ้นมาเมื่อปี 2000 ทำให้ยังไม่มีมาตรฐานที่กำหนดให้บังคับใช้งานเหมือน Protocol SOAP เดิม ซึ่งจะสังเกตุได้ว่าจะไม่มี REST Specification อยู่บน W3C และไม่มี REST Developer Toolkit ถ้าหากอยากใช้ RESTful ก็ต้องกำหนดเงื่อนไขจากภาษาที่เขียนเอาเอง โดยภาษาในยุคปัจจุบันนี้มีการรองรับ RESTful กันเป็นส่วนใหญ่แล้ว

หลักการของ REST

- **ข้อมูล (Resources)** จะถูกดำเนินการจาก Directory ตามโครงสร้าง URIs ที่สามารถเข้าใจได้ง่าย
- **ส่วนของการแสดงผล (Representations)** ข้อมูลจะถูกส่งมาในรูปแบบของ JSON หรือ XML เพื่อใช้ในการแสดงผล
- **ข้อความ (Messages)** ที่ใช้ในการส่งข้อมูลจะใช้วิธีการ (Method) ของโพรโทคอล HTTP ที่เปิดให้ใช้ (ตัวอย่างเช่น วิธีการแบบ GET, POST, PUT, และ DELETE)
    - **ไม่คงสถานะของการเรียกใช้ (Stateless)** ระหว่าง ไคลเอนต์กับเซิร์ฟเวอร์ ตัวเซิร์ฟเวอร์จะไม่จดจำสถานะของการดำเนินการกับข้อมูล (Request)

2.2.1 รูปแบบการทำงานของ HTTP

ในการดำเนินการกับ REST API ที่ประกอบไปด้วย การสร้าง (CREATE), การเรียกดู (RETRIEVE), การแก้ไข (UPDATE), และการลบข้อมูลทิ้ง (DELETE) จะถูกใช้เรียกใช้งานตามรูปแบบการทำงาน (Methods) ของโพรโทคอล HTTP ซึ่งเกิดจากการร้องขอ (Request) จากผู้ใช้งาน ซึ่งประกอบไปด้วย

GET Method

GET เป็น Method ของ Protocol HTTP ที่ใช้ในการเรียกดูข้อมูล แต่ละ Request แบบ GET ควรจะต้องปลอดภัย และไม่มีจุดประสงค์อื่นแฝงเข้ามา request แบบ GET จะไม่สนใจว่ามีการเรียกใช้งานไปกี่ครั้ง ถ้าพารามิเตอร์ที่ส่งไปพร้อมกับ request เป็นตัวเดิม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลลัพจากการเรียกใช้ก็จะเหมือนเดิมตลอด request แบบ GET จะไม่ทำให้การดำเนินงานของระบบมีปัญหา นอกจากนี้ แต่ละ request สามารถมีเงื่อนไขในการเรียกดูข้อมูลได้

ตัวอย่างการใช้งาน

เรียกดูข้อมูลที่อยู่ (address) ด้วยรหัส (id) 1

GET /addresses/1

POST Method

POST เป็น Method ของ Protocol HTTP ที่ใช้สร้างข้อมูลตามตำแหน่งที่อยู่ของกลุ่มข้อมูล โดยจะเป็นการเพิ่มข้อมูลเข้าไปในกลุ่มข้อมูลดังกล่าว ซึ่งข้อมูลที่ถูกเพิ่มมาจาก request ที่ส่งมา

ตัวอย่างการใช้งาน

สร้างข้อมูลที่อยู่ (address)

POST /addresses

PUT Method

PUT เป็น Method ของ Protocol HTTP ที่ใช้ในการแก้ไขข้อมูลตามตำแหน่งเฉพาะ นั่นคือ ข้อมูลที่ถูกส่งมาแก้ไขจะไปแทนที่ข้อมูลเดิมตามที่อยู่ของข้อมูล ถ้าหากที่อยู่ดังกล่าวไม่มีข้อมูล หรือทรัพยากรใดๆ PUT Method จะทำการสร้างข้อมูลตามที่อยู่นั้นโดยใช้ข้อมูลจาก request ที่ส่งมา

ตัวอย่างการใช้งาน

สร้างข้อมูลที่อยู่ (address)

PUT /addresses/1

ข้อสังเกต : POST กับ PUT จะต่างกันตรงที่ ตำแหน่งของข้อมูล โดยที่ Method แบบ POST จะไม่เจาะจงรายละเอียดของข้อมูล จะสนเพียงแค่ว่าเป็นกลุ่มข้อมูลใด ในขณะที่ Method แบบ POST นั้น เจาะจงตำแหน่งของข้อมูลที่ต้องการเข้าถึง

DELETE Method

DELETE Method เป็น method ของ protocol HTTP ที่ใช้ในการค้นหาข้อมูลที่ผู้ใช้ต้องการ จากนั้นจึงทำการลบข้อมูลนั้นทิ้งไป

ตัวอย่างการใช้งาน

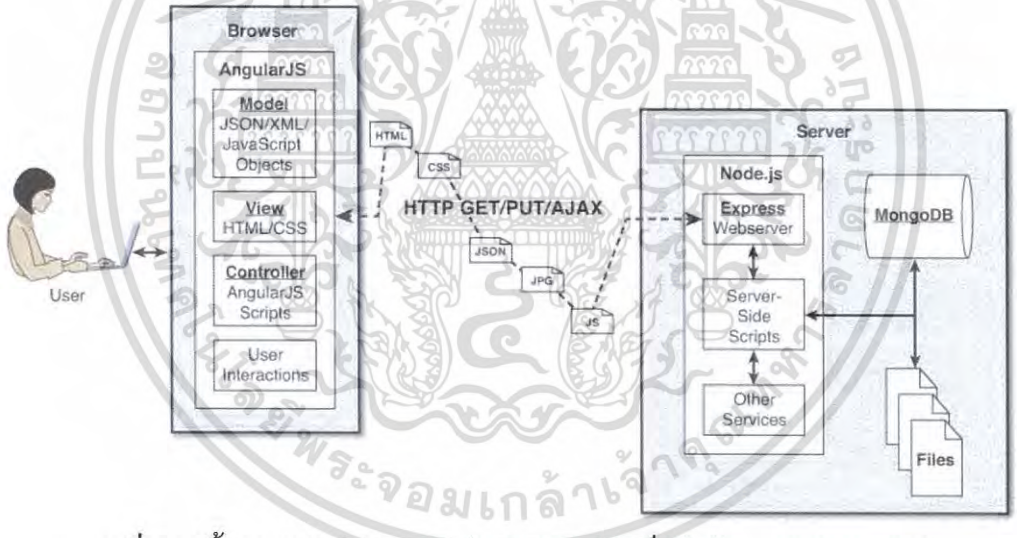
DELETE address/1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 การพัฒนาเว็บแอปพลิเคชันด้วย JavaScript

ภาษา JavaScript ได้เตรียม สภาพแวดล้อม และ framework ต่างๆ ที่สามารถใช้ในการพัฒนาเว็บแอพลิเคชันเอาไว้ โดยการพัฒนาเว็บแอปพลิเคชันด้วยภาษา JavaScript ที่นิยมในตอนนี้นั้นจะประกอบไปด้วย framework ต่างๆเช่น Node.js, AngularJS, Express และ MongoDB เป็นต้น [3]

- Node.js นั้นได้เตรียมสภาพแวดล้อมพื้นฐานในการพัฒนาเว็บแอปพลิเคชันเอาไว้ backend service และ server-side script ต่างๆนั้นล้วนถูกเขียนขึ้นด้วย Node.js
- MongoDB จะเป็นที่เอาไว้สำหรับเก็บข้อมูลของเว็บ ซึ่งจะสามารถเข้าถึงข้อมูลได้โดยผ่าน MongoDB driver ซึ่งเป็น module ของ Node.js
- เว็บเซิร์ฟเวอร์ จะถูกกำหนดโดย Express ซึ่งเป็นส่วนหนึ่งของ Node.js เช่นเดียวกัน
- การแสดงผลในเบราเซอร์จะถูกสร้างและควบคุมโดย AngularJS เฟรมเวิร์ก ซึ่งเป็น MVC (Model-View-Controller) เฟรมเวิร์กที่ช่วยเพิ่มความสามารถในการแสดงผลทางฝั่งไคลเอนต์ได้อย่างมีประสิทธิภาพ



รูปที่ 2.6 พื้นฐานการทำงานของเว็บแอปพลิเคชันที่สร้างโดย JavaScript [3]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4 Node.js

Node.js [4] คือ แพลตฟอร์มสำหรับการพัฒนาแอปพลิเคชันด้วยภาษา JavaScript ที่มีพื้นฐานการทำงานอยู่บน V8 JavaScript Engine Node.js เขียนด้วยภาษา JavaScript และจะถูก compile ให้เป็นภาษาเครื่องด้วย V8 Compiler นักพัฒนาสามารถใช้ภาษา JavaScript มาพัฒนาแอปพลิเคชันต่างๆด้วย Node.js ได้ ตัวอย่างเช่น การเขียนโปรแกรมทางฝั่งเซิร์ฟเวอร์เป็นต้น

Node.js ได้ถือกำเนิดขึ้นในปี 2009 โดยผู้พัฒนาก็คือ Ryan Dahl ซึ่งในตอนต้น Node.js ถูกพัฒนามาด้วยแนวคิดที่จะนำมาแก้ปัญหาความซับซ้อนที่เกิดจากการเข้าใช้งานแอปพลิเคชันพร้อมๆกัน โดยเฉพาะอย่างยิ่งเมื่อต้องทำงานกับเว็บเซิร์ฟเวอร์ต่างๆ จากนั้นเมื่อบริษัท Google ได้ทำการพัฒนา V8 JavaScript ขึ้นมาสำหรับใช้ในเว็บเบราเซอร์ที่ชื่อว่า Chrome ซึ่งเป็นตัวช่วยจัดการการทำงานต่างๆของเว็บได้เป็นอย่างดี Dahl จึงได้พัฒนา Node.js ให้สามารถทำงานได้บน V8 เพื่อที่จะให้ JavaScript สามารถทำงานได้ทางฝั่งเซิร์ฟเวอร์โดยมีสภาพแวดล้อมที่เหมือนกับฝั่งไคลเอนต์

ผลลัพธ์จากการพัฒนา Node.js บน V8 นั้นทำให้เซิร์ฟเวอร์สามารถรองรับการทำงานที่เพิ่มขึ้นได้เป็นอย่างดี เนื่องจาก Node.js จะช่วยลดช่องว่างของการพัฒนาทางฝั่งเซิร์ฟเวอร์กับไคลเอนต์ เนื่องจาก Node.js เขียนขึ้นด้วยภาษา JavaScript ทำให้นักพัฒนาสามารถเขียนสคริปต์ต่างๆในรูปแบบที่คล้ายกันทั้งฝั่งเซิร์ฟเวอร์และไคลเอนต์ และสามารถนำโค้ดจากสภาพแวดล้อมที่ต่างกันมาใช้งานซ้ำได้

Node.js เหมาะสำหรับอะไร

Node.js สามารถนำมาใช้งานได้หลากหลายรูปแบบ เนื่องจากว่ามันถูกสร้างมาให้ทำงานบน V8 JavaScript และสามารถปรับแต่งโค้ดให้ทำการควบคุม HTTP traffic ซึ่งเป็นการทำงานพื้นฐานของเว็บเซิร์ฟเวอร์ อย่างไรก็ตาม Node.js ยังสามารถนำไปใช้สำหรับบริการต่างๆเกี่ยวกับเว็บได้อีกด้วย เช่น

- เว็บเซอร์วิสต่างๆ เช่น REST เป็นต้น
- Real-time multiplayer games
- Back-end เว็บเซอร์วิส ตัวอย่างเช่น cross-domain, server-side request เป็นต้น
- การพัฒนาแอปพลิเคชันที่มีเว็บเป็นพื้นฐาน
- การสื่อสารของไคลเอนต์จำนวนมาก ตัวอย่างเช่น Instant messenger ต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.1 คุณสมบัติของ Node.js

การพัฒนาตั้งแต่ต้นจนจบใช้ภาษา JavaScript

สิ่งที่สำคัญของ Node.js มากที่สุดนั้นคือ มันได้ถูกออกแบบมาเพื่อให้ภาษา JavaScript สามารถทำงานได้ทั้งทางฝั่งเซิร์ฟเวอร์และฝั่งไคลเอนต์ ซึ่งการพัฒนาแอปพลิเคชันแบบก่อนๆนั้นเป็นสิ่งที่ยากในการตัดสินใจที่จะกำหนดการทำงานต่างๆเอาไว้ใน Client-side script หรือ Server-side script ตัวของ Node.js จะสามารถนำภาษาJavaScript ที่เขียนได้ง่ายทางฝั่งไคลเอนต์มาปรับใช้กับทางฝั่งเซิร์ฟเวอร์ได้หลากหลายวิธี รวมถึงความสามารถที่ฝั่งไคลเอนต์และเซิร์ฟเวอร์นั้น ทำการสื่อสารเป็นภาษาเดียวกัน ดังนั้น นาย b สามารถร่วมวงสนทนากับคนหลายๆคนได้พร้อมกัน และจะมีทำการโต้ตอบกับคนที่พูดคุยด้วย จะไม่มีการแบ่งการทำงานของสมอง เพราะว่าไม่มีร่างแยก ในกรณีที่นาย b ถูกถามด้วยคำถามที่ต้องใช้เวลาในการคิด นาย b ก็ยังสามารถที่จะโต้ตอบกับผู้อื่นไปพร้อมกับการประมวลผลคำถามภายในความคิดของตัวเอง ซึ่งคล้ายกับการควบคม request แบบ blocking I/O ในการทำงานพื้นหลังของ thread pool ซึ่งการทำงานใน thread pool จะส่งผลน้อยมากกับการประมวลผลของ thread หลักที่ใช้ในการประมวลผลของ event

เพิ่มความสามารถด้วย Event-Driven

Node.js มีกระบวนการควบคุมการใช้งานของ User (Request) ที่ทำการควบคุมโดยตัวควบคุมเพียงหนึ่งเดียว ซึ่งแตกต่างจากการประมวลผลแบบหลาย thread โดยการนำรูปแบบพื้นฐานของการจับเหตุการณ์ ซึ่งเป็นพื้นฐานการทำงานของภาษา JavaScript ในการรองรับการขยายของจำนวนผู้ใช้งานในแบบที่เว็บเซิร์ฟเวอร์สมัยก่อนทำไม่ได้

รองรับการขยายตัวที่เพิ่มมากขึ้น

Node.js มีนักพัฒนาจำนวนมาก และมีกลุ่มชุมชนที่จะคอยทำการคิดค้น module ของการทำงานต่างๆให้กับ Node.js อยู่เสมอ และ module ของ Node.js นั้นก็สามารถติดตั้งได้อย่างง่ายดายอีกด้วย

การพัฒนาที่เร็วขึ้น

การติดตั้งและพัฒนาแอปพลิเคชันด้วย Node.js นั้นสามารถทำได้ง่าย โดยใช้เวลาในการติดตั้งไม่นานก็สามารถดำเนินการกับเว็บเซิร์ฟเวอร์ได้ทันที

2.4.2 การทำงานของ Node.js

Node.js นั้นมีการทำงานในรูปแบบ event-driven ซึ่งเป็นรูปแบบที่สำคัญที่ช่วยให้สามารถรองรับการขยายของงานที่เพิ่มมากขึ้น และยังช่วยเพิ่มประสิทธิภาพของการทำงานให้ดียิ่งขึ้น ซึ่งแตกต่างกับการพัฒนาในรูปแบบดั้งเดิมที่ใช้รูปแบบของ thread ในการพัฒนาเว็บเซิร์ฟเวอร์

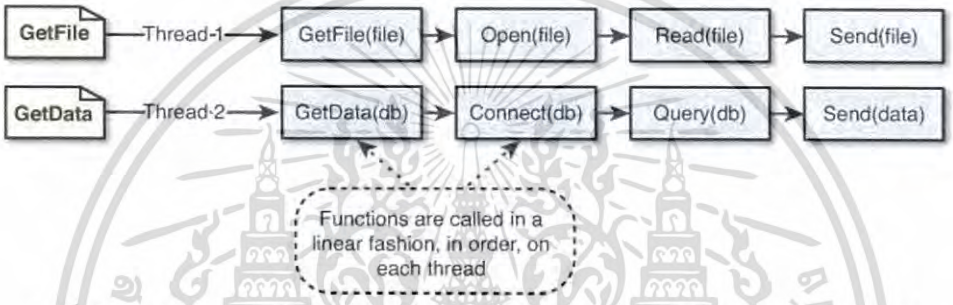
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Node.js Event Model

แอปพลิเคชันที่พัฒนาด้วย Node.js จะทำงานด้วย thread เดียว โดยใช้รูปแบบของ event-driven ในการทำงานของ thread ถึงแม้ว่า Node.js จะสร้าง thread pool ที่เอาไว้ใช้เป็นพื้นหลังการทำงาน แต่แอปพลิเคชันที่พัฒนาด้วย Node.js ก็ไม่ได้นำแนวคิดของแอปพลิเคชันแบบการทำงานหลาย thread มาใช้

Event Callback กับ รูปแบบการทำงานแบบ Thread

รูปแบบดั้งเดิมของการทำงานแบบ thread ผู้ใช้งานจะทำการส่ง request จะเข้ามายังเว็บเซิร์ฟเวอร์และจะกำหนดให้แต่ละ thread ที่สามารถทำงานได้รับผิดชอบแต่ละ request จนกระทั่งการทำงานเสร็จและสร้าง response คืนกลับไปยังผู้ใช้งาน



รูปที่ 2.7 เปรียบเทียบ Event Callback กับ รูปแบบการทำงานแบบ Thread [4]

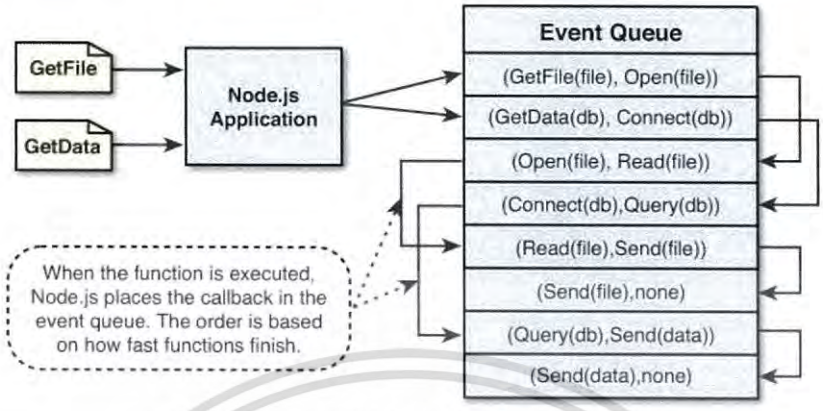
จากรูปที่ 2.7 แสดงให้เห็นลำดับการทำงานในรูปแบบ thread ทำการประมวลผลจำนวน 2 request นั่นคือ GetFile และ GetData โดย GetFile จะทำการ request เพื่อทำการเปิด file อ่านข้อมูล และจากนั้นส่งข้อมูลกลับไปยังผู้ request โดยการสร้าง response ซึ่งทุกการทำงานทั้งหมดจะถูกทำงานโดย thread เพียง thread เดียว และส่วนของ GetData จะทำการส่ง request เพื่อทำการเชื่อมต่อไปยังฐานข้อมูล ทำการค้นหาข้อมูลและส่งข้อมูลนั้นคืนกลับไปยังผู้ request โดยการสร้าง response

กระบวนการทำงานทั้งหมดนั้น สามารถทำโดยโดย Node.js เช่นเดียวกัน โดย Node.js จะแทนที่การทำงานโดยใช้รูปแบบของ event ทำงานเหล่านั้นแทนที่การใช้ thread โดย Node.js จะทำการเพิ่มงานเข้าไปใน event queue จากนั้น จะมี thread เพียงหนึ่ง thread ทำหน้าที่ในการนำ event ที่อยู่ในลำดับแรกสุดของ event queue ไปทำการประมวลผล โดยกระบวนการนำแต่ละ event ไปประมวลผลจะวนรอบไปเรื่อยๆ โดยเรียกกระบวนการวนรอบดังกล่าวว่า event loop

แต่ละ event ที่ทำการประมวลผลนาน หรืออยู่ในรูปแบบของ blocking I/O จะไม่ถูกสั่งให้ทำงานโดยตรงผ่าน event loop แต่จะเป็นการเพิ่มฟังก์ชันไปยัง event queue โดยฟังชันดังกล่าวจะเรียกใช้งาน event เหล่านั้น พร้อมกับ callback กลับมา เมื่อการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำงานของฟังก์ชันนั้นเสร็จสิ้น หลังจากที่ Node.js ดำเนินการกับ event ทั้งหมดที่อยู่ใน event queue เสร็จจะถือว่าเป็นการสิ้นสุดกรทำงานของNode.js



รูปที่ 2.8 การทำงานของ Node.js ที่ทำการควบคุม request [4]

จากรูปที่ 2.8 แสดงการทำงานของ Node.js ที่ทำการควบคุม request คือ GetFile และ GetData โดย Node.js จะทำการเพิ่ม GetFile และ GetData ลงใน event queue ของ event ในขั้นตอนแรก Node.js จะทำการนำ request แบบ GetFile ไปดำเนินการ และเมื่อสิ้นสุดการดำเนินการจะทำการเพิ่มฟังก์ชัน open() ซึ่งเป็น callback กลับไปยัง event queue ของ event จากนั้นจึงทำการนำ request แบบ GetData ไปดำเนินการ และเมื่อสิ้นสุดการดำเนินการจะทำการเพิ่มฟังก์ชัน connect() ซึ่งเป็น callback กลับไปยัง event queue ของ event เช่นกัน จากนั้น event loop จะทำการสั่งให้ฟังก์ชัน callback ทำงาน ซึ่งกระบวนการจะดำเนินไปเรื่อยๆจนกระทั่งไม่มี callback ให้ดำเนินการอีก ตัวอย่างในรูปแสดงให้เห็นถึง Connect ที่มีการทำงานที่นานกว่า Read ดังนั้น Send(file) จึงถูกเรียกใช้งานก่อน Query(db)

Blocking I/O ใน Node.js

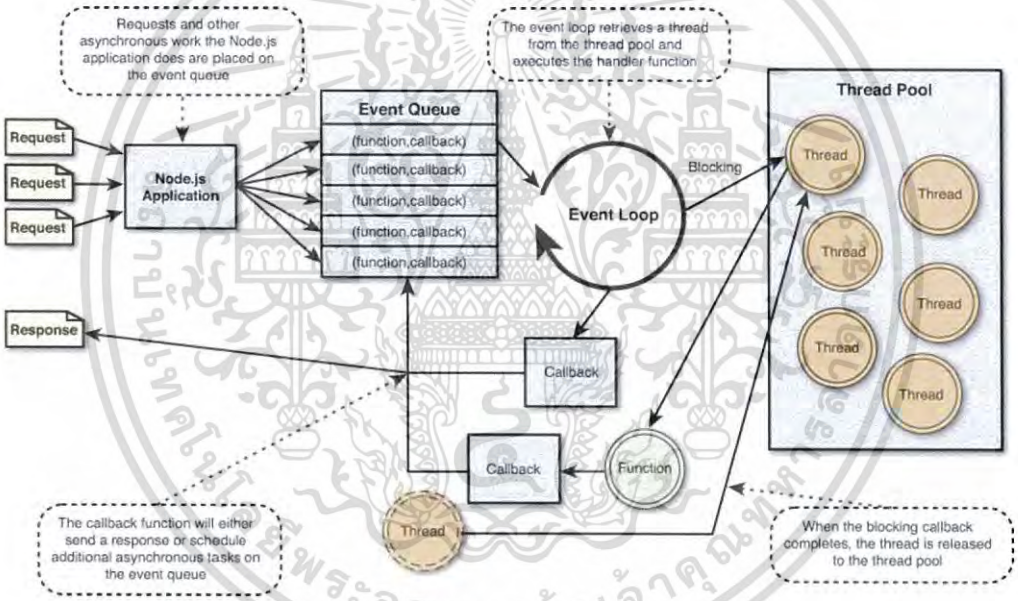
การทำงานของ Node.js โดยใช้รูปแบบของ event นั้นจะเกิดปัญหาขึ้นในกรณีที่ฟังก์ชันที่จะดำเนินการด้วย Node.js เป็นฟังก์ชันแบบ Blocking I/O ซึ่ง ฟังก์ชันแบบ Blocking I/O จะหยุดการทำงานของ thread ที่สั่งดำเนินการฟังก์ชันนั้น และรอให้ให้การ response ก่อนที่จะดำเนินการต่อไป ตัวอย่างของฟังก์ชันแบบ Blocking I/O เช่น

- อ่านไฟล์
- การค้นข้อมูลจากฐานข้อมูล
- การร้องขอใช้งาน socket
- การเข้าถึงเซอร์วิส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Node.js จะใช้ event callback ในการจัดการกับ Blocking I/O ดังนั้น request ต่างๆที่มีการทำงานแบบ Blocking I/O นั้นจะถูกทำงานที่ thread อื่นๆ ซึ่งทำงานอยู่บนพื้นหลังที่ Node.js สร้าง thread pool ไว้ เมื่อ thread .ใน event loop นั้นรับ event ที่มีรูปแบบ Blocking I/O จาก event queue Node.js จะทำการเรียกใช้ thread จาก thread pool และดำเนินการกับ ฟังก์ชัน แบบ blocking I/O นั้น แทนการใช้ thread ของ event loop ดำเนินการ เพื่อป้องกันการหยุดทำงานของ event loop โดยเรียก thread ที่ทำหน้าที่นี้ว่า blocking thread

ฟังก์ชันที่ทำงานอยู่บน blocking thread จะสามารถเพิ่ม event กลับไปยัง event queue เพื่อทำการประมวลผลต่อไปได้ ตัวอย่างเช่น การเรียกข้อมูลจากฐานข้อมูล โดยทั่วไปจะทำการส่ง ฟังก์ชัน callback ที่ทำการเก็บผลลัพธ์เอาไว้ไปยัง event queue ก่อนที่ส่ง response ไปยังผู้ใช้งาน



รูปที่ 2.9 การทำงานของ Node.js ที่มีลักษณะการทำงานในรูปแบบ event [4]

จากรูปที่ 2.9 แสดงให้เห็นถึงการทำงานของ Node.js ที่มีลักษณะการทำงานในรูปแบบ event ซึ่งประกอบไปได้การทำงานของ event queue, event loop และ thread pool สังเกตได้ว่า event loop จะทำหน้าที่ทั้งดำเนินการกับฟังก์ชันโดยใช้ thread ของ event loop ได้ทันที กับการสั่งให้ดำเนินการกับฟังชันแบบ blocking I/O บน blocking thread

การควบคุมข้อมูล I/O ใน Node.js

เว็บแอปพลิเคชัน และเซอร์วิสส่วนใหญ่มักจะมีข้อมูลจำนวนมากมายไหลเวียนอยู่ในระบบ ซึ่งข้อมูลนั้นอยู่ในรูปแบบของ ข้อความ, JSON, binary buffer, และ data stream

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังนั้น Node.js จึงมีวิธีจัดการกับข้อมูลที่หลากหลาย ค่อยสนับสนุนการทำงานของข้อมูล I/O จากระบบหนึ่ง ไปสู่อีกระบบหนึ่ง

การทำงานของ Node.js กับ JSON

JSON คือประเภทของข้อมูลที่จะต้องทำงานด้วยทุกครั้งที่ทำการสร้างเว็บแอปพลิเคชันด้วย Node.js ตัวของ JSON นั้น มาจากคำว่า JavaScript Object Notation ซึ่งจะมีความคล้ายคลึงกับ object ในภาษา JavaScript แต่ JSON นั้น จะอยู่ในรูปแบบของ String โดยสามารถที่จะแปลงข้อมูลจาก JSON เป็น JavaScript object และจาก JavaScript object เป็น JSON ได้อย่างง่ายดาย JSON ยังทำงานได้ดีกับการรวมกลุ่มของข้อมูลจากไคลเอนต์แล้วส่งไปยังเซิร์ฟเวอร์

การแปลงข้อมูล JSON เป็น JavaScript object

ในภาษา JavaScript ข้อมูล JSON จะอยู่ในรูปแบบของ String ซึ่งทำให้ง่ายต่อการเขียน และสามารถทำการแปลง JSON String นั้นให้เป็น JavaScript object ได้อย่างง่ายดาย โดยใช้ method ที่ชื่อว่า JSON.parse(string) ตัวอย่างเช่น รูปภาพที่ 2.10 แสดงให้เห็นถึงการแปลง JSON ไปเป็น JavaScript object โดยกำหนดให้ accountStr เป็น String ที่อยู่ในรูปแบบของ JSON จากนั้นทำการแปลงข้อมูลด้วย method JSON.parse() และจะทำให้สามารถเข้าถึงข้อมูลของ JavaScript ได้ด้วยเครื่องหมาย จุลภาค (dot notation)

```
var accountStr = '{"name":"Jedi", "members":["Yoda","Obi Wan"],' +
                 '"number":34512, "location": "A galaxy far, faraway"}';
var accountObj = JSON.parse(accountStr);
console.log(accountObj.name);
console.log(accountObj.members);
```

รูปที่ 2.10 วิธีการแปลงข้อมูล JSON ให้อยู่ในรูป JavaScript object

ผลลัพธ์จากรูปที่ 2.10 คือ

```
Jedi
[ 'Yoda', 'Obi Wan' ]
```

รูปที่ 2.11 ผลลัพธ์ที่ได้จากการแปลง JSON เป็น JavaScript object

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การแปลง JavaScript object เป็น JSON

Node.js สามารถแปลง JavaScript object ให้อยู่ในรูปแบบของ JSON String ได้ นั่นหมายความว่าสามารถทำการเก็บข้อมูลของ JSON ไว้ในไฟล์ หรือ ฐานข้อมูลได้เช่นกัน โดยส่งข้อมูลดังกล่าวผ่านการเชื่อมต่อของ HTTP โดย method ที่ใช้แปลงข้อมูล JavaScript object ให้เป็น JSON นั้นคือ JSON.stringify(object) ที่จะทำการแปลงข้อมูล JavaScript object ให้อยู่ในรูปแบบของ JSON String ตัวอย่างเช่นส่วนของโค้ดด้านล่างที่กำหนด JavaScript object ที่มีข้อมูลประกอบไปด้วย ข้อความ, ตัวเลข, และ array ซึ่ง method JSON.stringify() จะทำการแปลงข้อมูลนั้นให้อยู่ในรูปแบบ JSON String

```
var accountObj = {
    name: "Baggins",
    number: 10645,
    members: ["Frodo, Bilbo"],
    location: "Shire"
};

var accountStr = JSON.stringify(accountObj);
console.log(accountStr);
```

รูปที่ 2.12 การแปลง JavaScript object เป็น JSON

ผลลัพธ์จากรูปที่ 2.12 คือ

```
{
    "name":"Baggins","number":10645,
    "members":["Frodo, Bilbo"],"location":"Shire"
}
```

รูปที่ 2.13 ผลลัพธ์ที่ได้จากการแปลง JavaScript object เป็น JSON

การเข้าถึง File System โดย Node.js

ในการทำงานร่วมกับ file ข้อมูลต่างๆนั้นถือเป็นสิ่งสำคัญของ Node.js โดยเฉพาะอย่างยิ่งถ้าต้องการจัดการกับไฟล์ที่ข้อมูลมีการเปลี่ยนแปลงอยู่ตลอดเวลา เช่นไฟล์ข้อมูลในเว็บแอปพลิเคชัน หรือเซอร์วิสต่างๆ Node.js ได้เตรียม module ที่ช่วยในการทำงานกับ file ไว้ให้ซึ่งมีชื่อ module ว่า “fs module” โดย module ดังกล่าวได้เตรียม API ที่ใช้ใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
149252

การเข้าถึงไฟล์ในรูปแบบต่างๆ เช่น open, read, write, หรือ modify เป็นต้น แต่ก่อนที่จะใช้งาน จำเป็นที่จะต้องเรียกใช้ module สำหรับเข้าถึงไฟล์ก่อน โดยสามารถเรียกใช้งาน module ได้จากคำสั่ง

```
var fs = require('fs');
```

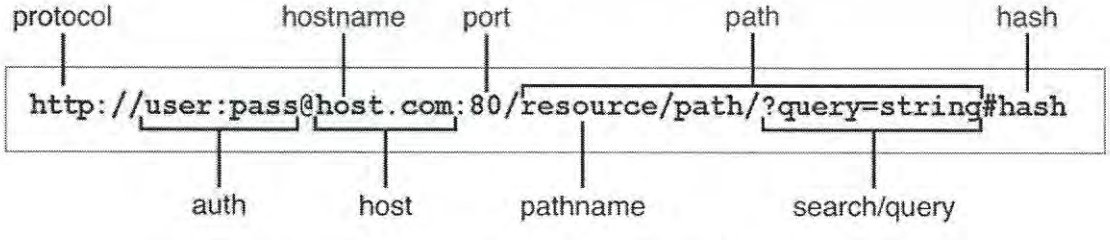
การสร้าง HTTP เซอร์วิสใน Node.js

Node.js นั้นยังมีความสามารถที่จะสร้าง HTTP, HTTPS เซิร์ฟเวอร์และเซอร์วิสต่างๆ ได้ อย่างรวดเร็ว โดย Node.js ได้เตรียม module สำหรับการทำงานดังกล่าวเอาไว้ นั่นคือ module ที่ชื่อว่า http และ https โดยทั้งสอง module นั้นได้เตรียมสิ่งที่จำเป็นสำหรับสร้างเว็บเซิร์ฟเวอร์และเว็บเซอร์วิสเอาไว้ช่วยอำนวยความสะดวกในการสร้างเว็บเซิร์ฟเวอร์ได้เป็นอย่างดี แต่อย่างไรก็ตาม http module นั้นถือว่ามีการทำงานที่ยังไม่มีประสิทธิภาพมากพอ ซึ่งสิ่งที่ http module ไม่สามารถทำได้นั้นก็คือ การควบคุม route, cookie, cache, และอีกหลายๆอย่าง แต่ก็ยังมีผู้พัฒนา module ที่สามารถนำมาใช้แทน http module ได้อย่างมีประสิทธิภาพมากขึ้น ไม่ว่าจะเป็น express หรือ loopback ซึ่งเป็น module ที่ทำให้การสร้างเว็บเซิร์ฟเวอร์นั้นง่ายขึ้น และยังมีประสิทธิภาพดีอีกด้วย

การประมวลผล URL

Uniform Resource Location (URL) ทำหน้าที่เป็นที่อยู่ ซึ่ง HTTP server ใช้ในการควบคุม request จาก client URL จะเตรียมข้อมูลที่จำเป็นสำหรับใช้ในการ request ไปยังเซิร์ฟเวอร์ที่ถูกต้องบนพอร์ตที่กำหนด และเข้าถึงข้อมูลที่เหมาะสม URL สามารถแบ่งย่อยออกเป็นหลายๆส่วนด้วยกัน ซึ่งแต่ละส่วนนั้นจะเตรียมข้อมูลสำหรับเว็บเซิร์ฟเวอร์ใช้ในการควบคุม HTTP request จากไคลเอนต์

รูปด้านล่างแสดงให้เห็นโครงสร้างพื้นฐานของ URL และส่วนประกอบของตัว URL แต่ละ HTTP request ไม่จำเป็นต้องมีส่วนประกอบครบทุกส่วน ตัวอย่างเช่น request ส่วนใหญ่ ไม่จำเป็นต้องมีส่วนของการยืนยันตัวตน และอีกหลายๆ request นั้นไม่จำเป็นที่จะต้องมีส่วนของการค้นหาข้อมูล



รูปที่ 2.14 ส่วนประกอบของ Uniform Resource Location (URL) [4]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

HTTP request จากไคลเอนต์นั้นประกอบไปด้วย URL ที่มีข้อมูลดังรูปด้านบน เพื่อที่จะสามารถใช้ข้อมูลเหล่านั้นได้อย่างมีประสิทธิภาพ Node.js ได้เตรียม URL module ซึ่งเตรียมฟังก์ชันต่างๆในการแปลง URL ให้เป็น URL object

ตารางที่ 2.1 รายละเอียดส่วนประกอบของ Uniform Resource Location (URL)

ส่วนประกอบ	**คำอธิบาย**
protocol	เป็น protocol ที่ใช้สำหรับการ request
host	ส่วนที่บอกว่า URL นั้นเป็น URL จากที่ใด
auth	ส่วนของการยืนยันข้อมูลของ host นั้นๆ
hostname	ชื่อของ host
port	หมายเลข port ของ host
path	คือส่วนที่บอกที่อยู่ของ URL ที่ทำการเก็บข้อมูลต่างๆ
pathname	ชื่อของ path
search	เป็นข้อมูลที่ใช้ในการกำหนดการค้นหา โดยจะนำหน้าด้วยเครื่องหมายคำถาม (?) เสมอ
query	เปรียบเหมือน parameter ที่ใช้ทำการ search
hash	เป็นส่วนของ URL ที่ทำหน้าที่เชื่อมโยงข้อมูลในหน้าเดียวกันโดยมีเครื่องหมาย # นำหน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5 Strongloop

Strongloop [5] เป็นแพลตฟอร์มที่รวบรวม library และ framework ต่างๆที่พัฒนาขึ้นมาจาก Node.js โดยแพลตฟอร์มจะเตรียมสภาพแวดล้อมที่ทำให้สามารถสร้าง API ไว้สำหรับใช้งานในแอปพลิเคชัน โดย framework หลักที่ใช้ในการสร้าง API นั้นก็คือ LoopBack framework

2.5.1 LoopBack framework

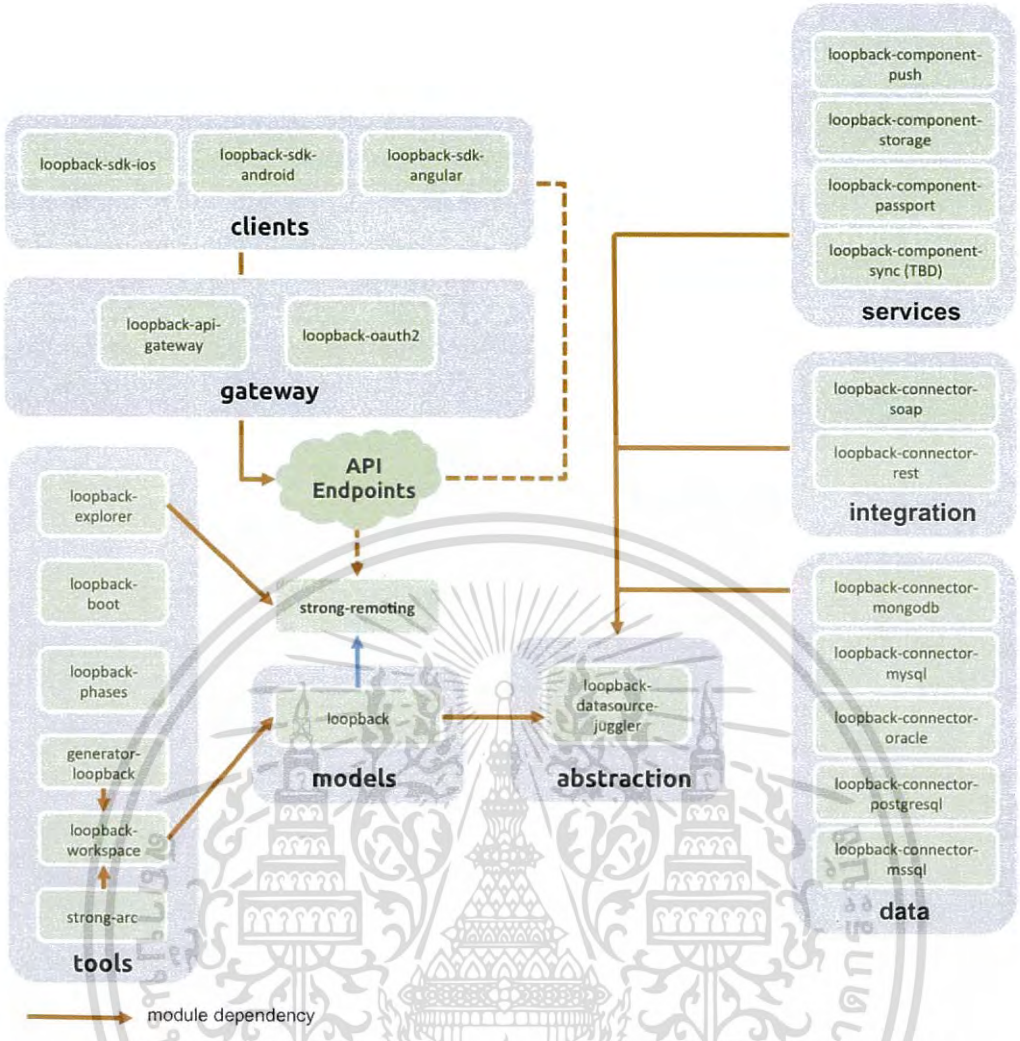
LoopBack framework เป็น open-source framework ของ Node.js ที่ประกอบไปด้วย module ต่างๆ ของ Node.js ซึ่งทำให้สามารถสร้างแอปพลิเคชันได้อย่างรวดเร็ว โดยความสามรถของ LoopBack framework มีดังนี้

- สร้าง REST API โดยไม่จำเป็นต้องเขียนโค้ด หรือ เขียนเพียงเล็กน้อยเท่านั้น
- สามารถเข้าถึงแหล่งข้อมูลได้หลายประเภท ไม่ว่าจะเป็น relational database, MongoDB, API แบบ SOAP และ REST เป็นต้น
- สามารถกำหนดรูปแบบของความสัมพันธ์ และ การควบคุมการใช้งาน สำหรับ API ต่างๆที่มีความซับซ้อน
- เตรียม เซอร์วิสต่างๆสำหรับ ใช้งานในโมบายแอปพลิเคชัน อาทิ geolocation service, file service, push service เป็นต้น
- สามารถสร้างแอปพลิเคชันได้โดย loopback เตรียม SDK สำหรับ โมบายแอปพลิเคชัน ได้แก่ Android กับ iOS และ JavaScript SDK สำหรับ เว็บแอปพลิเคชัน
- แอปพลิเคชันสามารถทำงานได้ทั้งแอปพลิเคชันเซิร์ฟเวอร์และบนคลาวด์

แอปพลิเคชันที่สร้างด้วย LoopBack จะดำเนินการต่างๆกับแหล่งข้อมูลต่างๆ (Data Sources) ผ่าน API ของ Model ที่สร้างโดย LoopBack โดยสามารถใช้งาน API ได้โดย LoopBack จะเตรียม Node.js API ไว้ให้หรือใช้งานผ่าน API ของ เซอร์วิสแบบ REST อีกทั้งยังเตรียม Native Client API สำหรับใช้งานบน iOS, Android และ JavaScript ไว้ให้อีกด้วย ซึ่งการใช้ API นั้น ทำให้แอปพลิเคชันสามารถเรียกดูข้อมูลจากฐานข้อมูล, จัดเก็บข้อมูล, อัพโหลดไฟล์, ส่งอีเมล, สร้าง push notification, ลงทะเบียนผู้ใช้งาน, และเรียกใช้ความสามารถต่างๆที่แหล่งข้อมูลหรือบริการนั้นๆเตรียมไว้ให้ได้

ผู้ใช้งานสามารถเรียกใช้ API ต่างๆที่สร้างโดย LoopBack ได้ทันทีโดยใช้งาน Strongloop Remoting ที่เชื่อมต่อเข้ากับ transport layer ที่ทำให้สามารถใช้งาน API ได้ผ่านทาง RESTful เว็บเซอร์วิส, WebSockets, และการรับส่งข้อมูลแบบอื่นๆได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.15 module ที่สำคัญของ LoopBack ในการสร้างแอปพลิเคชัน [5]

ตารางที่ 2.2 Module ของ LoopBack framework

Module	ประเภท	คำอธิบาย	การใช้งาน
LoopBack	Model	Model และ API Server	ใช้กำหนดโครงสร้างของ model และเปิด API สำหรับเรียกใช้งานโดยไม่สนใจข้อมูลที่จัดเก็บ
loopback-datasource-juggler	Abstraction	Model แบบ Abstraction ที่ทำการเก็บข้อมูล	เป็นตัวที่ทำให้ Model ที่สร้างมีความสามารถ CRUD โดยขึ้นอยู่กับแหล่งข้อมูลนั้นๆ
loopback-boot	Initialization	เริ่มต้นการทำงานแอปพลิเคชัน	กำหนดค่าเริ่มต้นการทำงาน และสั่งเริ่มการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.2(ต่อ) Module ของ LoopBack framework

Module	ประเภท	คำอธิบาย	การใช้งาน
loopback-phase	Sequencing	ตัวกลางของการดำเนินการ	กำหนดตัวกลาง (Middleware) ที่ใช้การดำเนินการตามขั้นตอนของ แอปพลิเคชัน
loopback-connector-mongodb loopback-connector-mysql loopback-connector-postgresql loopback-connector-msssql loopback-connector-oracle	Data	ตัวเชื่อมต่อเข้ากับแหล่งข้อมูล แบบ RDBMS และ NoSQL	ใช้เพื่อเป็นตัวเชื่อมต่อแต่ละ model เข้ากับแหล่งข้อมูลประเภท RDBMS และ NoSQL
loopback-connector-rest loopback-connector-soap	Integration	ตัวเชื่อมต่อเข้ากับเซอร์วิสที่มีอยู่	ใช้เพื่อเป็นตัวเชื่อมต่อเข้ากับเซอร์วิสที่มีอยู่แล้วผ่านทาง API ของเซอร์วิสนั้นๆ
generator-loopback strong-arc	Tooling	เครื่องมือที่ใช้สร้างแอปพลิเคชัน	ใช้เพื่อเป็นเครื่องมือในการโดยประกอบไปด้วย Yeoman generator สำหรับ CLI และ Strongloop Arc ในรูปแบบ GUI
loopback-component-push loopback-component-storage loopback-component-passport loopback-component-sync	Services	เซอร์วิสต่างๆที่เตรียมไว้ให้ใช้งาน	ใช้ตามจุดประสงค์ของแต่ละแอปพลิเคชัน โดย loopback ได้เตรียมเซอร์วิสไว้ให้สำหรับใช้งานในรูปแบบของ component

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.2(ต่อ) Module ของ LoopBack framework

Module	ประเภท	คำอธิบาย	การใช้งาน
strong-gateway loopback-component-oauth2	Gateway	ทางเข้า-ออก ของ API	ใช้เพื่อความปลอดภัยของการใช้งาน API
loopback-sdk-ios loopback-sdk-android loopback-sdk-angular	Clients	ชุด SDK ของไคลเอนต์ต่างๆ	เพื่อนำมาพัฒนา native client ของแอปพลิเคชัน ตามแพลตฟอร์มต่างๆ

2.5.2 แนวคิดของ LoopBack

LooBack framework ประกอบไปด้วยองค์ประกอบต่างๆ ดังต่อไปนี้

Model

Model คือหัวใจสำคัญของ LoopBack framework ซึ่งแสดงให้เห็นถึงแหล่งข้อมูล เช่นฐานข้อมูลหรือเซอร์วิสต่างๆ (เช่น SOAP, REST) โดย LoopBack Model นั้นคือ Object ของ JavaScript ที่มี API แบบ Node.js และ API แบบ REST ให้ใช้งาน

ประเภทของ Model

คุณสมบัติสำคัญที่ทำให้ LoopBack มีประสิทธิภาพในการทำงานนั่นคือ เมื่อเราทำการกำหนด Model แล้ว LoopBack จะทำการกำหนด REST API มาให้ พร้อมกับกลุ่มคำสั่ง CREATE, READ, UPDATE, และ DELETE ในการดำเนินการกับข้อมูล

Basic Model Object

Basic Model Object คือ ต้นแบบของทุก Model โดยมี คลาส สำหรับเรียกใช้ฟังก์ชันเสริมในการดำเนินการกับข้อมูล เช่น Hooks Operation ซึ่งเป็นฟังก์ชันเสริมของคลาสใน Basic Model Object ที่ใช้ในการดำเนินการกับข้อมูล และ function สำหรับการตรวจสอบความถูกต้องของข้อมูล (Validating model data) ตัว Basic Model object นี้ จะเป็นพื้นฐานในการสืบทอดคุณสมบัติของ Model ต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Connected Model

Connected Model คือ model ที่มีการสืบทอดมาจาก Basic Model Object แล้วทำการเชื่อมต่อ model ที่สืบทอดมาเข้ากับแหล่งข้อมูลซึ่งจะทำให้ model นั้นสามารถดำเนินการตาม CRUD โดย model ที่มีรูปแบบ Connected Model จะสืบทอดคลาสที่ชื่อว่า Relation ที่ใช้ในการกำหนดความสัมพันธ์ของ แต่ละ model และคลาส PersistedModel ที่ใช้ในการดำเนินการกับข้อมูล

Built-in Model

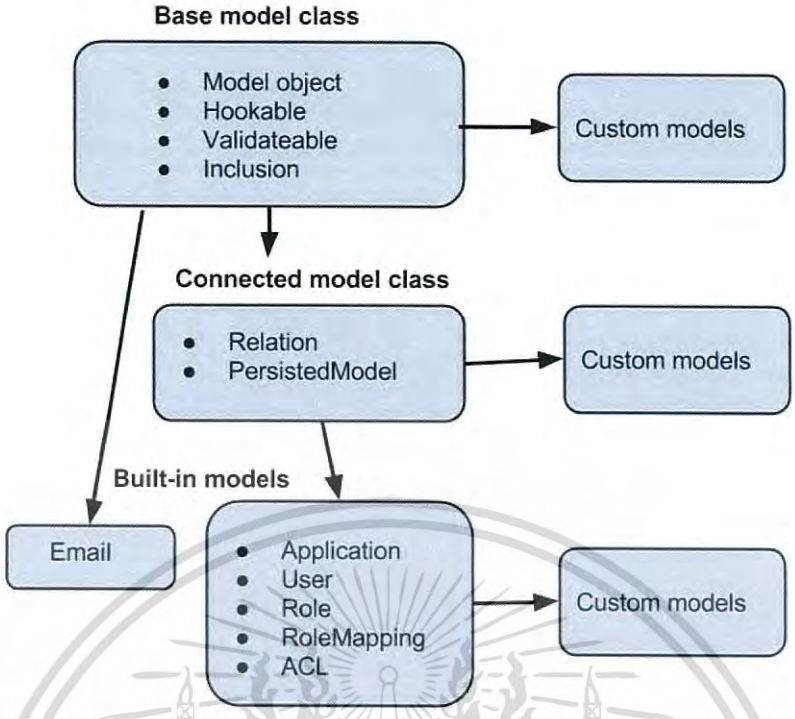
ทุกแอปพลิเคชันที่สร้างด้วย LoopBack framework จะมีการกำหนด built-in Model มาด้วยทุกครั้ง ตัวอย่างของ Built-in Model เช่น User Model ที่ใช้ในการลงทะเบียน-เข้าสู่ระบบ,บทบาทModel ที่ใช้ในการกำหนดบทบาทหน้าที่ของผู้ใช้, Access Control Model ที่ใช้ในการกำหนดสิทธิ์ในการเข้าถึงข้อมูลของผู้ใช้งาน, E-mail Model ที่ใช้กำหนดการตั้งค่าสำหรับการส่ง E-mail ไปยังผู้ใช้งานแอปพลิเคชัน และ Application Model ใช้ในการกำหนด metadata ของแอปพลิเคชัน

Built-in Model ยกเว้น E-mail Model จะทำการสืบทอดคุณสมบัติมาจาก PersistedModel ดังนั้น Built-in Model จึงมีคุณสมบัติ CRUD เช่นเดียวกับ PersistedModel ด้วย

Custom Model

Custom Model คือ Model ที่นักพัฒนาสามารถกำหนดรายละเอียดของตัว Model เองได้ตามความต้องการ โดย Model ที่กำหนดขึ้นมานั้นจะสามารถเลือกได้ว่าจะสืบทอดคุณสมบัติจาก Model แบบใด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.16 ลำดับการสืบทอดของ Model [5]

ความสัมพันธ์ของ Model

แต่ละ Model สามารถกำหนดความสัมพันธ์ระหว่างกันได้ ซึ่งความสัมพันธ์จะมีลักษณะคล้ายกับฐานข้อมูลแบบมีความสัมพันธ์ (Relational Database) ความสัมพันธ์ของ model ประกอบไปด้วย BelongsTo, HasMany, และ HasAndBelongsToMany

การดำเนินการ CRUD กับ Model

เมื่อเชื่อมต่อ Model เข้ากับ แหล่งข้อมูลที่ใช้สำหรับดำเนินการ เช่น ฐานข้อมูล แล้วนั้น Model ดังกล่าวจะถูกเรียกว่าเป็น Connected Model ซึ่งจะมีฟังก์ชันในการดำเนินการกับข้อมูลคือ CEATE, READ, UPDATE, DELETE (CRUD) เพิ่มเข้ามา ซึ่งฟังก์ชันการดำเนินการเหล่านี้ มาจากคลาสที่ชื่อว่า PersistedModel

ตารางที่ 2.3 ฟังก์ชันการทำงานของ คลาส PersistedModel

การดำเนินการ	REST	Node.js API	ความสอดคล้องกับการดำเนินการของ SQL
Update (แก้ไข)	POST /modelName PUT /modelName	updateAll()*	UPDATE
Read (เรียกดูข้อมูล)	GET /modelName?filter=...	find()*	SELECT

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.3(ต่อ) ฟังก์ชันการทำงานของ คลาส PersistedModel

การดำเนินการ	REST	Node.js API	ความสอดคล้องกับการดำเนินการของ SQL
Delete (ลบข้อมูล)	DELETE /modelName/modelID	destroyById()*	DELETE
Create (สร้างข้อมูล)	PUT /modelName POST /modelName	create()*	INSERT

*หมายเหตุ: API ที่แสดงในตารางเป็นเพียงตัวอย่างของ API เท่านั้น

Application Logic

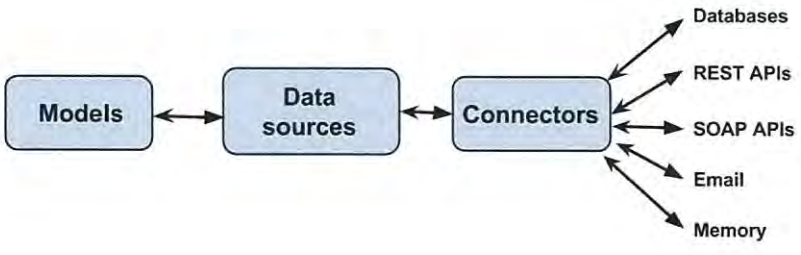
นักพัฒนาสามารถกำหนด Application logic เองได้หลายวิธี เช่น

- การเพิ่ม Application logic ไปยัง model ผ่าน remote method ต่างๆ (กำหนดปลายทางของ REST เอง), remote hooks ที่จะถูกเรียกให้ทำงาน โดย remote method ต่างๆ, และ Operation hooks ที่จะถูกเรียกให้ทำงาน เมื่อมีการดำเนินการ CRUD กับ model
- เพิ่ม boot script ซึ่งเป็น script ที่จะทำงานตั้งแต่เริ่มการทำงานของแอปพลิเคชัน
- สร้าง middleware มาใช้งานเอง ซึ่งการสร้างจะคล้ายกับ Express framework และยังสามารถเพิ่มโค้ดสำหรับการตรวจสอบข้อมูล ก่อนที่จะทำการบันทึกไปยังแหล่งข้อมูลได้อีกด้วย

Data source และ connector

LoopBack สามารถสื่อสารกับ Back-end เซอร์วิสต่างๆ ได้ ไม่ว่าจะเป็น ฐานข้อมูล , REST API, เว็บเซอร์วิส แบบ SOAP, และ เซอร์วิสอื่นๆ ที่ให้บริการข้อมูล

แหล่งข้อมูลต่างๆนั้นจะถูกเชื่อมต่อโดยตัวเชื่อมต่อ (Connector) ที่จะสื่อสารไปยังฐานข้อมูลหรือเซอร์วิสต่างๆ ซึ่งแอปพลิเคชันจะไม่สามารถใช้งานตัวเชื่อมต่อนั้นได้โดยตรง แต่จะสามารถเข้าถึงแหล่งข้อมูลได้โดยใช้ DataSource API และ PersistedModel API



รูปที่ 2.17 การเชื่อมต่อระหว่าง model, datasources, และconnectors [5]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5.3 การเชื่อมต่อ Model เข้ากับ Data Source

LoopBack model สามารถเชื่อมต่อเข้ากับ backend system ต่างๆเช่น ฐานข้อมูล ผ่าน data source ที่จะทำการเตรียมฟังก์ชัน Create, Retrieve, Update, Delete เอาไว้ และยังสามารถที่จะเชื่อมต่อเข้ากับเซอร์วิสอื่นๆได้อีกเช่น REST API, SOAP web service, Storage service และอื่นๆอีกมากมาย

Data source ต่างๆ จะทำการเชื่อมต่อกับ connector ที่ทำการกำหนด คำสั่งต่างๆที่ดำเนินการกับข้อมูลโดยใช้ database driver หรือ API ของเซอร์วิสนั้นๆ โดยทั่วไปแล้ว แอปพลิเคชันจะไม่ทำการใช้ connector โดยตรง แต่จะเรียกใช้ connector ผ่าน data source โดยใช้ DataSource API และ PersistedModel API แทน

รายละเอียดของ Data source

รายละเอียดของ Data source จะถูกกำหนดเอาไว้ในไฟล์ที่ชื่อว่า datasource.json ซึ่งมีรายละเอียดดังนี้

```
{
  "mysql-corp": {
    "connector": "mysql",
    "host": "your-mysql-server.foo.com",
    "user": "db-username",
    "password": "db-password",
    "database": "your-db-name"
  }
}
```

รูปที่ 2.18 รายละเอียดของไฟล์ datasources.json

ตารางที่ 2.4 Data source ต่างๆ ที่สามารถเชื่อมต่อเข้ากับ API ได้

Property	Type	Description
connector	String	กำหนด connector ที่ใช้งานในการเชื่อมต่อ datasource
database	String	ชื่อของฐานข้อมูลที่ทำการเชื่อมต่อ
debug	Boolean	ถ้ามีค่าเป็น true จะทำการเข้าสู่โหมด debug
host	String	ชื่อ host ของฐานข้อมูล
password	String	Password สำหรับเชื่อมต่อเข้าฐานข้อมูลนั้นๆ
port	Number	หมายเลข TCP port ของฐานข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.4(ต่อ) Data source ต่างๆ ที่สามารถเชื่อมต่อเข้ากับ API ได้

Property	Type	Description
url	String	ฐานข้อมูลในรูปแบบของ url
username	String	ชื่อผู้ใช้งานในการเชื่อต่อเข้าสู่ฐานข้อมูล

ตารางที่ 2.5 Connector ต่างๆ ที่ใช้เชื่อมต่อ Data Source เข้ากับ Model

Database Connector		
Connector	**Module**	**การติดตั้งผ่าน CLI**
Memory Connector	Built-in module	-
MongoDB	loopback-connector-mongodb	npm install –save loopback-connector-mongodb
MySQL	loopback-connector-mysql	npm install –save loopback-connector-mysql
Oracle	loopback-connector-oracle	npm install –save loopback-connector-oracle
PostgreSQL	loopback-connector-postgresql	npm install –save loopback-connector-postgresql
SQL Server	loopback-connector-mssql	npm install –save loopback-connector-mssql

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.5(ต่อ) Connector ต่างๆ ที่ใช้เชื่อมต่อ Data Source เข้ากับ Model

Connector อื่นๆ		
Connector	**Module**	**การติดตั้งผ่าน CLI**
Email connector	Built-in module	-
Push connector	loopback-component-push	npm install –save loopback-component-push
Remote connector	loopback-connector-remote	npm install –save loopback-connector-remote
REST	loopback-connector-rest	npm install –save loopback-connector-rest
SOAP	loopback-connector-soap	npm install –save loopback-connector-soap
Storage connector	loopback-component-storage	npm install –save loopback-component-storage

2.5.4 การดำเนินการกับข้อมูล

เมื่อทำกำหนดรายละเอียดของ model แล้วจะสามารถดำเนินการ create, read, update, และ delete (CRUD) เพื่อทำการเพิ่มข้อมูลไปยัง model, จัดการกับ model, และทำการเรียกดูข้อมูลจาก model นั้นได้ ทุกๆ Loopback model ที่เชื่อมต่อกับ ที่เก็บข้อมูลต่างๆ (เช่นฐานข้อมูล) จะมีฟังก์ชันที่ใช้ในการดำเนินการ CRUD จาก PersistedModel class ทันที

PersistedModel

PersistedModel คือ class ที่ประกอบไปด้วย method ที่ใช้สำหรับ creating, updating, และ deleting ข้อมูล โดยข้อมูลของ model จะถูกเรียกว่า model instance เมื่อทำการเปรียบเทียบ model กับ data base นั้น model จะเปรียบเสมือนกับ table และ model instance จะเปรียบเสมือน row หรือ record ของ table นั้นๆ

Creating data (model instance)

การสร้าง model instance จะใช้ method ของ PersistedModel ในการเพิ่มข้อมูล ซึ่งจะหมายถึงการ insert หรือ create model instance ซึ่งจะประกอบไปด้วย

- **Create** สร้าง model instance (record)
- **upsert** ตรวจสอบว่า ถ้า model instance (record) มีอยู่แล้ว ข้อมูลที่ทำการเพิ่มเข้ามาจะไป update instance ที่มีอยู่แล้วโดยจะ update ตาม ID ของ instance นั้นๆ ถ้ายังไม่มี instance จะทำการสร้างขึ้นมาใหม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **findOrCreate** จะทำการค้นหา instance ที่ตรงกับเงื่อนไขที่ระบุไว้ใน filter ที่ชื่อว่า where ถ้าพบข้อมูลที่ตรงกัน จะคืนค่าของ object ที่มีข้อมูลของ instance นั้นๆมาให้ ถ้าไม่พบ จะทำการสร้าง instance (record) ขึ้นมาใหม่
- **save** ทำการบันทึก model instance ถ้า instance ไม่มี ID จะทำการเรียก method create ให้ทำงานแทน

Update data (model instance)

การ update model instance นั้น จะทำการเรียก method ผ่าน object ของ model ได้เลย เนื่องจาก update เป็น method แบบ static โดย update method ประกอบไปด้วย

- **updateAll** ทำการ update instance (record) หลายจำนวนที่ตรงกับเงื่อนไข where พร้อมกันในการเรียกใช้ method ครั้งเดียว
- **updateAttribute** update ข้อมูลของ attribute (property) เพียงหนึ่งตัว
- **updateAttributes** update ข้อมูลของ attribute (property) จำนวนหลายตัว

Delete data (model instance)

การ delete model instance นั้น จะทำการเรียก method ผ่าน object ของ model ได้เลย เนื่องจาก delete update เป็น method แบบ static โดย delete method ประกอบไปด้วย

- **destroyAll** จะทำการลบทุก model instance ที่ตรงกับเงื่อนไข where
- **destroyById** จะทำการลบ model instance ที่ตรงตาม ID ที่ระบุเอาไว้

Querying data (model instance)

Query คือการดำเนินการแบบ read ของ model ที่จะทำการคืนค่ากลุ่มของข้อมูล หรือ ผลลัพธ์ จากการเรียกดูของผู้ใช้ โดยสามารถเรียกใช้งาน query ผ่าน Node API และ REST API และใช้ filter ซึ่งจะทำหน้าที่เป็นตัวกรองข้อมูลให้ตรงตามที่ผู้ใช้งานต้องการ โดยข้อมูลที่ return กลับคืนมานั้นจะอยู่ในรูปแบบของ JSON

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.6 คำสั่ง query ข้อมูลผ่านทาง REST API

Query	Model API (Node)	REST API
ค้นหาทุก model instance ตามรายละเอียดที่กำหนดไว้ใน filter	Find(filter, callback)	GET /modelName?filter...
ค้นหา model instance ตัวแรกที่ตรงตามรายละเอียดที่กำหนดไว้ใน filter	findOne(filter, callback)	GET /modelName/findOne?filter...
ค้นหา model instance ตาม ID	findById(id, [filter], callback) (filter จะมีหรือไม่มีก็ได้)	GET /modelName/modelID

Filters

Filter คือ object ที่จะทำการกรองข้อมูลของ query ตามที่เรากำหนด โดยทั้ง REST API และ Node API จะสามารถกำหนด filter ได้ทั้งคู่ ซึ่ง loopback ยังสนับสนุน filter ที่มีรายละเอียดเฉพาะ ที่มีลักษณะคล้ายกับ condition ของภาษา SQL แต่ filter จะถูกออกแบบมาเพื่อป้องกันการ injection โดยใช้ภาษา JavaScript ในการออกแบบ โดย filter ที่มีลักษณะเฉพาะสามารถใช้ได้กับ method PersistedModel.find() (และ model ที่เกี่ยวข้อง)

ตารางที่ 2.7 รายการ filter สำหรับกรองข้อมูลที่ทำการเรียกดู

Filter type	Type	Description
Fields	Object, Array, หรือ String	กำหนดรายละเอียดของ field ที่ต้องการเมื่อมี response ส่งกลับมา
Include	String, Object, หรือ Array	ทำการเพิ่มผลลัพธ์จาก model ที่เกี่ยวข้องเนื่องจากมีความสัมพันธ์กันระหว่าง model
limit	Number	จำกัดจำนวนของ model instance ที่คืนค่ากลับมา
order	String	กำหนดรายละเอียดของการเรียงลำดับข้อมูลว่าเรียงจากมากไปน้อย หรือ น้อยไปมาก
skip (offset)	Number	ข้าม model instance ตามรายละเอียดที่ระบุไว้
Where	Object	กำหนดรายละเอียดของการค้นหาข้อมูล มีลักษณะคล้ายกับประโยค WHERE ในภาษา SQL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 MongoDB

MongoDB [6] คือฐานข้อมูลที่มีการจัดเก็บข้อมูลในรูปแบบ NoSQL โดยมีพื้นฐานของข้อมูลที่จัดเก็บเป็น document model โดยที่ object ของ data นั้นจะถูกจัดเก็บเป็น document แต่ละ document จะถูกเก็บรวมกัน รวมเป็น 1 collection และแต่ละ collection รวมกันได้เป็น 1 ฐานข้อมูลโดย MongoDB จะทำการสร้างที่เก็บข้อมูลที่มีประสิทธิภาพ และสามารถขยายพื้นที่จัดเก็บข้อมูลได้อัตโนมัติ ขึ้นอยู่กับข้อมูลที่จัดเก็บว่ามีจำนวนเท่าใด และ MongoDB ยังง่ายต่อการติดตั้งอีกด้วย

รูปที่ 2.19 สัญลักษณ์ MongoDB [6]

2.6.1 กลุ่มของเอกสาร (Collection)

กลุ่มของเอกสาร (Collection) คือการรวมกลุ่มของ document ที่มีจุดประสงค์เหมือนกัน หรือคล้ายคลึงกัน collection จะทำหน้าที่คล้ายกับ table ในฐานข้อมูล SQL แต่อย่างไรก็ตาม ก็มีสิ่งที่แตกต่างกันที่สำคัญด้วย ใน MongoDB collection จะไม่บังคับในการกำหนด schema สำหรับเก็บข้อมูลซึ่งจะทำให้ข้อมูลใน collection นั้น มีการจัดเก็บข้อมูลโดยมีโครงสร้างที่แตกต่างกันออกไป ขึ้นอยู่กับว่าต้องการจะจัดเก็บข้อมูลที่มีโครงสร้างแบบไหน ถือว่าเป็นข้อแตกต่างที่สำคัญในการลดปริมาณของจำนวนตารางในฐานข้อมูล SQL

2.6.2 เอกสาร (Document)

Document หมายถึงข้อมูลจำนวนหนึ่ง entity ใน MongoDB collection จะถูกสร้างขึ้นจาก document ตั้งแต่ 1 หรือมากกว่าขึ้นไป เปรียบเทียบกับ SQL database นั้น document จะเหมือนกับ row หรือ record ใน table หนึ่ง แต่สิ่งที่แตกต่างกันระหว่าง MongoDB และฐานข้อมูล SQLนั้นคือ row ในฐานข้อมูล SQLนั้นจะทำการเก็บข้อมูลใดๆ ได้เพียงแค่ใน column นั้นสำหรับข้อมูลใน 1 row แต่ MongoDB ตัวของ document จะสามารถเก็บ document ซ้อนกันอยู่ได้

2.6.3 ประเภทของข้อมูลที่จัดเก็บใน MongoDB

ข้อมูลที่จัดเก็บใน MongoDB จะเป็นข้อมูลประเภท BSON ซึ่งเป็นข้อมูล JSON ในรูปแบบ binary ทำให้ขนาดของข้อมูลมีลักษณะเบา ยิ่งไปกว่านั้นรูปแบบของข้อมูลที่เป็นแบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

field/value มีความคล้ายคลึงกับ object ในภาษา JavaScript ซึ่งหมายความว่าการแปลงข้อมูลที่ได้รับมาจาก MongoDB ไปเป็น JavaScript object นั้นสามารถทำได้ง่ายขึ้นอีกด้วย

```
{
    "name": "New Project",
    "version": 1,
    "languages": [
        "JavaScript",
        "HTML",
        "CSS"
    ],
    "admin": {
        "name": "Brad",
        "password": "****"
    },
    "paths": {
        "temp":"/tmp",
        "project": "/opt/project",
        "html": "/opt/project/html"
    }
}
```

รูปที่ 2.20 ตัวอย่างของข้อมูลใน Document

document ดังกล่าวประกอบไปด้วย field name, version, languages, admin, และ path โดยชื่อของ field ไม่สามารถเป็นตัวอักษรคำว่า null, เครื่องหมายจุด (.), หรือตัวอักษร dollar signs ($) รวมถึงการตั้งชื่อ field ว่า _id ที่ถูกจองไว้สำหรับ object ID ค่าของ _id จะเป็นค่าที่ไม่ซ้ำกันในระบบ

ขนาดของ document ใน MongoDB จะมีค่ามากที่สุดคือ 16MB เพื่อเป็นการป้องกันการ query ที่เกินขนาดของ document ที่อาจจะส่งผลต่อหน่วยความจำ RAM ที่ใช้งานไม่เพียงพอ หรือ อาจจะส่งผลกับ file system ทำงานผิดพลาดได้

2.7.6 การกำหนดรูปแบบของข้อมูล

ก่อนที่จะทำการสร้างฐานข้อมูลด้วย MongoDB นั้น นักพัฒนาจำเป็นที่จะต้องทำความเข้าใจข้อมูลที่จะถูกทำการจัดเก็บเสียก่อนว่าข้อมูลเหล่านั้นจัดเก็บไปเพื่ออะไร และจะทำการเข้าถึงข้อมูลที่ทำการจัดเก็บด้วยวิธีไหน การทำความเข้าใจแนวคิดของข้อมูลจะช่วยลดเวลาในการกำหนดโครงสร้างข้อมูลของแอปพลิเคชันไปได้อย่างมาก และทำให้แอปพลิเคชันทำงานได้อย่างมีประสิทธิภาพมากขึ้นอีกด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การจัดการข้อมูลให้อยู่ในรูปอย่างง่ายด้วยการอ้างอิง document

การทำข้อมูลให้อยู่ในรูปอย่างง่าย (Data normalization) คือกระบวนการที่จะทำการลดความซ้ำซ้อน (redundancy) และการขึ้นต่อกัน (dependency) ของ document และ collection การทำข้อมูลให้อยู่ในรูปอย่างง่าย สามารถทำได้โดยการระบุ property ขอบ object ที่เป็น subobject และทำการจัดเก็บ value ของ property นั้นแยกกันในอีก collection ซึ่งจะทำให้มี property ที่มี value เหมือนกันในทั้งสอง collection โดยทั่วไป การจัดการข้อมูลในลักษณะนี้จะใช้สำหรับ object ที่มีความสัมพันธ์แบบ one-to-many หรือ many-to-many

ข้อดี

ข้อดีของการทำข้อมูลให้อยู่ในรูปอย่างง่ายนั้นคือ ฐานข้อมูลจะมีขนาดที่เล็กลง เนื่องจากการสร้าง collection ขึ้นมาเก็บข้อมูลที่ใช้อ้างอิงแยกไว้เป็น 1 collection แทนที่การเก็บข้อมูลซ้ำซ้อนภายใน collection เดียวกัน ข้อดีอีกอย่างหนึ่งคือช่วยในการแก้ไขข้อมูล ในกรณีที่ subobject มีการแก้ไขข้อมูลบ่อยครั้ง ก็จะทำการแก้ไขเพียงครั้งเดียวแทนที่การแก้ไขทุก object ที่มี subobject

ข้อเสีย

ข้อเสียที่เห็นได้ชัดเจนของการทำข้อมูลให้อยู่ในรูปอย่างง่ายนั้นคือ เมื่อทำการเรียกดูข้อมูลจาก object ใดๆ ที่ทำการ ทำข้อมูลให้อยู่ในรูปอย่างง่ายกับ subobject หนึ่ง จะต้องทำการเรียกดูข้อมูลของ subobject เพื่อที่จะได้ข้อมูลที่ object ใช้อ้างอิง เพื่อทำการเรียกดูข้อมูลใน object นั้นได้ตรงตามที่ต้องการ ซึ่งจะทำให้ประสิทธิภาพลดลง หากมีการเข้าถึงข้อมูลจำนวนหลายครั้ง

ตัวอย่าง

ระบบหนึ่ง ประกอบไปด้วย “ผู้ใช้งาน” (user) ที่มี “ร้านขายสินค้าที่ชื่นชอบ” (favoriteStore) โดยที่แต่ละ user นั้นมี object ที่ประกอบไปด้วย name, phone, และ favoriteStore เป็น property ซึ่ง favoriteStore เป็น subobject ที่ประกอบไปด้วย name, street, city, และ zip เป็น property

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อย่างไรก็ตาม ในกรณีที่มี User จำนวน 1,000 คนที่มี Favorite store เหมือนกัน จะทำให้เกิดความสัมพันธ์แบบ one-to-many เป็นจำนวนมาก ดังนั้น การจัดเก็บ Favorite store ซึ่งเป็น object ไว้ในแต่ละ user นั้นจะทำให้เกิดความซ้ำซ้อนของข้อมูลขึ้น ดังนั้น จะต้องจัดการทำข้อมูลให้ อยู่ในรูปอย่างง่าย คือ ทำการกำหนด _id ให้แต่ละ document ที่สามารถใช้ในการอ้างอิงถึง favoriteStore นั้นๆได้ว่าเป็นของ document ใดใน favoriteStore collection ดังนั้น แอปพลิเคชันจะใช้ ID ของ favoriteStore อ้างอิงข้อมูล จาก user collection ไปยัง favoriteStore collection แทนที่การจัดเก็บทั้ง object ของ favoriteStore แทน

รูปที่ 2.21 การอ้างอิง favouriteStoreId ใน User Collection [6]

จากรูปที่ 2.21 แสดงให้เห็นโครงสร้าง collection ของ User และ FavoriteStore ที่ทำการเก็บข้อมูลแยก collection กัน และใช้ property ที่ชื่อว่า favoriteStore เป็นตัวอ้างอิงถึง collection นั้น

การทำข้อมูลให้อยู่ในรูปแบบมาตรฐานด้วย Embedded Documents

การทำข้อมูลให้อยู่ในรูปแบบมาตรฐานคือกระบวนการระบุว่า subobject เป็นของ object ใด และจะทำการนำข้อมูลของ subobject ไป ใส่ไว้ใน object นั้นๆ โดยปกติการทำข้อมูลให้อยู่ในรูปอย่างง่ายนั้นมักจะใช้กับ object ที่มีความสัมพันธ์แบบ one-to-one, -การขึ้นต่อกันของ object ไม่มาก และมีการแก้ไขข้อมูลไม่เยอะมาก

ข้อดี

ข้อดีของการทำข้อมูลให้อยู่ในรูปแบบมาตรฐานคือ ในการเรียกดูข้อมูลเพียงครั้งเดียว ก็สามารถได้รับข้อมูลทั้งหมดที่เกี่ยวข้องกันมาด้วย โดยไม่จำเป็นที่จะต้องเรียกดูข้อมูลที่ใช้อ้างอิงจาก collection ของ subobject มาก่อน ซึ่งจะช่วยเพิ่มประสิทธิภาพในการทำงานมากขึ้น

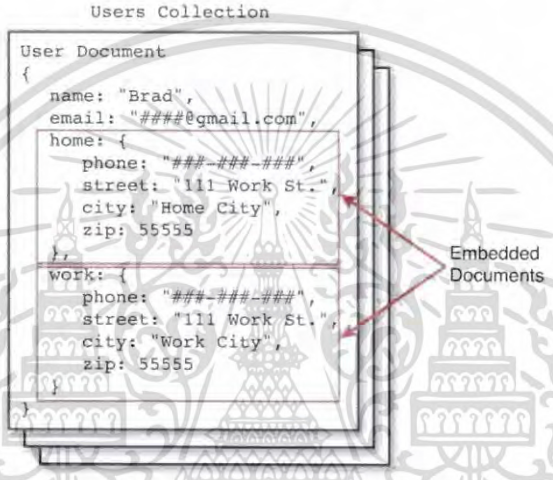
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อเสีย

ในกรณีที่ข้อมูลมีความสัมพันธ์แบบ one-to-many จะต้องทำการเก็บข้อมูลของ subobject ทุกตัวที่มีความสัมพันธ์กัน ซึ่งจะเป็นการเพิ่มความซ้ำซ้อนของข้อมูลขนาดใหญ่ และทำให้เปลืองพื้นที่จัดเก็บข้อมูล

ตัวอย่าง

จากระบบที่ประกอบไปด้วย User, Home, และ Work โดยที่ property ของ User ประกอบไปด้วย name, home, และ work ซึ่ง home และ work เป็น subobject ที่ประกอบไปด้วย property ที่มีคุณสมบัติ phone street, city และ zip



รูปที่ 2.22 การอ้างอิงแบบ Embedded Documents [6]

Home และ work property ไม่สามารเปลี่ยนแปลงค่าได้บ่อยนัก เนื่องจากอาจจะมี User หลายคนที่มี subobject ที่เหมือนกัน ถ้าหากข้อมูล subobject มีการเปลี่ยนแปลงแล้ว จะต้องแก้ไขข้อมูลของ object ทั้งหมดที่มีความสัมพันธ์กันด้วย ซึ่งจากรูปที่ 2.2 แสดงให้เห็น User collection ที่มี property ที่ชื่อว่า home และ work ทำการเก็บข้อมูลเป็น subobject ที่ประกอบไปด้วยข้อมูลของ home และ work

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7 AngularJS

AngularJS [7] คือ client-side script framework ที่ถูกพัฒนาโดย Google ที่เขียนขึ้นด้วยภาษา JavaScript ซึ่งใช้แนวคิดของ MVW (Model-View-Whatever) ในการออกแบบโครงสร้างของแอปพลิเคชันโดย AngularJS ได้เตรียมการทำงานต่างๆเอาไว้สำหรับการควบคุมการป้อนข้อมูลจากผู้ใช้งานผ่านทาง browser, จัดการข้อมูลทางฝั่งไคลเอนต์, และควบคุม element ในการแสดงผลต่างๆผ่านทาง browser

2.7.1 คุณสมบัติของ AngularJS

คุณสมบัติต่างๆของ AngularJS ที่เป็นประโยชน์ในการพัฒนาแอปพลิเคชันต่างๆ ประกอบไปด้วยคุณสมบัติดังนี้

Two-way Data binding

AngularJS ได้เตรียมวิธีการผูกข้อมูลทางฝั่ง controller เข้ากับ view ทำให้ข้อมูลที่ใช้งานเสมือนเป็นข้อมูลชุดเดียวกัน เมื่อมีการเปลี่ยนแปลงของข้อมูลทางฝั่งใด อีกฝั่งก็จะเปลี่ยนแปลงตามไปด้วย

Template

AngularJS จะทำการเปลี่ยน HTML page ธรรมดาให้กลายเป็น template HTML ที่เพิ่มประสิทธิภาพด้วย Directive ต่างๆ ซึ่งเป็น component ที่ AngularJS เตรียมไว้ให้ และยังสามารถสร้าง Directive ใช้งานเองได้อีกด้วย

MVW (Model-View-Whatever)

ในการพัฒนาแอปพลิเคชันต่างๆจะมีรูปแบบของการออกแบบที่แตกต่างๆกันออกไป ตัวอย่างเช่น MVC (Model-View-Controller), MVVM (Model-View-View-Model), และอีกมากมาย ซึ่ง AngularJS สามารถทำงานได้ไม่ว่าแอปพลิเคชันจะถูกออกแบบมาให้มีโครงสร้างแบบใดก็ตาม จึงเรียกการออกแบบแอปพลิเคชันที่พัฒนาด้วย AngularJS ว่า Model-View-Whatever หรือ MVW นั่นเอง

Dependency Injection

ในการพัฒนา AngularJS Application นั้นสามารถพัฒนาแยกเป็น module ต่างๆ และแต่ละ module สามารถเรียกใช้งานโดย module อื่นๆได้ ในลักษณะการเรียกใช้แบบเดียวกับการส่งค่า parameter เรียกกระบวนใช้งานแต่ละ module ต่างๆว่า Dependency Injection

Testing

AngularJS ได้ถูกออกแบบมาให้ทดสอบได้ง่ายขึ้น ซึ่งหมายความว่า จะสามารถทดสอบแอปพลิเคชันได้ขณะใดก็ได้ที่ต้องการทดสอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.2 Module

ในการสร้าง Angular Application นั้นสามารถรวมการทำงานต่างๆที่เกี่ยวข้องกับ AngularJS เข้าด้วยกันเป็น module ทำให้จำเป็นที่จะต้องประกาศการใช้งานใน HTML ก่อนจึงจะสามารถใช้คำสั่งต่างๆเกี่ยวกับ AngularJS ได้ ข้อดีของการรวมคำสั่งต่างๆเป็น module คือ แต่ละ module จะสามารถเรียกใช้ module อื่นๆได้ ผ่านกระบวนการ Dependency Injection, reuse การใช้งาน module ได้, และจำกัด scope ของส่วนที่มีการทำงานด้วย AngularJS ได้

2.7.3 Directive

AngularJS Directive คือ component ที่ประกอบไปด้วยกลุ่มของโค้ดภาษา HTML และ JavaScript ที่เตรียมฟังก์ชันการทำงานเฉพาะต่างๆเอาไว้ ด้วยคุณสมบัติของ Directive ทำให้สามารถนำ component กลับมาใช้งานใหม่ได้ภายในแอปพลิเคชัน Directive สามารถเป็นได้ทั้ง attribute ของ HTML tag หรือว่าจะเป็น HTML tag เองก็ได้ ในแอปพลิเคชันที่สร้างถ้าหากว่ามีการใช้งานแต่ละ component ที่ซ้ำๆกันเกิดขึ้น การสร้าง Directive สำหรับ component เหล่านั้นจะเป็นการช่วยลดการเขียนโค้ดที่ซ้ำๆกันได้

AngularJS นั้นได้เตรียม Directive พื้นฐานที่เตรียมไว้สำหรับใช้งานได้ทันที ซึ่งมี component ที่สำคัญต่อการพัฒนาแอปพลิเคชันดังนี้

ng-app

ng-app เป็น Directive ที่มีไว้สำหรับเริ่มต้นการทำงานของแอปพลิเคชันของ AngularJS ตัว ng-app นั้นจะอยู่ในลักษณะที่เป็น attribute ของ HTML tag ซึ่ง tag ที่อยู่ภายใต้ tag ที่มี ng-app เป็น attribute นั้น จะสามารถเรียกใช้การทำงานต่างๆของ AngularJS ได้เมื่อแอปพลิเคชันได้เริ่มต้นการทำงาน ในหนึ่งแอปพลิเคชันนั้นสามารถที่จะมีการประกาศ ng-app ได้มากกว่า 1 ครั้ง แต่การเรียกการใช้งาน ng-app หลายๆที่ จะทำให้แอปพลิเคชันเกิดความซับซ้อนโดยไม่จำเป็น ดังนั้นควรจะมีการประกาศ ng-app แค่ครั้งเดียวในแอปพลิเคชันถ้าหากมีการประกาศ ng-app เพิ่มให้พิจารณาจากการใช้งานแอปพลิเคชันนั้นๆ

ng-controller

ng-controller เป็น directive ที่อยู่ในลักษณะ attribute ของ HTML tag ซึ่งจะทำหน้าที่ในการระบุขอบเขตของ HTML ที่สามารถใช้งานฟังก์ชันต่างๆของ controller ที่ทำการระบุเอาไว้ใน value ของ ng-controller ในการประกาศ ng-controller นั้นสามารถมีได้มากกว่า 1 เช่นเดียวกับ ng-app และยังสามารถประกาศ ng-controller ซ้อนกันได้

ng-model

ng-model เป็น Directive ที่สำคัญอย่างยิ่งสำหรับ AngularJS โดย ng-model จะมีหน้าที่สำหรับ “ผูกข้อมูล” (bind) จาก input (input tag ใน HTML) เข้ากับ ตัวแปร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

scope ของ controller ซึ่งตัวแปรดังกล่าวจะถูกอ้างถึงด้วย $scope ซึ่งเป็น object ของ AngularJS ที่ใช้ในการผูกข้อมูล ตัวอย่างเช่น $scope.myVar ซึ่ง ng-model จะทำการอ้างถึง myVar เพื่อทำการผูกข้อมูลเข้าด้วยกัน เมื่อไรก็ตามที่ข้อมูลมีการเปลี่ยนแปลงบน HTML ข้อมูลของตัวอ้างอิง $scope ใน controller ก็จะเปลี่ยนแปลงตามด้วยเช่นกัน

ng-repeat

ในกระบวนการเรียกดูข้อมูลใน array หนึ่ง AngularJS นั้นจะมี Directive ที่ชื่อว่า ng-repeat ที่เมื่อนำไปประกาศเป็น attribute ให้กับ HTML tag ใดแล้วนั้น ก็จะเกิดการสร้าง tag ขึ้นมาซ้ำตามจำนวนความยาวของ array นั้นๆ แต่ข้อมูลจะแตกต่างกันออกไปตามข้อมูลใน array รวมถึงยังมีฟังก์ชัน filter ที่ใช้สำหรับการกรองข้อมูล และฟังชัน order ที่ใช้ในการเรียงลำดับของข้อมูลอีกด้วย

ng-show และ ng-hide

- **ng-show** เป็น Directive ที่ถูกใช้มากที่สุดเช่นเดียวกัน โดยจะทำหน้าที่ในการแสดง DOM เมื่อ ng-show มีค่าเท่ากับ true
- **ng-hide** จะทำหน้าที่ในการซ่อน DOM เมื่อ ng-hide มีค่าเท่ากับ true

ng-if

ng-if จะแตกต่างกับ ng-show และ ng-hide โดยที่ ng-if จะทำการแสดง DOM element ก็ต่อเมื่อมีค่า เป็น true

2.7.4 Controller

รูปที่ 2.23 การทำงานร่วมกันระหว่าง Controller และ View

Controller คือโค้ดภาษา JavaScript ที่ทำการควบคุมกระบวนการต่างๆที่เกี่ยวข้องกับข้อมูล ซึ่งจะทำงานร่วมกับ view (หรือ HTML) โดยการประกาศใช้งาน controller นั้นจะต้องประกาศผ่าน ng-controller และตามด้วยชื่อของ controller นั้นๆ เนื่องจากว่า controller ทำงานร่วมกันกับ view จึงสามารถผูกข้อมูลจากทั้งสองฝั่งได้ และเมื่อมีข้อมูลจากฝั่งใด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เปลี่ยนแปลง ข้อมูลของอีกฝั่งก็จะเปลี่ยนตามไปด้วย เรียกลักษณะการทำงานแบบนี้ว่า two-way data binding

2.7.5 เซอร์วิส

ในการพัฒนา Angular Application บางครั้งในหลาย controller อาจมีการใช้งานฟังก์ชันต่างๆที่เหมือนกัน ทำให้เกิดการเขียนฟังก์ชันการทำงานแบบเดิมขึ้นซ้ำๆ ทำให้เกิดความซ้ำซ้อนของโค้ดที่เขียนไป Angular Service จึงเป็นตัวที่ช่วยในการนำฟังก์ชันนั้นมาเขียนแยกออกเป็น component และเมื่อใดก็ตามที่ controller ต้องการจะใช้ฟังก์ชัน ก็ให้ใช้วิธีการ Dependency Injection Service เข้ามาใช้งานในแต่ละ controller ซึ่งจะทำให้ลดความซ้ำซ้อนของการเขียนโค้ด เนื่องจากเป็นการเขียนโค้ดเพียงแค่ครั้งเดียว ซึ่ง AngularJS ก็ได้ทำการเตรียมเซอร์วิสต่างๆที่จำเป็นไว้ให้ใช้งานเช่นเดียวกัน ตัวอย่างเช่น $route, $http, $window เป็นต้น

2.7.6 Router

Router คือ ฟังก์ชันที่ใช้กำหนดเส้นทางการเข้าถึงแอปพลิเคชันที่สร้างด้วย AngularJS โดยใช้ Provider ชื่อว่า $routeProvider กำหนดข้อมูลของ route โดยใช้ method ที่ชื่อว่า when ทำการตั้งค่าต่างๆของ route จากนั้น เมื่อ user ทำการคลิก link ที่สอดคล้องกับ route นั้นๆ ก็จะทำการเปลี่ยน view ตาม route ที่ตรงกัน ซึ่ง view จะแสดงผลผ่าน directive ที่ชื่อว่า ui-view

2.7.7 เปรียบเทียบ AngularJS กับ jQuery

jQuery คือ JavaScript library ที่ช่วยเพิ่มความสามารถของ webpage ให้มีความสามารถต่างๆ โดยวิธีการทำงานของ jQuery คือเมื่อ browser ทำการโหลด resource ที่เป็น HTML จากเว็บเซิร์ฟเวอร์แล้วทำการสร้าง DOM (Document Object Model) ซึ่งเป็นโครงสร้างที่ browser จะทำการแปลงข้อมูล แล้วแสดงผลให้กับ user ตัว jQuery จะเข้าไปจัดการกับ DOM เหล่านั้น เรียกวิธีการนี้ว่า DOM manipulation เพื่อเพิ่มความสามารถต่างๆ ของ webpage ที่แสดงผล ตัวอย่างเช่น จัดการ HTML event, เรียกใช้ request แบบ AJAX, เพิ่ม Animation และ Effect ต่างๆ

ในขณะที่ AngulaJS นั้น เป็น JavaScript framework ที่มีพื้นฐานแบบ MVW ซึ่งการทำงานของ AngularJS นั้นจะไม่ได้ใช้วิธี DOM Manipulation แต่จะทำการเพิ่มคุณสมบัติ หรือสร้าง DOM ขึ้นมาใหม่โดย Directive ซึ่งเป็น component ของ AngularJS เพื่อที่จะจัดการกับ HTML element ต่างๆ และด้วยคุณสมบัติของ scope ภายใน controller ทำให้เกิดลักษณะการทำงานที่เรียกว่า two-way data binding นั่นหมายความว่า AngularJS จะมอง HTML เป็น template ที่ต้องทำการ render HTML บวกกับ Directive ที่ถูกเพิ่มเข้าไป เพื่อแสดงผลให้กับผู้ใช้ แต่กับ jQuery ต้องรอให้ browser ทำการเปลี่ยน HTML element เป็น DOM เพื่อที่ jQuery จะได้เข้าไปจัดการกับ DOM นั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.8 Module เสริมของ AngularJS

ในปัจจุบัน ได้มีนักพัฒนาทำการสร้าง module สำหรับการทำงานต่างๆด้วย AngularJS เกิดขึ้นมากมาย ซึ่งช่วยอำนวยความสะดวกในการนำมาใช้งาน โดยตัวอย่างของ module ที่ใช้งานมีดังนี้

Angular-Facebook

angular-facebook เป็น module ที่มีไว้สำหรับการทำ authentication กับ Facebook โดยต้องทำการสร้าง Facebook Application ขึ้นมาก่อน เพื่อที่จะทำการร้องขอข้อมูลส่วนตัวจากบัญชีผู้ใช้ Facebook มาดำเนินการต่างๆ ในแอปพลิเคชันเราได้

Angular File Upload

Angular File Upload เป็น module ของ AngularJS ที่สนับสนุนการ upload ไฟล์ต่างๆ, การลากและวางเพื่อทำการอัพโหลดไฟล์, การสร้าง queue เพื่อทำการอัพโหลด อีกทั้งยังสามารถตรวจสอบไฟล์ที่ใช้ในการอัพโหลด เพื่อคัดกรองไฟล์นั้นได้อีกด้วย

ตัว module นั้นสนับสนุนการ upload ด้วย native HTML5 แต่ได้ลดระดับไปอยู่ในรูปแบบของการ upload ด้วย iframe สำหรับการสนับสนุน browser รุ่นเก่าด้วย และยังสามารถทำงานร่วมกับ server-side script ที่สนับสนุน HTML form upload ได้เช่นกัน

เมื่อ file ได้ถูกเลือก หรือ ถูกลากมาวางบนพื้นที่ upload filter ซึ่งเป็น function ของ module จะถูกเรียกให้ทำงาน เพื่อคัดกรอง file ตามที่กำหนดเอาไว้ file ที่ผ่าน filter มาได้นั้นจะถูกเพิ่มไปยัง queue เพื่อรอดำเนินการ upload ทันทีที่พร้อม

AngularUI Router

AngularUI Router คือ AngularJS framework ที่ใช้ในการสร้าง state สำหรับการ routing โดยการนิยาม state machine สำหรับ interface ต่างๆของ web ซึ่งแตกต่างจาก $route service ที่ AngularJS เตรียมไว้ให้ตรงที่ $route service นั้นจะทำการสร้าง URL สำหรับการ routing

Angular-UTF8-Base64

Angular-UTF8-Base64 เป็น module ของ AngularJS framework ที่ถูกพัฒนาขึ้นมาเพื่อใช้ในการเข้ารหัสและถอดรหัสของข้อมูลแบบ 64บิต และสนับสนุนข้อมูลที่อยู่ในรูปแบบ utf8 อีกด้วย

2.8 GoJS

GoJS [8] คือ JavaScript Library ที่พัฒนาโดย Northwoods Software มีคุณสมบัติที่เกี่ยวกับการสร้าง diagram ต่างๆโดยใช้งานผ่าน browser ที่สามารถทำงานได้หลาย platform GoJS ได้เตรียมโครงสร้างของ diagram ที่ซับซ้อนเอาไว้สำหรับนำมาสร้าง ประกอบไปด้วย node (รูปทรงของ diagram), link (เส้นเชื่อมโยงของ diagram), และกลุ่มของ template และ layout ที่สามารถปรับแต่งได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

GoJS ยังได้เตรียมคุณสมบัติที่สูงขึ้นไปอีก นั่นคือคุณสมบัติ drag-and-drop, copy-and-paste, in-place text editing, template, data binding, transactional state และ undo management, palette, overview, การจัดการกับ event, คำสั่ง และเครื่องมือต่างๆสำหรับกำหนดการดำเนินการต่างๆเอง

GoJS เขียนขึ้นด้วยภาษา JavaScript ดังนั้น การทำงานจะไม่ข้องกี่ยวกับเซิร์ฟเวอร์โดย GoJS จะทำงานบน browser โดยผ่านการ render ด้วย HTML5 canvas element หรือ SVG โดยที่ไม่ต้องทำงานร่วมกับ server-side script อีกทั้งยังไม่ขึ้นกับ JavaScript Framework หรือ Library ใดๆ ดังนั้น มันจึงทำงานร่วมกับ Framework หรือ Library ใดๆก็ได้

2.8.1 การเริ่มต้นการทำงานของ GoJS

การเริ่มต้นการทำงานของ GoJS library จะต้องทำการเรียกใช้งานฟังก์ชันที่ใช้สำหรับการเริ่มต้นการทำงานของ GoJS library ก่อน โดยสามารถเรียกใช้งานได้ดังนี้

```
var gojs = go.GraphObject.make;
```

รูปที่ 2.24 การประกาศตัวแปรเริ่มต้นการทำงานของ GoJS

2.8.2 การเรียกใช้งานฟังก์ชันของ GoJS

หลังจากทำการเริ่มต้นการทำงานด้วยการเรียกฟังก์ชัน go.GraphObject.make แล้วนั้น จะทำให้สามารถเรียกใช้งานฟังก์ชันของ GoJS ได้ ซึ่งฟังก์ชันของ GoJS จะมีพารามิเตอร์ที่ใช้ทำการระบุถึงประเภทของสิ่งที่กำลังจะสร้างต่างๆ เช่น แผนภาพ, รูปทรงต่างๆที่ใช้ในแผนภาพ, เส้นเชื่อม เป็นต้น

```
gojs(ประเภทของสิ่งที่จะสร้าง, css selector, [option ต่างๆ]);
```

รูปที่ 2.25 คำสั่งสร้างส่วนต่างๆ ของ GoJS

ตารางที่ 2.8 พารามิเตอร์ของ GoJS ที่ใช้ในการพัฒนาแอปพลิเคชัน

พารามิเตอร์	คำอธิบาย
ประเภทของสิ่งที่จะสร้าง	เป็นประเภทของสิ่งที่จะสร้างโดย GoJS จำเป็นที่จะต้องใส่ทุกครั้ง ในการสร้าง
css selector	Class หรือ id ที่ใช้ระบุ element ของ HTML ที่จะทำการแสดงผลจาก GoJS จะมีหรือไม่มีก็ได้
Option ต่างๆ	ตัวเลือก ที่สามารถกำหนดให้กับส่วนที่ใช้ในการแสดงผลต่างๆ ซึ่งจะมี หรือไม่มีก็ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
วิธีการดำเนินงานสหกิจศึกษา

วิธีดำเนินงานการพัฒนาแอปพลิเคชันด้วยภาษา JavaScript มีลำดับขั้นตอนการพัฒนาต่างๆ แบ่งเป็นขั้นตอนต่างๆ ดังนี้

3.1 รายละเอียด และขอบเขตของแอปพลิเคชัน

Application ที่ทดลองพัฒนา มีรายละเอียด และขอบเขตของแอปพลิเคชันดังต่อไปนี้

3.1.1 รายละเอียดของแอปพลิเคชัน

Application ที่ทดลองพัฒนาด้วยภาษา JavaScript เป็นแอปพลิเคชันที่เกี่ยวกับการสร้าง diagram ไม่ว่าจะเป็น diagram อย่างง่าย เช่น diagram ที่มีโครงสร้างแบบ tree ไปจนถึง diagram ที่มีความซับซ้อนที่ใช้ในการออกแบบการพัฒนาต่างๆ

ระบบบัญชีผู้ใช้ และการลงทะเบียนเข้าใช้งาน

Application นั้นมีระบบบัญชีผู้ใช้งาน สำหรับทำการบันทึกข้อมูล diagram เพื่อระบุว่า diagram เป็นของผู้ใดสร้าง โดยจะต้องทำการสมัครสมาชิกเพื่อทำการลงทะเบียนเข้าใช้งานก่อน โดยการสมัครสมาชิก สามารถสมัครได้ 2 วิธี ได้แก่สมัครโดยตรงกับทางแอปพลิเคชันและการสมัครโดยเชื่อมโยงข้อมูลบัญชีผู้ใช้งานจากสื่อสังคมออนไลน์ คือ Facebook จากนั้นเมื่อผู้ใช้งาน ทำการลงทะเบียนเข้าใช้งานแล้วจะเข้ามาสู่หน้าแอปพลิเคชันโดยส่วนของแอปพลิเคชันจะประกอบไปด้วยพื้นที่สามส่วน ซึ่งส่วนแรกจะเป็นส่วนที่ทำการเก็บ template ของ diagram ที่จะสร้าง ส่วนที่สองคือส่วนของพื้นที่ที่จะใช้สร้าง diagram ส่วนที่สามคือส่วนของรายการ diagram ที่เคยสร้างไว้

การสร้าง Diagram

ในการสร้าง diagram สามารถเลือกประเภทของ diagram ที่ต้องการจะสร้าง ได้จากนั้นเมื่อทำการเลือก diagram ที่ต้องการจะสร้างแล้ว สามารถเลือก template ของ diagram จาก palette โดยวิธีการ drag-and-drop ในกระบวนการสร้าง diagram นั้นสามารถที่จะ redo และ undo การกระทำต่างๆ และมีฟังก์ชัน copy-and-paste ข้อมูลที่ทำการสร้างได้

การบันทึกข้อมูลของ Diagram

หลังจากที่ทำการปรับแต่ง diagram ได้ตามที่ต้องการแล้ว เมื่อต้องการที่จะบันทึกข้อมูล จะสามารถบันทึกข้อมูลลงบนฐานข้อมูลได้ โดยวิธีการบันทึกข้อมูลจะบันทึกข้อมูลของ diagram, ประเภทของ diagram, และชื่อของผู้สร้าง diagram

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเรียกดูข้อมูลของ Diagram

ข้อมูลที่ถูกบันทึกเก็บไว้ในฐานข้อมูลนั้นสามารถที่จะเรียกดูได้ โดยข้อมูลจะแสดงอยู่ในส่วนของรายการข้อมูล ซึ่งจะแสดงรายการของ diagram ตามผู้ใช้งาน และประเภทของ diagramนั้นๆ เมื่อทำการเลือกข้อมูลจากรายการแล้ว จะทำการแสดง diagram ที่ทำการเลือกนทันที โดยการเลือกข้อมูล มีข้อจำกัดคือ สามารถเลือกข้อมูลได้เพียง 1 diagram เท่านั้น

การแก้ไข Diagram

การแก้ไข diagram สามารถดำเนินการ เพิ่ม, ลบ, redo, undo diagram ได้ ในกรณีที่ต้องการแก้ไขข้อมูลของ diagram ที่เคยบันทึกเอาไว้ จะต้องทำการเลือก diagram จากรายการข้อมูล diagram เสียก่อน จึงจะสามารถทำการแก้ไข diagram ได้

การลบข้อมูลของ Diagram

การลบข้อมูล จะเป็นการลบข้อมูลของ diagram ที่ทำการบันทึกเอาไว้ในฐานข้อมูล สามารถทำได้โดยการเลือก diagram ที่ต้องการจะลบทิ้ง จากนั้นทำการลบข้อมูล โดยจะมี dialog เตือนว่าต้องการลบข้อมูลนั้นๆหรือไม่ ถ้าตอบตกลง ก็จะทำการลบข้อมูล diagram ที่เลือก ออกจากระบบฐานข้อมูล

การ Import Diagram

ข้อมูล diagram ที่ทำงานบนแอปพลิเคชันนั้น เป็นข้อมูลในรูปแบบ object ของ JavaScript ดังนั้น ข้อมูล diagram สามารถ import เข้ามาได้ โดยประเภทของข้อมูลต้องอยู่ในรูปแบบของ JSON file เนื่องจาก JSON สามารถแปลงข้อมูลเป็น object ของ JavaScript ได้ โดยใช้เพียงแค่คำสั่งเดียวในการทำการแปลงข้อมูล ซึ่งสะดวกต่อการใช้งานเป็นอย่างมาก

การ Export Diagram

การ export ข้อมูลสามารถทำได้ 2 วิธี ได้แก่ การ export ข้อมูลเป็น JSON file เนื่องจากข้อมูล diagram ที่แสดงให้เห็นนั้น แท้จริงแล้วเป็นข้อมูลในรูปแบบ object ของ JavaScript ซึ่งสามารถเปลี่ยนเป็นข้อมูลในรูปแบบ JSON ได้ การ export ข้อมูล จึงสามารถ export ออกมาได้ในรูปแบบ JSON file และการ export อีกวิธีคือ export ออกมาในรูปแบบ รูปภาพชนิด png การ export ข้อมูลไม่จำเป็นต้องทำการบันทึกข้อมูลลงในฐานข้อมูลก่อน ก็สามารถทำการ export ได้

การใช้งานในส่วนของผู้ที่ไม่ทำการลงทะเบียน

ส่วนผู้ใช้งานที่ไม่ได้สมัครสมาชิกจะทำได้เพียงทดลองใช้งานแอปพลิเคชันได้เพียงอย่างเดียว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การบันทึกการทำงานต่างๆลงบน log

ในการดำเนินการต่างๆกับ diagram จะมีการบันทึกข้อมูลการดำเนินการเก็บไว้บน log ด้วย

3.1.2 ขอบเขตของแอปพลิเคชัน

ขอบเขตของแอปพลิเคชันแสดงไว้ในตารางที่ 3.1

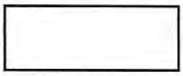
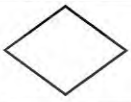
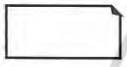
ตารางที่ 3.1 ขอบเขตของแอปพลิเคชัน

กระบวนการ	คำอธิบาย	การอนุญาติ	
		ผู้ใช้งานทั่วไป	ผู้ใช้งานลงทะเบียน
ลงทะเบียน	ลงทะเบียนสมัครสมาชิก ผ่านแอปพลิเคชันหรือ Facebook	ใช่	ใช่
ลงชื่อเข้าใช้งาน	ลงชื่อเข้าใช้งานโดยใช้ชื่อบัญชี และรหัสผ่าน หรือลงทะเบียนด้วย Facebook	ใช่	ใช่
สร้าง diagram	สร้าง diagram จากประเภท diagram ที่เลือก	ไม่ใช่	ใช่
แก้ไข diagram	แก้ไข diagram จาก diagram ที่สร้างหรือ diagram ที่มาจากแหล่งข้อมูล	ไม่ใช่	ใช่
เรียกดูข้อมูล diagram	เรียกดูข้อมูล diagram ที่บันทึกไว้จากฐานข้อมูล	ไม่ใช่	ใช่
บันทึกข้อมูล diagram	บันทึกข้อมูล diagram ลงบนฐานข้อมูล	ไม่ใช่	ใช่
ลบ diagram	ลบ diagram ที่บันทึกออกจากฐานข้อมูล	ไม่ใช่	ใช่
Import diagram	Import diagram จาก JSON file ที่เก็บข้อมูลของ diagram เอาไว้	ไม่ใช่	ใช่
Export diagram	Export diagram ที่สร้างไว้ โดยสามารถ export ได้ 2 รูปแบบคือ JSON file และ Image PNG file	ไม่ใช่	ใช่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับรูปทรงต่างๆ ที่ใช้ในการพัฒนาแอปพลิเคชัน แสดงไว้ในตารางที่ 3.2

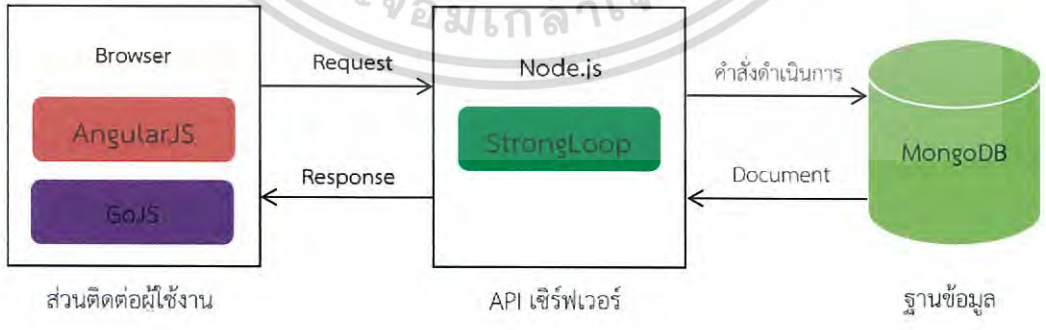
ตารางที่ 3.2 รูปทรงต่างๆ ที่ใช้ในการพัฒนาแอปพลิเคชัน

รูปทรง	ชื่อ	คำอธิบาย
	Start	จุดเริ่มต้นการทำงาน
	Process	กระบวนการต่างๆ (process)
	Decision	การตัดสินใจ (decision)หรือการเปรียบเทียบ (comparision)
	Connector	จุดเชื่อมต่อ (connector)
	End	จุดสิ้นสุดการทำงาน
	Comment	คอมเม้นต์

3.2 การออกแบบแอปพลิเคชัน

3.2.1 สถาปัตยกรรมของแอปพลิเคชัน

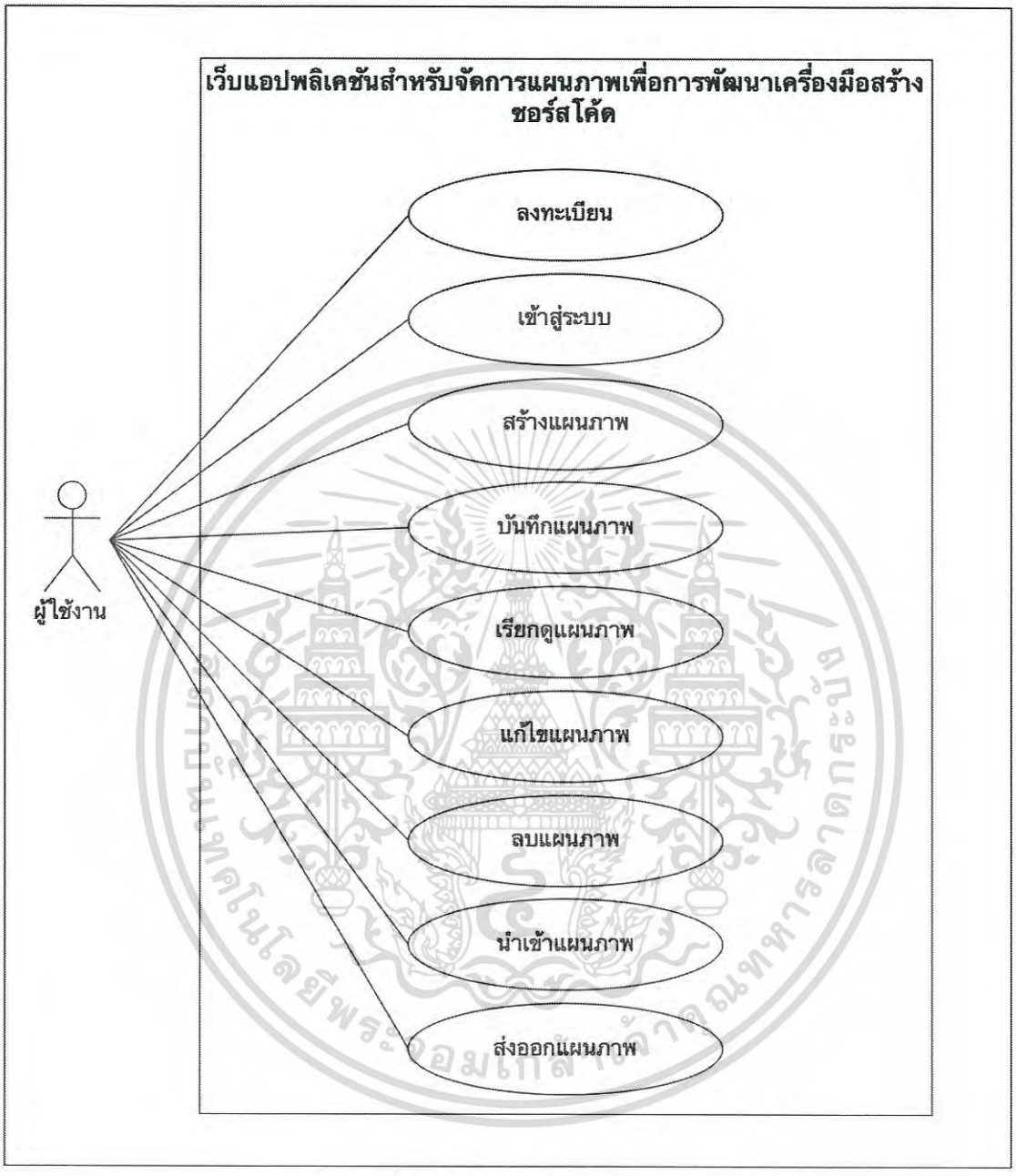
สถาปัตยกรรมของแอปพลิเคชันจะมีโครงสร้างแบ่งเป็น 3 ส่วนที่สำคัญ ประกอบไปด้วย ส่วนติดต่อผู้ใช้งาน ส่วนของเซิร์ฟเวอร์ และส่วนของฐานข้อมูล โดยส่วนติดต่อผู้ใช้งาน จะใช้ AngularJS ในการควบคุมการทำงานของแอปพลิเคชัน และ GoJS ในการกำหนดรายละเอียดของเทมเพลตที่ใช้สร้างแผนภาพ ส่วนของ API จะพัฒนาบน Node.js โดยใช้ StrongLoop เฟรมเวิร์กเป็นตัวสร้าง API และฐานข้อมูล จะใช้ MongoDB ในการเก็บข้อมูล ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 สถาปัตยกรรมของแอปพลิเคชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.2 แผนภาพ Use Case

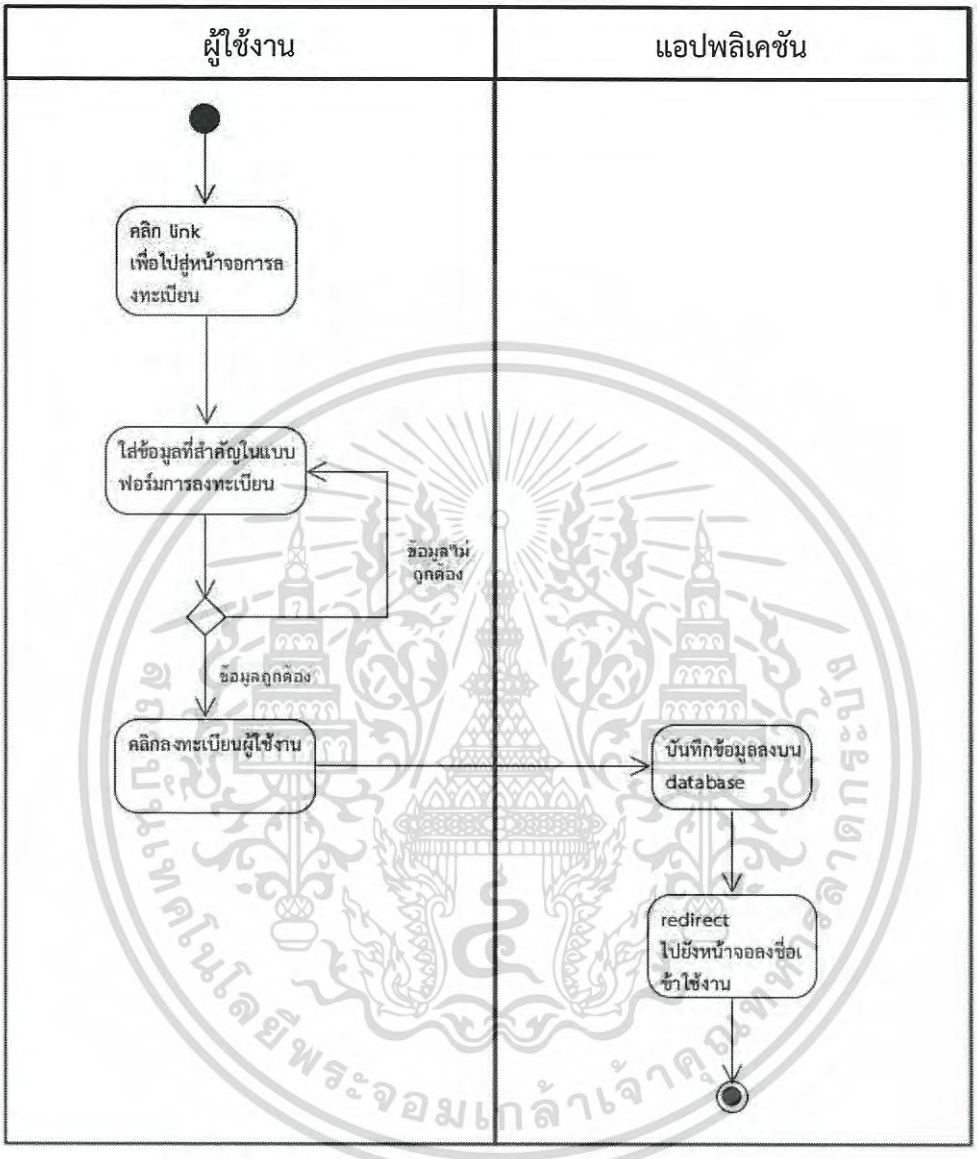


รูปที่ 3.2 แผนภาพ Use Case

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.3 แผนภาพกิจกรรม

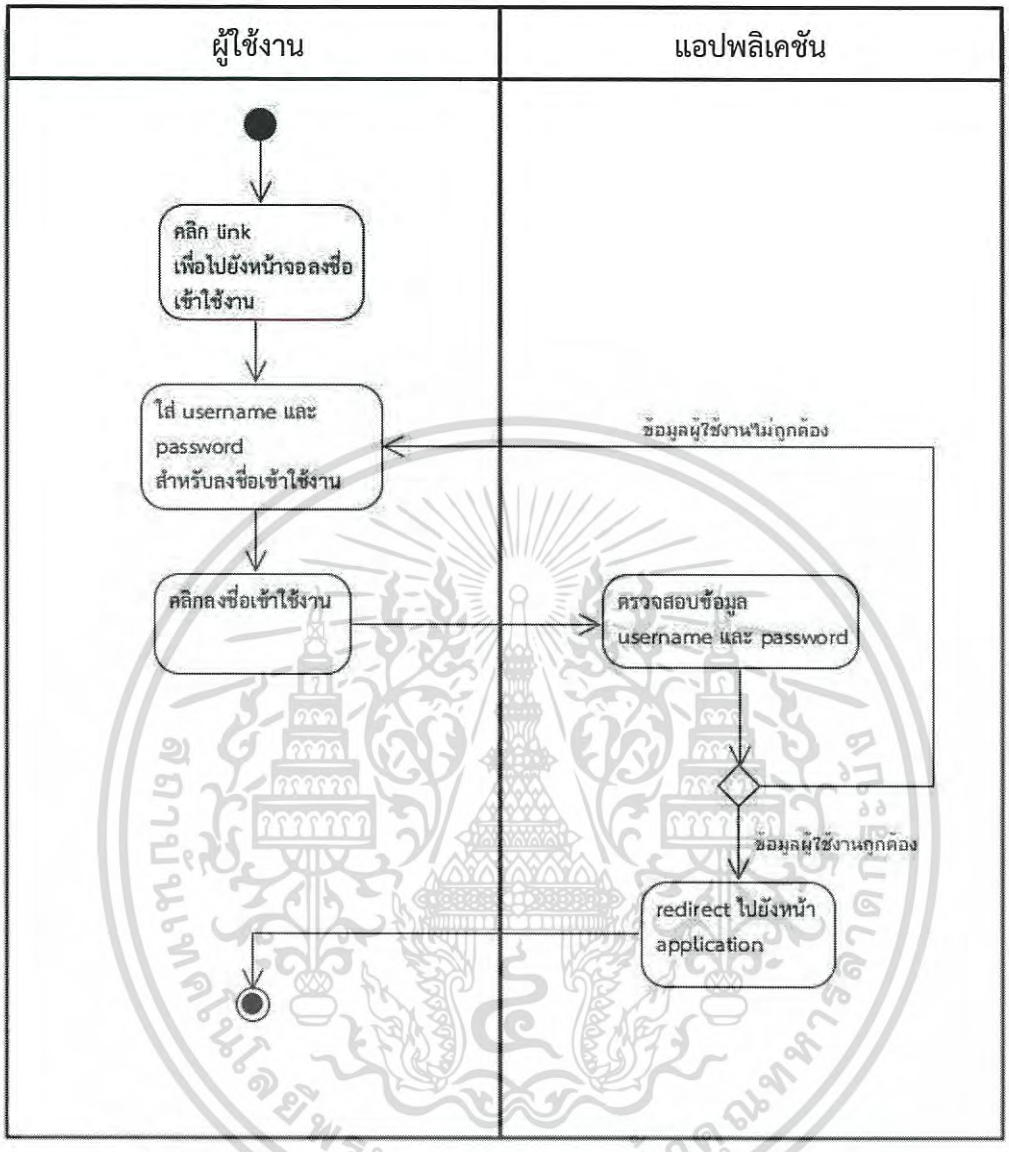
การลงทะเบียนเข้าใช้งาน



รูปที่ 3.3 Activity Diagram การลงทะเบียนเข้าใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

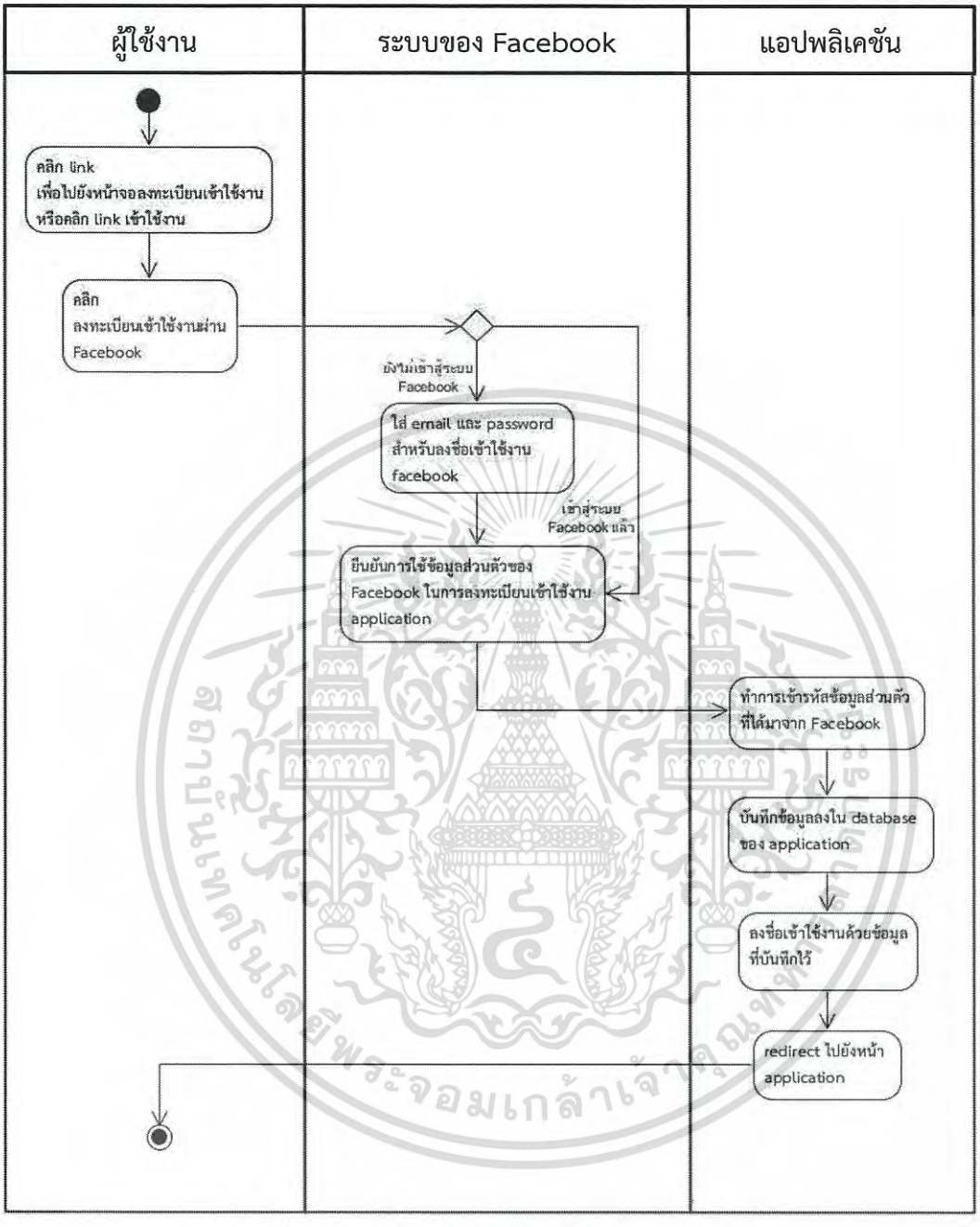
การเข้าใช้งานด้วยบัญชีผู้ใช้ของแอปพลิเคชัน



รูปที่ 3.4 Activity Diagram การเข้าใช้งานด้วยบัญชีผู้ใช้ของแอปพลิเคชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

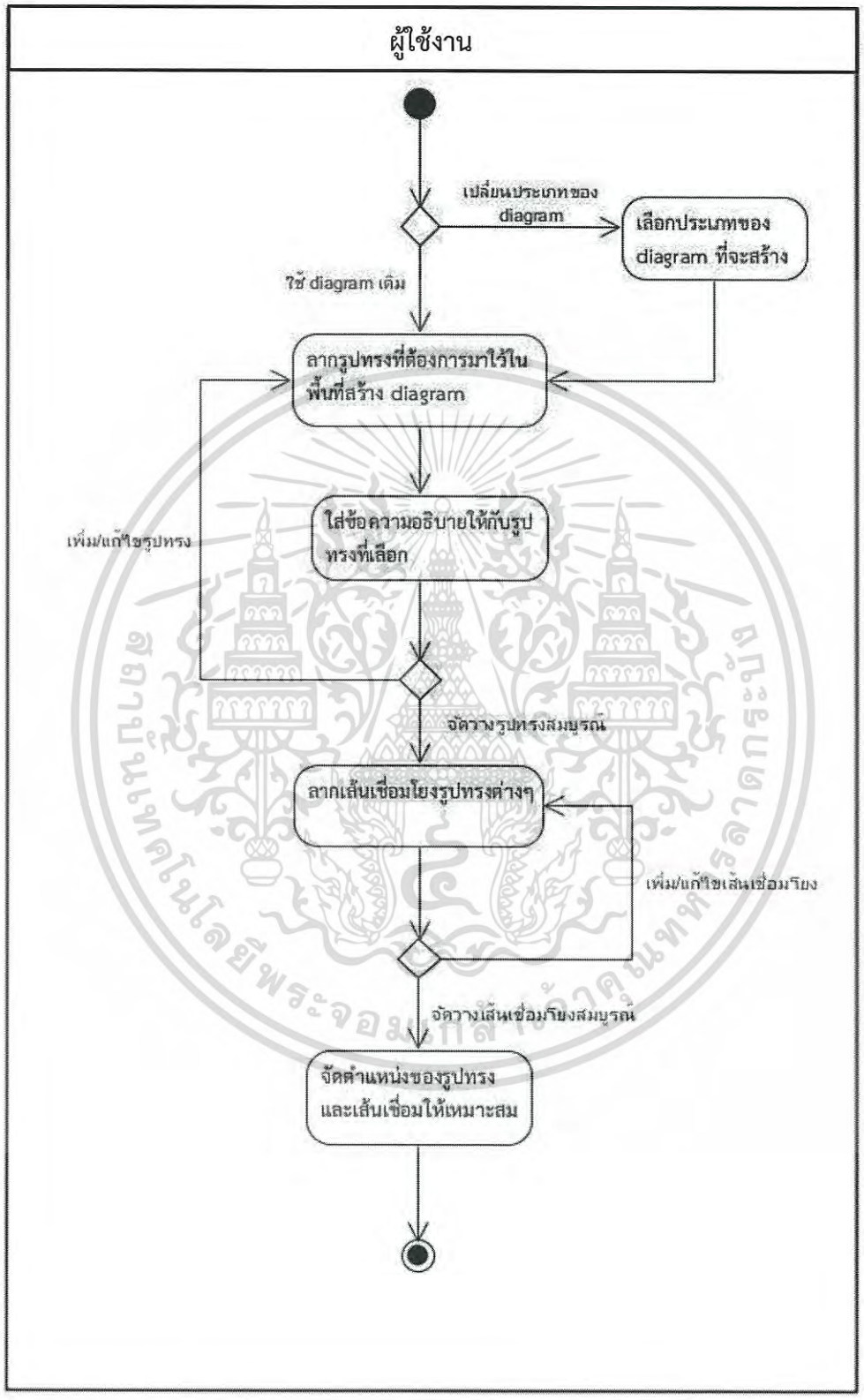
การเข้าใช้งานด้วยบัญชีผู้ใช้ของ Facebook



รูปที่ 3.5 การเข้าใช้งานด้วยบัญชีผู้ใช้ของ Facebook

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

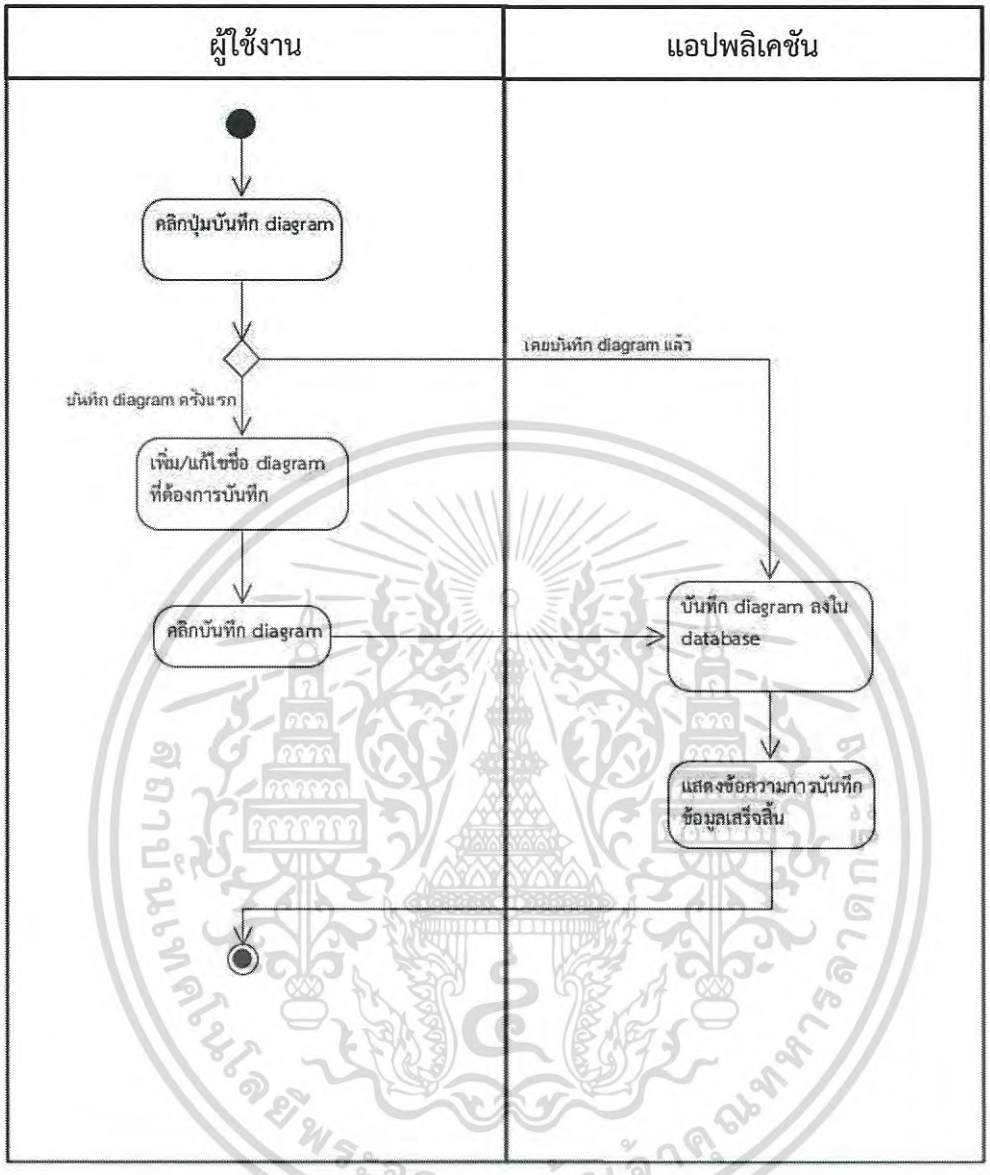
การทำงานของแอปพลิเคชัน



รูปที่ 3.6 Activity Diagram การทำงานของแอปพลิเคชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

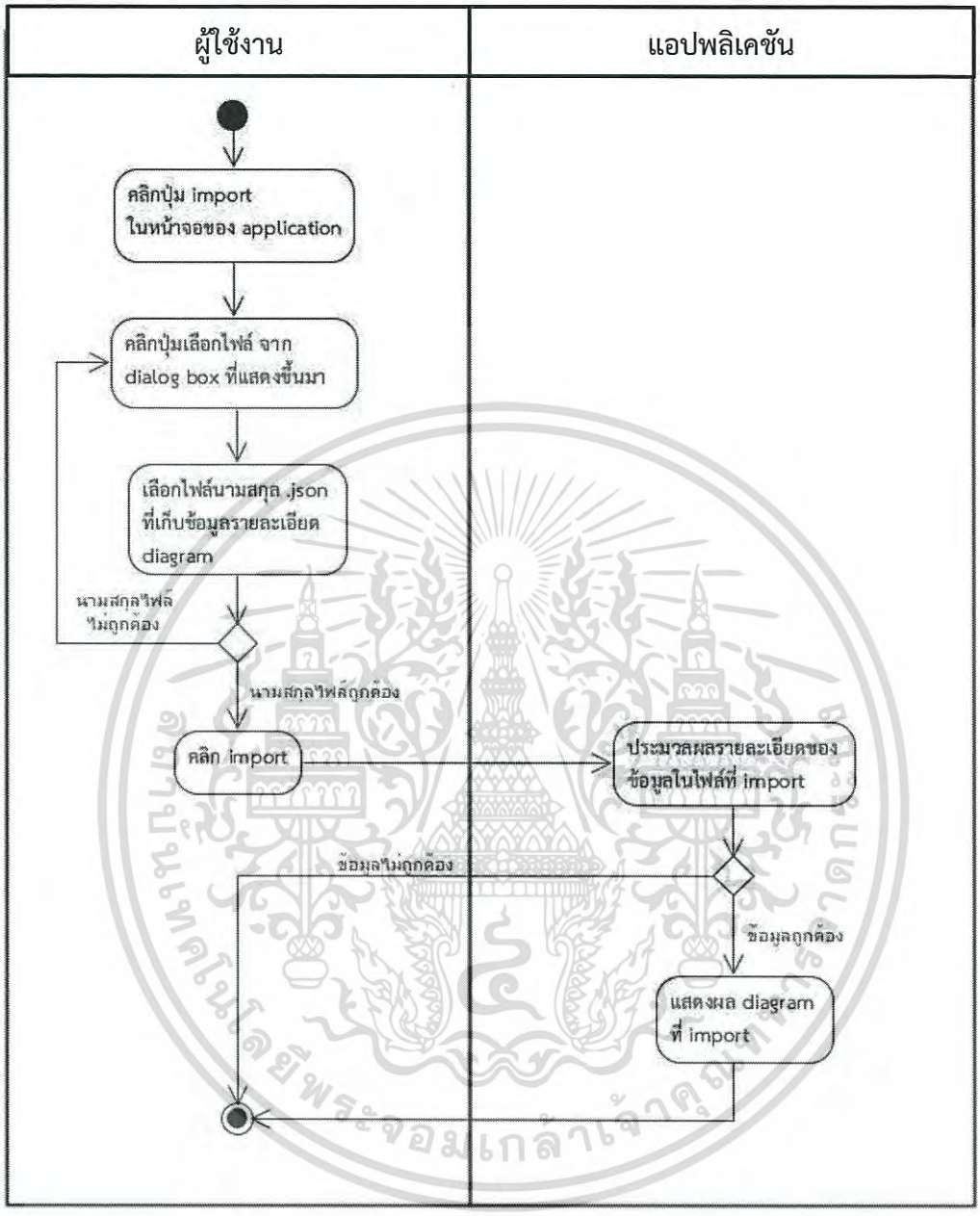
การบันทึกข้อมูล



รูปที่ 3.7 Activity Diagram การบันทึกข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

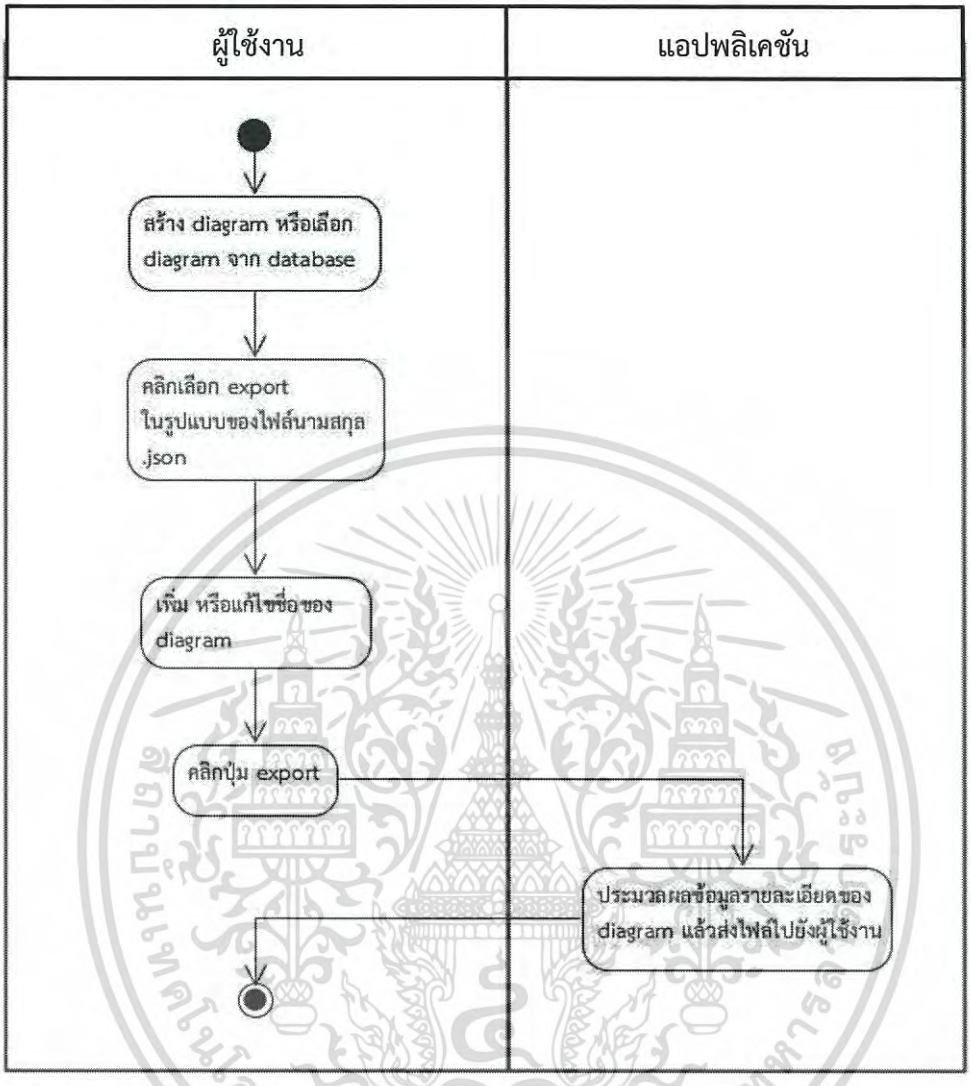
การนำเข้างาน (Import)



รูปที่ 3.8 Activity Diagram การนำเข้างาน (Import)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

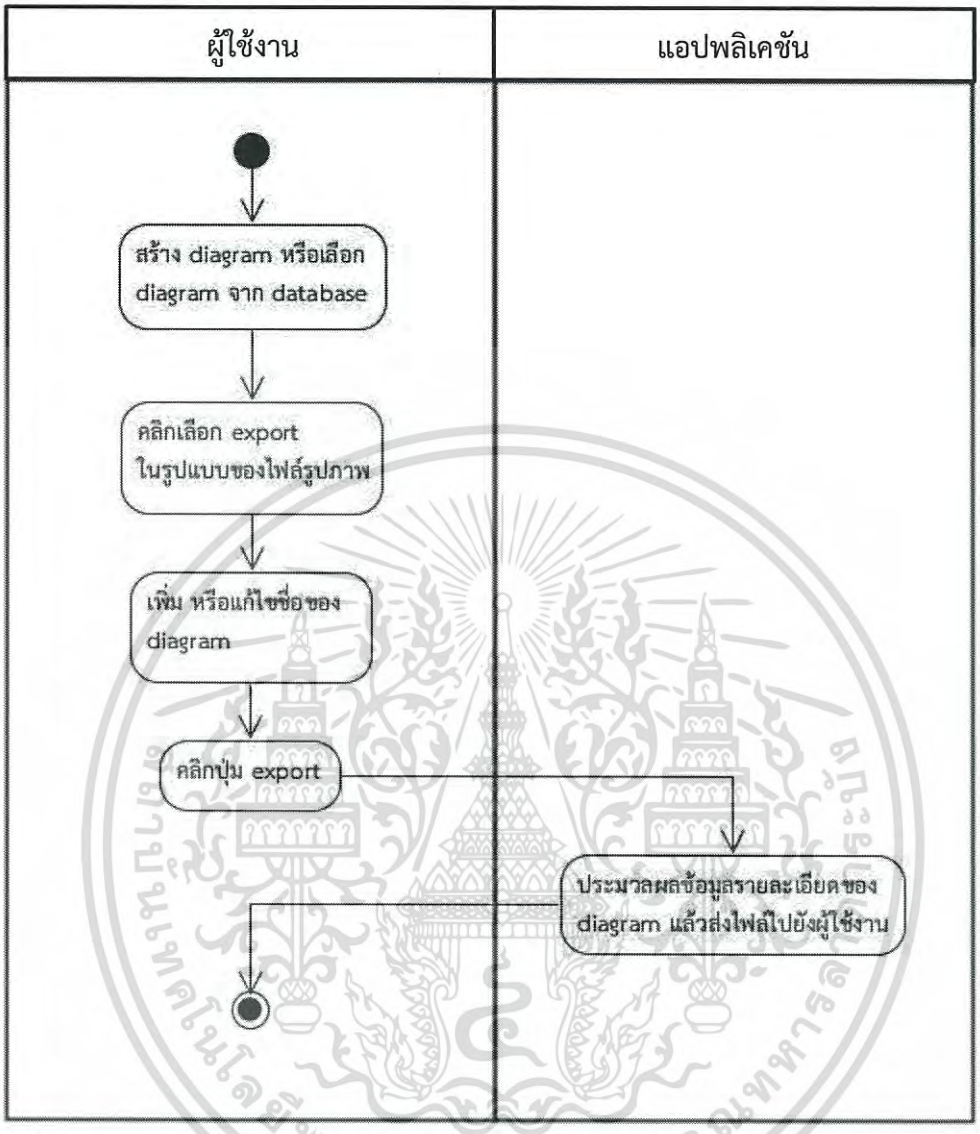
การส่งออกงานในรูปแบบ JSON file (Export to JSON file)



รูปที่ 3.9 Activity Diagram การส่งออกงานในรูปแบบ JSON file (Export to JSON file)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

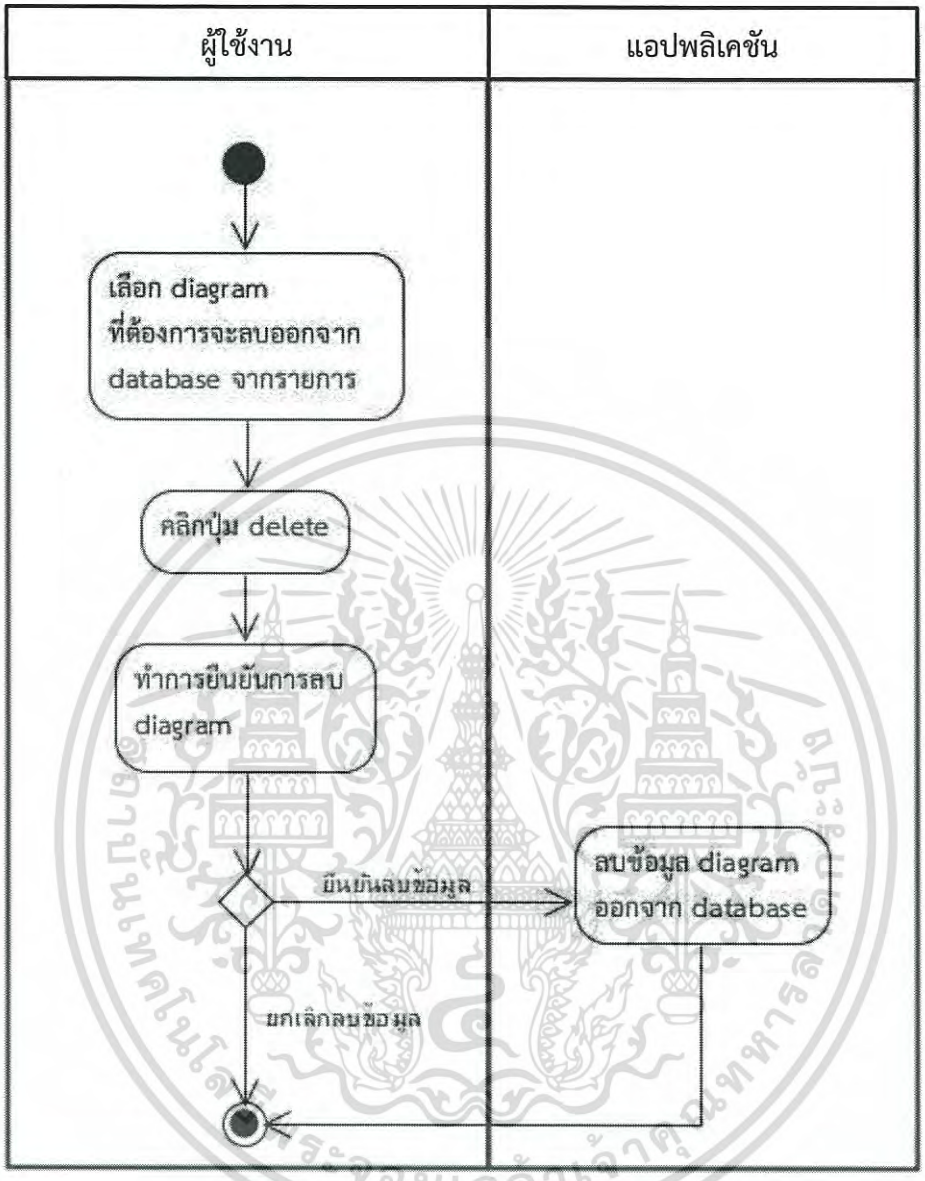
การส่งออกงานในรูปแบบ Image file (Export to PNG file)



รูปที่ 3.10 Activity Diagram การส่งออกงานในรูปแบบ Image file (Export to PNG file)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

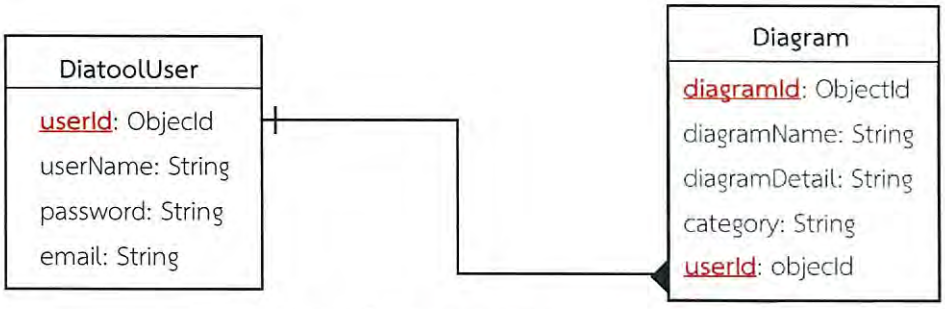
การลบข้อมูล



รูปที่ 3.11 Activity Diagram การลบข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.4 โครงสร้าง และรูปแบบของข้อมูลที่ใช้ในการจัดเก็บ



รูปที่ 3.12 โครงสร้างของข้อมูล

เนื่องจาก StrongLoop ทำงานด้วย LoopBack framework ที่ไม่สนใจว่า data sourceจะอยู่ในรูปแบบใด แต่จะกำหนดโครงสร้างของข้อมูลจาก model ซึ่งมีลักษณะคล้ายคลึงกับโครงสร้างของ Relational Database โดยโครงสร้างของแอปพลิเคชันที่ได้กำหนดไว้ดังรูปที่ 3.16 มีรายละเอียดดังนี้

DiatoolsUser model และ Diagram model เป็น model ของแอปพลิเคชันที่สร้างด้วย LoopBack framework ซึ่งเปรียบเสมือนกับ Entity ใน relational database และทั้งสอง model มีความสัมพันธ์ซึ่งกันและกัน โดยที่ DiatoosUser model มีความสัมพันธ์แบบ HasMany กับ Diagram model ส่วน Diagram model มีความสัมพันธ์แบบ BelongsTo กับ DiatoolsUser model ซึ่งเปรียบเทียบกัน relational database และคือ DiatoolsUser มีความสัมพันธ์แบบ one-to-many กับ Diagram นั่นเอง

ข้อมูลจะอยู่ในรูปแบบของ JSON สอดคล้องกับชนิดข้อมูลในฐานข้อมูล MongoDB โดยฐานข้อมูลที่ใช้จัดเก็บ จะประกอบไปด้วย Collection จำนวน 2 กลุ่ม ได้แก่ DiatoolsUser collection และ Diagram collection รายละเอียดโครงสร้างของข้อมูล แสดงไว้ในตารางที่ 3.3 และ 3.4

ตารางที่ 3.3 โครงสร้างข้อมูลของ DiatoolUser collection

ชื่อข้อมูล	ชนิดข้อมูล	คำอธิบาย
userId	ObjectId	หมายเลขระบุตัวตนของผู้ใช้งาน
userName	String	ชื่อของผู้ใช้งาน
password	String	รหัสผ่านในการลงชื่อเข้าใช้งาน
email	String	อีเมลที่ใช้ในการลงชื่อเข้าใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.4 โครงสร้างขอมูลของ Diagram collection

ชื่อข้อมูล	ชนิดข้อมูล	คำอธิบาย
diagramId	ObjectId	หมายเลขที่ใช้ในการอ้างถึงแผนภาพ
diagramName	String	ชื่อของแผนภาพตามที่ผู้ใช้สร้าง
diagramDetail	Object	รายละเอียดของแผนภาพ
category	String	ประเภทของแผนภาพที่สร้าง
userId	ObjectId	หมายเลขที่ใช้ในการอ้างอิงถึงผู้ใช้งานที่สร้างแผนภาพ

จากตารางที่ 3.3 และตารางที่ 3.4 จะมีความสัมพันธ์แบบ one-to-many กล่าวคือ ในผู้ใช้งาน 1 คน สามารถสร้างแผนภาพได้มากกว่า 1 ภาพ หรือไม่มีแผนภาพใดๆ โดยข้อมูลที่ใช้ในการอ้างอิงคือ userId และเนื่องจากคุณสมบัติของชนิดข้อมูลที่ใช้ในการจัดเก็บในฐานข้อมูล MongoDB นั้นสามารถมีชนิดข้อมูลแบบ object ที่มีลักษณะคล้ายกับ object ของภาษา JavaScript จึงทำการจัดเก็บข้อมูล diagramDetail ในรูปแบบของ object โดยมีรายละเอียดของ object แสดงในตาราง 3.5

ตารางที่ 3.5 โครงสร้างข้อมูลของ DiagramDetail object

ชื่อข้อมูล	ชนิดข้อมูล	คำอธิบาย
class	String	ระบุประเภทของแผนภาพ
linkFromPortIdProperty	String	ประเภทการเชื่อมต่อของแผนภาพตั้งต้น
linkToPortIdProperty	String	ประเภทการเชื่อมต่อของแผนภาพปลายทาง
nodeDataArray	Array	กลุ่มของแผนภาพที่ทำการสร้าง
linkDataArray	Array	กลุ่มของเส้นเชื่อมต่อระหว่างแผนภาพ

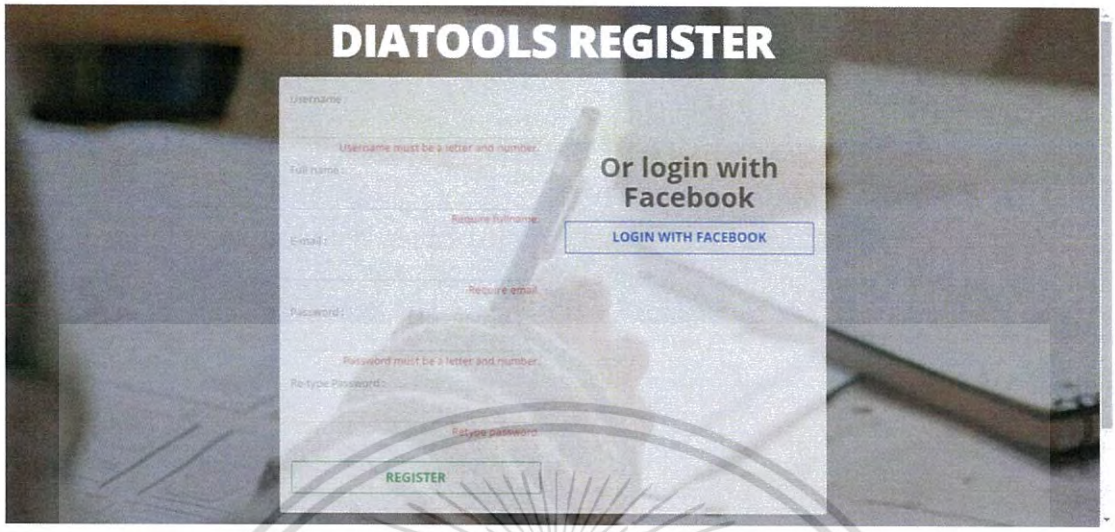
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.5 หน้าจอผู้ใช้งาน

รูปที่ 3.13 หน้าจอ Landing page

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Register page



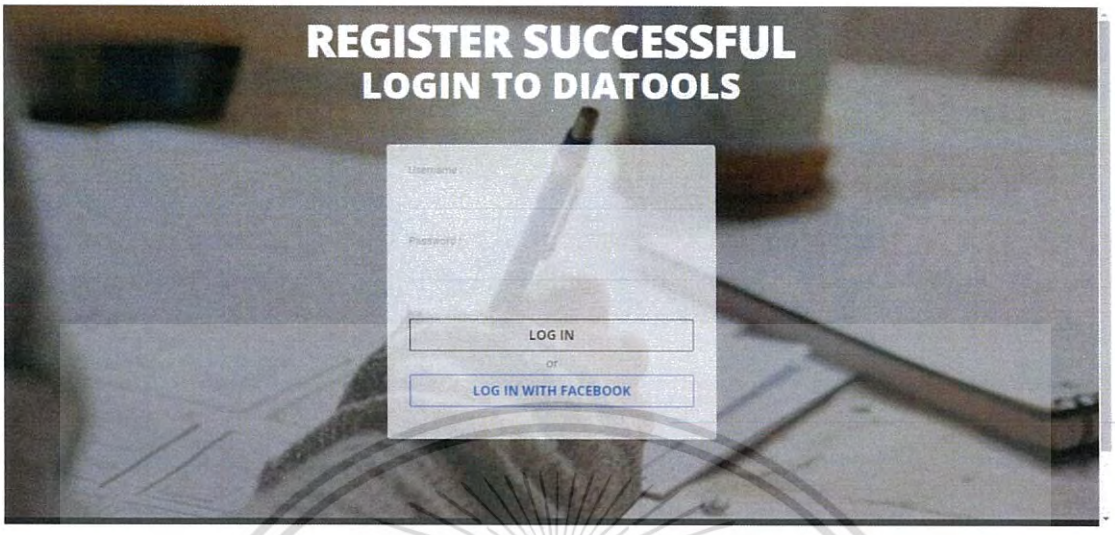
รูปที่ 3.14 หน้าจอ Register page

Login dialog

รูปที่ 3.15 หน้าจอ Login dialog

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Login page



รูปที่ 3.16 หน้าจอ Login page

Application page

รูปที่ 3.17 หน้าจอแอปพลิเคชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 กระบวนการพัฒนาแอปพลิเคชัน

3.3.1 การสร้างแอปพลิเคชัน

ในขั้นตอนการสร้างแอปพลิเคชันนั้นจะใช้เฟรมเวิร์กที่ชื่อว่า LoopBack ซึ่งเป็นเฟรมเวิร์กที่ใช้สร้างแอปพลิเคชัน และสามารถใช้สร้าง API ที่ใช้ดำเนินการกับข้อมูลได้ โดยสามารถสร้างแอปพลิเคชันด้วยเฟรมเวิร์ก LoopBack ซึ่งจะได้โครงสร้าง ตามตารางที่ 3.6

ตารางที่ 3.6 รายละเอียดโครงสร้างของแอปพลิเคชัน

File หรือ Directory	คำอธิบาย
Root directory	
package.json	JSON file ที่ทำการระบุรายละเอียด package ต่างๆที่ใช้ในแอปพลิเคชัน
/node_modules	directory ที่เก็บ package ต่างๆที่ต้องใช้งานในการพัฒนาแอปพลิเคชันตามรายละเอียดที่กำหนดไว้ใน package.json
README.md	File ที่แสดงรายละเอียดของ project
/server directory	
server.js	File หลักที่แอปพลิเคชันใช้ในการทำงาน
config.json	ใช้สำหรับตั้งค่าต่างๆของแอปพลิเคชัน
datasource.json	กำหนดรายละเอียดของ datasource ที่ใช้ในแอปพลิเคชัน
model-config.json	ตั้งค่า model ที่ใช้ในแอปพลิเคชันรวมถึงกำหนดว่า model เชื่อมต่อกับ datasource ใด
/server directory	
middleware.json	ใช้สำหรับกำหนด middleware
/boot	Directory ที่ทำการเก็บ file ต่างๆที่จะทำงานเมื่อเริ่มต้นการทำงานของแอปพลิเคชัน
/client directory	
README.md	File ที่แสดงรายละเอียดของ project ในส่วนของไคลเอนต์
File อื่นๆ	เป็น file ที่เกี่ยวข้องกับการดำเนินการกับเว็บไซต์
/common directory	
/Model	ทำการเก็บ JSON file และ JavaScript file ที่กำหนดรายละเอียดของ model ที่ใช้ในแอปพลิเคชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.2 การพัฒนาแอปพลิเคชันทางฝั่งเซิร์ฟเวอร์

ติดตั้ง และกำหนดรายละเอียดของ Data Source

ในการที่จะเก็บข้อมูลลงไปในฐานข้อมูล จำเป็นที่จะต้องติดตั้งตัวเชื่อมต่อเข้ากับฐานข้อมูล และต้องระบุรายละเอียดของฐานข้อมูลที่ใช้ในการเชื่อมต่อ ซึ่งมีรายละเอียดดังนี้

1) เนื่องจากแอปพลิเคชันที่ทำการพัฒนานั้นทำการเก็บข้อมูลในฐานข้อมูล MongoDB จึงต้องติดตั้งตัว connector ของ MongoDB โดยไปที่ไดเรกทอรีของโปรเจค จากนั้นพิมพ์คำสั่งดังต่อไปนี้ใน Command Line Interface

```
npm install loopback-connector-mongodb
```

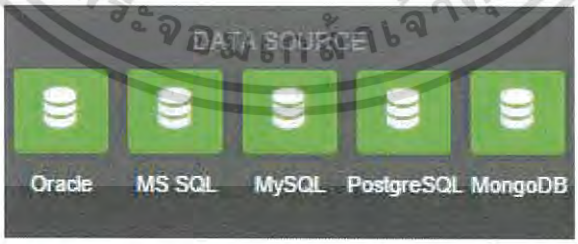
รูปที่ 3.18 คำสั่งติดตั้งตัวเชื่อมต่อฐานข้อมูล

2) ทำการเข้าสู่หน้าจอการจัดการแอปพลิเคชัน StrongLoop Arc โดยพิมพ์คำสั่งดังต่อไปนี้ใน Command Line Interface

```
E:\projects\project-diagram-tool>slc arc
```

รูปที่ 3.19 คำสั่งเข้าใช้งาน StrongLoop Arc

3) ทำการเข้าสู่ระบบจัดการแอปพลิเคชัน จากนั้นคลิกที่ composer แล้วดูในส่วนของการจัดการ data source ให้เลือกฐานข้อมูลแบบ MongoDB

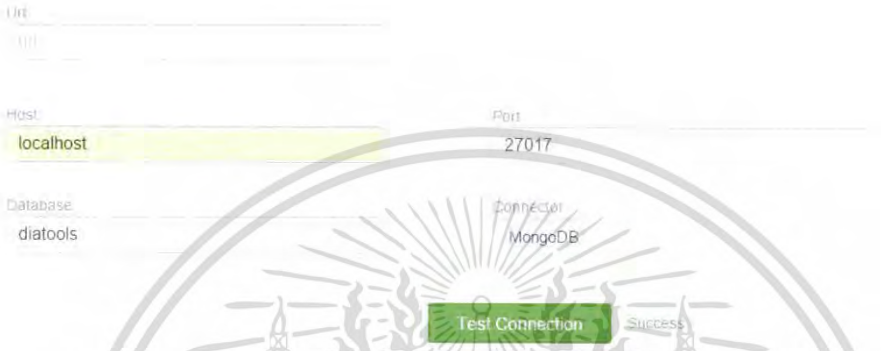


รูปที่ 3.20 ประเภทของฐานข้อมูลที่สามารถเชื่อมต่อได้

4) ทำการกำหนดรายละเอียดของฐานข้อมูล MongoDB โดยใส่ชื่อผู้ใช้งาน และรหัสผ่านของฐานข้อมูลตามที่ตั้งไว้ และกำหนดชื่อของฐานข้อมูลที่จะทำการจัดเก็บข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3.21 หน้าจอการกำหนดรายละเอียดของฐานข้อมูล(1)



รูปที่ 3.22 หน้าจอการกำหนดรายละเอียดของฐานข้อมูล(2)

5) หลังจากกำหนดรายละเอียดของฐานข้อมูลเรียบร้อยแล้ว ให้คลิกปุ่ม Test Connection ถ้าผลลัพธ์จากการทดสอบผ่าน จะขึ้นข้อความว่า Success นั่นหมายความว่า ฐานข้อมูล พร้อมใช้งานแล้ว

รูปที่ 3.23 การทดสอบการเชื่อมต่อฐานข้อมูล

หมายเหตุ: ไม่จำเป็นที่จะต้องสร้างฐานข้อมูลเตรียมไว้ เพราะ MongoDB จะทำการสร้างฐานข้อมูล และ คอลเล็กชันให้ทันทีที่มีการเพิ่มข้อมูลเข้าสู่ฐานข้อมูล

การกำหนดรายละเอียดของ Model ที่จะนำมาสร้าง API

การดำเนินการกับข้อมูลในแอปพลิเคชันจะกระทำการผ่าน API ซึ่งการทำงานของ API จะขึ้นอยู่กับการกำหนดรายละเอียดของ model โดยสามารถกำหนดรายละเอียดของ model ได้ผ่านทาง StrongLoop Arc ซึ่งวิธีการสร้างมีรายละเอียดดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

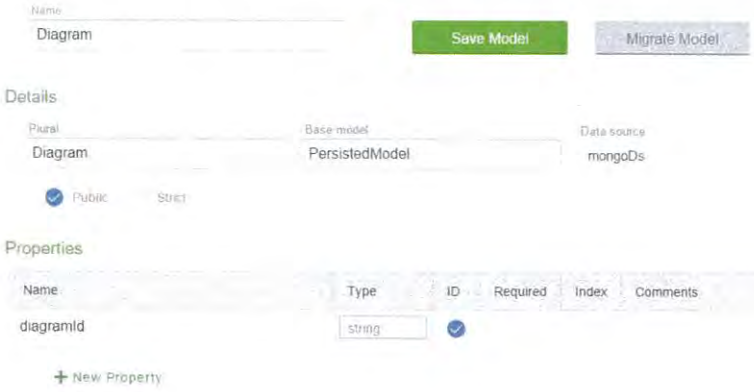
1. ไปยังหน้าจอการจัดการแอปพลิเคชันของ StrongLoop Arc คลิกที่ Composer จากนั้นดูในส่วนของการสร้าง model แล้วทำการคลิก New

รูปที่ 3.24 ปุ่มสำหรับสร้าง model ใหม่

2. ทำการกำหนดรายละเอียดของ model โดยจะสร้าง model 2 ตัวด้วยกัน model แรก เป็น model ที่กำหนดรายละเอียดของผู้ใช้งานของแอปพลิเคชัน มีชื่อ model ว่า DiatoolsUser ซึ่งมีรายละเอียดแสดงดังรูปที่ 3.25

รูปที่ 3.25 หน้าจอกำหนดรายละเอียดของ DiatoolsUser model

Model ตัวที่สอง เป็น model ที่กำหนดรายละเอียดของแผนภาพต่างๆ มีชื่อ model ว่า Diagram มีรายละเอียดแสดงดังรูปที่ 3.26



รูปที่ 3.26 หน้าจอกำหนดรายละเอียดของ Diagram model

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในส่วนของ properties ให้ทำการสร้างแค่ ID ของแต่ละ model เท่านั้น เนื่องจากว่าฐานข้อมูลที่ทำการเก็บ เป็นแบบ NoSQL ซึ่งสามารถเพิ่ม properties ได้ โดยขึ้นอยู่กับข้อมูลที่จัดเก็บ

การกำหนดความสัมพันธ์ให้กับ Model

การกำหนดความสัมพันธ์ให้กับ model เป็นการกำหนดความสัมพันธ์แบบ HasMany ให้กับ model DiatoolsUser และ Diagram โดยมีความสัมพันธ์กันคือ DiatoolsUser HasMany Diagram

กำหนดรายละเอียดให้กับ Middleware

การกำหนดรายละเอียดให้กับ middleware เป็นการกำหนดสิ่งที่แอปพลิเคชันจะตอบสนอง เมื่อมีผู้ใช้งานทำการเรียกใช้แอปพลิเคชัน ซึ่งการกำหนด middleware มีรายละเอียดดังต่อไปนี้

1) กำหนดทรัพยากรต่างๆที่ใช้ในการทำงานของแอปพลิเคชันเมื่อถูกเรียกใช้งาน

```
app.use('/js', loopback.static(__dirname + '../../../client/js'));
app.use('/bower_components', loopback.static(__dirname + '../../../client/bower_components'));
app.use('/css', loopback.static(__dirname + '../../../client/css'));
app.use('/view', loopback.static(__dirname + '../../../client/view'));
```

รูปที่ 3.27 รายละเอียดการกำหนดทรัพยากรต่างๆ ของแอปพลิเคชัน

2) กำหนด route ให้กับแอปพลิเคชันเพื่อทำการตอบสนองต่อผู้ที่เรียกใช้งาน โดยจะทำการส่งข้อมูลกลับไปตามที่ผู้ใช้งานร้องขอ

```
app.get("/", function(req, res){
  res.sendFile(path.resolve(__dirname, "../../client/index.html"));
});

app.get("/register", function(req, res){
  res.sendFile(path.resolve(__dirname, "../../client/index.html"));
});

app.get("/tools*", function(req, res){
  res.sendFile(path.resolve(__dirname, "../../client/index.html"));
});

app.get("/login*", function(req, res){
  res.sendFile(path.resolve(__dirname, "../../client/index.html"));
});
```

รูปที่ 3.28 เส้นทางสำหรับใช้แสดงผลของแอปพลิเคชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.3 การพัฒนาแอปพลิเคชันทางฝั่งไคลเอนต์

การสร้างแอปพลิเคชันทางฝั่งไคลเอนต์ด้วย AngularJS

ในการสร้างแอปพลิเคชันทางฝั่งไคลเอนต์จะใช้ AngularJS ในการสร้างแอปพลิเคชัน โดยทำการกำหนด AngularJS module ที่ชื่อว่า app สำหรับจัดการแอปพลิเคชัน ดังแสดงในรูปที่ 3.29

```
angular.module("app", ['ui.router', 'ngResource', 'ngAnimate',
'ngMessages','lbServices', 'angularFileUpload', "facebook",
"ab-base64", "flash", "angular-loading-bar", "angularSmoothscroll"]);
```

รูปที่ 3.29 รายการของ AngularJS module ต่างๆ

โดย AngularJS module จะทำการ injection ไลบรารีที่จำเป็นเข้ามาใช้งานในแอปพลิเคชัน ซึ่งจะถูกกำหนดเอาไว้ในอาร์เรย์ ซึ่งเป็นพารามิเตอร์ตัวที่สองในการสร้าง module รายการของไลบรารีที่ทำการ injection เข้ามามีรายละเอียดดังแสดงไว้ในตารางที่ 3.7

ตารางที่ 3.7 ไลบรารีของ AngularJS ที่ใช้ในแอปพลิเคชัน

ไลบรารี	คำอธิบาย
ui.router	จัดการ route ของแอปพลิเคชัน ทำให้แอปพลิเคชันมีการทำงานแบบ Single Page ได้
ngResource	ทำให้แอปพลิเคชันสามารถเรียกใช้งาน RESTful เซอร์วิสได้
ngAnimate	ช่วยในการจัดการแอนิเมชันของแอปพลิเคชันได้
ngMessage	จัดการข้อความที่แสดง เมื่อมีการป้อนข้อมูลเกิดขึ้น
lbServices	เป็นไลบราลีที่ทำให้สามารถเรียก API ทางฝั่งเซิร์ฟเวอร์ได้
angularFileUpload	ใช้ในการ import ไฟล์เข้าสู่แอปพลิเคชัน
facebook	ใช้ในการเชื่อมต่อแอปพลิเคชันโดยบัญชีผู้ใช้งานของ Facebook
Ab-base64	ใช้ในการเข้ารหัสของผู้ใช้งาน
flash	ใช้ในการแสดงข้อความ
angular-loading-bar	ใช้สำหรับแสดงแถบการโหลดข้อมูล
angularSmoothScroll	ใช้สำหรับการเชื่อมโยงภายในหน้าเดียวกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กำหนดเส้นทางที่ใช้ในการเข้าถึงแอปพลิเคชันด้วย AngulaJS Config

เป็นการกำหนด route ในการแสดงผลให้กับแอปพลิเคชัน ซึ่งจะทำให้แอปพลิเคชันมีการทำงานแบบ Single Page Application โดยใช้ไลบรารีของ AngularJS ที่ชื่อว่า Angular ui.router ทำการกำหนดรูปแบบของ route

การสร้าง AngularJS Service เพื่อจัดการกับข้อมูลของแอปพลิเคชัน

การสร้าง AngularJS Service เพื่อใช้ในการจัดการกับข้อมูลมีรายละเอียดดังแสดงในตารางที่ 3.8

ตารางที่ 3.8 เซอร์วิสของ AngularJS ที่ใช้ในแอปพลิเคชัน

เซอร์วิส	คำอธิบาย
authService	จัดการข้อมูลของผู้ใช้งาน
diagramService	จัดการข้อมูลของแผนภาพที่สร้าง
fbService	จัดการการเชื่อมต่อโดยใช้บัญชีผู้ใช้งาน Facebook
gojsService	จัดการรูปแบบการทำงานของแผนภาพ
lbService	เป็นเซอร์วิสที่ใช้ติดต่อกับฐานข้อมูล สร้างโดย LoopBack
modalService	จัดการในส่วนของการแสดงผลแบบหน้าต่าง
savefileService	จัดการบันทึกข้อมูลให้อยู่ในรูปแบบ JSON และ ไฟล์ PNG
uploadService	ช่วยทำให้สามารถนำข้อมูลของแผนภาพที่อยู่ในรูปแบบ JSON เข้าสู่แอปพลิเคชัน

สร้าง AngularJS Controller สำหรับจัดการแอปพลิเคชัน

การสร้าง AngularJS controller เป็นการกำหนดการจัดการต่างๆ ให้กับแอปพลิเคชัน ประกอบไปด้วย controller ต่างๆ ดังแสดงในตารางที่ 3.9

ตารางที่ 3.9 Controller ของ AngularJS ที่ใช้ในแอปพลิเคชัน

Controller	คำอธิบาย
authentication	เป็น controller ที่ใช้จัดการเกี่ยวกับการลงชื่อเข้าใช้งาน, การออกจากระบบ, และ การลงทะเบียนเข้าใช้งาน
diagram	เป็น controller ที่ใช้จัดการเกี่ยวกับการดำนเนินการ CRUD ของแผนภาพ รวมถึง การจัดการแผนภาพไม่ว่าจะเป็น การย้อนการกระทำของแผนภาพ การเรียกคืนการกระทำของแผนภาพ, การ import ข้อมูลของแผนภาพที่อยู่ในรูปแบบของ JSON, การ export ข้อมูลออกไปในรูปแบบ JSON และ ไฟล์ PNG

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การสร้าง AngularJS Directive สำหรับการแสดงผลแม่แบบของรูปทรงต่างๆ

การสร้าง AngularJS Directive สำหรับการแสดงผลแม่แบบของรูปทรงต่างๆ เป็นการสร้างส่วนของการแสดงผลแม่แบบที่จะใช้ในการสร้างแผนภาพ แม่แบบของแผนภาพจะถูกกำหนดโดยไลบรารีที่ชื่อว่า GoJS ซึ่งมีรายละเอียด ดังนี้

1) กำหนด metadata ให้กับ directive ซึ่งเป็นออบเจ็คของ JavaScript ที่กำหนดรูปแบบของ directive ที่ต้องการให้แสดงผลในแอปพลิเคชัน

```
restrict: "E",
scope: false,
link: flowchart,
template: '<div class="panel-body" id="palette" style="width: Auto; height: 500px;"></div>'
```

รูปที่ 3.30 การกำหนดรายละเอียดของ AngularJS directive

Metadata ของ Directive ประกอบไปด้วยคุณสมบัติต่างๆ แสดงไว้ในตารางที่ 3.10

ตารางที่ 3.10 คุณสมบัติของ AngularJS Directive

คุณสมบัติ	**คำอธิบาย**
restrict	เป็นข้อความ ที่ระบุว่า directive ที่ใช้แสดงใน HTML อยู่ในรูปแบบใด โดยสามารถเป็นได้ทั้ง E (Element) และ A (Attribute)
scope	เป็นคุณสมบัติที่เอาไว้กำหนดขอบเขตการทำงานของตัวแปรที่ใช้กับ directive ถ้าเป็น false คือไม่จำกัดขอบเขต และ true คือจำกัดขอบเขต ให้ใช้ได้แต่ใน directive นี้เท่านั้น
link	เป็นคุณสมบัติที่ใช้เชื่อมโยงกับฟังก์ชัน ที่ทำการกำหนดการทำงานของ directive
template	เป็นคุณสมบัติที่เก็บข้อมูลภาษา HTML ซึ่งอยู่ในรูปแบบข้อความ เอาไว้บอกว่า directive มีการแสดงผลอย่างไร เมื่อแอปพลิเคชันทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2) ทำการกำหนดรายละเอียดเทมเพลตของรูปทรงต่างๆ ตามที่แสดงไว้ในรูปที่ 3.31

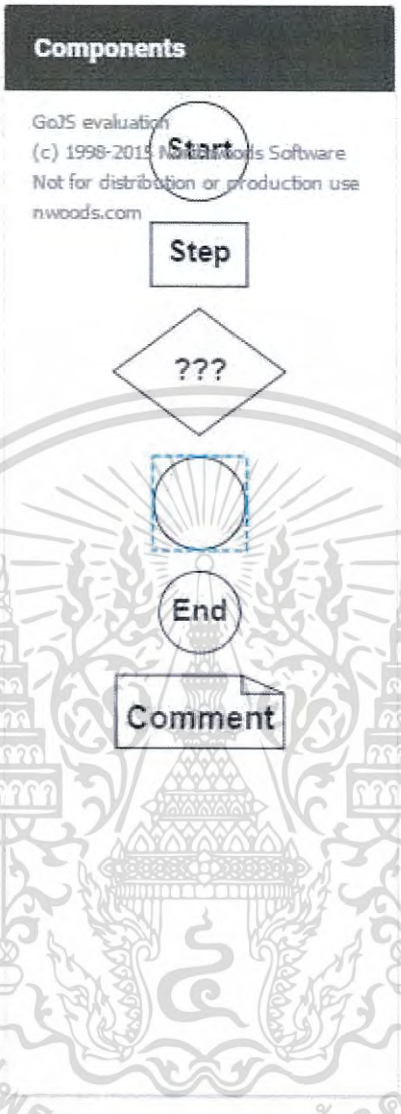
```
scope.myPalette =
  scope.g(go.Palette, "palette",
    {
      "animationManager.duration": 800,
      nodeTemplateMap: diagram.nodeTemplateMap,
      model: new go.GraphLinksModel([
        {
          category: "Start",
          text: "Start"
        }, {
          text: "Step"
        }, {
          text: "???",
          figure: "Diamond"
        }, {
          figure: "Circle"
        }, {
          category: "End",
          text: "End"
        }, {
          category: "Comment",
          text: "Comment"
        }
      ])
    });
```

รูปที่ 3.31 การกำหนดรูปทรงต่างๆ ของแผนภาพ

จากรูปที่ 3.31 แสดงถึงวิธีการกำหนดรูปทรงต่างๆ ของเทมเพลต ซึ่ง category จะหมายถึงประเภทของรูปทรงที่ไลบรารีมีให้แล้ว ส่วน text คือข้อความที่อยู่บนรูปทรงนั้นๆ และ figure คือส่วนที่กำหนดรูปทรงตามต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ได้ผลลัพธ์ ตามรูปที่ 3.32



รูปที่ 3.32 ผลลัพธ์ของรูปทรงต่างๆ ตามที่กำหนดเอาไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
ผลการดำเนินงานและอภิปรายผล

จากการพัฒนาแอปพลิเคชันด้วย JavaScript ได้ผลเป็นแอปพลิเคชันที่สามารถดำเนินการต่างๆได้ตามข้อกำหนดและขอบเขตของแอปพลิเคชันได้อย่างครบถ้วน โดยผลการทดลองพัฒนาแอปพลิเคชันมีรายละเอียดดังต่อไปนี้

4.1 ผลการทดสอบแอปพลิเคชัน

จากการพัฒนาแอปพลิเคชันด้วยภาษา JavaScript สามารถทดสอบแอปพลิเคชันโดยใช้เครื่องของผู้พัฒนาเป็นเครื่องทดสอบ และจัดการทดสอบตามรายละเอียด และขอบเขตของแอปพลิเคชันได้ผลของการทดสอบแบ่งเป็นหัวข้อได้ดังนี้

4.1.1 การเข้าถึงแอปพลิเคชัน

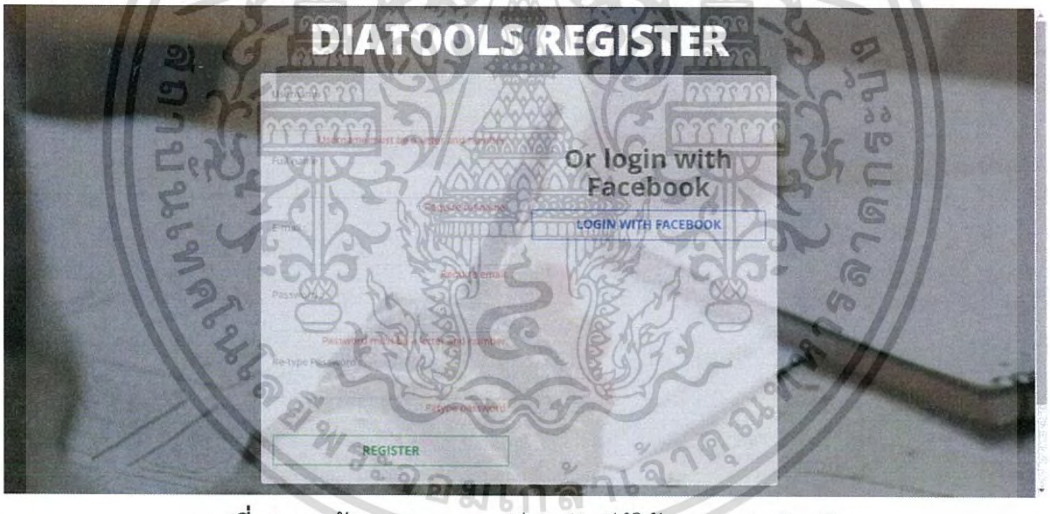
เนื่องจากเครื่องที่ใช้เป็นเซิร์ฟเวอร์ของแอปพลิเคชันกับเครื่องทดสอบการเข้าถึงของแอปพลิเคชันนั้นเป็นเครื่องเดียวกัน การทดสอบจึงเป็นการส่ง request ของ URL ไปที่ http://localhost:3000/ ซึ่งเป็น route ที่ส่ง request ไปยังเครื่องของตัวผู้พัฒนาเอง โดยกำหนด port ที่เป็นช่องทางในการเข้าถึงแอปพลิเคชันเท่ากับ 3000 และ ทำการทดสอบการเข้าถึงแอปพลิเคชันด้วย REST end-point ได้ผลลัพธ์แสดงออกมาได้ดังตารางที่ 4.1

ตารางที่ 4.1 ผลลัพธ์จากการเข้าถึง URI ของแอปพลิเคชัน

URI ที่ใช้เข้าถึง	คำอธิบาย
/	จากการเรียกดูแอปพลิเคชันจาก http://localhost:3000/ และมี end-point คือ / จะได้ผลลัพธ์ที่แสดงออกมาบน browser ได้เป็น landing page ดังที่แสดงในรูป 4.1
/register	จากการเรียกดูแอปพลิเคชันจาก http://localhost:3000/ และมี end-point คือ /register จะได้ผลลัพธ์ที่แสดงออกมาบน browser ได้เป็นหน้าจอการลงทะเบียนบัญชีผู้ใช้ของแอปพลิเคชันดังที่แสดงในรูป 4.2
/login	จากการเรียกดูแอปพลิเคชันจาก http://localhost:3000/ และมี end-point คือ /register จะได้ผลลัพธ์ที่แสดงออกมาบน browser ได้เป็นหน้าจอการลงชื่อเข้าใช้งานแอปพลิเคชันดังที่แสดงในรูป 4.2
/tools/{diagramName}	จากการเรียกดูแอปพลิเคชันจาก http://localhost:3000/ และมี end-point คือ /register จะได้ผลลัพธ์ที่แสดงออกมาบน browser ได้เป็นหน้าจอของแอปพลิเคชันโดยประเภทของ diagram ที่จะสร้างขึ้นอยู่กับ diagramName

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.1 การแสดงผลของ landing page



รูปที่ 4.2 หน้าจอการลงทะเบียนบัญชีผู้ใช้ของแอปพลิเคชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.3 หน้าจอการลงชื่อเข้าใช้งานแอปพลิเคชัน

รูปที่ 4.4 หน้าจอแอปพลิเคชัน

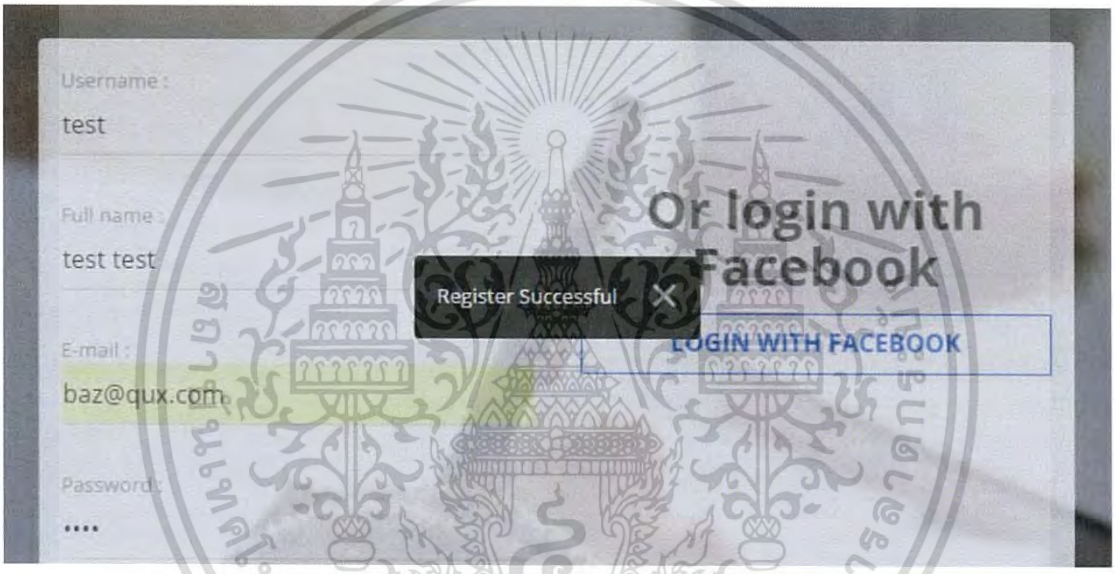
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.2 การแสดงผลของแอปพลิเคชัน

ในแอปพลิเคชันได้มีการสร้างการเคลื่อนไหวขององค์ประกอบต่างๆ ของแอปพลิเคชันขึ้น ซึ่งองค์ประกอบของแอปพลิเคชันที่เคลื่อนไหวประกอบไปด้วย dialog box ต่างๆ เช่น login, save diagram, import diagram, export diagram, logout เป็นต้น ซึ่งสามารถแสดงผลของภาพเคลื่อนไหวได้อย่างถูกต้อง

4.1.3 การลงทะเบียนบัญชีผู้ใช้ของแอปพลิเคชัน

การลงทะเบียนบัญชีผู้ใช้ของแอปพลิเคชันนั้นจำเป็นที่จะต้องใส่ข้อมูลต่างๆให้ถูกต้อง และข้อมูลที่ใช้ต้องเป็นข้อมูลที่ยังไม่เคยทำการลงทะเบียนบัญชีผู้ใช้งานของแอปพลิเคชันมาก่อน มิเช่นนั้นจะไม่สามารถลงทะเบียนได้

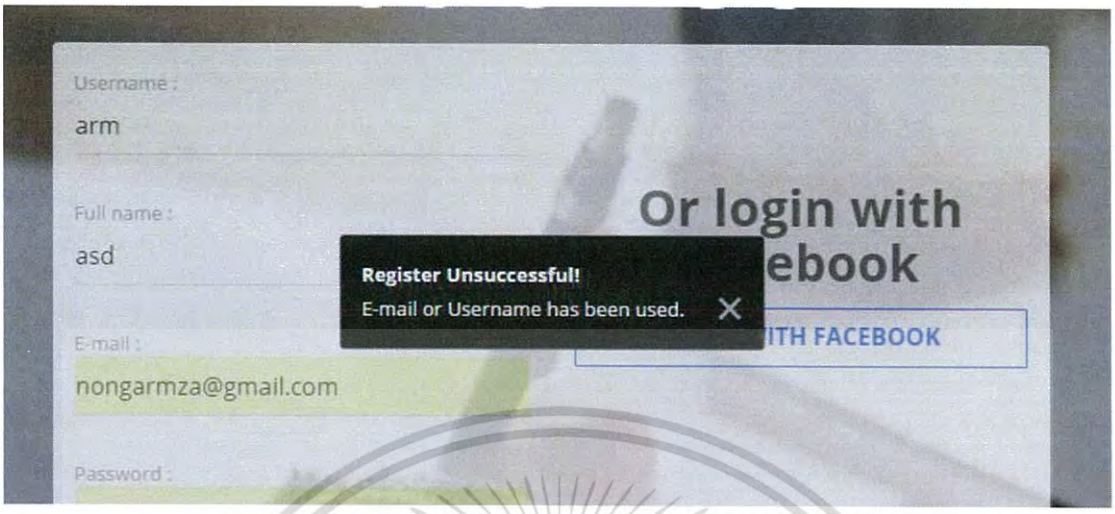


รูปที่ 4.5 ผลลัพธ์การลงทะเบียนผู้ใช้งานสำเร็จ

เมื่อทำการลงทะเบียนสำเร็จ จะแสดง dialog box ที่มีข้อความว่า Register Successful ดังที่แสดงในรูปที่ 4.5 และจะเชื่อมโยงไปยังหน้าจอการลงทะเบียนเข้าใช้งานแอปพลิเคชันทันที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในกรณีที่ไม่สามารถลงทะเบียนได้ จะแสดง dialog box ที่แสดงดังรูปที่ 4.6



รูปที่ 4.6 ผลลัพธ์การลงทะเบียนผู้ใช้งานไม่สำเร็จ

4.1.4 การลงชื่อเข้าใช้ด้วยบัญชีผู้ใช้ของแอปพลิเคชัน

การลงชื่อเข้าใช้ด้วยบัญชีผู้ใช้ของแอปพลิเคชันหลังจากที่ได้ทำการลงทะเบียนไปแล้วนั้น สามารถเข้าใช้งานได้ 2 แห่ง นั่นคือลงชื่อเข้าใช้จาก landing page และ ลงชื่อเข้าใช้จากหน้าจอลงชื่อเข้าใช้งานหลังจากการลงทะเบียนผู้ใช้ ในกรณีที่บัญชีผู้ใช้ และรหัสผ่านนั้นถูกต้อง จะเชื่อมโยงไปยังหน้าจอของแอปพลิเคชันทันที และในกรณีที่ไม่สามารถลงชื่อเข้าใช้งานได้นั้น จะแสดง dialog box ขึ้นมาแจ้งเตือน ดังแสดงในรูปที่ 4.9

รูปที่ 4.7 dialog box ที่ใช้ในการลงชื่อเข้าใช้งานในหน้า landing page

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.8 หน้าจอการลงชื่อเข้าใช้งานแอปพลิเคชัน

รูปที่ 4.9 ข้อความของการลงชื่อเข้าใช้งานที่ไม่ถูกต้อง

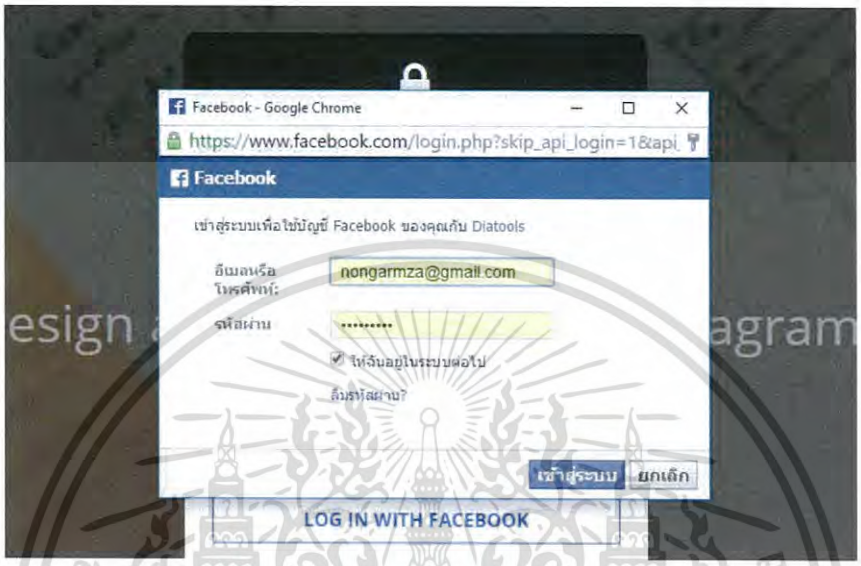
4.1.5 การลงชื่อเข้าใช้ด้วยบัญชีผู้ใช้ของ Facebook

การลงชื่อเข้าใช้งานแอปพลิเคชันด้วยบัญชีผู้ใช้ของ Facebook นั้นสามารถกระทำได้ 2 แห่งเหมือนกันกับการลงชื่อเข้าใช้งาน ด้วยบัญชีผู้ใช้ของแอปพลิเคชันแต่ไม่จำเป็นต้องระบุข้อมูลบัญชีผู้ใช้งาน ให้ทำการคลิกปุ่มลงชื่อเข้าใช้งานด้วย Facebook ได้ทันที

รูปที่ 4.10 ปุ่มการลงชื่อเข้าใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในกรณีที่ยังไม่ได้ทำการลงชื่อเข้าใช้งาน Facebook นั้น เมื่อทำการคลิกลงชื่อเข้าใช้งานด้วยบัญชีผู้ใช้ของ Facebook Application จะทำการแสดง dialog box ให้ทำการระบุชื่อผู้ใช้และรหัสผ่านสำหรับเข้าใช้งาน Facebook ก่อน จากนั้นจึงจะเชื่อมโยงไปยังหน้าจอการขออนุญาติใช้ข้อมูลส่วนตัวของ Facebook ในการใช้งานแอปพลิเคชัน



รูปที่ 4.11 การลงชื่อเข้าใช้งาน Facebook

ส่วนในกรณีที่ได้ทำการลงชื่อเข้าใช้งาน Facebook ไปแล้ว จะทำการแสดง dialog box การขออนุญาติใช้ข้อมูลส่วนตัวของ Facebook และเมื่อทำการอนุญาติให้ใช้งาน จะทำการเชื่อโยงไปยังหน้าจอแอปพลิเคชันทันที โดยกระบวนการดังที่กล่าวมานี้ จะเกิดขึ้นครั้งเดียวในครั้งแรกที่ทำการลงชื่อเข้าใช้ ในครั้งต่อไปแอปพลิเคชันจะจำข้อมูลที่ได้จาก Facebook เมื่อทำการคลิกเข้าใช้งานด้วย Facebook จะเชื่อมโยงไปสู่หน้าจอแอปพลิเคชันทันที

4.1.6 การใช้งานแอปพลิเคชัน

หลังจากที่มีการลงชื่อเข้าใช้งานแอปพลิเคชันแล้วนั้นก็จะเข้าสู่หน้าจอของแอปพลิเคชันซึ่งจะสามารถดำเนินการต่างๆกับ diagram ไม่ว่าจะเป็นการสร้าง, แก้ไข, ลบรูปทรง, ย้อนการกระทำ, เรียกคืนการกระทำ เป็นต้น โดยผลการทดสอบการใช้งานแสดงได้ดังนี้

การสร้าง

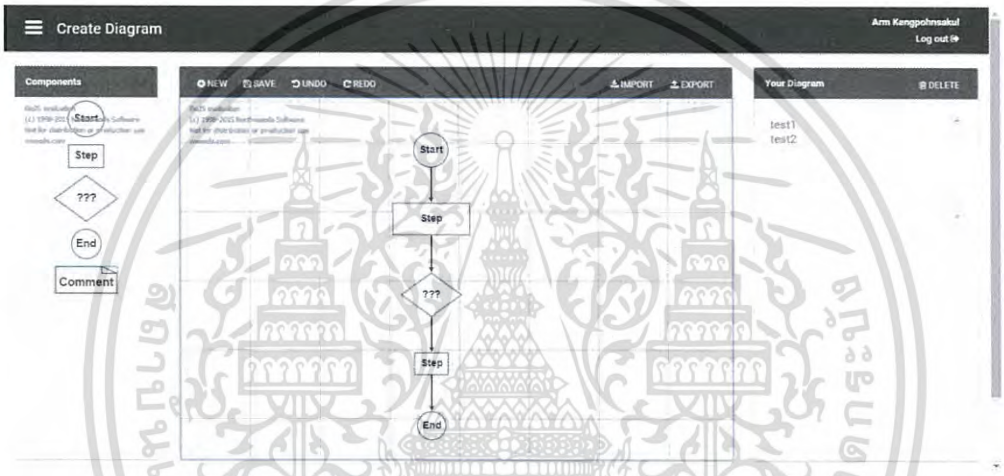
ในการสร้าง diagram จากแอปพลิเคชันนั้นทำได้โดยการลาก รูปทรงต่างๆจาก template ที่ได้ทำการเลือกไว้ โดย template จะอยู่ใน panel ทางด้านซ้ายของจอ ซึ่งผู้ใช้งานสามารถเลือกรูปทรงที่ต้องการ จากนั้นลากรูปทรงนั้นวางไว้ใน panel ตรงกลางของหน้าจอ ในส่วนของรูปทรงที่สามารถทำการเชื่อมต่อกันด้วยเส้นเชื่อมโยง ก็สามารถที่จะลากเส้นเชื่อมโยงจากรูปทรงหนึ่ง ไปยังอีกรูปทรงหนึ่ง รวมถึงการเพิ่มคำอธิบายให้กับรูปทรงก็สามารถทำได้เช่นกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแก้ไข

หลังจากที่ทำการสร้างรูปทรงตามที่ต้องการแล้ว ก็ยังสามารถที่จะแก้ไข diagram ต่างๆได้ โดยวิธีแก้ไขข้อมูลนั้นสามารถกระทำได้หลายวิธี ประกอบไปด้วย

- การลด/เพิ่ม ขนาดของรูปทรง และเส้นเชื่อมระหว่างรูปทรง
- การลบรูปทรง และเส้นเชื่อมระหว่างรูปทรง
- การย้อนการกระทำ (undo) สามารถใช้คีย์ลัด Ctrl+Z ได้
- การเรียกคืนการกระทำ (redo) สามารถใช้คีย์ลัด Ctrl+Y ได้
- การล้างข้อมูลให้กลับเป็นเริ่มต้น
- การตัด/คัดลอก-วาง รูปทรงและเส้นเชื่อมระหว่างรูปทรง

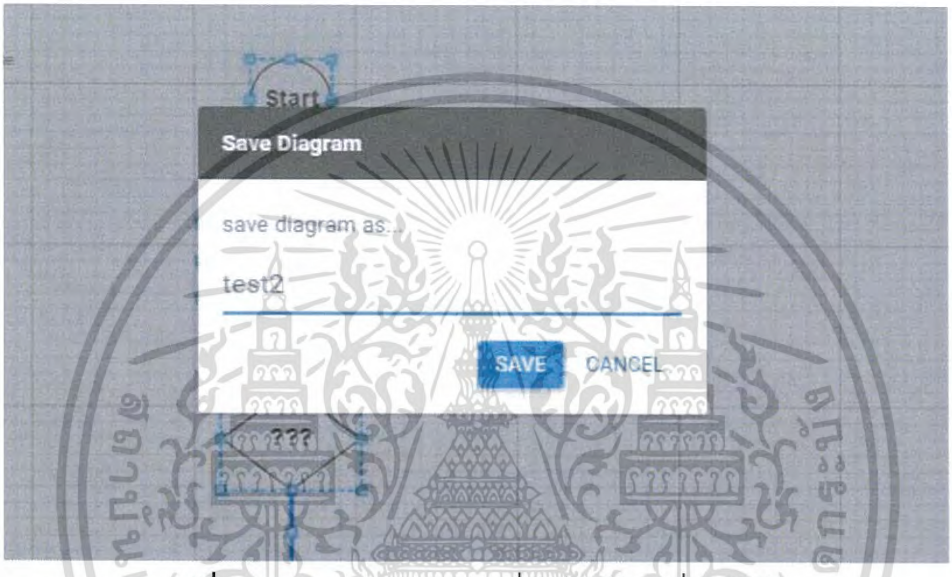


รูปที่ 4.12 หน้าจอของแอปพลิเคชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การบันทึกข้อมูลลงฐานข้อมูล

หลังจากทำดำเนินการสร้าง และแก้ไขข้อมูลเสร็จเรียบร้อยแล้วนั้น ก็สามารถที่จะบันทึกข้อมูลที่สร้างไว้ลงไปในฐานข้อมูลได้ โดยวิธีการบันทึกข้อมูลจะมีอยู่ 2 แบบคือ การบันทึกข้อมูลใหม่ไปยังฐานข้อมูลกับการบันทึกข้อมูลที่เคยบันทึกไว้ก่อนหน้านั้นอยู่แล้ว โดยที่การบันทึกข้อมูลใหม่จะแสดง dialog box สำหรับการตั้งชื่อข้อมูลที่จะบันทึก แล้วเมื่อบันทึกเสร็จสิ้น จะแสดง dialog แจ้งว่าบันทึกข้อมูลสำเร็จ ส่วนในกรณีของการบันทึกข้อมูลที่เคยมีอยู่แล้วในฐานข้อมูลจะแสดง dialog แจ้งว่าบันทึกข้อมูลสำเร็จแล้วเท่านั้น



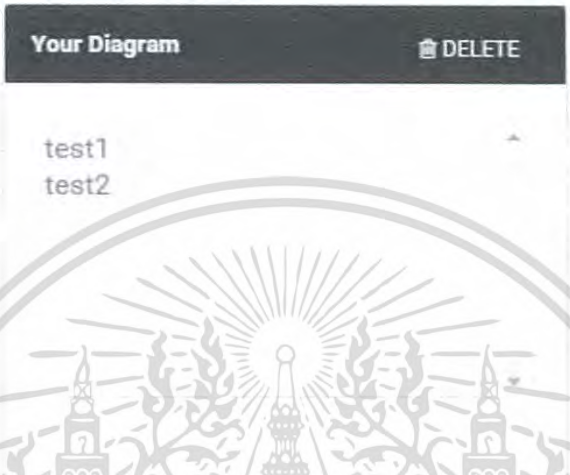
รูปที่ 4.13 dialog การกำหนดชื่อให้กับข้อมูลที่จะบันทึก

รูปที่ 4.14 dialog แสดงข้อความการบันทึกสำเร็จ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเรียกดูข้อมูลจากฐานข้อมูล

หลังจากที่ทำการบันทึกข้อมูลลงฐานข้อมูลแล้ว สามารถที่จะทำการเรียกดูข้อมูลที่บันทึกได้ โดยการ double click ที่ชื่อของข้อมูลที่บันทึกจาก list ของข้อมูลที่แสดงอยู่บน panel ทางด้านขวาของแอปพลิเคชันจากนั้นระบบจะทำการแสดงข้อมูลที่เรียกมาไว้ใน panel กลางของแอปพลิเคชัน



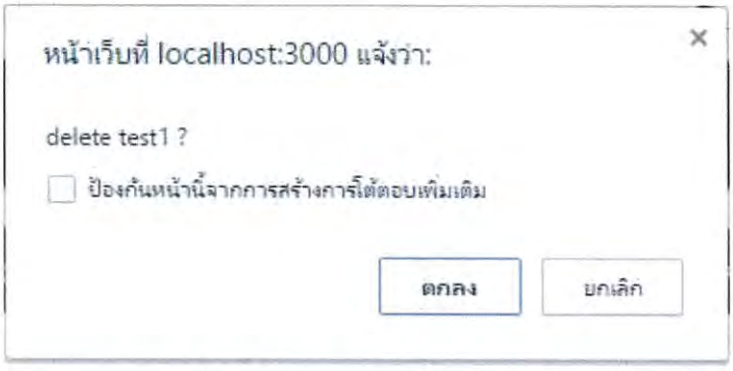
รูปที่ 4.15 รายการของข้อมูลที่บันทึกลงฐานข้อมูล

การลบข้อมูลออกจากฐานข้อมูล

ถ้าหากผู้ใช้งานไม่ต้องการข้อมูลที่บันทึกไว้ในฐานข้อมูลแล้ว สามารถที่จะลบข้อมูลดังกล่าวทิ้งไปได้ โดยทำการเลือกข้อมูลจากรายการข้อมูลที่แสดงอยู่ใน panel ด้านขวาโดยการคลิกที่ชื่อของข้อมูลดังกล่าว 1 ที จากนั้นทำการคลิกที่ปุ่มลบข้อมูล ซึ่งจะแสดง dialog box เพื่อยืนยันการลบข้อมูล เมื่อทำการยืนยันการลบข้อมูลแล้วนั้น ระบบจะทำการไปลบข้อมูลออกจากฐานข้อมูลและจะทำการแสดง dialog box แสดงข้อความการลบข้อมูลเสร็จสิ้น

รูปที่ 4.16 รายการของข้อมูลและแถบที่แสดงปุ่ม delete ข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.17 dialog ยืนยันการลบข้อมูล

รูปที่ 4.18 dialog แสดงข้อความการลบข้อมูลสำเร็จ

การ export ข้อมูลออกเป็น JSON file

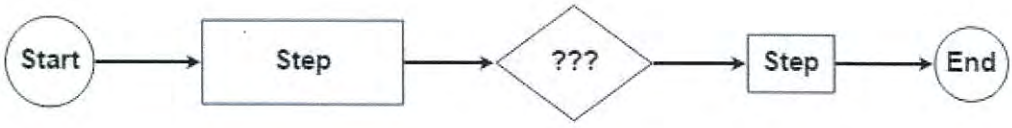
Application สามารถที่จะ export ข้อมูลที่สร้างไว้ออกเป็น JSON file ซึ่ง JSON file ดังกล่าวจะทำการเก็บรายละเอียดของข้อมูลที่ส่งออกไป การ export จะมีวิธีการเช่นเดียวกับการบันทึกข้อมูล แต่การ export ข้อมูลจะทำการแสดง dialog ที่ให้กำหนดชื่อก่อนทุกครั้ง

การ export ข้อมูลออกเป็น image file (PNG)

การ export ข้อมูลออกเป็น image file มีวิธีการเช่นเดียวกันกับการ export ข้อมูลออกเป็น JSON file แต่ผลลัพธ์ที่ได้ออกมานั้นจะอยู่ในรูปแบบ PNG

รูปที่ 4.19 การ export ข้อมูลในรูปแบบต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.20 ผลลัพธ์ของการ export ในรูปแบบ JSON

```
{ "class": "go.GraphLinksModel",
  "linkFromPortIdProperty": "fromPort",
  "linkToPortIdProperty": "toPort",
  "nodeDataArray": [
{"category":"Start", "text":"Start", "key":-1, "loc":"10 -120"},
{"category":"Comment", "text":"Comment", "key":-5, "loc":"10 30"}
 ],
  "linkDataArray": []}
```

รูปที่ 4.21 ผลลัพธ์ของการ export ในรูปแบบ image

การ import ข้อมูลจาก JSON file

ไฟล์ JSON ที่ทำการ export ออกไป นั้นสามารถ import เข้ามาในแอปพลิเคชันได้ โดยไฟล์ที่มีรายละเอียดของข้อมูลที่ถูกต้อง จะแสดงข้อมูลดังกล่าวนั้นในลักษณะของรูปทรงและเส้นเชื่อมระหว่างรูปทรงใน panel กลางของแอปพลิเคชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
สรุปผลสหกิจศึกษาและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

ภาพรวมของแอปพลิเคชัน

หลังจากที่ได้มีการพัฒนาแอปพลิเคชันที่สามารถจัดการเกี่ยวกับแผนภาพ flowchart ซึ่งผลของการพัฒนานั้นตรงตามข้อกำหนดที่ได้วางเอาไว้ เช่น สามารถสร้างแผนภาพ, แก้ไขแผนภาพ, ลบแผนภาพ, นำเข้าแผนภาพ, และส่งออกแผนภาพ ซึ่งเป็นการดำเนินการผ่านทางหน้าจอการใช้งาน และทำการส่งข้อมูลไปยังเซอร์เวอร์ เพื่อที่จะดำเนินการกับฐานข้อมูลอีกทีหนึ่ง

ประสิทธิภาพของภาษา JavaScript

ภาษา JavaScript เมื่อถูกนำมาใช้ในการพัฒนาอยู่บนแพลตฟอร์มของ Node.js นั้น ทำให้สามารถสร้างแอปพลิเคชันที่ทำงานอยู่บนฝั่งของเซิร์ฟเวอร์ได้ โดยที่ Node.js นั้นมี module ต่างๆ ที่ช่วยอำนวยความสะดวกในการพัฒนาแอปพลิเคชันได้อย่างรวดเร็ว และประสิทธิภาพของแอปพลิเคชันที่พัฒนานั้นก็สามารถที่จะทำงานได้ดี เนื่องด้วยการทำงานของ Node.js จะเป็นการทำงานแบบ Event-Driven ทำให้สามารถรองรับการใช้งานที่เพิ่มขึ้นได้ดีกว่าภาษาสคริปต์ทางฝั่งเซิร์ฟเวอร์ต่างๆ ที่มีอยู่ในปัจจุบัน

การนำไปใช้ในการพัฒนาแอปพลิเคชันขนาดใหญ่

ในการพัฒนาแอปพลิเคชันขนาดใหญ่ (Enterprise Application) ที่นำมาใช้ในองค์การต่างๆ นั้น สามารถนำภาษา JavaScript มาพัฒนาได้ โดยการพัฒนาบนแพลตฟอร์ม Node.js ซึ่งมี module ต่างๆ ที่สามารถทำการพัฒนาแอปพลิเคชันขนาดใหญ่ได้ ซึ่งก็ได้มีบางบริษัทนำ module ต่างๆ เหล่านั้น มารวมกันเป็นเฟรมเวิร์ก ตัวอย่างเช่น StrongLoop, Sail เป็นต้น ซึ่งทำให้การพัฒนาแอปพลิเคชันขนาดใหญ่ด้วยภาษา JavaScript นั้นสามารถทำได้ง่ายขึ้นอีกด้วย

5.2 ข้อเสนอแนะ

เนื่องจากแอปพลิเคชันนี้ ถูกพัฒนาขึ้นบนแพลตฟอร์ม Node.js ซึ่งเป็นแพลตฟอร์มที่เพิ่งกำเนิดขึ้นมาได้ไม่นาน และยังไม่มีมาตรฐานในการพัฒนาให้เป็นไปในรูปแบบเดียวกัน ส่งผลให้ในภายภาคหน้า เมื่อเฟรมเวิร์ก หรือไลบรารีต่างๆ ที่ใช้ในการพัฒนาแอปพลิเคชันนี้ ได้มีการปรับปรุง หรือเปลี่ยนแปลงวิธีการใช้งาน อาจทำให้มีการแก้ไขตัวแอปพลิเคชัน เพื่อรองรับการทำงานของเฟรมเวิร์ก หรือไลบรารีที่ใช้งานทีหลังอีกด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารอ้างอิง

[1] **“JSON,”** [ออนไลน์].สืบค้นเมื่อ: 20 กันยายน 2558. เข้าถึงได้จาก: http://www.json.org/.

[2] **"Understanding REST,"** [ออนไลน์]. สืบค้นเมื่อ: 18 กรกฎาคม 2558. เข้าถึงได้จาก: https://spring.io/understanding/REST.

[3] B. Dayley, **"Introducing the Node.js-to-AngularJS Stack,"** in *Node.js, MongoDB and AngularJS Web Development*, Michigan, Addison-Wesley Professional, 2014, p. 696.

[4] B. Dayley, **"Learning Node.js,"** in *Node.js, MongoDB and AngularJS Web Development*, Michigan, Addison-Wesley Professional, 2014, p. 696.

[5] **"StrongLoop Documentation,"** IBM, 2013. [ออนไลน์]. สืบค้นเมื่อ: 20 กรกฎาคม 2558. เข้าถึงได้จาก: https://docs.strongloop.com.

[6] B. Dayley, **"Learning MongoDB,"** in *Node.js, MongoDB and AngularJS Web Development*, Michigan, Addison-Wesley Professional, 2014, p. 696.

[7] B. Dayley, **"Learning AngularJS,"** in *Node.js, MongoDB and AngularJS Web Development*, Michigan, Addison-Wesley Professional, 2014, p. 696.

[8] **"GoJS Diagrams for JavaScript and HTML,"** Northwoods Software, [ออนไลน์]. สืบค้นเมื่อ: 2 สิงหาคม 2558. เข้าถึงได้จาก: http://gojs.net/.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก

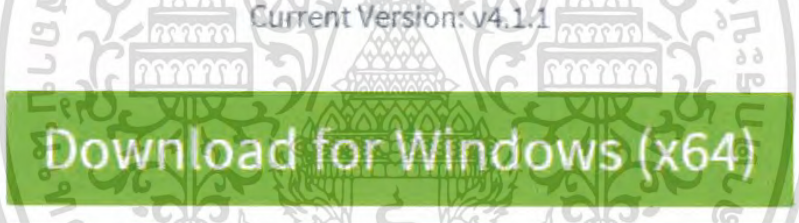
การติดตั้งสภาพแวดล้อมสำหรับการพัฒนาแอปพลิเคชัน

การติดตั้งสภาพแวดล้อมสำหรับการพัฒนาแอปพลิเคชันด้วยภาษา JavaScript ประกอบไปด้วย

- การติดตั้ง Node.js
- การติดตั้ง MongoDB และ Robomongo
- การติดตั้ง Atom Text Editor

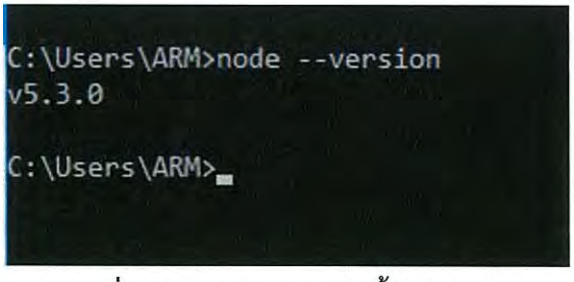
ขั้นตอนการติดตั้ง และตรวจสอบเวอร์ชันของ Node.js

ทำการดาวน์โหลดตัวติดตั้ง Node.js จาก http://www.nodejs.org และทำการติดตั้งเหมือนโปรแกรมทั่วไป



รูปที่ ก.1 ดาวน์โหลด Node.js

ตรวจสอบการติดตั้งโดยเปิด command line ขึ้นมา แล้วพิม node –version



รูปที่ ก.2 ตรวจสอบการติดตั้ง Node.js

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การติดตั้ง Node.js จะติดตั้ง Node Pakage Manager ซึ่งเป็นตัวจัดการส่วนเสริมต่างๆ ของ Node.js มาด้วย สามารถตรวจสอบได้โดยพิมพ์ npm –version

รูปที่ ก.3 ตรวจสอบการติดตั้งของ Node Package Manager

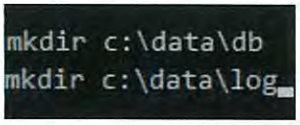
การติดตั้ง MongoDB และ Robomongo

การติดตั้ง MongoDB

ดาวน์โหลด MongoDB installer จาก URL https://www.mongodb.org/downloads

รูปที่ ก.4 ดาวน์โหลด MongoDB

ในส่วนของการติดตั้ง MongoDB จะมีการติดตั้งคล้ายกับโปรแกรมทั่วไป ซึ่งหลังจากที่ทำการติดตั้งเสร็จเรียบร้อยแล้ว ให้ทำการกำหนด path สำหรับเก็บข้อมูลที่ทำการบันทึก โดยทำการสร้างไดเรกทอรีที่ทำการเก็บข้อมูลของฐานข้อมูลและ log ของฐานข้อมูลที่ drive C โดยพิมพ์คำสั่ง mkdir ตามด้วยไดเรกทอรีที่ต้องการสร้าง



รูปที่ ก.5 การสร้างไดเรกทอรีสำหรับเก็บข้อมูลฐานข้อมูลและ log

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สร้างไฟล์ mongod.cfg ที่ไดเรกทอรี “C:\Program Files\MongoDB” โดยมีรายละเอียดของไฟล์ดังนี้

```
1  ##store data here
2  dbpath=C:\data\db
3
4  ##all output go here
5  logpath=C:\data\log\mongo.log
6
```

รูปที่ ก.6 รายละเอียดของไฟล์ config MongoDB

บรรทัดที่ 2: กำหนดเส้นทางที่ทำการเก็บข้อมูลของฐานข้อมูล
บรรทัดที่ 5: กำหนด log path เก็บข้อมูลการดำเนินการต่างๆ

ไปที่ไดเรกทอรี “C:\Program Files\MongoDB\Server\3.2\bin” จากนั้นพิมพ์คำสั่ง “mongod --config "C:\Program Files\MongoDB\mongod.cfg" --install” เพื่อสร้างเซอร์วิสของ MongoDB

```
mongod --config "C:\Program Files\MongoDB\mongod.cfg" --install
```

รูปที่ ก.7 คำสั่งสร้างเซอร์วิสของ MongoDB

ติดตั้ง Robomongo

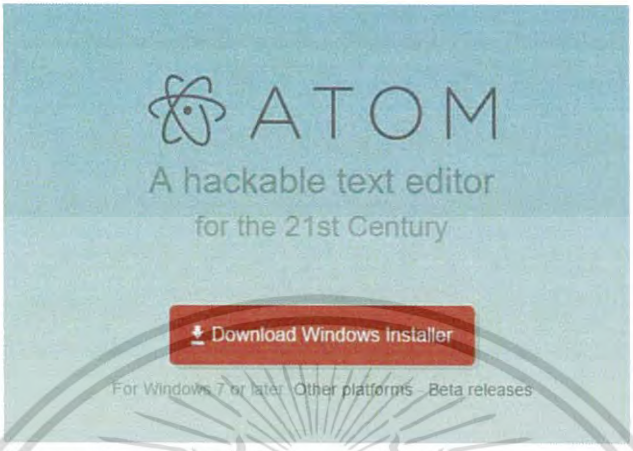
ทำการดาวน์โหลดตัวติดตั้งจาก URL http://app.robomongo.org/download.html

รูปที่ ก.8 หน้าจอแสดงการดาวน์โหลด Robomongo

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

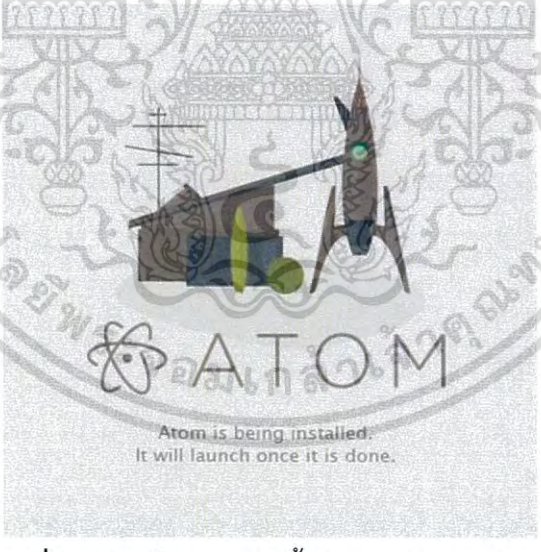
การติดตั้ง Atom Text Editor

ทำการดาวน์โหลดตัวติดตั้งจาก URL https://atom.io/



รูปที่ ก.9 หน้าจอแสดงพื้นที่การดาวน์โหลด Atom Text Editor

ติดตั้ง Atom Text Editor



รูปที่ ก.10 หน้าจอการติดตั้ง Atom Text Editor

โปรแกรมจะถูกเปิดขึ้นมาอัตโนมติ เมื่อการติดตั้งเสร็จสิ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข

การติดตั้ง framework และ library ต่างๆ

การติดตั้ง framework และ library ต่างๆ

Framework และ library ต่างๆ จะถูกติดตั้งผ่านทาง Node Package Manager ซึ่งเป็นตัวจัดการที่ถูกติดตั้งมาพร้อมกับ Node.js โดย Framework ที่ทำการติดตั้ง ประกอบไปด้วย

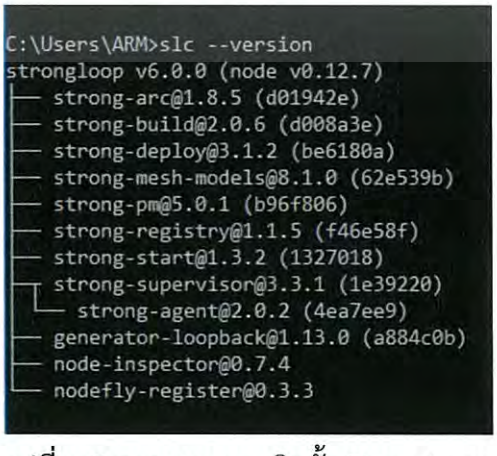
- StrongLoop framework สำหรับจัดการ API
- Bower สำหรับจัดการ front-end library
- Library เสริมของ AngularJS

StrongLoop framework สำหรับจัดการ API

ติดตั้ง StrongLoop framework ผ่าน CLI โดยพิมพ์คำสั่ง npm install –g strongloop

รูปที่ ข.1 ติดตั้ง StrongLoop

ทำการตรวจสอบการติดตั้ง โดยพิม slc –version ใน CLI จะต้องแสดงเวอร์ชันของ StrongLoop ที่ติดตั้งไว้

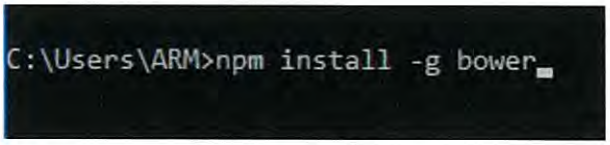


รูปที่ ข.2 ตรวจสอบการติดตั้ง StrongLoop

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

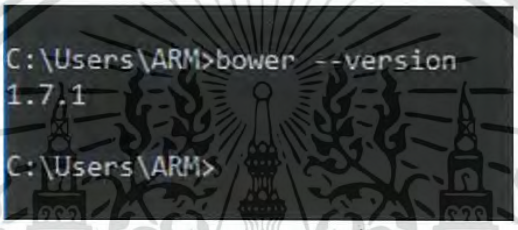
Bower สำหรับจัดการ front-end library

ติดตั้ง Bower ผ่านทาง CLI โดยพิมพ์คำสั่ง npm install –g bower



รูปที่ ข.3 การติดตั้ง Bower

ทำการตรวจสอบการติดตั้งโดยพิมพ์คำสั่ง bower –version ใน CLI จะต้องแสดงเวอร์ชันของ Bower ที่ได้ทำการติดตั้งไว้



รูปที่ ข.4 ตรวจสอบการติดตั้ง Bower

Library เสริมของ AngularJS

Library เสริมของ AgularJS ที่ใช้ในการพัฒนาแอปพลิเคชันจะถูกติดตั้งผ่านทาง Bower ซึ่งเป็นตัวจัดการ library มีรายละเอียด แสดงไว้ในตารางที่ ข.1

ตารางที่ ข.1 Library เสริมของ AngularJS ที่ใช้ในการพัฒนา

ชื่อ library	การติดตั้ง	คำอธิบาย
Angular Facebook	bower install angular-facebook	ใช้ในการเชื่อมต่อแอปพลิเคชันเข้ากับ Facebook
Angular File Upload	bower install angular-file-upload	ทำให้สามารถอัพโหลดไฟล์เข้าสู่แอปพลิเคชันได้
Angular UI Router	bower install angular-ui-router	เป็นตัวช่วยในการทำ route สำหรับ Single Page Application
Angular UTF8 Base64	bower install angular-utf8-base64	ใช้ในการเข้ารหัสข้อมูลที่สำคัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ค

การใช้งาน framework และ library ต่างๆ ในการพัฒนาแอปพลิเคชัน

การใช้งาน framework และ library ต่างๆ ในการพัฒนาแอปพลิเคชันประกอบไปด้วย

- การสร้างฐานข้อมูลสำหรับใช้เก็บข้อมูลของแอปพลิเคชัน
- การสร้าง Project สำหรับจัดการ API ด้วย StrongLoop
- การสร้าง data source สำหรับเก็บข้อมูล
- การสร้าง API ด้วย StrongLoop

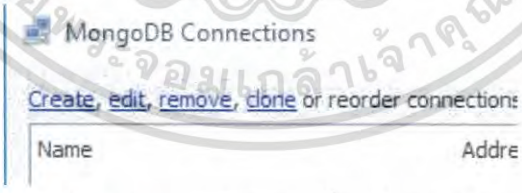
การสร้างฐานข้อมูลสำหรับใช้ในการเก็บข้อมูลของแอปพลิเคชัน

สร้างฐานข้อมูลของ MongoDB ด้วย Robomongo

เปิดโปรแกรม Robomongo ขึ้นมา

รูปที่ ค.1 โปรแกรม Robomongo

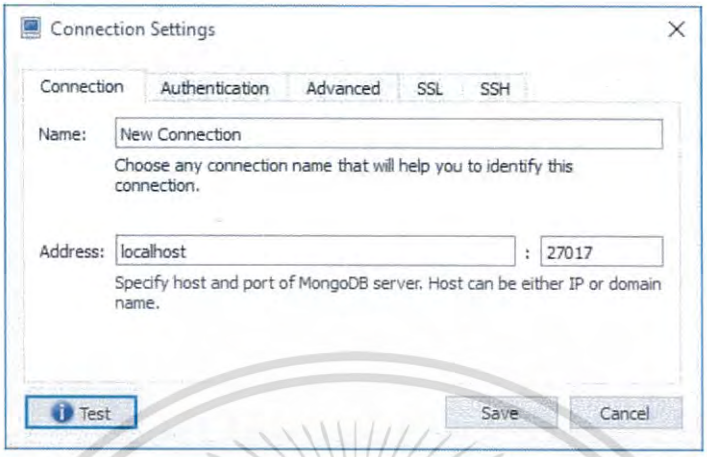
ในหน้าต่างที่แสดง เลือก Create



รูปที่ ค.2 ส่วนของการจัดการการเชื่อมต่อของ MongoDB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กำหนดรายละเอียดการเชื่อมต่อ ตั้งชื่อให้กับ connection ที่จะสร้าง ในช่อง Address กำหนดเป็น localhost และ port กำหนดเป็น 27017 จากนั้นกด Save

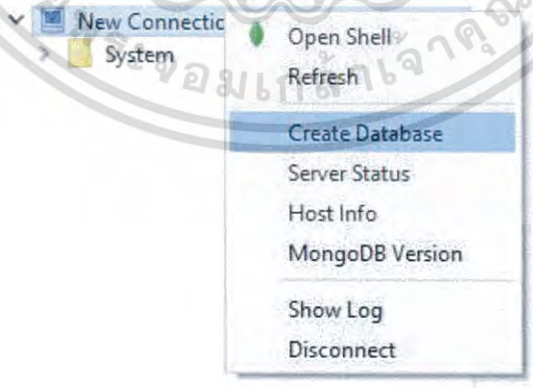


รูปที่ ค.3 กำหนดรายละเอียดของการเชื่อต่อกับ MongoDB

เลือก connection ที่ต้องการจะเชื่อมต่อ จากนั้นกดปุ่ม Connect

รูปที่ ค.4 ปุ่ม Connect เชื่อมต่อกับฐานข้อมูล

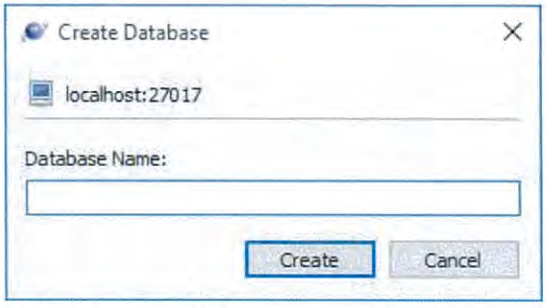
คลิกขวาที่รายชื่อของการเชื่อมต่อ จากนั้นเลือก Create Database



รูปที่ ค.5 การสร้างฐานข้อมูล MongoDB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

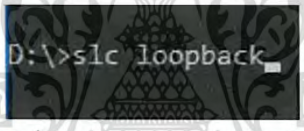
ตั้งชื่อฐานข้อมูล จากนั้นกด Create



รูปที่ ค.6 หน้าจอตั้งชื่อให้กับฐานข้อมูล MongoDB

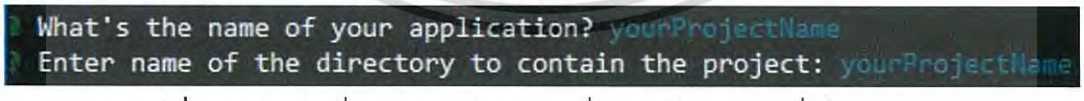
การสร้าง Project สำหรับจัดการ API ด้วย StrongLoop

ในการใช้ StrongLoop ซึ่งเป็น framework สำหรับสร้าง project จะใช้คำสั่งใน CLI (Command Line Interface) ในการสร้าง ซึ่งในโฟลเดอร์ project จะเก็บไฟล์ที่สำคัญต่างๆสำหรับการสร้าง API ไว้ คำสั่งที่ใช้สร้าง project คือ



รูปที่ ค.7 คำสั่งสร้าง Project ของ StrongLoop

จากนั้นรอสักครู่ จะขึ้นข้อความให้ใส่ชื่อ project ใส่ชื่อ project ที่ต้องการ จากนั้นกด Enter จะขึ้นข้อความให้ใส่ชื่อ directory ของ project สามารถตั้งได้ตามใจชอบ หรือถ้าจะตั้งให้เหมือนกันกับชื่อ project สามารถทำได้โดยกดปุ่ม Enter



รูปที่ ค.8 กำหนดชื่อของ Project และ ชื่อของ Directory ที่เก็บ Project

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากนั้นกดปุ่ม Enter แล้วทำการรอจนกว่าจะขึ้นข้อความดังรูป

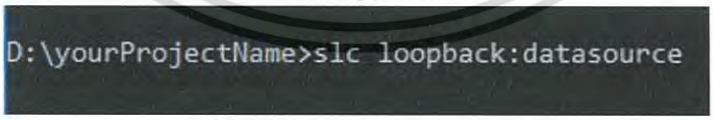
รูปที่ ค.9 หน้าจอแสดงการสร้าง Project เสร็จสิ้น

การสร้าง data source สำหรับเก็บข้อมูล และการติดตั้ง connector สำหรับเชื่อมต่อ model เข้ากับ data source

ในการใช้งาน API บางครั้งอาจมีการดำเนินการกับข้อมูลเพื่อที่จะนำไปบันทึกลงบนฐานข้อมูลหรืออาจจะเรียกใช้งานเซอร์วิสต่างๆจากภายนอก โดย ฐานข้อมูลและเซอร์วิสต่างๆนั้นจะถูกมองว่าเป็น data source ถ้าหากต้องการใช้งานนั้น จะต้องทำการสร้าง data source ขึ้นมาเสียก่อน โดย data source จะเป็นตัวจัดการกำหนดรายละเอียดต่างๆ เพื่อที่จะให้ model สามารถเรียกใช้งานเซอร์วิสหรือฐานข้อมูลเหล่านั้นได้ โดยสามารถสร้าง data source ได้สองทางคือ ทาง CLI และ StrongLoop Arc

การสร้าง data source ผ่าน CLI

ไปยัง directory ของ project จากนั้นทำการพิมพ์คำสั่ง slc loopback:datasource



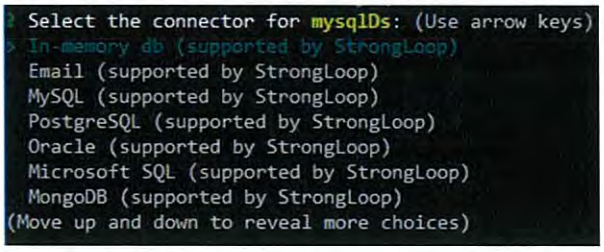
รูปที่ ค.10 การสร้าง data source

กำหนดชื่อของ data source ที่จะสร้าง

รูปที่ ค.11 กำหนดชื่อของ data source

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เลือกประเภท connector ของ datasource



รูปที่ ค.12 รายชื่อประเภทของ datasource ที่สามารถสร้างได้

การสร้าง data source ผ่าน StrongLoop Arc

ทำการสร้าง data source ผ่าน arc โดยพิมพ์คำสั่ง slc arc

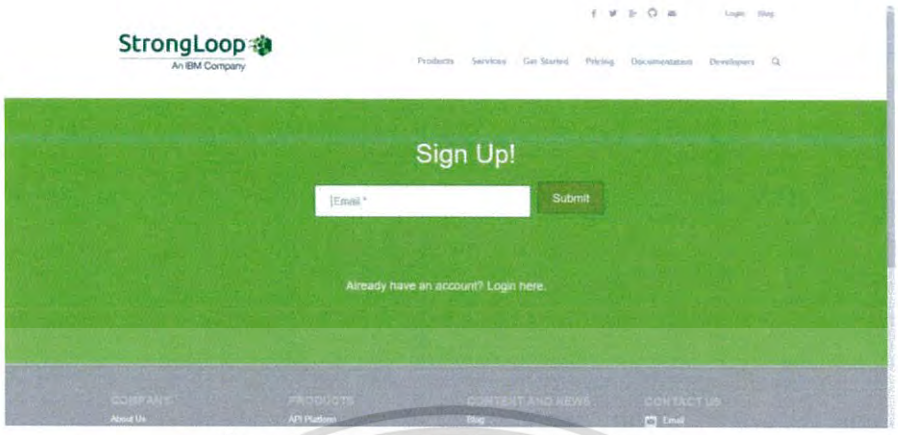
รูปที่ ค.13 การเข้าใช้งาน StrongLoop Arc

Log in เข้าสู่ระบบ arc .ในกรณีที่ยังไม่บัญชีผู้ใช้ ให้ทำการสมัครสมาชิกก่อน โดยคลิก Register

รูปที่ ค.14 หน้าจอการลงชื่อเข้าใช้งาน StrongLoop Arc

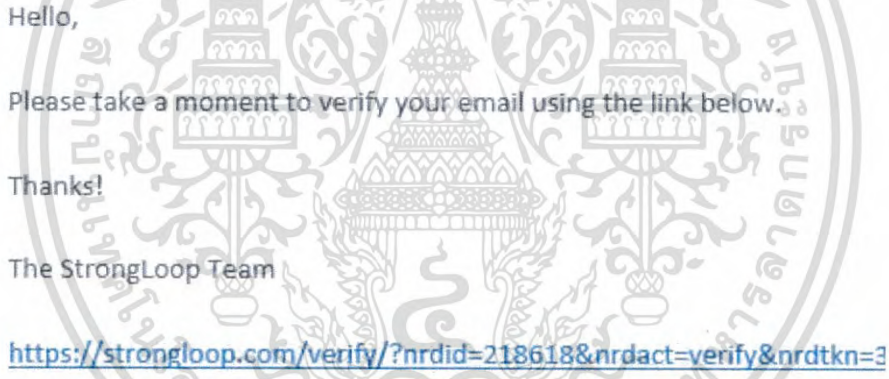
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำการกรอก Email สำหรับลงทะเบียน จากนั้นไปยัง Email ที่ทำการลงทะเบียนไว้



รูปที่ ค.15 หน้าจอการลงทะเบียน StrongLoop

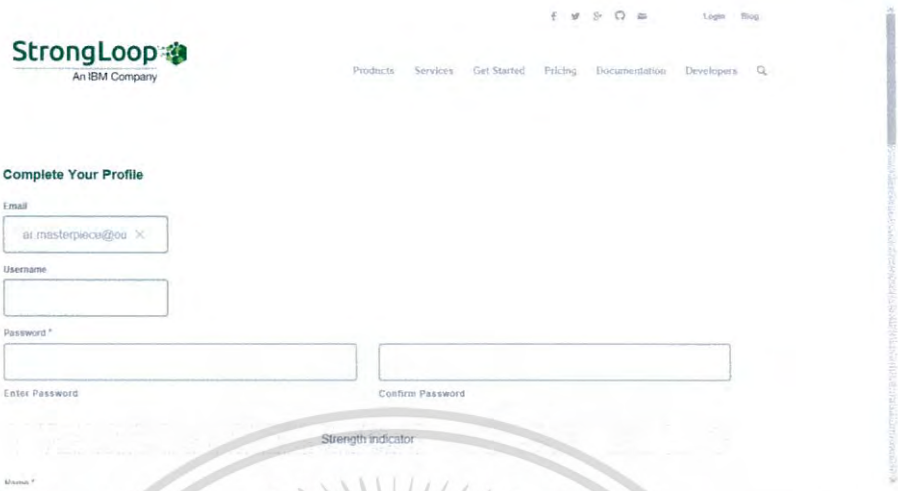
กดลิ้งที่แนบมากับ Email เพื่อทำการกรอกข้อมูลที่ใช้ในการลงทะเบียน



รูปที่ ค.16 email ตอบกลับการลงทะเบียน StrongLoop

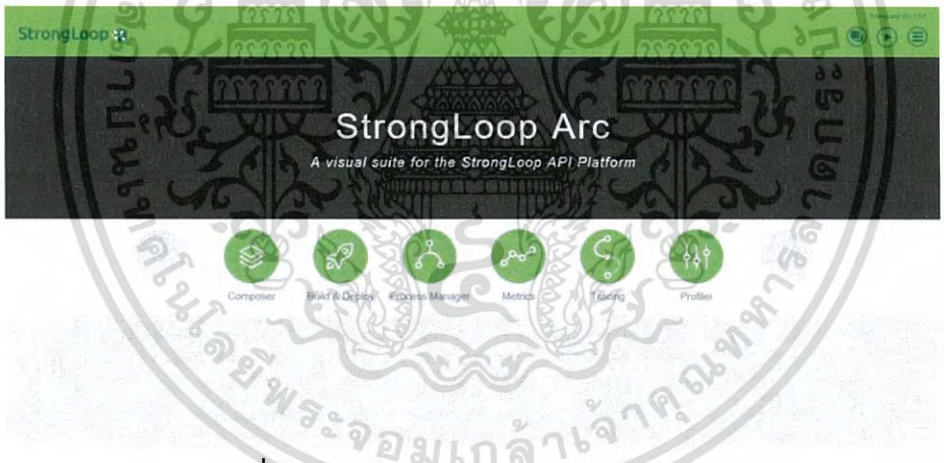
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กรอกข้อมูลที่จำเป็นสำหรับการลงทะเบียน จากนั้นคลิก Register



รูปที่ ค.17 หน้าจอการกรอกข้อมูลที่ใช้ในการลงทะเบียนใช้งาน StrongLoop Arc

เมื่อลงทะเบียนเสร็จสิ้น ให้นำชื่อผู้ใช้งาน และรหัสผ่าน log in เข้าสู่ระบบ



รูปที่ ค.18 หน้า Index ของ StrongLoop Arc

คลิก Composer

รูปที่ ค.19 สัญลักษณ์ Composer

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เลือก data source ที่ต้องการจะสร้าง

รูปที่ ค.20 ประเภทของฐานข้อมูลที่สามารถนำมาสร้าง data source ได้

ใส่รายละเอียดของ data source ที่ต้องการจะสร้าง

รูปที่ ค.21 หน้าจอแสดงส่วนของการกำหนดรายละเอียดของ data source

หลังจากที่ทำการสร้าง data source เสร็จแล้วนั้น การที่ model จะสามารถเชื่อมต่อกับ data source ต่างๆได้จำเป็นที่จะต้องติดตั้งตัว connector เสียก่อนโดยสามารถติดตั้งได้ผ่านทาง CLI เท่านั้น โดยคำสั่งที่ใช้ติดตั้งคือ npm install --save module ที่ใช้ในการติดตั้ง

```
D:\yourProjectName>npm install loopback-connector-mysql
```

รูปที่ ค.22 การติดตั้งตัว Connector

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ ค.1 แสดงรายชื่อของ connector ที่สามารถติดตั้งได้

ตารางที่ ค.1 รายชื่อ Connector ที่สามารถติดตั้งได้

Connector ของฐานข้อมูล	
ชื่อ connector	**Module ที่ใช้ในการติดตั้ง**
MongoDB	loopback-connector-mongodb
MySQL	loopback-connector-mysql
Oracle	loopback-connector-oracle
PostgreSQL	loopback-connector-postgresql
SQL Server	loopback-connector-mssql
Connector ของเซอร์วิสอื่นๆ	
Push connector	loopback-component-push
Remote connector	loopback-connector-remote
REST	loopback-connector-rest
SOAP	loopback-connector-soap
Storage connector	loopback-component-storage

การสร้าง API ด้วย StrongLoop

หลังจากสร้าง project ที่ทำการเก็บข้อมูลต่างๆสำหรับใช้ในการสร้าง API แล้ว สามารถสร้าง API ได้ 2 วิธี คือ ใช้ CLI ในการสร้าง หรือใช้ StrongLoop Arc ในการสร้าง

การสร้าง API จาก model โดยใช้ CLI ในการสร้าง

ทำการสร้าง model โดยพิมพ์คำสั่ง slc loopback:model

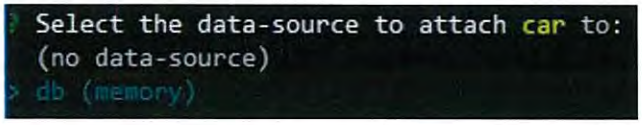
รูปที่ ค.23 การสร้าง model

ใส่ชื่อของ model ที่ต้องการจะสร้าง

รูปที่ ค.24 กำหนดชื่อให้กับ model

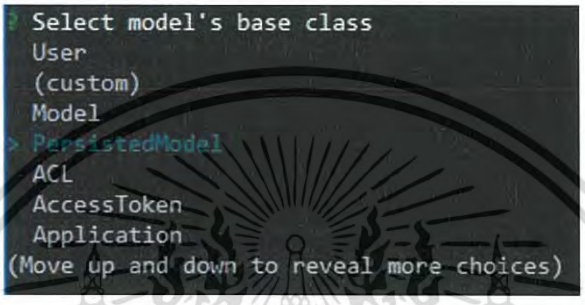
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เลือก datasource ที่ต้องการให้ model เชื่อมต่อ



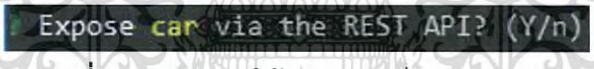
รูปที่ ค.25 รายการของ data source ที่ model สามารถเชื่อมต่อได้

กำหนด base model ให้กับ model ที่จะสร้าง



รูปที่ ค.26 รายการประเภทของ model

เปิดการใช้งาน REST API ให้กับ model

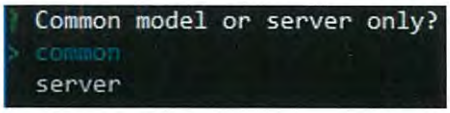


รูปที่ ค.27 กำหนดให้ model เป็น REST API

กำหนดชื่อในรูปแบบ plural ที่ใช้ใน REST API ให้กับ model

รูปที่ ค.28 กำหนดชื่อของ model ในการสร้าง REST URL

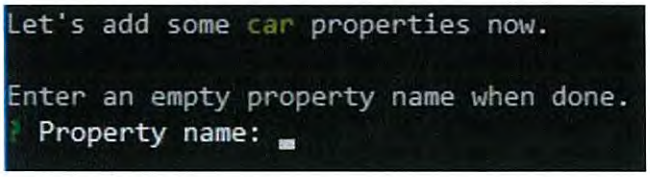
เลือกประเภทของ model ว่าจะให้เป็น model ที่สามารถเรียกใช้งานได้จากที่ไหนบ้าง โดย common จะสามารถใช้งานได้ทั้งไคลเอนต์และเซิร์ฟเวอร์ ส่วนเซิร์ฟเวอร์จะใช้งานได้แค่ฝั่งเซิร์ฟเวอร์อย่างเดียว



รูปที่ ค.29 ประเภทของ model ที่สร้าง

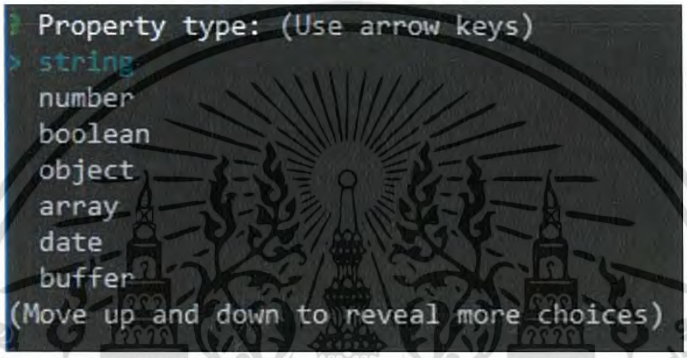
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กำหนด property ให้กับ model



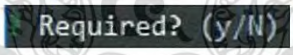
รูปที่ ค.30 การกำหนด property ของ model

เลือกประเภทของข้อมูลให้กับ property



รูปที่ ค.31 ประเภทของ property

ทำการกำหนดว่า property นี้จำเป็นต้องมีทุกครั้งในการดำเนินการ CRUD หรือไม่

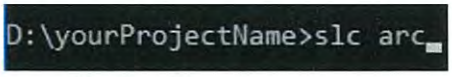


รูปที่ ค.32 กำหนดความต้องการของ property

หากสร้าง property สำหรับ model ได้ครบแล้วนั้น ให้ทำการกดปุ่ม Enter เพื่อทำการออกจากขั้นตอนการสร้าง model

การสร้าง API จาก model โดยใช้ StrongLoop Arc

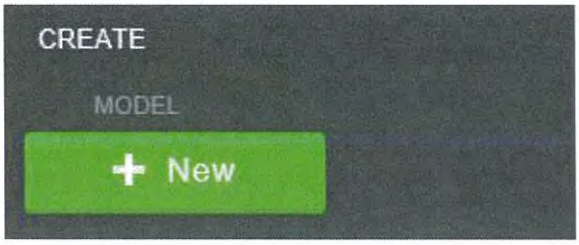
พิมพ์คำสั่ง slc arc ใน CLI เพื่อเข้าสู่ StrongLoop Arc



รูปที่ ค.33 เข้าสู่ StrongLoop Arc

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Log in เข้าสู่ระบบ จากนั้นทำการสร้าง model โดยคลิก New



รูปที่ ค.34 ปุ่ม New สำหรับสร้าง model

ทำการใส่ข้อมูลต่างๆ

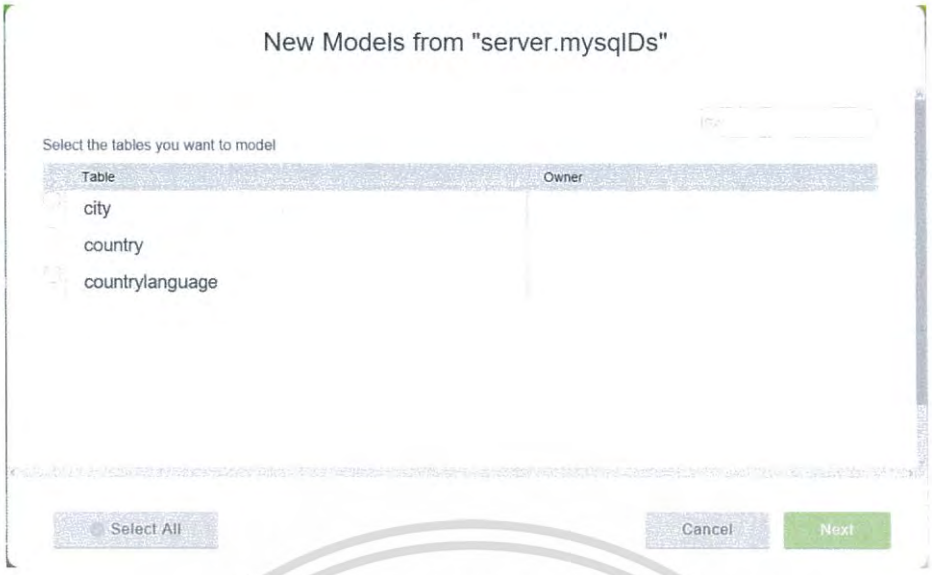
รูปที่ ค.35 ส่วนที่ใช้ในการกำหนดรายละเอียดของ model

สามารถทำการสร้าง model ได้จาก table ของ ฐานข้อมูลแบบ RDBMS ได้โดยคลิก discover Model

รูปที่ ค.36 เมนู discover Model

ทำการเลือก table ที่ต้องการจะนำมาสร้าง model จากนั้นคลิก Next

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ค.37 รายการ table สำหรับสร้าง model

ทำการเลือก field ที่ต้องการ จากนั้นคลิก Next

รูปที่ ค.38 รายการ column ของ ตารางสำหรับกำหนดรายละเอียด model

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

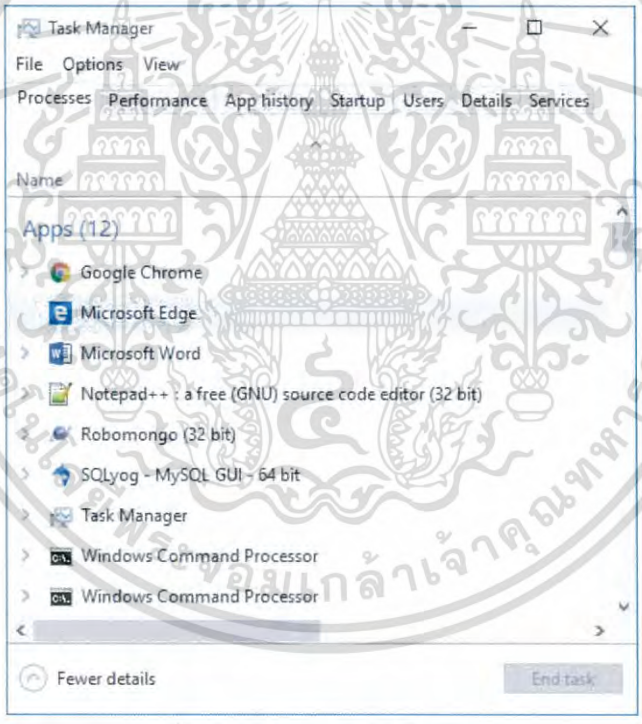
ภาคผนวก ง
การใช้งานแอปพลิเคชัน

การใช้งานแอปพลิเคชันมีรายละเอียดดังนี้

- การเริ่มต้นการทำงานของ database service
- การเริ่มการทำงานของแอปพลิเคชัน
- การเรียกดูรายการของ REST API ที่สามารถใช้งานได้
- การหยุดการทำงานของแอปพลิเคชัน

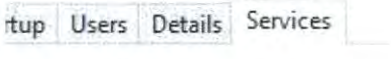
การเริ่มต้นการทำงานของ database service

เปิด task manager ขึ้นมา โดยกดปุ่ม Ctrl+Alt+Delete แล้วเลือก Task Manager



รูปที่ ง.1 หน้าต่าง Task Manager

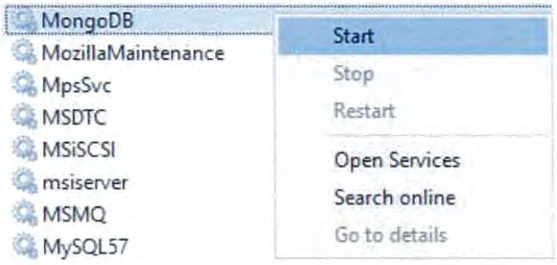
ไปที่ tab เซอร์วิส



รูปที่ ง.2 แถบเซอร์วิส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หารเซอร์วิสของฐานข้อมูลจากนั้นคลิกขวา แล้วเลือก Start



รูปที่ ง.3 การเริ่มต้นการทำงานของ MySQL service และ MongoDB service

การเริ่มการทำงานของแอปพลิเคชัน

ไปยัง directory ของ StrongLoop project ที่สร้างไว้ จากนั้นพิมพ์คำสั่ง node . ผ่าน CLI เพื่อเริ่มการทำงาน

รูปที่ ง.4 เริ่มต้นการทำงานของแอปพลิเคชัน

เมื่อเริ่มการทำงานสำเร็จ จะแสดงข้อความดังรูปที่ ง.5

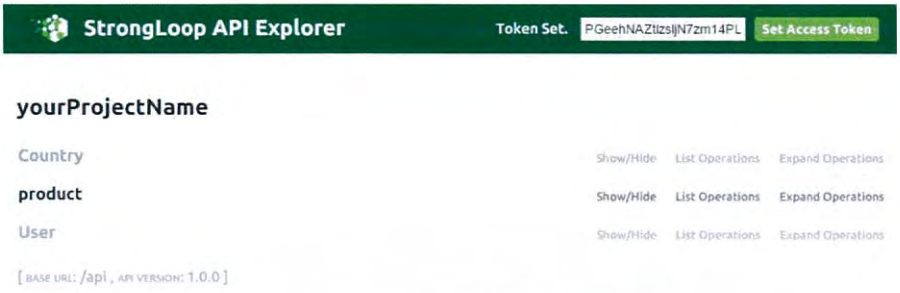
รูปที่ ง.5 การทำงานของแอปพลิเคชัน

หมายเหตุ: ในขณะที่เริ่มต้นการทำงานของแอปพลิเคชันไม่ควรปิดการทำงานของ CLI

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเรียกดูรายการของ REST API ที่สามารถใช้งานได้

เมื่อเริ่มต้นการทำงานแอปพลิเคชันแล้วไปที่ URL http://localhost:3000/explorer



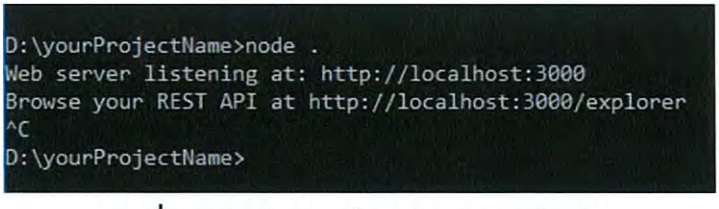
รูปที่ ง.6 รายการของ model ที่สามารถเรียกใช้ REST API ได้

จะสามารถเรียกดู API ที่สามารถใช้งานได้

รูปที่ ง.7 รายการของ API ที่สามารถเรียกใช้งานได้

การหยุดการทำงานของแอปพลิเคชัน

ไปยังหน้าจอของ CLI ที่เริ่มการทำงาน จากนั้นกดปุ่ม Ctrl + C จะทำการยกเลิกการทำงานของแอปพลิเคชัน



รูปที่ ง.8 ยกเลิกการทำงานของแอปพลิเคชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้