รายงานวิจัยสหกิจศึกษา
เรื่องการพัฒนาระบบจัดการกล้องวงจรปิดบนเว็บ
ปฏิบัติงาน ณ บริษัท ทัชเทคโนโลยี จำกัด

นางสาวณัฐชรีณ ไปยะโพธิ์ศรี	รหัสนักศึกษา	6340207201
นายทศพร แซ่อึ้ง	รหัสนักศึกษา	6340208114
นายธนกร ทองคล้าย	รหัสนักศึกษา	6340208115
นายปรวรรธ จำปาพันธุ์	รหัสนักศึกษา	6340208121

รายงานนี้เป็นส่วนหนึ่งของการศึกษารายวิชาสหกิจศึกษา

สาขาเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์และเทคโนโลยี

ภาคการศึกษาที่ 2 ปีการศึกษา 2566

มหาวิทยาลัยราชภัฏนครราชสีมา

รายงานการปฏิบัติงานสหกิจศึกษา

เรื่องการพัฒนาระบบจัดการกล้องวงจรปิดบนเว็บ

นางสาวณัฐชรีณ ไปยะโพธิ์ศรี	**รหัสนักศึกษา**	**6340207201**
นายทศพร แซ่อึ้ง	**รหัสนักศึกษา**	**6340208114**
นายธนกร ทองคล้าย	**รหัสนักศึกษา**	**6340208115**
นายปรวรรธ จำปาพันธุ์	**รหัสนักศึกษา**	**6340208121**

ปฏิบัติงาน ณ บริษัททัชเทคโนโลยี จำกัด

โทรศัพท์ +66(0)2 115 0518 เลขที่ 1127 ถ.สุรนารายณ์ เขตในเมือง

อำเภอเมือง นครราชสีมา รหัสไปรษณีย์ 30000

https://www.touchtechnologies.co.th/

กิตติกรรมประกาศ

ตามที่ข้าพเจ้านางสาวณัฐชรีณ ไปยะโพธิ์ศรี นายทศพร แซ่อึ้ง นายธนกร ทองคล้าย และนายปรวรรธ จำปาพันธุ์ ได้มาปฏิบัติงานสหกิจศึกษา ณ บริษัท ทัชเทคโนโลยี จำกัด ระหว่างวันที่ 4 ธันวาคม พ.ศ.2566 ถึง 29 มีนาคม พ.ศ.2567 ในระหว่างการปฏิบัติงานข้าพเจ้าได้รับความรู้ประสบการณ์ต่างๆ ในการทำงานจริงอันหาค่ามิได้ จากมหาวิทยาลัย ทั้งการทำงานและการจัดทำรายงานฉบับนี้ สำเร็จลงได้ด้วยดี ด้วยความช่วยเหลือ สนับสนุน ให้คำปรึกษาในปัญหาต่างๆ จากบุคลากรหลายผ่าย ดังนี้

1.คุณคเณศวร์ ศรีชุม ตำแหน่ง General manager

2.คุณมยุรี รุนสูงเนิน ตำแหน่ง ที่ปรึกษา System Analysis

3.คุณศรายุทธ ไกรษร ตำแหน่ง ที่ปรึกษา Frontend Developer

4.คุณสุธน เสริฐกระโทก ตำแหน่ง ที่ปรึกษา Development Operations

5.คุณจตุพร เหรียญทอง ตำแหน่ง ที่ปรึกษา Backend Developer

นอกจากนี้ยังมีบุคคลท่านอื่นๆ ที่ไม่ได้กล่าวไว้ ณ ที่นี้ ซึ่งได้อบรมสั่งสอน ให้คำแนะนำที่ดี ในการทำงานและการจัดทำรายงานฉบับนี้ ข้าพเจ้าขอขอบพระคุณทุกท่านเป็นอย่างสูงและหากเนื้อหาฉบับนี้มีข้อผิดพลาดประการใด ข้าพเจ้ากราบขออภัย มา ณ โอกาสนี้

คณะผู้จัดทำ

12 กุมภาพันธ์ 2567

ชื่อรายงาน การพัฒนาระบบจัดการกล้องวงจรปิดบนเว็บ

คณะผู้จัดทำ นายทศพร แซ่อึ้ง รหัสนักศึกษา 6340208114
นายธนกร ทองคล้าย รหัสนักศึกษา 6340208115
นายปรวรรธ จำปาพันธุ์ รหัสนักศึกษา 6340208121
นางสาวณัฐชรีณ ไปยะโพธิ์ศรี รหัสนักศึกษา 6340207201

อาจารย์ที่ปรึกษา ผศ.ดร.ทิพยา ถินสูงเนิน

ปีการศึกษา 2566

บทคัดย่อ

โครงการวิจัยนี้เริ่มต้นขึ้นด้วยวัตถุประสงค์ที่ชัดเจนในการพัฒนาเว็บไซต์ดูกล้องวงจรปิดและระบบจัดการกล้องวงจรปิดอย่างมีประสิทธิภาพ โดยใช้เครื่องมือที่ทันสมัยและมีประสิทธิภาพเช่น Frigate เพื่อให้การแสดงภาพจากกล้องวงจรปิดเป็นไปอย่างราบรื่นบนเว็บไซต์ นอกจากนี้ยังมีการพัฒนาระบบจัดการกล้องวงจรปิดที่ให้ความสะดวกสบายในการเพิ่ม ลบ ดู และแก้ไขกล้องต่างๆ อีกทั้งยังมีการให้บริการเว็บไซต์ที่สามารถรองรับการใช้งานใน 2 ภาษา ไทยและอังกฤษ รวมถึงการสร้างระบบสมัครสมาชิกที่มีการยืนยันตัวตนและการส่งรหัส OTP เพื่อความปลอดภัยเมื่อมีการลืมรหัสผ่าน โครงการนี้มุ่งหวังที่จะเสริมสร้างทั้งความสะดวกสบายและความปลอดภัยให้กับผู้ใช้งานทั้งหลายในการใช้งานระบบกล้องวงจรปิดออนไลน์นี้ ด้วยความพยายามและการพัฒนาที่ต่อเนื่อง หวังว่าโครงการนี้จะมีผลการดำเนินงานที่ประสบความสำเร็จและเป็นประโยชน์ต่อผู้ใช้งานในระยะยาว

การปฏิบัติงานในครั้งนี้ คณะได้รับมอบหมายให้ปฏิบัติงานในตำแหน่ง Frontend , Backend , System Analysis และ Development Operations ในโครงการการจัดการกล้องวงจรปิดบนเว็บ โดยคณะผู้จัดทำจะทำงานร่วมมือกันในแต่ละตำแหน่ง ซึ่งจากการปฏิบัติสหกิจศึกษาในครั้งนี้ได้สร้างประโยชน์ทั้งทางด้านวิชาการในแง่การเพิ่มพูนความรู้ในสาขาวิชาเทคโนโลยีสารสนเทศ สาขาวิทยาการคอมพิวเตอร์ และในสาขาวิชาอื่นๆ อีกทั้งยังเป็นการนำความรู้ที่ได้เรียนมาในมหาวิทยาลัยมาประยุกต์ใช้ในการทำโครงการเป็นอย่างดี โดยผลการพัฒนาพบว่าระบบทำงานได้ตรงตามการทดสอบฟังก์ชั่นที่ได้กำหนด

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

บริษัท ทัชเทคโนโลยี จำกัด ประกอบธุรกิจการพัฒนาเทคโนโลยีเกี่ยวกับสุขภาพ Platform Digital Healthcare จำหน่ายผลิตภัณฑ์ประเภทสินค้าเทคโนโลยี และพัฒนาระบบให้กับหน่วยงานภาครัฐ เช่น โครงการระบบสารสนเทศเพื่อการบริหารการจัดการเรื่องร้องเรียนและบริการสาธารณะ , แอปพลิเคชั่น Mana Traffic Reporter สำหรับเลือกเส้นทางการเดินทางในกรุงเทพ , และอื่นๆ

บริษัท ทัชเทคโนโลยี จำกัด ที่ตั้ง ชั้น 2 บจก ทัช เทคโนโลยี จำกัด เลขที่ 1127 ถ.สุรนารายณ์ ในเมือง เมือง นครราชสีมา 30000 โดยบริษัทที่ตั้ง ณ นครราชสีมา เป็นทีมพัฒนาระบบสาขานครราชสีมา โดย บริษัทหลังตั้งอยู่ ณ กรุงเทพมหานคร โดยบริษัท ณ นครราชสีมา รับทำระบบให้หน่วยงาน เช่น โครงการระบบสารสนเทศเพื่อการบริหารการจัดการเรื่องร้องเรียนและบริการสาธารณะ ที่ร่วมมือกับเทศบาลนครราชสีมา โดยมี Line ChatBot เพื่อรับเรื่องและดูคำร้องอื่นๆ มีเว็บสำหรับจัดการคำร้อง มี Chat Service สำหรับตอบคำถามกับผู้ร้องเรียน โครงการการทำแอปพลิเคชั่นสำหรับเลือกเส้นทางการเดินทางในกรุงเทพ โครงการการพัฒนาเทคโนโลยีเพื่อสุขภาพ และอื่นๆ

วัตถุประสงค์ของการปฏิบัติงาน

1.เพื่อศึกษาการทำงานภายในบริษัท

2.เพื่อศึกษาการพัฒนาเว็บไซต์ด้วยเครื่องมือที่ใช้ในการทำงาน

3.เพื่อศึกษาระบบการจัดการกล้องวงจรปิด

ประวัติและรายละเอียดของหน่วยงาน

1. ชื่อและสถานที่ตั้งของสถานประกอบการ

บริษัท ทัชเทคโนโลยี จำกัด

ที่ตั้ง ชั้น 2 บจก ทัช เทคโนโลยี จำกัด เลขที่ 1127 ถ.สุรนารายณ์ ในเมือง เมือง นครราชสีมา 30000

2. ลักษณะการประกอบการ ผลิตภัณฑ์บริการของสถานประกอบการ

บริษัท ทัชเทคโนโลยี จำกัด เป็นบริษัทที่รับทำระบบให้หน่วยงาน เช่น โครงการระบบสารสนเทศเพื่อการบริหารการจัดการเรื่องร้องเรียนและบริการสาธารณะ แอปพลิเคชั่น Ma Traffic Reporter สำหรับเลือกเส้นทางการเดินทางในกรุงเทพ และระบบที่เกี่ยวกับเทคโนโลยีเพื่อสุขภาพ เครื่องตรวจสุขภาพ และอื่นๆ

3. ตำแหน่งและลักษณะงานที่สถานประกอบการมอบหมาย

นายทศพร แซ่อึ้ง ตำแหน่ง (System Analysis)

วิเคราะห์และออกแบบความต้องการระบบ ดูแลและจัดการเอกสารให้คนในทีม ออกแบบและวางแผนตารางการทำงาน Timeline วางกำหนดงานและวันส่งหมอบงาน การออกแบบระบบต่างๆ ออกแบบฐานข้อมูล กำหนด API Specification สื่อสารและให้คำแนะนำกับคนในทีม

นายธนกร ทองคล้าย ตำแหน่ง (Development Operations)

การติดตั้ง Service ต่างๆที่จำเป็นต้องใช้ การติดตั้งและใช้งาน Frigate ที่ใช้ในการแปลงกล้องวงจรปิด เป็นเส้น Api และลิ้ง Https เพื่อนำไปใช้บนเว็บ การ Deploy และ Build เว็บไซต์ และการจดโดเมน Website เพื่อในไปใช้งานได้จริง

นายปรวรรธ จำปาพันธุ์ ตำแหน่ง (Backend Developer)

พัฒนาระบบส่วนของการทำงานหลังบ้านโดยใช้ภาษา Golang วิเคราะห์ข้อมูลตาม Requirement เพื่อมาพัฒนาระบบ, วางโครงสร้างข้อมูลหลังบ้านตาม API Specification, สร้างฟังก์ชั่นการ เพิ่ม ลบ แก้ไข ข้อมูล กล้องและผู้ใช้ ด้วย API โดยใช้ฐานข้อมูล MongoDB, สร้างฟังก์ชั่นจัดการข้อมูลกล้อง cctv ผ่าน api ของ frigate, สร้างฟังก์ชั่นสำหรับระบบ Authentication เช่น token และ OTP, ออก api แต่ละเส้นเพื่อให้ frontend นำข้อมูลไปใช้งาน

นางสาวณัฐชรีณ ไปยะโพธิ์ศรี ตำแหน่ง (Frontend Developer)

สักษณะงาน การออกแบบหน้าเว็บทุกหน้าโดยใช้ Figma การเขียนโลจิกหน้าบ้านสำหรับการกำหนดการรับข้อมูลหน้าบ้าน การเขียนการทำงานเว็บ 2 ภาษา (ไทย – อังกฤษ) การเขียนหน้าเว็บด้วย javascript และ react

4. ชื่อและตำแหน่งงานของพนักงานที่ปรึกษา

นางสาวมยุรี รุนสูงเนิน ตำแหน่ง System Analysis

นายสุธน เสริฐกระโทก ตำแหน่ง Development Operations

นายจตุพร เหรียญทอง ตำแหน่ง Backend Developer

นายศรายุทธ ไกรษร ตำแหน่ง Frontend Developer

5. ระยะเวลาในการปฏิบัติงาน

5.1 ระยะเวลาในการปฏิบัติงาน

ตั้งแต่วันที่ 4 ธันวาคม พ.ศ.2566 ถึงวันที่ 29 มีนาคม พ.ศ.2567 ระยะเวลา 4 เดือน

5.2 วันในการปฏิบัติงาน

จันทร์ - ศุกร์

5.3 เวลาในการปฏิบัติงาน

09:00 – 18:00

บทที่ 2
รายละเอียดของการปฏิบัติงาน

จากการเรียนรู้ทฤษฎีการพัฒนาเว็บไวต์จากมหาวิทยาลัยราชภัฏนครราชสีมา จนกระทั่งได้มีโอกาศออกมาฝึกสหกิจศึกษากับ บริษัท ทัชเทคโนโลยี จำกัด เรื่องการพัฒนาระบบจัดการกล้องวงจรปิดบนเว็บ ก็ได้นำทฤษฎีที่ได้เรียนมาใช้ในการทำงานในหลายเรื่อง และได้ศึกษาเรื่องใหม่ควบคู่กับการทำงานไปด้วย เอกสารที่ใช้ได้แก่

1. ทฤษฎีเกี่ยวกับ MongoDB
2. ทฤษฎีเกี่ยวกับ Golang
3. ทฤษฎีเกี่ยวกับ Visual Studio Code
4. ทฤษฎีเกี่ยวกับ RESTful API
5. ทฤษฎีเกี่ยวกับ Postman
6. ทฤษฎีเกี่ยวกับ FrigateNVR
7. ทฤษฎีเกี่ยวกับ Gin framework
8. ทฤษฎีเกี่ยวกับ Docker
9. ทฤษฎีเกี่ยวกับ Jenkins
10. ทฤษฎีเกี่ยวกับ Nginx
11. ทฤษฎีเกี่ยวกับ Gitlab
12. ทฤษฎีเกี่ยวกับ ReactJS
13. ทฤษฎีเกี่ยวกับ JavaScript
14. ทฤษฎีเกี่ยวกับ CSS
15. ทฤษฎีเกี่ยวกับ HTML
16. Figma
17. Google Cloud Server
18. Portainer

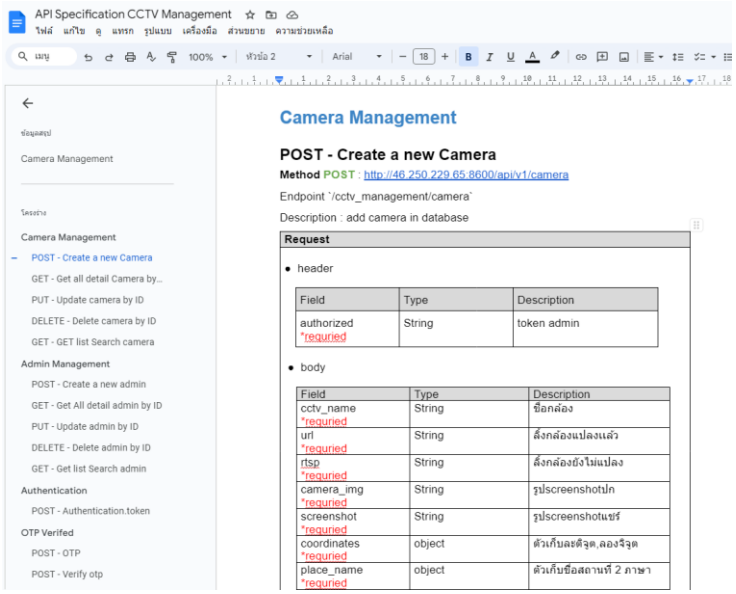
รายละเอียดของงานที่ปฏิบัติ

การพัฒนาระบบการจัดการกล้องวงจรปิดบนเว็บ จะมีผู้ใช้งาน 3 ระดับ คือ 1. ผู้ใช้งานทั่วไป สามารถดูกล้องวงจรปิดได้ทุกตัว 2. ผู้ดูแลทั่วไป สามารถดู แก้ไขข้อมูลกล้อง เพิ่มกล้อง และลบ ได้โดยการเข้าสู่ระบบไปยังในส่วนหลังบ้าน 3. ผู้ดูและระบบสูงสุด สามารถ เพิ่ม ลบ แก้ไข ดู ข้อมูลกล้อง และข้อมูลผู้ดูแลทั่วไปได้ จากนั้นทำการสร้างฐานข้อมูลโดยใช้ ฐานข้อมูล Mongodb

ขั้นตอนในการปฏิบัติงาน

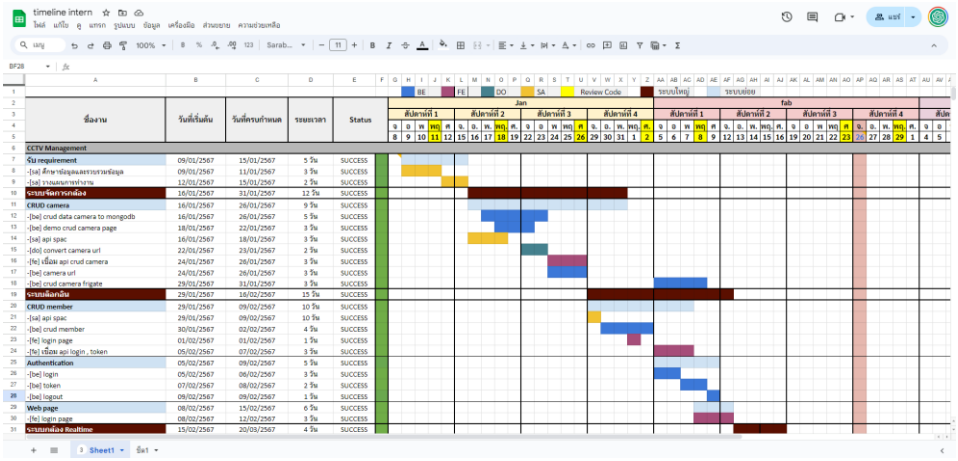
1. การวิเคราะห์ (System Analysis)

นายทศพร แซ่อึ้ง ตำแหน่ง System Analysis รับหน้าที่ในการศึกษาและวิเคราะห์จากความต้องการระบบที่ผู้หมอบหมายงานหมอบให้เพื่อนำมาวิเคราะห์ และวางแผนการทำงาน โดยการกำหนดตารางการทำงาน จากนั้นจัดทำเอกสาร API Specification วิเคราะห์และสื่อสารกับตำแหน่ง backend เพื่อสร้าง api ตามความต้องการของระบบได้อย่างถูกต้องโดยกำหนด Request , Success response , Errors response การกำหนด POST GET UPDATE DELETE และ HTTP status resnponse code และอื่นๆ



ภาพที่ 2.1 ตัวอย่างเอกสาร Api Spacification

timeline การทำงานให้กับตำแหน่งต่างๆ โดยกำหนดวันในการทำโดยจะมีรายละเอียดระบบฟังก์ชั่นหลักและฟังกชันย่อย หน้าที่ในแต่ละตำแหน่ง ระยะเวลาทำ สถานการณ์ทำงาน



ภาพที่ 2.2 ตัวอย่างเอกสาร Timeline

เอกสาร Document Infomation รวบรวมลิ้งในการเข้าถึง Service ต่างๆใน server Url เอกสาร และ URl ทั้งหมดในระบบ เพื่อให้คนในทีมเรียกใช้งานภายหลังได้ง่ายยิ่งขึ้น

Document Infomation

name	url	username	password
See it live 2			
See it live 2			
Mongodb		admin	P@ssword1234
Kongka			
Portainer		admin	P@ssword1234
Jenkins		admin	admin123
Frigate			

name	url
rtsp touch 1	
rtsp touch 2	
http cctv touch 1	
cctv test 1	
cctv test 2	
cctv test 3	

ภาพที่ 2.3 ตัวอย่างเอกสาร Document Information

2. การออกแบบ (Design)

2.1 เชิงตระกะ (Logical Design)

นายทศพร แซ่อึ้ง ตำแหน่ง System Analysis รับหน้าการออกแบบลักษณะการทำงานของระบบ Flow Overview และ Flow การทำงานในระบบ OTP Verified ระบบ OTP forget โดยการออกแบบการทำงานเพื่อให้ backend เข้าใจ flow ของระบบและให้เข้าใจระบบตามความต้องการ

ภาพที่ 2.4 ตัวอย่างการออกแบบ Flow ระบบ OTP

จากนั้นออกแบบ Data Dictionary ของฐานข้อมูล mongodb ในการเก็บประเภทข้อมูลต่างๆ จำนวนที่เก็บ และเก็บข้อมูล 2 ภาษา

2.2 การออกแบบเชิงกายภาพ (Physical Design)

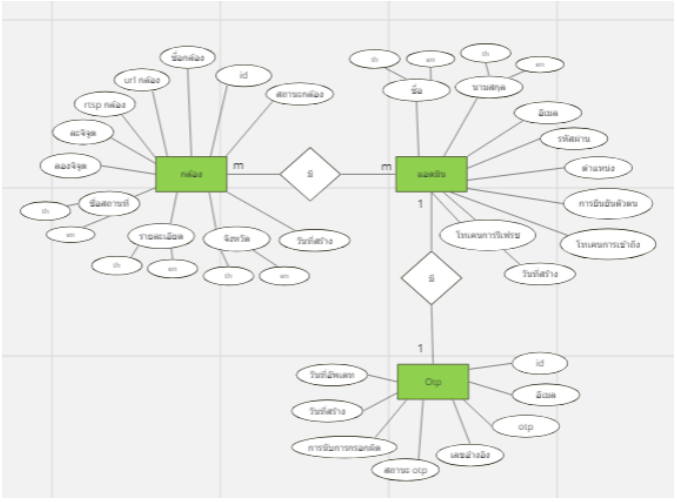
นายทศพร แซ่อึ้ง ตำแหน่ง System Analysis รับหน้าที่ในการออกแบบ Data Dictionary ของฐานข้อมูล mongodb ในการเก็บประเภทข้อมูลต่างๆ จำนวนที่เก็บ และเก็บข้อมูล 2 ภาษา

Collection Name (cctv admin)

รายการ(ไทย)	Field	ประเภทข้อมูล	จำนวน	request	หมายเหตุ
ไอดี	_id	string	255	Yes	
ชื่อ	first_name	string	255	Yes	แสดงชื่อผู้เข้าใช้งาน
นามสกุล	last_name	string	255	Yes	
อีเมล	email	string	255	Yes	
รหัสผ่าน	password	string	100	Yes	พิมพ์ใหญ่+เล็กใหญ่
เบอร์โทร	telephone	string	10	Yes	
ตำแหน่ง	role	string	100	Yes	cctv_admin , super admin
ยืนยันตัวตน	verify	string	20		

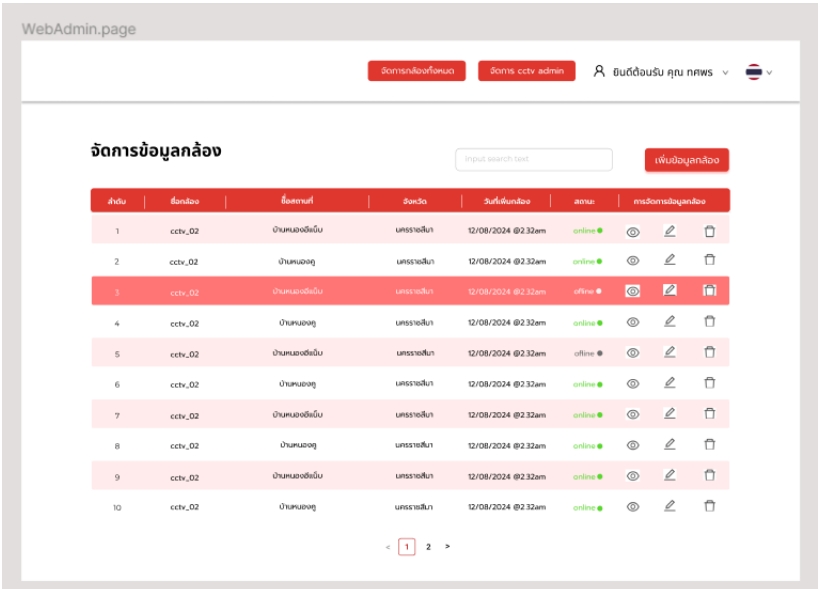
ภาพที่ 2.5 ตัวอย่างเอกสาร Data Dictionary

ตัวอย่างตาราง ER diagram แบบจำลองที่ใช้อธิบายโครงสร้างของฐานข้อมูลซึ่งเขียนออกมาในลักษณะของรูปภาพ การอธิบายโครงสร้างและความสัมพันธ์ของข้อมูล (Relationship)



ภาพที่ 2.6 ตัวอย่างเอกสาร ER diagram

การออกแบบหน้าเว็บต่างๆ โดยใช้เครื่องมือ Figma ในการออกแบบหน้าเว็บโดยสื่อสารและร่วมมือกันกับ Frontend Developer ในการออกแบบหน้าเว็บจะใช้ Ant Design เพื่อนำมาเปรียบเทียบเพื่อออกแบบ



ภาพที่ 2.7 ตัวอย่างการออกแบบ หน้าการจัดการกล้อง

3 การพัฒนาระบบ (ส่วนหลังบ้าน)

นายปรวรรธ จำปาพันธุ์ ตำแหน่ง Backend Developer ภาระงานที่ได้รับหมอบหมาย การพัฒนาโปรแกรมระบบหลังบ้าน การเขียน logic การทำงานหลังบ้านโดยใช้ภาษา Golang เขียนฟังฟังก์ชั่นการจัดการข้อมูลพื้นฐาน เพิ่ม ลบ แก้ไข กล้องและผู้ใช้ ด้วย RestAPI โดยใช้ฐานข้อมูล MongoDB โดยมี API ทั้งหมด 18 เส้น โดยแบ่งออกดังนี้

จัดการข้อมูลกล้องฝั่ง Server ประกอบไปด้วย Create, Update, Delete, Read, List

จัดการข้อมูลแอดมินฝั่ง Server ประกอบไปด้วย Create, Update, Delete, Read, List

หน้าแสดงข้อมูลกล้องฝั่ง Client ประกอบไปด้วย Client, ClientID, AllProvince, Snapshot

Authentication ประกอบไปด้วย Login, SendOTP, VerifyOTP, UpdatePass

```
{
    //Server
    adminRoutes := apiRoutes.Group("", authMiddleware)
    {
        adminRoutes.GET("/camera", app.camera.List)
        adminRoutes.POST("/camera", app.camera.Create)
        adminRoutes.GET("/camera/:id", app.camera.Read)
        adminRoutes.PUT("/camera/:id", app.camera.Update)
        adminRoutes.DELETE("/camera/:id", app.camera.Delete)

        adminRoutes.GET("/admin", app.admin.List)
        adminRoutes.POST("/admin", app.admin.Create)
        adminRoutes.GET("/admin/:id", app.admin.Read)
        adminRoutes.PUT("/admin/:id", app.admin.Update)
        adminRoutes.DELETE("/admin/:id", app.admin.Delete)
    }

    //Client
    apiRoutes.GET("/camera/client", app.camera.Client)
    apiRoutes.GET("/camera/client/:id", app.camera.ClientID)
    apiRoutes.GET("/camera/client/all_province", app.camera.AllProvince)

    apiRoutes.POST("/login", app.auth.Login)
    apiRoutes.POST("/snapshot", app.camera.SnapShot)

    //OTP
    apiRoutes.POST("/send-otp", app.otp.Create)
    apiRoutes.POST("/verify-otp", app.otp.VerifyOTP)

    //Password
    apiRoutes.PUT("/admin/set-password/:id", app.admin.UpdatePass)
}
router.GET("/swagger/*any", ginSwagger.WrapHandler(swaggerFiles.Handler))
```

ภาพที่ 2.8 การจัดการ API Routes ของฝั่งหลังบ้าน

3.1 การจัดการข้อมูลกล้อง Server

สำหรับโครงสร้างข้อมูลของกล้อง จะประกอบไปด้วย id, cctv_name, rtsp, camera_img, url, screenshot, coordinates, place_name, detail, province, cctv_status, created_at, updated_at โดยจะมีข้อมูลบางตัวที่เก็บทั้ง ภาษาไทย และ ภาษาอังกฤษ จะแยกฟิลข้อมูลโดยใช้ th,en

```go
package domain

type Camera struct {
    ID          string       `bson:"_id"`
    Name        string       `bson:"cctv_name"`
    Url         string       `bson:"url"`
    Rtsp        string       `bson:"rtsp"`
    Img         string       `bson:"camera_img"`
    Screenshot  string       `bson:"screenshot"`
    Coordinates *Coordinates `bson:"coordinates"`
    PlaceName   *PlaceName   `bson:"place_name"`
    Detail      *Detail      `bson:"detail"`
    Province    *Province    `bson:"province"`
    CctvStatus  string       `bson:"cctv_status"`
    CreatedAt   int64        `bson:"created_at"`
    UpdatedAt   int64        `bson:"updated_at"`
}

type Coordinates struct {
    Latitude  float64 `bson:"latitude"`
    Longitude float64 `bson:"longitude"`
}

type Detail struct {
    Detail_th string `bson:"th"`
    Detail_en string `bson:"en"`
}

type PlaceName struct {
    PlaceName_th string `bson:"th"`
    PlaceName_en string `bson:"en"`
}

type Province struct {
    Province_th string `bson:"th"`
    Province_en string `bson:"en"`
}
```

ภาพที่ 2.9 โครงสร้างข้อมูลของข้อมูลกล้อง

3.1.1 การเพิ่มข้อมูลกล้อง (Server)

การเพิ่มข้อมูลกล้องจะมีการเพิ่มข้อมูลผ่าน API เส้น Create ของ Camera โดยจะมีการเขียนฟังก์ชั่นสำหรับ Generate ID และ Validation ตัว cctv_name เพื่อให้ไม่สามารถเพิ่มชื่อซ้ำกัน

และ เพิ่มข้อมูล Rtsp ไปยัง API ของ Frigate เพื่อเพิ่มไปยัง Config Frigate โดยการ Call Func Frigate ในฝั่งหลังบ้าน เข้ามาใน Func Create กล้อง เมื่อทำการเพิ่มข้อมูลระบบจะบันทึกในฐานข้อมูล

service > camera > implement > create.go > ...

```go
package implement

import (
    "context"
    "log"
    "template/service/camera/camerain"
    "template/service/util"
)

func (impl *implementation) Create(ctx context.Context, input *camerain.CreateInput) (ID string, err error) {
    // Validate input
    err = impl.validator.Validate(input)
    log.Println(err)
    if err != nil {
        return "", util.ValidationCreateErr(err)
    }

    // Generate ID
    input.ID = impl.uuid.Generate()

    // Create camera domain object
    camera := camerain.CreateInputToCameraDomain(input)

    // Create camera in repository
    _, err = impl.repo.Create(ctx, camera)
    if err != nil {
        return "", util.RepoCreateErr(err)
    }

    // Call frigateCamera function
    go frigateCamera()

    // Return ID of created camera
    return camera.ID, nil
}
```

ภาพที่ 2.10 ฟังก์ชั่นสำหรับเพิ่มข้อมูลกล้อง


```
_id: "1757641696402731008"
cctv_name : "camera_1"
url : "https://frigate-intern.touchdevops.com/api/camera_1"
rtsp : "rtsp://touch:Touch1234@f03e0ec59b99.sn.mynetname.net:5555"
camera_img : "https://api-gateway-intern.touchdevops.com/images/9c8c2a53-3451-4253-b…"
screenshot : ""
▸ coordinates : Object
▸ place_name : Object
▸ detail : Object
▸ province : Object
created_at : 1707889410
updated_at : 1710149795
cctv_status : "online"
```


ภาพที่ 2.11 ข้อมูลกล้องในฐานข้อมูล mongodb

3.1.2 การอัพเดทข้อมูลกล้อง (Server)

การอัพเดทข้อมูลกล้องจะมีการอัพเดทข้อมูลผ่าน API เส้น Update ของ Camera โดยจะอัพเดทข้อมูลในฐานข้อมูลและใน Config ของ Frigate

service > camera > implement > update.go > ...

```go
package implement

import (
    "context"

    "template/service/camera/camerain"

    "template/domain"
    "template/service/util"
)

func (impl *implementation) Update(ctx context.Context, input *camerain.UpdateInput) (err error) {
    // Validate input
    err = impl.validator.Validate(input)
    if err != nil {
        return util.ValidationUpdateErr(err)
    }

    // Read camera information from repository
    camera := &domain.Camera{}
    filters := makeCameraIDFilters(input.ID)
    err = impl.repo.Read(ctx, filters, camera)
    if err != nil {
        return util.RepoReadErr(err)
    }

    // Map update input to camera domain object
    update := camerain.UpdateInputToCameraDomain(input)
    camera.ID = update.ID
    camera.Name = update.Name
    camera.Url = update.Url
    camera.Rtsp = update.Rtsp
    camera.Img = update.Img
    camera.Screenshot = update.Screenshot
    camera.Coordinates = &domain.Coordinates{
        Latitude:  update.Coordinates.Latitude,
        Longitude: update.Coordinates.Longitude,
    }
    camera.Detail = &domain.Detail{
        Detail_th: update.Detail.Detail_th,
        Detail_en: update.Detail.Detail_en,
    }
    camera.PlaceName = &domain.PlaceName{
        PlaceName_th: update.PlaceName.PlaceName_th,
        PlaceName_en: update.PlaceName.PlaceName_en,
    }
    camera.Province = &domain.Province{
        Province_th: update.Province.Province_th,
```

ภาพที่ 2.12 ฟังก์ชั่นสำหรับอัพเดทข้อมูลกล้อง

3.1.3 การลบข้อมูลกล้อง (Server)

การอัพเดทข้อมูลกล้องจะมีการอัพเดทข้อมูลผ่าน API เส้น Delete ของ Camera โดยจะลบข้อมูลในฐานข้อมูลและใน Config ของ Frigate

```go
package implement

import (
    "context"
    "template/domain"
    "template/service/camera/camerain"
    "template/service/util"
)

func (impl *implementation) Delete(ctx context.Context, input *camerain.DeleteInput) (err error) {
    // Read camera information from repository
    camera := &domain.Camera{}
    filters := makeCameraIDFilters(input.ID)
    err = impl.repo.Read(ctx, filters, camera)
    if err != nil {
        return util.RepoReadErr(err)
    }

    // Delete camera from repository
    err = impl.repo.Delete(ctx, filters)
    if err != nil {
        return util.RepoDeleteErr(err)
    }

    // Call frigateCamera function to update Frigate configuration
    go frigateCamera()

    return nil
}

```

ภาพที่ 2.13 ฟังก์ชั่นสำหรับลบข้อมูลกล้อง

3.2 การจัดการข้อมูลผู้ดูแล (Server)

สำหรับโครงสร้างข้อมูลของผู้ดูแล จะประกอบไปด้วย id, first_name, last_name, email, password, telephone, role, verified, access_token, refresh_token, expired_at, created_at, updated_at โดยจะมีข้อมูลบางตัวที่เก็บทั้ง ภาษาไทย และ ภาษาอังกฤษ จะแยกฟิลข้อมูลโดยใช้ th,en

```go
package domain

type Admin struct {
    ID           string     `bson:"_id"`
    Firstname    *Firstname `bson:"first_name"`
    Lastname     *Lastname  `bson:"last_name"`
    Email        string     `bson:"email"`
    Password     string     `bson:"password"`
    Telephone    string     `bson:"telephone"`
    Role         string     `bson:"role"`
    Verified     bool       `bson:"verified"`
    AccessToken  string     `bson:"access_token"`
    RefreshToken string     `bson:"refresh_token"`
    ExpiredAt    int64      `bson:"expired_at"`
    CreatedAt    int64      `bson:"created_at"`
    UpdatedAt    int64      `bson:"updated_at"`
}

type Firstname struct {
    Firstname_th string `bson:"th"`
    Firstname_en string `bson:"en"`
}

type Lastname struct {
    Lastname_th string `bson:"th"`
    Lastname_en string `bson:"en"`
}

type TokenAdmin struct {
    AccessToken  string `bson:"access_token"`
    RefreshToken string `bson:"refresh_token"`
    ExpiredAt    int64  `bson:"expired_at"`
}
```

ภาพที่ 2.14 โครงสร้างข้อมูลของข้อมูลผู้ดูแล

3.2.1 การเพิ่มข้อมูลผู้ดูแล (Server)

การเพิ่มข้อมูลผู้ดูแลจะมีการเพิ่มข้อมูลผ่าน API เส้น Create ของ Admin โดยจะมีการเขียนฟังก์ชั่นสำหรับ Generate ID และ Validation ฟิล Email เพื่อให้ไม่สามารถใช้อีเมลซ้ำกัน และเพิ่มฟังก์ชั่นสำหรับการส่งรหัส OTP ไปยังอีเมลที่ทำการเพิ่มข้อมูล เพื่อทำการ Verified บัญชีผู้ใช้งาน โดยใช้ SMTP Server Gmail

service > admin > implement > create.go > (*implementation).Create

```go
package implement

import (
    "bytes"
    "context"
    "html/template"
    "net/smtp"
    "strconv"
    "template/service/admin/adminin"
    "template/service/util"

    "github.com/jordan-wright/email"
    "golang.org/x/crypto/bcrypt"
)

func (impl *implementation) Create(ctx context.Context, input *adminin.CreateInput) (ID string, err error) {
    if err := impl.validator.Validate(input); err != nil {
        return "", util.ValidationCreateErr(err)
    }

    hashedPassword, err := HashPassword(input.Password)
    if err != nil {
        return "", err
    }

    input.ID = impl.uuid.Generate()
    admin := adminin.CreateInputToAdminDomain(input)
    admin.Password = hashedPassword
    admin.Verified = false

    if _, err := impl.repo.Create(ctx, admin); err != nil {
        return "", util.RepoCreateErr(err)
    }

    if err := SendEmail(input.Email, "https://seeitlive-intern.touchdevops.com/One-time-password-Reqeust"); err != nil {
        return "", err
    }

    return admin.ID, nil
}

const (
    senderEmail    = "porawatmik2001@gmail.com"
    senderPassword = "rvbm jzhi kuag pwtw"
    smtpServer     = "smtp.gmail.com"
    smtpPort       = 587
    templateFile   = "asset/register.html"
)
```

ภาพที่ 2.15 ฟังก์ชั่นสำหรับเพิ่มข้อมูลผู้ดูแล

3.2.2 การอัพเดทข้อมูลผู้ดูแล (Server)

การอัพเดทข้อมูลผู้ดูจะมีการอัพเดทข้อมูลผ่าน API เส้น Update ของ Admin โดยจะอัพเดทข้อมูลและบันทึกลงในฐานข้อมูล

```go
package implement

import (
    "context"

    "template/service/admin/adminin"

    "template/domain"
    "template/service/util"
)

func (impl *implementation) Update(ctx context.Context, input *adminin.UpdateInput) (err error) {
    err = impl.validator.Validate(input)
    if err != nil {
        return util.ValidationUpdateErr(err)
    }

    admin := &domain.Admin{}
    filters := makeAdminIDFilters(input.ID)

    err = impl.repo.Read(ctx, filters, admin)
    if err != nil {
        return util.RepoReadErr(err)
    }

    update := adminin.UpdateInputToAdminDomain(input)
    admin.ID = update.ID
    admin.Firstname = &domain.Firstname{
        Firstname_th:  update.Firstname.Firstname_th,
        Firstname_en: update.Firstname.Firstname_en,
    }
    admin.Lastname = &domain.Lastname{
        Lastname_th:  update.Lastname.Lastname_th,
        Lastname_en: update.Lastname.Lastname_en,
    }
    admin.Email = update.Email
    // admin.Password = update.Password
    admin.Telephone = update.Telephone
    admin.Role = update.Role
    admin.UpdatedAt = update.UpdatedAt

    err = impl.repo.Update(ctx, filters, admin)
    if err != nil {
        return util.RepoUpdateErr(err)
    }

    return nil
}
```

ภาพที่ 2.16 ฟังก์ชั่นสำหรับอัพเดทข้อมูลผู้ดูแล

3.1.3 การลบข้อมูลกล้อง (Server)

การอัพเดทข้อมูลกล้องจะมีการอัพเดทข้อมูลผ่าน API เส้น Delete ของ Admin โดยจะลบข้อมูลในฐานข้อมูล

```go
package implement

import (
    "context"
    "template/domain"
    "template/service/admin/adminin"
    "template/service/util"
)

func (impl *implementation) Delete(ctx context.Context, input *adminin.DeleteInput) (err error) {
    admin := &domain.Admin{}
    filters := makeAdminIDFilters(input.ID)

    err = impl.repo.Read(ctx, filters, admin)
    if err != nil {
        return util.RepoReadErr(err)
    }

    err = impl.repo.Delete(ctx, filters)
    if err != nil {
        return util.RepoDeleteErr(err)
    }

    return nil
}
```

ภาพที่ 2.17 ฟังก์ชั่นสำหรับลบข้อมูลผู้ดูแล

3.2.4 การอัพเดทรหัสผ่านของผู้ดูแล (Server)

การอัพเดทข้อมูลผู้ดูจะมีการอัพเดทข้อมูลผ่าน API เส้น UpdatePass ของ Admin โดยจะอัพเดทรหัสผ่านของผู้ดูแล เพื่อนำไปใช้ในระบบ เช่น การอัพเดทรหัสผ่าน, ลืมรหัสผ่าน

```go
package implement

import (
    "context"

    "template/service/admin/adminin"

    "template/domain"
    "template/service/util"

    "golang.org/x/crypto/bcrypt"
)

func (impl *implementation) UpdatePass(ctx context.Context, input *adminin.UpdatePassInput) (err error) {
    err = impl.validator.Validate(input)
    if err != nil {
        return util.ValidationUpdateErr(err)
    }

    // Generate hashed password
    hashedPassword, err := bcrypt.GenerateFromPassword([]byte(input.Password), bcrypt.DefaultCost)
    if err != nil {
        return err // handle error
    }

    admin := &domain.Admin{}
    filters := makeAdminIDFilters(input.ID)

    err = impl.repo.Read(ctx, filters, admin)
    if err != nil {
        return util.RepoReadErr(err)
    }

    update := adminin.UpdatePassInputToAdminDomain(input)
    admin.ID = update.ID
    admin.Password = string(hashedPassword) // Store hashed password
    admin.UpdatedAt = update.UpdatedAt

    err = impl.repo.Update(ctx, filters, admin)
    if err != nil {
        return util.RepoUpdateErr(err)
    }

    return nil
}
```

ภาพที่ 2.18 ฟังก์ชั่นสำหรับอัพเดทรหัสผ่านของผู้ดูแล

3.3 การแสดงข้อมูลกล้อง (Client)

สำหรับการแสดงข้อมูลกล้องของฝั่งผู้ใช้งาน หรือ Client จะประกอบไปด้วย API 3 เส้น ดังนี้

- Client คือ API ที่ไว้แสดงข้อมูลกล้องทั้งหมด หรือ เส้น List รายการกล้องทั้งหมดจากฐานข้อมูล

- ClientID คือ API ที่ไว้แสดงข้อมูลกล้อง ID นั้นๆ
- AllProvince คือ API ที่แสดงจังหวัดในฐานข้อมูลโดยไม่ซ้ำกัน

3.3.1 ฟังก์ชั่นสำหรับแสดงข้อมูลกล้องทั้งหมด (Client)

สำหรับฟังก์ชั่นนี้จะใช้สำหรับแสดงข้อมูลของกล้องในฝั่งของผู้ใช้งานจะแสดงข้อมูลออกไปบางส่วน ไม่เหมือนกันฝั่ง Server โดยฝั่ง Client จะแสดงข้อมูล เช่น id, camera_img, place_name, province, และ cctv_status

```go
package out

import (
    "template/domain"
)

type CameraViewClient struct {
    ID         string     `json:"id"`
    Img        string     `json:"camera_img"`
    PlaceName  *PlaceName `json:"place_name"`
    Province   *Province  `json:"province"`
    CctvStatus string     `json:"cctv_status"`
} // @Name CameraViewClient

func CameraToViewClient(camera *domain.Camera) (view *CameraViewClient) {

    var place *PlaceName

    if camera.PlaceName != nil {
        place = &PlaceName{
            PlaceName_th: camera.PlaceName.PlaceName_th,
            PlaceName_en: camera.PlaceName.PlaceName_en,
        }
    }

    var pro *Province

    if camera.Province != nil {
        pro = &Province{
            Province_th: camera.Province.Province_th,
            Province_en: camera.Province.Province_en,
        }
    }

    view = &CameraViewClient{
        ID:         camera.ID,
        Img:        camera.Img,
        PlaceName:  place,
        Province:   pro,
        CctvStatus: camera.CctvStatus,
    }
    return
}
```

ภาพที่ 2.19 ฟังก์ชั่นสำหรับแสดงข้อมูลกล้องทั้งหมดของฝั่ง Client

3.3.2 ฟังก์ชั่นสำหรับแสดงข้อมูลกล้อง by ID (Client)

สำหรับฟังก์ชั่นนี้จะใช้สำหรับแสดงข้อมูลของกล้อง by ID หรือ แสดงกล้องตัวที่เลือก โดยจะมีการแสดงข้อมูล id, url, coordinates, place_name, detail, province

service > camera > out > client_id.go > ...

```go
package out

import (
    "log"
    "template/domain"
)

type CameraViewClientID struct {
    ID string `json:"id"`
    Url         string       `json:"url"`
    Coordinates *Coordinates `json:"coordinates"`
    PlaceName   *PlaceName   `json:"place_name"`
    Detail      *Detail      `json:"detail"`
    Province    *Province    `json:"province"`
}

func CameraToViewClientID(camera *domain.Camera) (view *CameraViewClientID) {
    log.Println(camera.Coordinates)
    var coor *Coordinates

    if camera.Coordinates != nil {
        coor = &Coordinates{
            Latitude:  camera.Coordinates.Latitude,
            Longitude: camera.Coordinates.Longitude,
        }
    }

    var detail *Detail

    if camera.Detail != nil {
        detail = &Detail{
            Detail_th: camera.Detail.Detail_th,
            Detail_en: camera.Detail.Detail_en,
        }
    }

    var place *PlaceName

    if camera.PlaceName != nil {
        place = &PlaceName{
            PlaceName_th: camera.PlaceName.PlaceName_th,
            PlaceName_en: camera.PlaceName.PlaceName_en,
        }
    }

    var pro *Province

    if camera.Province != nil {
```

ภาพที่ 2.20 ฟังก์ชั่นสำหรับแสดงข้อมูลกล้อง by ID ของฝั่ง Client

3.3.3 ฟังก์ชั่นสำหรับแสดงข้อมูลจังหวัด (Client)

สำหรับฟังก์ชั่นนี้จะใช้สำหรับแสดงข้อมูลจังหวัดที่มีในฐานข้อมูลโดยไม่ซ้ำกัน โดยจะมีฟังก์ชั่นสำหรับเช็ค สถานะ cctv_status ต้องเป็น online ถึงจะแสดงข้อมูล เพื่อนำไปใช้ใน Dropdown จังหวัดของฝั่งหน้าบ้าน

```go
package implement

import (
    "context"
    "template/domain"
    "template/service/camera/out"
    "template/service/util"
)

func (impl *implementation) AllProvince(ctx context.Context, opt *domain.PageOption) (total int, items []*out.CameraViewAllProvince, err error) {

    uniqueProvinces := make(map[string]struct{})

    total, records, err := impl.repo.List(ctx, opt, &domain.Camera{})
    if err != nil {
        return 0, nil, util.RepoClientErr(err)
    }

    for _, record := range records {
        camera := record.(*domain.Camera)

        if camera.CctvStatus == "online" {
            if _, found := uniqueProvinces[camera.Province.Province_th]; !found {
                uniqueProvinces[camera.Province.Province_th] = struct{}{}
                uniqueProvinces[camera.Province.Province_en] = struct{}{}
                items = append(items, out.CameraToViewAllProvince(camera))
            }
        }
    }

    return total, items, nil
}
```

ภาพที่ 2.21 ฟังก์ชั่นสำหรับแสดงข้อมูลจังหวัด Client

3.4 Authentication System

สำหรับระบบ Authentication จะเป็นการยืนยัน เช่น การระบุตัวตนของผู้ใช้ระบบ โดยในระบบจะแบ่งออกเป็น 2 ส่วน คือ ส่วนของ Token และ OTP

3.4.1 ฟังก์ชั่นสำหรับ Generate token

สำหรับฟังก์ชั่น Generate token นั้นมีหน้าที่สำหรับสร้าง Token ที่ใช้ในการรับรองตัวตน (authentication) และการทำงานร่วมกันระหว่างระบบต่าง ๆ ในอินเทอร์เน็ต โดยส่วนมากใช้ในการรับรองตัวตนของผู้ใช้ (user authentication) และการแลกเปลี่ยนข้อมูลระหว่างระบบ ณ ที่นี้จะใช้เป็น Library ของตัว JWT Token

```go
func (impl *implementation) loginEmailAdmin(username string) (adminID string, role string, firstnameTH string, firstnameE
    admin := &domain.Admin{}
    filters := []string{
        fmt.Sprintf("email:eq:%s", username),
    }
    err = impl.adminRepo.Read(context.Background(), filters, admin)
    if err != nil {
        return "", "", "", "", false, util.Unauthorized(err)
    }

    return admin.ID, admin.Role, admin.Firstname.Firstname_th, admin.Firstname.Firstname_en, admin.Verified, nil
}

func (impl *implementation) checkPASS(input *authenticationin.LoginInput) (err error) {
    // Check Email
    admin := &domain.Admin{}
    filters := fmt.Sprintf("email:eq:%s", input.Username)
    err = impl.adminRepo.Read(context.Background(), []string{filters}, admin)
    if err != nil {
        return util.Unauthorized(errors.New("email not found"))
    }

    // // Check Password
    if err := bcrypt.CompareHashAndPassword([]byte(admin.Password), []byte(input.Password)); err != nil {
        return util.Unauthorized(errors.New("incorrect password"))
    }

    return nil
}

func (impl *implementation) getTokenEmailAdmin(ctx context.Context, adminID string, role string, th string, en string, jw
    // เวลาหมดอายุสำหรับ Access Token
    expirationTime := time.Now().Add(2 * time.Hour)
    claims := &authenticationin.ClaimsWithGenToken{
        UserID: adminID,
        RegisteredClaims: jwt.RegisteredClaims{
            ExpiresAt: jwt.NewNumericDate(expirationTime),
        },
        Role:         role, // ใช้ข้อมูล Role ที่ส่งมา
        Firstname_th: th,
        Firstname_en: en,
    }
    accessToken := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
    accessString, err := accessToken.SignedString(jwtKey)
    if err != nil {
        return nil, util.Unauthorized(errors.New("error"))
```

ภาพที่ 2.22 ฟังก์ชั่นสำหรับ Generate token

เมื่อทำการ Login ด้วย username และ password สำเร็จ ระบบจะทำการ generate token มาให้ โดยจะมีเวลาหมดอายุของ token อยู่ที่ 2 ชั่วโมงเมื่อครบ token นั้นจะ expired ไม่สามารถใช้งานได้ ต้องทำการส่ง request เพื่อขอ token ใหม่


```
{
    "status": "OK",
    "code": 200,
    "data": {
        "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
            eyJ1c2VyX2lkIjoiMTc3MzU1NTE4NzI0ODUyNTMxMiIsInJvbGUiOiJzdXBlcmF
            ojxK5LILW0tjlnQ0BwYVo8UHNQjMbckJO8DgChP1FRI",
        "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
            eyJ1c2VyX2lkIjoiMTc3MzU1NTE4NzI0ODUyNTMxMiIsInJvbGUiOiJzdXBlcmF
            IP2psTtBFf3URNAA-6rCASlibpnj7DGK3Ywa9mtP_qI",
        "expired_at": 1712519386
    }
}
```


ภาพที่ 2.23 token เมื่อ login

3.4.2 ฟังก์ชั่นสำหรับ SendOTP

สำหรับฟังก์ชั่น SendOTP นั้นมีหน้าที่สำหรับ Generate ตัว เลข OTP และ ตัว ของ RefCode เพื่อใช้ในการยืนยันตัวบัญชีผู้ใช้ โดย หมายเลข OTP จะมี 6 หลัก และ RefCode จะมีตัวอักขระ 4 หลัก และ ตั้งเวลาสำหรับหมดอายุของ OTP เป็นเวลา 3 นาที

```go
const (
    otpLength      = 6
    refCodeLength  = 4
    expiredTime    = 3 * time.Minute
    templateFile   = "asset/sendotp.html"
    senderEmail    = "porawatmik2001@gmail.com"
    senderPassword = "rvbm jzhi kuag pwtw"
    smtpServer     = "smtp.gmail.com"
    smtpPort       = 587
    charset        = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
)

// GenerateOTP generates a random 6-digit OTP
func generateOTP() string {
    otp := rand.Intn(999999)
    return fmt.Sprintf("%06d", otp)
}

// GenerateRefCode generates a random 6-character reference code
func generateRefCode() string {
    b := make([]byte, refCodeLength)
    for i := range b {
        b[i] = charset[rand.Intn(len(charset))]
    }
    return string(b)
}

// SendEmail sends an email with the OTP
func sendEmail(recipient, otp, refCode string) error {
    // Read template file
    tplContent, err := os.ReadFile(templateFile)
    if err != nil {
        return fmt.Errorf("error reading template file: %v", err)
    }

    tpl, err := template.New("emailTemplate").Parse(string(tplContent))
    if err != nil {
        return fmt.Errorf("error parsing template: %v", err)
    }

    // Create email content
    var body bytes.Buffer
    err = tpl.Execute(&body, struct{ OTP, RefCode string }{OTP: otp, RefCode: refCode})
    if err != nil {
        return fmt.Errorf("error executing template: %v", err)
    }

```

ภาพที่ 2.24 ฟังก์ชั่นสำหรับ SendOTP

3.4.3 ฟังก์ชั่นสำหรับ VerifyOTP

สำหรับฟังก์ชั่น VerifyOTP นั้นมีหน้าที่สำหรับ ยืนยันตัว OTP ที่ได้จากการ SendOTP โดยฟังก์ชั่นนี้จะมีการตรวจสอบหากกรอก OTP ผิดเกิน 3 ครั้ง OTP นั้นจะหมดอายุ และไม่สามารถใช้งานได้ต้องทำการ SendOTP เพื่อขอเลข OTP ใหม่ หากกรอบ OTP ถูกต้องก็จะทำการปรับสถานะ Verified ของข้อมูล Admin เป็น true และ สถานะ OTP เป็น active

```
52      if input.Code != otp.Code {
53          otp.Attempts++
54          if otp.Attempts > 3 {
55              otp.Status = "expired"
56              if err = impl.otpRepo.Update(ctx, filters, otp); err != nil {
57                  return nil, util.RepoUpdateErr(err)
58              }
59              return nil, util.OTPExpiredErr(err)
60          }
61          if err = impl.otpRepo.Update(ctx, filters, otp); err != nil {
62              return nil, util.RepoUpdateErr(err)
63          }
64          return nil, util.OTPNotFound(err)
65      }
66
67      adminFilters := []string{
68          "email:eq:" + otp.Email,
69      }
70
71      if err = impl.adminRepo.Read(ctx, adminFilters, admin); err != nil {
72          return nil, util.RepoReadErr(err)
73      }
74
75      updateInput := &adminInput.UpdateVerifyInput{
76          Email:    otp.Email,
77          Verified: true,
78      }
79
80      if err = impl.adminService.UpdateVerify(ctx, updateInput); err != nil {
81          return nil, util.RepoUpdateErr(err)
82      }
83
84      otp.Status = "active"
85
86      if err = impl.otpRepo.Update(ctx, filters, otp); err != nil {
87          return nil, util.RepoUpdateErr(err)
88      }
89
90      view = out.OTPToViewVerify(admin, otp)
91      return view, nil
```

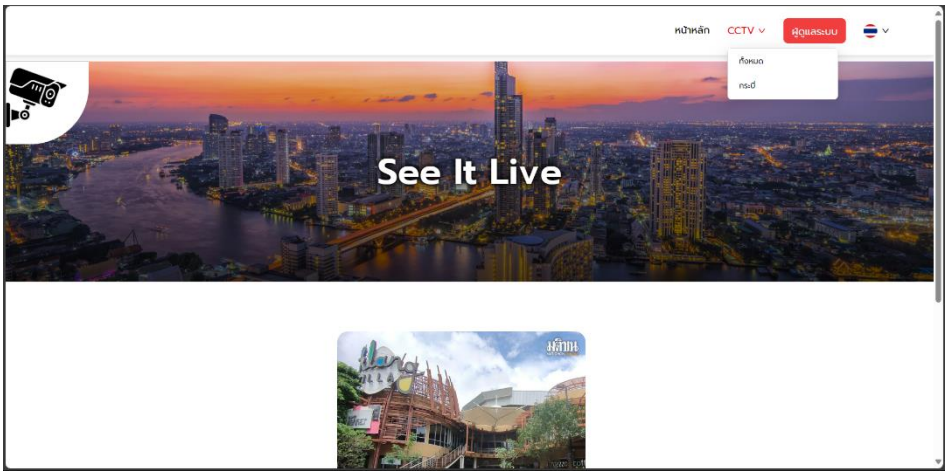
ภาพที่ 2.25 ฟังก์ชั่นสำหรับ VerifyOTP

4 การพัฒนาระบบ (ส่วนหน้าบ้าน)

นางสาวณัฐชรีณ ไปยะโพธิ์ศรี รับผิดชอบในส่วนการพัฒนาเว้บไซต์ส่วน front-end ทั้งหมด สร้างหน้าตามprocess การทำงานตามที่ออกแบบ คือการเลือกหน้าเลือกปุ่มแล้วไปที่ใดต่อนั้นเอง และการนำ lib ต่างๆมาใช้งาน เพื่อช่วยให้การทำงานดีขึ้น สดวกขึ้น การใช้งาน css เพื่อให้ผลลที่แสดงออกมาป็นไปตามความต้องการระบบและตามที่ออกแบบไว้ พัฒนาด้วนภาษา react js ได้แบ่งการทำงานออกเป็น 2 ส่วนใหญ่ๆคือ 1.ส่วนของ client คือผู้ใช้งานทั่วไป 2.ส่วนของการจัดการของมูลของ admin และเว็บไซต์ทุกหน้าใช้งานได้ 2 ภาษาทั้งหมด รวมทั้งการสร้าง mock data เพื่อใช้งาน

1.ส่วนของ client จะเป็นการนำข้อมูลออกมาแสดงเพื่อให้ผู้ใช้งานได้ใช้งาน ทำการเชื่อมต่อ api เพื่อนำข้อมูลออกมาใช้งาน รวมถึงการจำกัดการแสดงข้อมูล การเข้าถึงต่างๆการใช้งาน

1.1 ตัวอย่างการแสดงผลหน้าแรก และการเข้าถึง dropdown

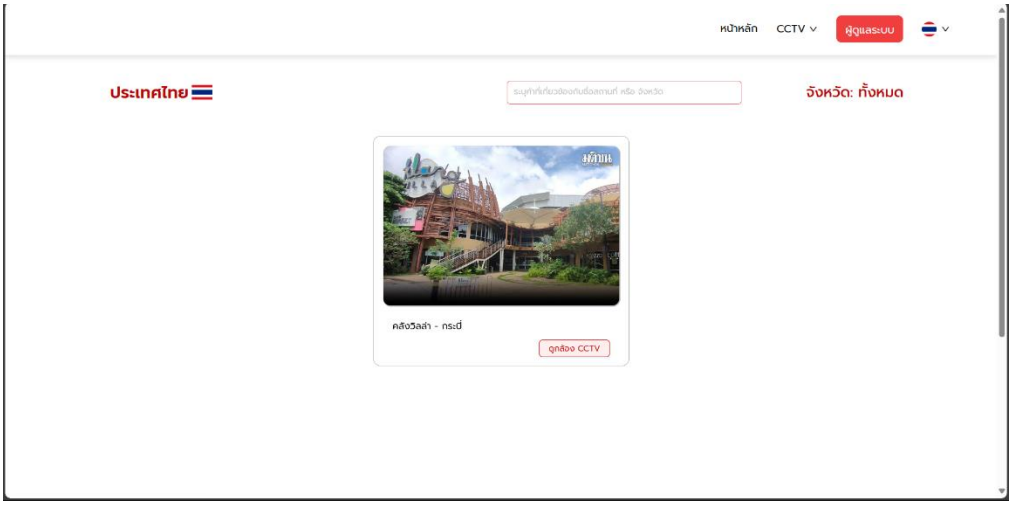


ภาพที่ 2.26 ตัวอย่างการแสดงผลหน้าแรก

1.2 ตัวอย่างโค้ดการเข้าถึง api เพื่อนำข้อมูลออกมาใช้งานทำงานตามเงื่อนไขการเลือกจังหวัดและการค้นหา

```
const CCTVPage = () => {
  });

  const selectedProvince = location.state?.selectedProvince || "";
  const selectedProvinceList = location.state?.selectedProvinceList || {};

  const handleNavigate = (camera) => {
    const selectedCameraId = camera.id;
    navigate(`/Camera-detail/${selectedCameraId}`);
  };

  const fetchCameras = async () => {      You, 3 weeks ago • function search relaod if 401
    setIsLoading(true);
    try {
      let response;

      if (selectedProvince === "all") {
        if (searchValue) {
          response = await axios.get(
            `${process.env.REACT_APP_CLIENT}?filters=province.${selectedLanguage}|place_name.${selectedLanguage}:or:like:${searchValue}&
            filters=cctv_status:eq:online&page=${pagination.current}&per_page=${pagination.pageSize}`
          );
        } else {
          response = await axios.get(
            `${process.env.REACT_APP_CLIENT}?filters=cctv_status:eq:online&page=${pagination.current}&per_page=${pagination.pageSize}`
          );
        }
      } else {
        response = await axios.get(
          `${process.env.REACT_APP_CLIENT}?filters=province.${selectedLanguage}:like:${selectedProvinceList[0]?.province[selectedLanguage]}`
        );
      }


      if (response.status === 200 || response.status === 201) {
        setCameras(response.data.data);
        const contentLength = response.headers["x-content-length"];
        if (contentLength) {
          setPagination((prevPagination) => ({
            ...prevPagination,
            total: contentLength,
```

ภาพที่ 2.27 ตัวอย่างโค้ดการเลือกใช้ api พร้อม filter ตามเงื่อนไข

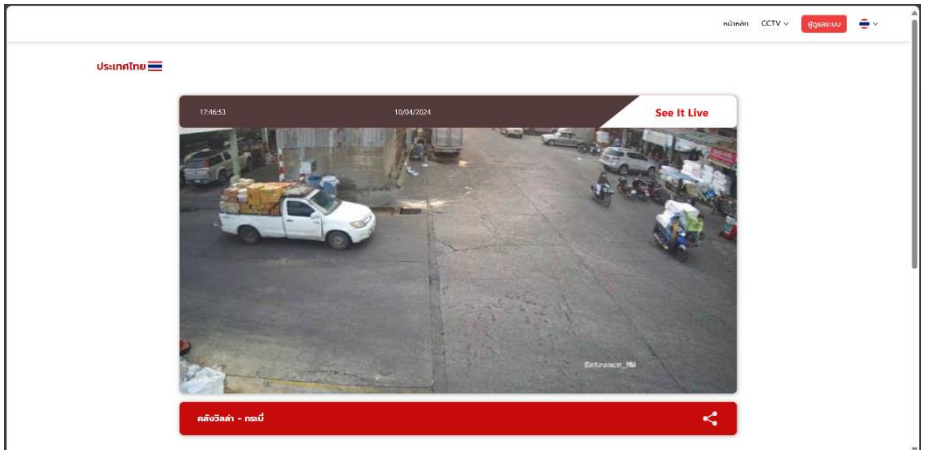


ภาพที่ 2.28 ตัวอย่างการแสดงผลจากโค้ด

1.3 ตัวอย่างการแสดงข้อมูลเมื่อเลือกสถานที่ที่ต้องการ จะแสดงรายละเอียดต่างๆ ตามการออกแบบ

```
// );
const fetchCameraDetails = async () => {
  try {
    const response = await axios.get(
      `${process.env.REACT_APP_GET_ID.replace("_id", id)}`
    );
    setSelectedCamera(response.data.data);
    //console.log("response.data.data", response.data.data);
  } catch (error) {
    console.error("Error fetching camera details:", error);
  }
};

useEffect(() => {
  if (id) {
    fetchCameraDetails();
  }
}, [id]);

if (!selectedCamera) {
  return (
    <div
      style={{
        alignContent: "center",
        fontSize: "18px",
        marginTop: "23%",      You, 4 weeks ago • share facebook
        marginBottom: "23%",
        fontFamily: "Prompt",
        display: "flex",
        justifyContent: "center",
      }}
    >
      {t("navbar.nodata")}
    </div>
  );
}
```

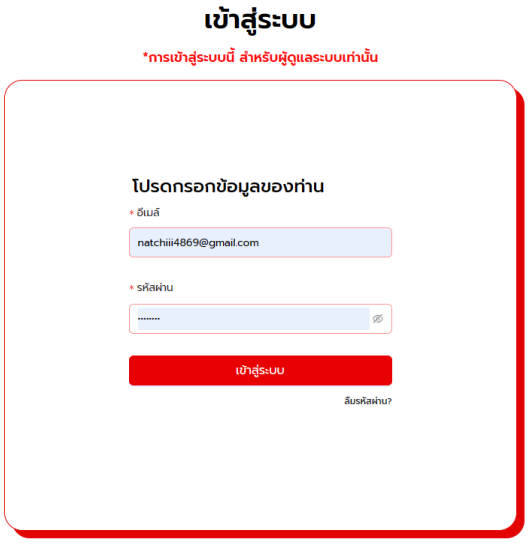
ภาพที่ 2.29 ตัวอย่างโค้ดการเลือกใช้ api พร้อมการเลือกข้อมูลมาใช้โดยการส่ง id ที่ต้องการไป



ภาพที่ 2.30 ตัวอย่างการแสดงผลจากโค้ดดังกล่าว

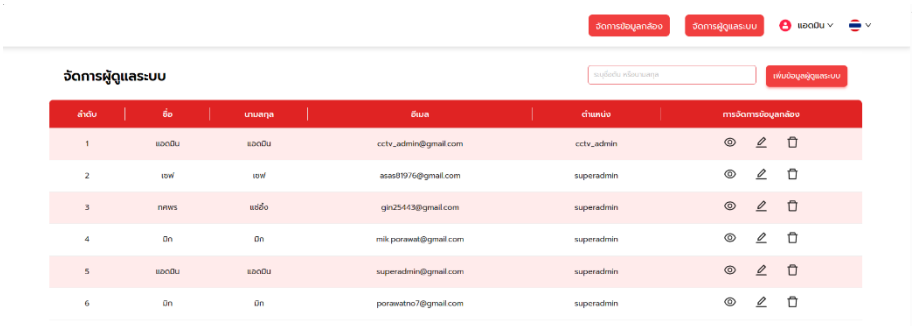
2.ส่วนของ Admin การเข้าถึงส่วนนี้จะต้องทำการล็อกอินเข้าสู่ระบบก่อนถึงจะสามารถเข้าถึงข้อมูล และการจัดการข้อมูลได้ จะเป็นการนำข้อมูลออกมาแสดงเพื่อให้ผู้ดูแลระบบได้ใช้งานจัดการข้อมูลได้ง่าย สดวก ลดการเข้าถึงฐานข้อมูลโดยตรงเพื่อลดการเกิดข้อผิดพลาดและปัญหาที่อาจจะเกิดขึ้น รายละเอียดดังนี้ การเข้าสู่ระบบ การแยกสิทธิ์เข้าถึงการใช้งานการจัดการข้อมูล CRUD ของ superadmin และ cctv_admin หน้าต่าง CRUD ทั้งหมด การจัดการข้อมูลกล้อง การจัดการข้อมูลผู้ดูแลระบบ

2.1 การเข้าสู่ระบบ จะมีช่องรับข้อมูล email และ password จะถูกส่งไปที่ api หากเข้าเงื่ออนไขจะได้รับ token เพื่อเข้าถึงตามสิทธิ์ที่กำหนดไว้



ภาพที่ 2.31 ตัวอย่างหน้าต่างการเข้าสู่ระบบ

ภาพที่ 2.32 ตัวอย่างหน้าต่างการเข้าถึงข้อมูลของ cctv_admin จัดการกล้องได้อย่างเดียว



ภาพที่ 2.33 ตัวอย่างหน้าต่างการเข้าถึงข้อมูลของ superadmin จัดการกล้องและผู้ดูแลระบบ

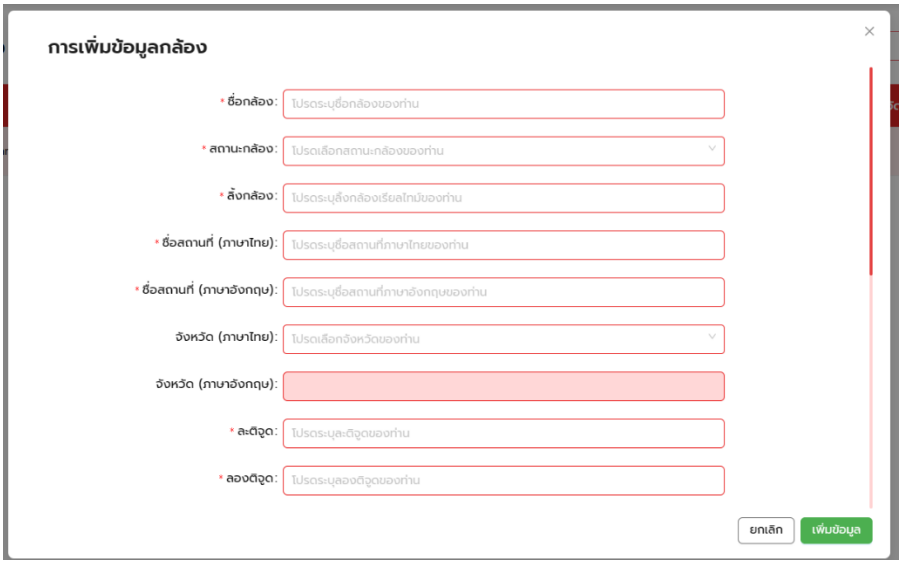
2.2 การสร้างหน้าต่าง CRUD ข้อมูลโดยทำเป็น modal เพื่อจัดการข้อมูล โดยการจัดการข้อมูลที่ทำร่วมกับฐานข้อมูลจะต้องการ Authorization token ที่ได้จาการเข้าสู่ระบบ เพื่อป้องกันการจัดการที่ไม่ถูกต้อง

2.2.1 การจัดการข้อมูลกล้อง

2.2.1.1 การสร้างหรือเพิ่มข้อมูล

```
const onFinish = async (values) => {
  setLoading(true);
  try {
    const res = await handleImageFileChange();

    //console.log("res", res);

    if (res.data.code === 200) {
      const datainput = {
        cctv_name: values.cctv_name,
        rtsp: values.rtsp,
        camera_img: res?.data?.data?.link?.[0],
        coordinates: {
          latitude: parseFloat(values["coordinates_latitude"]),
          longitude: parseFloat(values["coordinates_longitude"]),
        },
        place_name: {
          th: values["place_name_th"],
          en: values["place_name_en"],
        },
        detail: {
          th: values["detail_th"],
          en: values["detail_en"],
        },
        province: {
          th: values["province_th"],
          en: values["province_en"],
        },
        cctv_status: values.cctv_status,
      };
      // Create the camera using the prepared data
      const response = await axios.post(
        process.env.REACT_APP_POST_CAMERA,
        datainput,
        {
          headers: {
            Authorization: `Bearer ${access_token}`,
            "Content-Type": "application/json",
```

ภาพที่ 2.34 ตัวอย่างโค้ดการส่งข้อมูลไปเพิ่มที่ฐานข้อมูลผ่าน Api

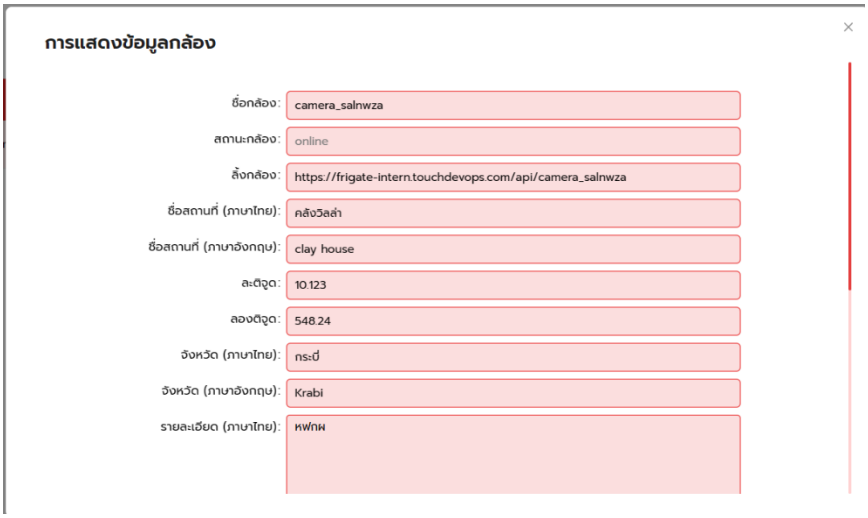


ภาพที่ 2.35 ตัวอย่างหน้าต่างการสร้างข้อมูล

2.2.1.2 การเรียกดูข้อมูล

```
useEffect(() => {
  const fetchData = async () => {
    if (isVisible) {
      try {
        // Constructing the URL
        const url = process.env.REACT_APP_GET_CAMERA+"/"+record.id

        //console.log("Constructed URL:", url); // Logging the constructed URL

        // Fetch camera data when modal is visible
        const response = await axios.get(url, {
          headers: {
            Authorization: `Bearer ${access_token}`,
          },
        });
        setCameraData(response?.data?.data);
        //console.log("Response:", response?.data?.data);
      } catch (error) {
        console.error("Error fetching camera data:", error);
      }
    }
  };

  fetchData();
}, [isVisible, record.id]);
```

ภาพที่ 2.36 ตัวอย่างโค้ดการเรียกดูข้อมูลที่ฐานข้อมูลผ่าน Api

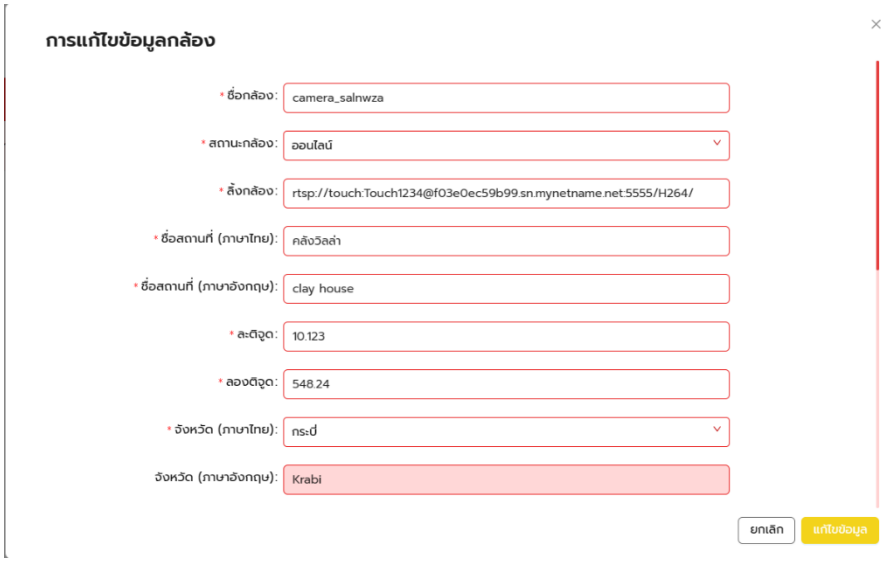


ภาพที่ 2.37 ตัวอย่างหน้าต่างการเรียกดูข้อมูล

2.2.1.3 การแก้ไขข้อมูล

```
const response = await axios.put(
  url,
  {
    cctv_name: form.getFieldValue("cctv_name"),
    rtsp: form.getFieldValue("rtsp"),
    camera_img: imageFile
      ? res?.data?.data?.link?.[0] || record?.camera_img
      : record.camera_img,
    coordinates: {
      latitude: parseFloat(form.getFieldValue("coordinates_latitude")),
      longitude: parseFloat(form.getFieldValue("coordinates_longitude")),
    },
    place_name: {
      th: form.getFieldValue("place_name_th"),
      en: form.getFieldValue("place_name_en"),
    },
    detail: {
      th: form.getFieldValue("detail_th"),
      en: form.getFieldValue("detail_en"),
    },
    province: {
      th: form.getFieldValue("province_th"),
      en: form.getFieldValue("province_en"),
    },
    cctv_status: updatedData.cctv_status,
  },
  {
    headers: {
      Authorization: `Bearer ${access_token}`,
    },
  }
);

if (response.status === 200 || response.status === 201) {
  message.success({
    content: t("alert.camera.successupdate"),
    style: {
      fontFamily: "prompt",
      fontSize: "16px",
    },
```

ภาพที่ 2.38 ตัวอย่างโค้ดการส่งข้อมูลไปแก้ไขที่ฐานข้อมูลผ่าน Api

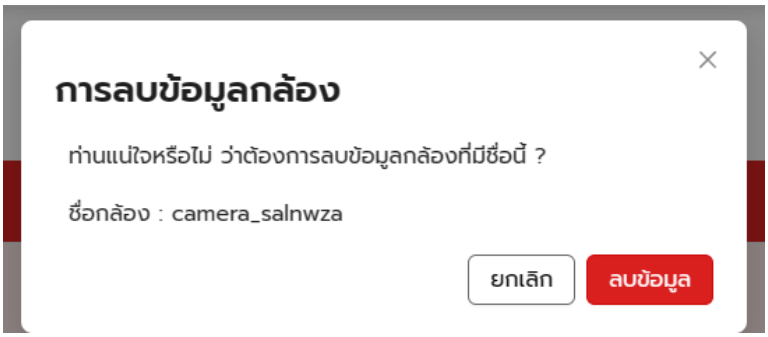


ภาพที่ 2.39 ตัวอย่างหน้าต่างการแก้ไขข้อมูลกล้อง

2.2.1.4 การลบข้อมูล

```
//console.log( loading , loading);

try {
  const Url = `${process.env.REACT_APP_DELETE_CAMERA.replace(
    "_id",
    record.id
  )}`;
  const response = await axios.delete(Url, {
    headers: {
      Authorization: `Bearer ${access_token}`,
    },
  });

  if (response.status === 200) {
    onClose(); // ปิดโมดัลเมื่อลบเสร็จสมบูรณ์
    onRefresh();
    message.success({
      content: t("alert.camera.successdelete"),
      style: {
        fontFamily: "prompt",
        fontSize: "16px",
      },
    });
  }
} catch (error) {
  message.error({
    content: t("alert.camera.errordelete"),
    style: {
      fontFamily: "prompt",
      fontSize: "16px",
    },
  });
} finally {
  setLoading(false); // สิ้นสุดการโหลด ไม่ว่าคำขอจะสำเร็จหรือไม่
}
};
```

ภาพที่ 2.40 ตัวอย่างโค้ดการลบข้อมูลกล้องผ่าน Api



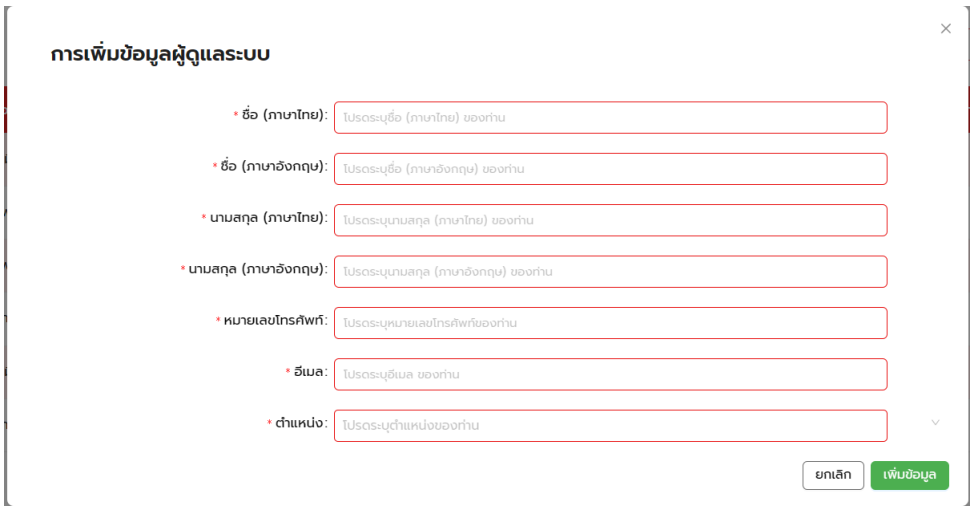
ภาพที่ 2.41 ตัวอย่างหน้าต่างการลบข้อมูลกล้อง

2.2.2 การจัดการข้อมูลผู้ดูแลระบบ

2.2.2.1 การสร้างหรือเพิ่มข้อมูลผู้ดูแลระบบ

```
const onFinish = async (values) => {
  setLoading(true);
  try {
    const response = await axios.post(
      process.env.REACT_APP_POST_ADMIN,
      {
        first_name: {
          th: values["first_name_th"],
          en: values["first_name_en"],
        },
        last_name: {
          th: values["last_name_th"],
          en: values["last_name_en"],
        },
        email: values.email,
        password: values.password,
        telephone: values.telephone,
        role: values.role,
      },
      {
        headers: {
          Authorization: `Bearer ${access_token}`,
        },
      }
    );

    if (response.status === 200 || response.status === 201) {
      message.success({
        content: t("alert.admin.successcreate"),
        style: {
          fontFamily: "prompt",
          fontSize: "16px",
        },
      });

      onClose();
      onRefresh();
      form.resetFields();
    } else {
      message.error(t("alert.admin.errorcreate"));       You, 2 week
```

ภาพที่ 2.42 ตัวอย่างโค้ดการส่งข้อมูลผู้ดูแลระบบไปเพิ่มที่ฐานข้อมูลผ่าน Api

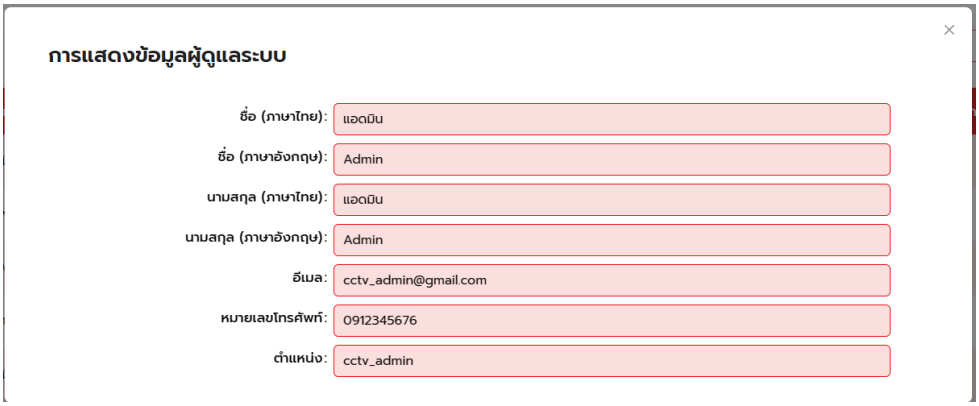


ภาพที่ 2.43 ตัวอย่างหน้าต่างการสร้างข้อมูลผู้ดูแลระบบ

2.2.2.2 การเรียกดูข้อมูลผู้ดูแลระบบ

```
export const ReadAdmin = ({ isVisible, onClose, record }) => {
  const [adminData, setAdminData] = useState(null);
  const access_token = localStorage.getItem("access_token");
  const { t } = useTranslation();

  const fetchAdminData = async () => {
    try {
      const response = await axios.get(
        `${process.env.REACT_APP_GET_ADMIN}/${record.id}`,
        {
          headers: {
            Authorization: `Bearer ${access_token}`,
          },
        }
      );
      setAdminData(response?.data?.data);
    } catch (error) {
      console.error("Error fetching admin data: ", error);
    }
  };

useEffect(() => {
  fetchAdminData();
}, [record.id]);
```

ภาพที่ 2.44 ตัวอย่างโค้ดการเรียกดูข้อมูลผู้ดูแลระบบที่ฐานข้อมูลผ่าน Api

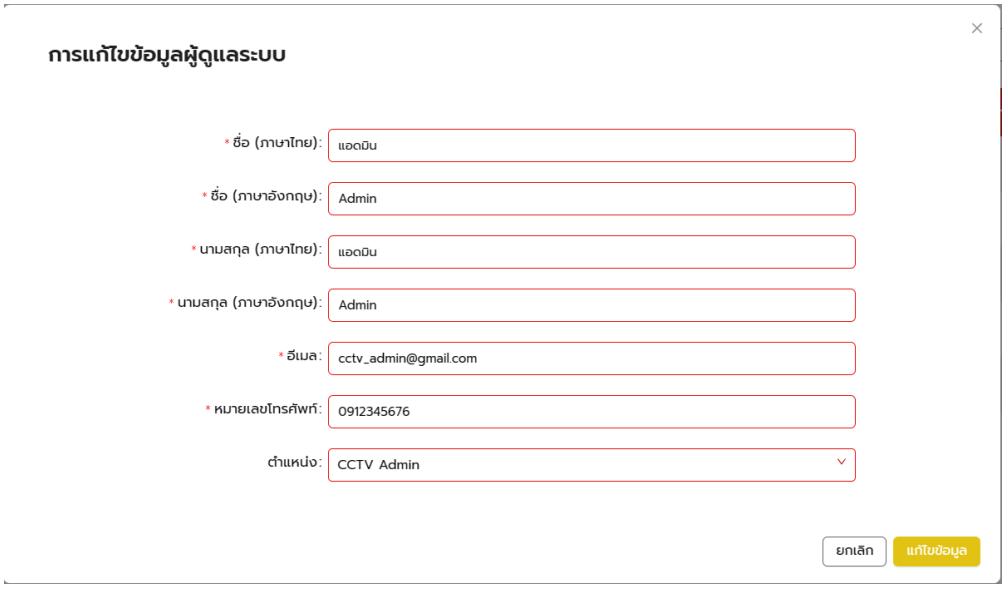


ภาพที่ 2.45 ตัวอย่างหน้าต่างการเรียกดูข้อมูลผู้ดูแลระบบ

2.2.2.3 การแก้ไขข้อมูลผู้ดูแลระบบ

```
const handleUpdate = async () => {
  try {
    setLoading(true);
    const url = `${process.env.REACT_APP_PUT_ADMIN.replace(
      "_id",
      record.id
    )}`;

    const response = await axios.put(
      url,
      {
        first_name: {
          th: form.getFieldValue("first_name_th"),
          en: form.getFieldValue("first_name_en"),
        },
        last_name: {
          th: form.getFieldValue("last_name_th"),
          en: form.getFieldValue("last_name_en"),
        },
        email: form.getFieldValue("email"),
        telephone: form.getFieldValue("telephone"),
        role: form.getFieldValue("role"),
      },
      {
        headers: {
          Authorization: `Bearer ${access_token}`,
        },
      }
    );

    if (response.status === 200 || response.status === 201) {
      message.success({      You, 2 weeks ago • Uncommitted char
        content: t("alert.admin.successupdate"),
        style: {
          fontFamily: "prompt",
          fontSize: "16px",
        },
      }).
```

ภาพที่ 2.46 ตัวอย่างโค้ดการส่งข้อมูลผู้ดูแลระบบไปแก้ไขที่ฐานข้อมูลผ่าน Api

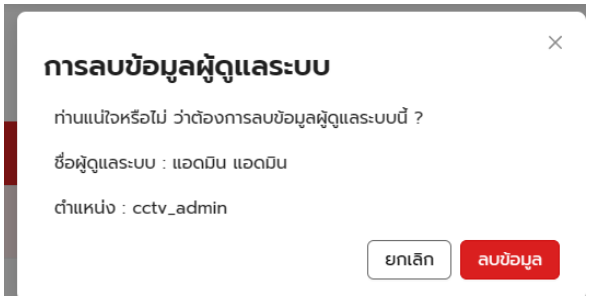


ภาพที่ 2.47 ตัวอย่างหน้าต่างการแก้ไขผู้ดูแลระบบ

2.2.2.4 การลบข้อมูลผู้ดูแลระบบ

```
try {
  if (record.id === currentUserID) {
    StyledErrorModal.error({
      title: t("alert.admin.erroraaa"),
      content: t("alert.admin.erroraaatext"),
      style: {
        fontFamily: "prompt",
      },
    });
  } else if (record.id !== currentUserID) {
    // เพิ่มเงื่อนไขด้วย else if
    const url = `${process.env.REACT_APP_DELETE_ADMIN.replace(
      "_id",
      record.id
    )}`;
    const response = await axios.delete(url, {
      headers: {
        Authorization: `Bearer ${access_token}`,
      },
    });

    if (response.status === 200 || response.status === 201) {
      onClose();
      onRefresh();
      message.success({
        content: t("alert.admin.successdelete"),
        style: {
          fontFamily: "prompt",
          fontSize: "16px",
        },
      });
    } else {
      message.error({
        content: t("alert.admin.errordelete"),
        style: {
          fontFamily: "prompt",
          fontSize: "16px",
        },
      });
```

ภาพที่ 2.48 ตัวอย่างโค้ดการลบข้อมูลผู้ดูแลระบบผ่าน Api

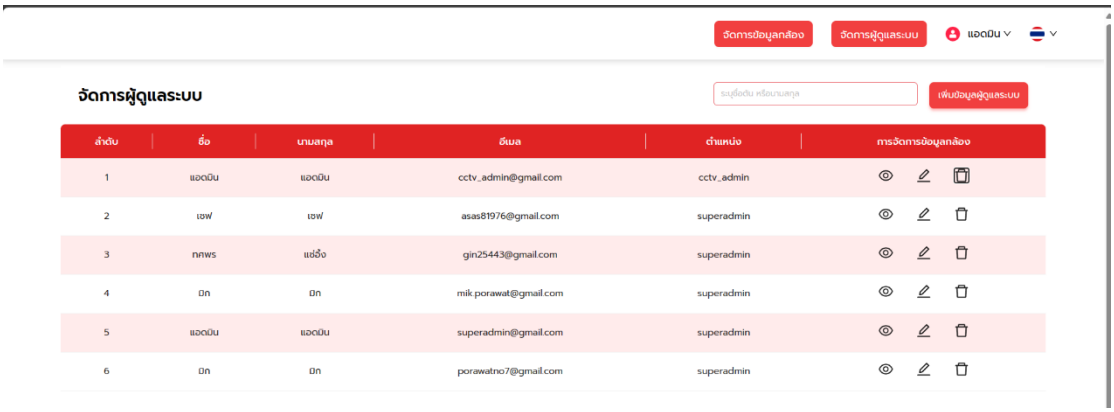


ภาพที่ 2.49 ตัวอย่างหน้าต่างการลบข้อมูลผู้ดูแลระบบ

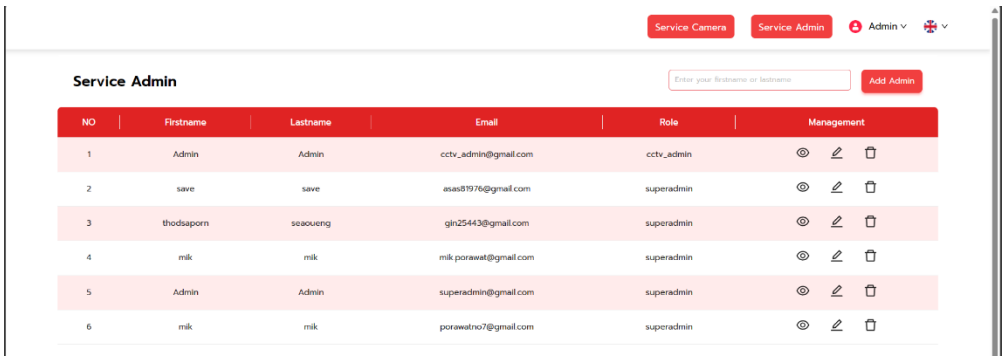
2.3 การใช้งาน 2 ภาษา

```
You, 2 weeks ago | 1 author (You)
import React, { createContext, useContext, useState, useEffect } from "react";
import i18n from "./i18n";

const LanguageContext = createContext();

export const LanguageProvider = ({ children }) => {
  const [selectedLanguage, setSelectedLanguage] = useState(
    localStorage.getItem("selectedLanguage") || "th"
  );

  // ฟังก์ชันเปลี่ยนภาษา
  const changeLanguage = (langKey) => {
    i18n.changeLanguage(langKey);
    setSelectedLanguage(langKey);
  };

  // เมื่อ selectedLanguage เปลี่ยน ให้บันทึกค่าลงใน localStorage
  useEffect(() => {
    localStorage.setItem("selectedLanguage", selectedLanguage);
  }, [selectedLanguage]);       You, 2 weeks ago • แก้ล่าสุด

  return (                                  (property) changeLanguage: (langKey: any) => void
    <LanguageContext.Provider value={{ selectedLanguage, changeLanguage }}>
      {children}
    </LanguageContext.Provider>
  );
};

export const useLanguage = () => {
  const context = useContext(LanguageContext);

  if (!context) {
    throw new Error("useLanguage must be used within a LanguageProvider");
  }

  return context;
};

```

ภาพที่ 2.50 ตัวอย่างโค้ดการใช้งาน 2 ภาษา



ภาพที่ 2.51 ตัวอย่างการแสดงข้อมูลตามภาษาเริ่มต้น



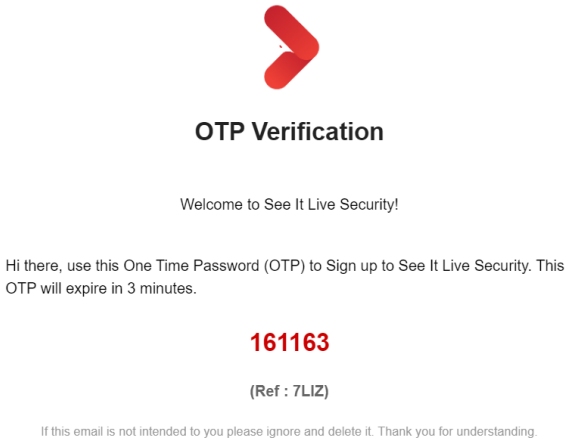
ภาพที่ 2.52 ตัวอย่างการแสดงข้อมูลเมื่อเลือกใช้ภาษาที่สอง

2.4 การใช้งาน otp

ในส่วนของระบบ otp จะใช้งานเมื่อ ผู้ดูแลระบบสูงสุดสร้างบัญชีให้กับผู้ดูแลโดย เมื่อสร้างบัญชีสำเร็จจะมีอีเมลเพื่อยืนยันโดยการกรอก otp 6 หลักเพื่อยืนยันตัวตน และเมื่อกดขอลืมรหัสผ่าน

ภาพที่ 2.53 ตัวอย่างการแสดงหน้าขอรับ Otp

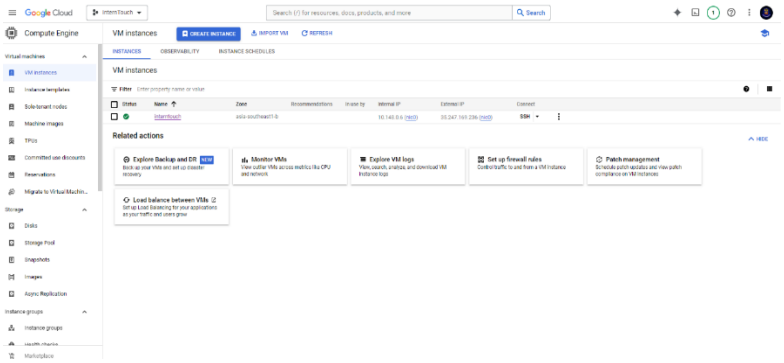
ภาพที่ 2.54 ตัวอย่างการแสดงหน้ากรอกรหัส otp



ภาพที่ 2.55 ตัวอย่าง Email otp

5 การใช้งานและการให้บริการ (ส่วนดูแลระบบหลังบ้าน)

นายธนกร ทองคล้าย ตำแหน่ง Development Operations งานที่ได้รับมอบหมาย การติดตั้งระบบต่างๆ ได้แก่ Docker,Jenkins,Portainer,Frigate,Uptime Kuma เพื่อให้ service ทุกอย่างของหน้าบ้านและหลังบ้านขึ้นบริการและใช้งานได้โดยขึ้นผ่าน google cloud และระบบปฏิบัติการ Linux Ubuntu server โดยสร้าง server จาก google cloud เพื่อที่จะ Deploy ส่วนหน้าบ้านและหลังบ้านและอื่นๆ

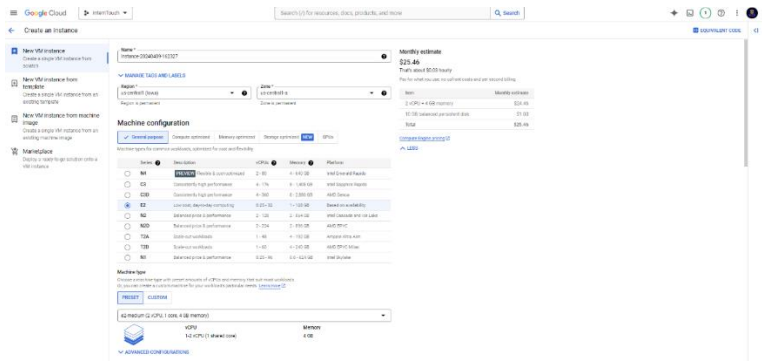


ภาพที่ 2.56 server ที่สร้างแล้วอยู่ใน google cloud

5.1 ติดตั้งระบบ Server Ubuntu version 22.04

5.1.1 สร้างเซิร์ฟเวอร์บน Google Cloud Platform

เข้าสู่ระบบ Google Cloud Console เลือกสร้างเซิร์ฟเวอร์โดยใช้ Compute Engine เลือกเครื่องคอมพิวเตอร์ที่ต้องการ เช่น CPU, RAM และอื่นๆที่อยากได้ และเลือกภูมิภาคที่ต้องการสร้างเซิร์ฟเวอร์ เลือกซอฟต์แวร์ของระบบปฏิบัติการ Ubuntu Server



ภาพที่ 2.57 สร้าง server ผ่าน google cloud

5.1.2 การติดตั้ง Docker

ติดตั้ง docker เพื่อช่วยในการสร้างและจัดการกับการทำงานของแอปพลิเคชันในรูปแบบของ containers หรือคอนเทนเนอร์ ทำให้ง่ายขึ้นในการสร้างและใช้งานแอปพลิเคชันต่าง ๆ

ติดตั้ง Docker บนเซิร์ฟเวอร์ Ubuntu Server โดยใช้คำสั่ง

```
sudo apt update
sudo apt install docker.io
sudo systemctl start docker
sudo systemctl enable docker
```

ภาพที่ 2.58 คำสั่งติดตั้ง docker บนเซิร์ฟเวอร์ Ubuntu

5.1.3 การติดตั้ง Jenkins

ติดตั้ง Jenkins เพื่อใช้ในการทำงานแบบ CI/CD (Continuous Integration/Continuous Deployment) ซึ่งช่วยให้การพัฒนาและการส่งมอบซอฟต์แวร์เป็นไปอย่างอัตโนมัติและต่อเนื่องได้ง่ายขึ้น ด้วยการใช้ Jenkins ทีมพัฒนาสามารถสร้าง, ทดสอบ, และส่งมอบโค้ดอัตโนมัติโดยมีการตรวจสอบและทดสอบอัตโนมัติทุกครั้งที่มีการเปลี่ยนแปลงในโค้ด ติดตั้ง Jenkins บนเซิร์ฟเวอร์ Ubuntu Server โดยใช้คำสั่ง

```
sudo apt update

sudo apt install openjdk-17-jdk

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]"
\
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update
```

ภาพที่ 2.59 คำสั่งติดตั้ง Jenkins บนเซิร์ฟเวอร์ Ubuntu

5.1.4 การติดตั้ง Portainer

Portainer เป็นเครื่องมือที่ใช้ในการจัดการ Docker ผ่านอินเทอร์เฟซกราฟิกที่ใช้งานง่ายและมีการตั้งค่าที่สะดวก ซึ่งช่วยให้ผู้ใช้สามารถควบคุมและจัดการ containers, images, volumes, networks และอื่น ๆ

ติดตั้ง Portainer บนเซิร์ฟเวอร์ Ubuntu Server โดยใช้คำสั่ง

```
sudo docker volume create portainer_data
docker run -d -p 8111:8000 -p 9443:9443 --name portainer --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ee:latest
docker ps -a
```

ภาพที่ 2.60 คำสั่งติดตั้ง Portainer บนเซิร์ฟเวอร์ Ubuntu

5.1.5 การติดตั้ง Frigate

ติดตั้ง Frigate เพื่อต้องการให้แสดงกล้องแบบเรียลทามและเพิ่มกล้องด้วยการใช้ API ของหลังบ้าน ใช้งานร่วมกับระบบกล้อง IP หรือกล้อง USB โดย Frigate มีความสามารถในการตรวจจับการเคลื่อนไหวที่อยู่ภายในภาพ

ติดตั้ง Frigate บนเซิร์ฟเวอร์ Ubuntu Server โดยใช้คำสั่ง

```
#install to server
version: "3.9"
services:
  frigate:
    container_name: frigate
    privileged: true
    restart: unless-stopped
    image: ghcr.io/blakeblackshear/frigate:stable
    shm_size: "64mb"
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /root/frigate/config/config.yml:/config/config.yml
      - /root/frigate/storage:/media/frigate
      - type: tmpfs # 1GB of memory
        target: /tmp/cache
        tmpfs:
          size: 1000000000
    ports:
      - "5000:5000" # Port used by the Web UI
      - "8554:8554" # RTSP feeds
      - "8555:8555/tcp" # WebRTC over tcp
      - "8555:8555/udp" # WebRTC over udp
    environment:
      FRIGATE_RTSP_PASSWORD: "admin123456789"
```

ภาพที่ 2.61 คำสั่งติดตั้ง Frigate บนเซิร์ฟเวอร์ Ubuntu

คำสั่ง config ของ Frigate เพื่อตั้งค่ากล้องและเพิ่ม ลบ แก้ไขกล้อง

```
mqtt:
  enabled: false
cameras:

  camera_3:
    ffmpeg:
      inputs:
        - path: rtsp://touch:Touch1234@f03e0ec59b99.sn.mynetname.net:5555
          roles:
            - detect
            - record
objects:
  track:
    - person
    - dog
    - car
detect:
  width: 640
  height: 480
  fps: 5
record:
  enabled: False
  expire_interval: 60
  retain:
    days: 0
    mode: all
snapshots:
  enabled: True
  clean_copy: False
  timestamp: False
  bounding_box: False
  crop: False
  height: 175
  retain:
    default: 1 # days
environment_vars:
  PLUS_API_KEY: c3e24240-80ec-4ad1-9030-65597cfc7b3f:0dfdb340f9735d6f9d1e37c32805ca75a37a9374
live:
  stream_name: camera_name
  height: 720
  quality: 8
detectors:
  cpu1:
    type: cpu
    num_threads: 2
```

ภาพที่ 2.62 คำสั่งตั้งค่า config ของ frigate บนเซิร์ฟเวอร์ Ubuntu

5.1.5 การติดตั้ง Uptime Kuma

Uptime Kuma เพื่อใช้ในการตรวจสอบและบันทึกข้อมูลเกี่ยวกับการใช้งานและประสิทธิภาพของเว็บไซต์หรือบริการต่าง ๆ ซึ่งมีไว้เพื่อการตรวจสอบว่าเว็บไซต์หรือบริการดังกล่าวมีการทำงานอย่างถูกต้องและเสถียรหรือไม่

ติดตั้ง Uptime Kuma บนเซิร์ฟเวอร์ Ubuntu Server โดยใช้คำสั่ง

```
docker run -d --name=uptimekuma -p 3001:3001 -e UPTIME_KUMA_USERNAME=admin -e
UPTIME_KUMA_PASSWORD=admin uptimekuma/uptimekuma
```

ภาพที่ 2.63 คำสั่งติดตั้ง Uptime Kuma บนเซิร์ฟเวอร์ Ubuntu

บทที่ 3
ผลการปฏิบัติงาน

รายงานวิจัยสหกิจศึกษา ณ บริษัท ทัชเทคโนโลยี จำกัด ระหว่างวันที่ 4 ธันวาคม พ.ศ.2566 ถึงวันที่ 29 มีนาคม พ.ศ 2567 มีรายละเอียดดังนี้

บทนำ

โครงการวิจัยฉบับนี้ มีวัตถุประสงค์พัฒนาเว็บไวต์ดูกล้องวงจรปิดและระบบจัดการกล้องวงจรปิด โดยการนำเครื่องมือ Frigate มาใช้ในการแปลงกล้องวงจรนำมาใช้แสดงบนเว็บไซต์ ระบบจัดการกล้องวงจรปิด เพิ่ม ลบ ดู แก้ไขกล้อง การทำเว็บไวต์ที่รับรองระบบ 2 ภาษา ไทย-อังกฤษ ระบบสมัครสมาชิก และ ระบบการส่ง otp เมื่อมีการขอลืมรหัส และการยืนยันตัวตน เมื่อสมัคร

ในส่วนของการออกแบบได้มีการนำ Api Specification มาใช้ในการออกแบบและกำหนด Api แต่ละเส้น Work Flow Diagram , Data Dictionary มาใช้งาน และ Mongodb ในการสร้างฐานข้อมูล

การปฏิบัติงานในครั้งนี้ คณะได้รับมอบหมายให้ปฏิบัติงานในตำแหน่ง Frontend , Backend , System Analysis และ Devops ในโครงการการจัดการกล้องวงจรปิดบนเว็บ โดยคณะผู้จัดทำจะทำงานร่วมมือกันในแต่ละตำแหน่ง ซึ่งจากการปฏิบัติสหกิจศึกษาในครั้งนี้ได้สร้างประโยชน์ทั้งทางด้านวิชาการในแง่การเพิ่มพูนความรู้ในสาขาวิชาเทคโนโลยีสารสนเทศ สาขาวิทยาการคอมพิวเตอร์ และในสาขาวิชาอื่นๆ อีกทั้งยังเป็นการนำความรู้ที่ได้เรียนมาในมหาวิทยาลัยมาประยุกต์ใช้ในการทำโครงการเป็นอย่างดี โดยผลการพัฒนาพบว่าระบบทำงานได้ตรงตามการทดสอบฟังก์ชั่นที่ได้กำหนด

แนวคิด ทฤษฎี และวรรณกรรมที่เกี่ยวข้อง

1. ทฤษฎีเกี่ยวกับ MongoDB

MongoDB เป็นฐานข้อมูลเอกสารแบบ NoSQL ที่เปิดเผยโค้ดแบบ open-source โดยไม่ใช้ภาษาคำสั่ง SQL และไม่เน้นการสร้างความสัมพันธ์ของข้อมูล เน้นการจัดเก็บข้อมูลในรูปแบบโครงสร้างที่เจ้าของฐานข้อมูลกำหนดขึ้นเอง โดยใช้ JSON (JavaScript Object Notation) เพื่อเก็บค่าเป็นคู่ key-value โดยจุดเด่นของ MongoDB อยู่ที่ความเร็วในการทำงานและการคิวรี่ข้อมูลที่เร็ว

ขึ้น การทำงานในส่วนของฐานข้อมูลลดลง แต่เน้นการทำงานในส่วนของโปรแกรมที่พัฒนาขึ้นมาแทน ฐานข้อมูลประเภทนี้เหมาะสำหรับข้อมูลขนาดใหญ่ที่ไม่ซับซ้อนและการทำงานในระบบ Real-Time ได้ดี

ภาพที่ 3.1 ฐานข้อมูล MongoDB

1) รูปแบบการจัดเก็บ

1.1) Collections: ข้อมูล document ใน MongoDB ถูกจัดเก็บใน Collections ซึ่งเปรียบเทียบได้กับ Table ใน Relational Database แต่ Collections ไม่จำเป็นต้องมี Schema เหมือนกันเพื่อบันทึกข้อมูล

1.2) Schemaless: ไม่จำเป็นต้องกำหนดโครงสร้างข้อมูลล่วงหน้า เช่น เราสามารถเพิ่มการเก็บข้อมูลเข้าไปใน Collection User ได้ตามต้องการ

2) ข้อดีของ MongoDB

2.1) MongoDB เป็นฐานข้อมูลแบบเอกสารที่ใช้ Json Style ในการเก็บข้อมูลโดยแต่ละเอกสารไม่จำเป็นต้องมีโครงสร้างข้อมูลเหมือนกัน

2.2) MongoDB ใช้ระบบการจัดการ memory แบบเดียวกับ cached memory ใน Linux ซึ่งให้ OS เป็นคนจัดการ Memory

2.3) ใช้ภาษา JavaScript เป็นคำสั่งในการจัดการข้อมูล

2.4) MongoDB เป็น Full Index สามารถค้นหาข้อมูลได้จากทุกส่วนของข้อมูล

2.5) รองรับการเพิ่มหรือลด field ข้อมูลได้อย่างรวดเร็วโดยไม่ต้องใช้คำสั่ง Alter Table

2.6) การอ่านและเขียนข้อมูลรวดเร็ว

2.7) การเขียนข้อมูลแบบ asynchronous ไม่ต้องรอ Insert เสร็จสิ้นก่อนที่จะทำงานต่อได้

2.8) มี Capped Collection ที่สามารถลบข้อมูลเก่าและเพิ่มข้อมูลใหม่ได้โดยอัตโนมัติ

2.9) การค้นหาข้อมูลรวดเร็ว

2.10) สามารถใช้เครื่อง server ที่ไม่ต้องมีคุณภาพสูงมากและแบ่งการทำงานได้หลายเครื่อง เพื่อประหยัดทรัพยากร

2.11) สามารถเขียนคำสั่งได้เป็นชุดโปรแกรมคล้ายกับ PL/SQL

3) ข้อเสียของ MongoDB

3.1) การใช้งาน MongoDB ในโปรเจกต์ที่มีการ JOIN ซับซ้อนอาจทำได้ยาก

3.2) การใช้พื้นที่จัดเก็บข้อมูลมากกว่า MySQL เนื่องจากไม่มี Schema และข้อมูลจะมี Schema อยู่ในแต่ละเอกสาร

3.3) การจัดการพื้นที่ disk เมื่อ disk เต็มอาจทำให้การลบข้อมูลไม่ทำให้ฐานข้อมูลเล็กลง ต้องทำการ compact เอง

3.4) การใช้งาน MongoDB เป็นฐานข้อมูลหลักอาจต้องมีการทำ Replication เพื่อเพิ่มความทนทานของข้อมูลในกรณีที่เครื่องดับไป โดย MongoDB มักจะเก็บข้อมูลใน Memory ไว้เป็นระยะเวลาหนึ่ง ซึ่งข้อมูลที่ยังค้างใน Memory และยังไม่ได้เขียนลง disk จะสูญหายในกรณีเครื่องดับ

2.ทฤษฎีเกี่ยวกับ Golang

ภาษา Go (หรือโดยทั่วไปเรียกว่า Golang) เป็นภาษาโปรแกรมมิ่งที่พัฒนาโดย Google เริ่มต้นโครงการเมื่อปี 2007 และเปิดตัวตัวแรกในปี 2009 เป็นภาษาที่ออกแบบมาเพื่อเป็นภาษาโปรแกรมเชิงพื้นฐานที่ใช้งานง่าย มีประสิทธิภาพสูง และเหมาะสำหรับการพัฒนาโปรแกรมขนาดใหญ่ มีความเข้าใจง่ายและโค้ดที่มีความสวยงาม

Go เน้นการงานขนาดใหญ่และโปรแกรมคอมพิวเตอร์ขนาดใหญ่ มีระบบการจัดการหน่วยความจำและการจัดการกับเส้นเวียน (Concurrency) ที่ช่วยให้ผู้พัฒนาสามารถเขียนโปรแกรมที่รับมือกับการทำงานพร้อมกันได้โดยมีประสิทธิภาพ

ภาพที่ 3.2 ภาษา Golang

คุณสมบัติและคุณประโยชน์สำคัญของ Go

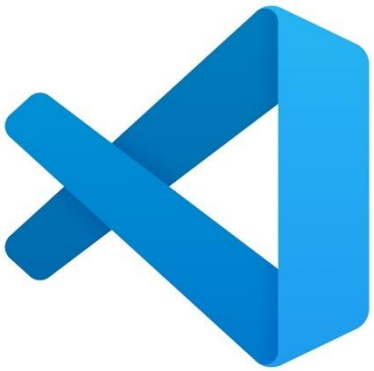
1. Concurrency: Go มีการรองรับการทำงานแบบ concurrent อย่างมีประสิทธิภาพด้วยโปรโตคอล Goroutines และ Channels ที่ทำให้การทำงานพร้อมกันของโค้ดเป็นเรื่องที่ง่ายและประสิทธิภาพสูง
2. Garbage Collection: Go มีระบบจัดการหน่วยความจำแบบอัตโนมัติ ทำให้นักพัฒนาไม่ต้องกังวลเกี่ยวกับการจัดการหน่วยความจำแบบชั้นต่ำ
3. สภาพแวดล้อมการพัฒนา (Development Environment): Go มีเครื่องมือสำหรับพัฒนาที่มีประสิทธิภาพและเป็นมาตรฐาน เช่น go fmt เพื่อจัดรูปโค้ด, go doc เพื่อสร้างเอกสาร API อัตโนมัติ, และ go test เพื่อทดสอบโค้ด
4. ความปลอดภัย: Go มีการเช็คชนิดของข้อมูลในช่วงเวลาคอมไพล์ และมีความปลอดภัยที่มากขึ้นเมื่อเทียบกับภาษาอื่น ๆ ที่ไม่มีการตรวจสอบชนิดข้อมูลในช่วงเวลาคอมไพล์
5. เร็วและมีประสิทธิภาพ: Go เร็วและมีประสิทธิภาพ ทำให้เป็นเลือกที่ดีสำหรับการพัฒนาโปรแกรมที่ต้องการประสิทธิภาพสูง
6. ไลบรารีมาตรฐาน (Standard Library): Go มีไลบรารีมาตรฐานที่มาพร้อมกับภาษา ซึ่งมีความหลากหลายและครอบคลุมความต้องการของนักพัฒนามากมาย ไม่ว่าจะเป็นเรื่องของการเขียนไฟล์, การทำงานกับเครือข่าย, การจัดการกับข้อมูล JSON, และอื่น ๆ
7. Cross-platform: Go รองรับการทำงานบนหลายแพลตฟอร์ม รวมถึง Windows, macOS, Linux, BSD, และอื่น ๆ
8. ร่วมมือกับภาษาอื่น: Go มีความสามารถในการทำงานร่วมกับภาษาอื่น ๆ อย่างง่ายดาย, เช่นการเรียกใช้งานฟังก์ชันจาก C, การใช้งาน HTTP และ RESTful services, และการใช้งานไลบรารีของภาษาอื่น ๆ

โดยสรุป, Go เป็นภาษาโปรแกรมมิ่งที่มีความรวดเร็วและมีประสิทธิภาพสูง มีความสามารถในการจัดการ concurrent อย่างมีประสิทธิภาพ และเหมาะสำหรับการพัฒนาโปรแกรมที่ต้องการประสิทธิภาพสูงและความปลอดภัย

3.ทฤษฎีเกี่ยวกับ Visual Studio Code

Visual Studio Code หรือ VSCode เป็นโปรแกรม Code Editor ที่ใช้ในการแก้ไขและปรับแต่งโค้ด โดยมาจากค่ายไมโครซอฟท์ มีการพัฒนาออกมาในรูปแบบ Open Source ซึ่งทำให้สามารถนำมาใช้งานได้ฟรี โปรแกรมนี้เป็นตัวแก้ไขซอร์สโค้ด (Source Code Editor) ที่ได้รับความนิยมมาก มีความเร็วและประสิทธิภาพในการทำงานที่ดี รวมถึงการรองรับหลายภาษา โปรเจคนี้เป็น

โปรเจค Open Source ของไมโครซอฟท์ที่ประสบความสำเร็จอย่างมาก รูปแบบการทำงานของ VSCode เหมือนกับ Text Editor ที่มีความสามารถเฉพาะในการทำแอปพลิเคชันให้ใช้งานง่ายขึ้น โดยเฉพาะฟีเจอร์การทำงานร่วมกับ Git ที่ทำให้สามารถดูและแก้ไขซอร์สโค้ดได้อย่างง่ายดาย Visual Studio Code เหมาะสำหรับนักพัฒนาโปรแกรมที่ต้องการใช้งานข้ามแพลตฟอร์ม รองรับการใช้งานทั้งบน Windows, macOS, และ Linux รวมถึงภาษา JavaScript, TypeScript, และ Node.js และสามารถเชื่อมต่อกับ Git ได้โดยง่ายไม่ซับซ้อน นอกจากนี้ยังมีเครื่องมือส่วนขยายต่าง ๆ ให้เลือกใช้อย่างมากมาย เช่น การเปิดใช้งานภาษาอื่น ๆ เช่น C++, C#, Java, Python, PHP หรือ Go, Themes, Debugger, และ Commands เป็นต้น



ภาพที่ 3.3 โปรแกรม VSCode Code editing

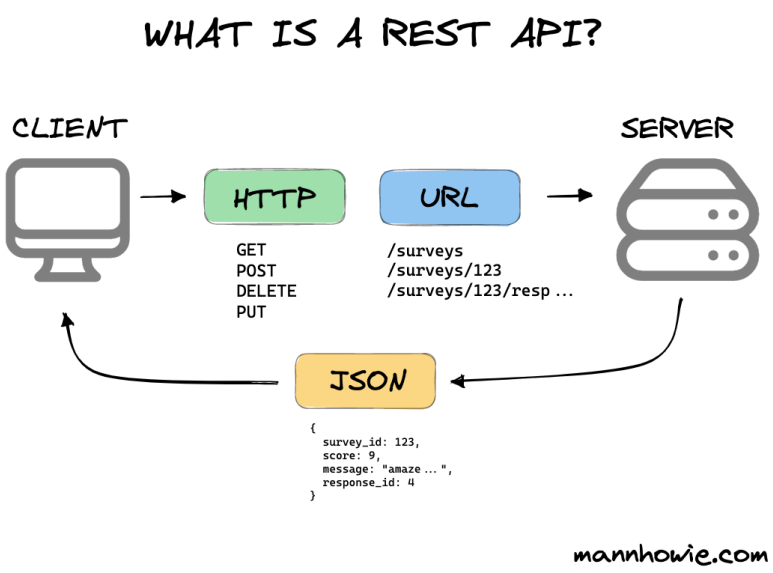
ความแตกต่างระหว่าง VSCode และ Visual Studio คือ VSCode ได้ทำการตัดส่วนของ GUI designer ออกไป เหลือแต่เพียงตัว Editor เท่านั้น ทำให้โปรแกรมนั้นมีน้ำหนักเบากว่า Visual Studio อย่างมาก

4.ทฤษฎีเกี่ยวกับ RESTful API

ทฤษฎีเกี่ยวกับ RESTful API นั้นเกิดจากแนวคิดของ Representational State Transfer (REST) ซึ่งเป็นรูปแบบการออกแบบและการสื่อสารระหว่างระบบคอมพิวเตอร์ โดย Roy Fielding ในวิทยานิพนธ์ของเขา "Architectural Styles and the Design of Network-based Software Architectures" ที่มหาวิทยาลัย California ในปี 2000 ซึ่งได้เผยแพร่บนอินเทอร์เน็ต เป็นที่รู้จักกันอย่างกว้างขวางในอุตสาหกรรมการพัฒนาระบบเครือข่ายและเว็บแอปพลิเคชันในปัจจุบัน

RESTful API เป็นแนวคิดการออกแบบและการพัฒนา API ที่เรียกใช้รูปแบบและพฤติกรรมของ REST โดยมีลักษณะเด่นคือการใช้ HTTP methods เพื่อดำเนินการต่าง ๆ ตามสถานะของข้อมูล

(state) และการใช้ URI (Uniform Resource Identifier) เพื่อระบุทรัพยากร (resource) ที่ต้องการเข้าถึง ซึ่งทำให้ API มีความสะดวกและมีประสิทธิภาพในการใช้งาน



ภาพที่ 3.4 ส่วนประกอบ RESTful API

REST ประกอบด้วยหลักการสำคัญต่อไปนี้

1.Client-Server Architecture (โครงสร้างระหว่างลูกค้าและเซิร์ฟเวอร์): REST มีการแยกแยะระหว่างลูกค้า (client) และเซิร์ฟเวอร์ (server) ทำให้สามารถพัฒนาและปรับปรุงแต่ละส่วนได้อิสระ โดยที่ไม่มีผลกระทบกับอีกฝั่งหนึ่ง

2.Statelessness (การไม่เก็บสถานะ): การสื่อสารระหว่างลูกค้าและเซิร์ฟเวอร์ไม่ควรจะเกี่ยวข้องกับข้อมูลสถานะ ทำให้การสื่อสารเป็นเหมือนแก่ลูกค้าใหม่ทุกครั้ง

3.Cacheability (ความสามารถในการใช้แคช): REST สนับสนุนการใช้งานแคชเพื่อเพิ่มประสิทธิภาพและลดการโหลดข้อมูลที่ไม่จำเป็น

4.Uniform Interface (อินเทอร์เฟซที่เป็นมาตรฐาน): มีการใช้แบบสองที่สำคัญคือ URI เพื่อระบุทรัพยากรและ HTTP methods เพื่อดำเนินการต่าง ๆ กับทรัพยากรนั้น ๆ

5.Layered System (ระบบชั้น): สามารถแบ่งระบบออกเป็นชั้นโดยที่แต่ละชั้นไม่รู้จักชั้นอื่น ๆ นอกจากชั้นที่อยู่ติดต่อกับตนเอง

6.Code on Demand (โค้ดตามคำสั่ง): ตัวลูกค้าสามารถร้องขอการดาวน์โหลดโค้ดเพิ่มเติมจากเซิร์ฟเวอร์เพื่อประมวลผลเพิ่มเติมบนเครื่องลูกค้าได้ (ตัวอย่างเช่น JavaScript)

การทำงานตามทฤษฎีนี้ช่วยให้ RESTful API เป็นแบบออกแบบที่มีประสิทธิภาพและยืดหยุ่น ทำให้เหมาะสมสำหรับการพัฒนาแอปพลิเคชันเว็บและเครือข่ายที่ใหญ่ขึ้นโดยเฉพาะในระบบที่มีการแจกแจงข้อมูลอย่างกว้างขวางและมีการประมวลผลที่สูงขึ้นในยุคปัจจุบัน

อีกสิ่งหนึ่งที่ควรพูดถึงเพิ่มเติมเกี่ยวกับ RESTful API คือหลักการของการออกแบบและการใช้งานของแต่ละเมธอด HTTP ซึ่งมีความสำคัญต่อการพัฒนา API:

GET: ใช้สำหรับการเรียกข้อมูลจากเซิร์ฟเวอร์โดยทั่วไป โดยไม่มีการเปลี่ยนแปลงสถานะของเซิร์ฟเวอร์หรือข้อมูล

POST: ใช้สำหรับการส่งข้อมูลใหม่ไปยังเซิร์ฟเวอร์ เช่น การสร้างข้อมูลใหม่

PUT: ใช้สำหรับการอัปเดตข้อมูลที่มีอยู่ในเซิร์ฟเวอร์ โดยระบุทรัพยากรที่ต้องการอัปเดต

DELETE: ใช้สำหรับการลบข้อมูลที่มีอยู่ในเซิร์ฟเวอร์ โดยระบุทรัพยากรที่ต้องการลบ

PATCH: ใช้สำหรับการอัปเดตเฉพาะส่วนหนึ่งของข้อมูลที่มีอยู่ในเซิร์ฟเวอร์ โดยระบุทรัพยากรที่ต้องการอัปเดตและข้อมูลที่ต้องการเปลี่ยนแปลง

HEAD: ใช้สำหรับการขอข้อมูลเฉพาะส่วนหัว (header) ของการตอบกลับจากเซิร์ฟเวอร์ โดยไม่ร้องขอเนื้อหาข้อมูล

การใช้เมธอด HTTP อย่างมีประสิทธิภาพและเหมาะสมจะช่วยให้ RESTful API มีการทำงานอย่างเป็นระบบและมีประสิทธิภาพในการให้บริการข้อมูลแก่ลูกค้าได้ดีขึ้น

5.ทฤษฎีเกี่ยวกับ Postman

Postman API คือเครื่องมือที่ช่วยในการทดสอบ APIs หรือ Application Programming Interfaces โดยออกแบบมาเพื่อช่วยให้ผู้พัฒนาสามารถทำการทดสอบ APIs ได้อย่างสะดวกและรวดเร็ว โดยไม่จำเป็นต้องเขียนโปรแกรมหรือสร้างโค้ดขึ้นมาเองทั้งหมด เครื่องมือนี้มีการให้บริการในรูปแบบของแอพพลิเคชันเว็บ ซึ่งมีการจัดเก็บข้อมูลการทดสอบ การสร้างสคริปต์การทดสอบอัตโนมัติ และการจัดการการทดสอบอื่นๆ ทั้งนี้ Postman API เป็นเครื่องมือที่เป็นที่นิยมในการทำงานด้านการพัฒนาซอฟต์แวร์และการทดสอบ APIs ในวงการไอทีและธุรกิจออนไลน์อย่างแพลตฟอร์มหรือเว็บเซอร์วิสต่างๆ อันที่จริง API ของ Postman เองก็ถือเป็นหนึ่งในตัวอย่างของ API ที่ถูกพัฒนาขึ้นเพื่อให้บริการในการทำงานนี้ด้วยตนเองด้วยความสามารถที่มีความสมบูรณ์และยืดหยุ่นสูง โดยสามารถทดสอบ APIs ต่างๆ ได้ทั้งในรูปแบบของ HTTP requests เช่น GET, POST, PUT, DELETE และ

อื่นๆ รวมถึงการจัดการข้อมูลและการทำงานร่วมกับตัวอื่นๆ ในกระบวนการพัฒนาซอฟต์แวร์แบบทีมได้อย่างมีประสิทธิภาพและสะดวกสบาย

ภาพที่ 3.5 โปรแกรม Postman

ความสามารถและคุณสมบัติ

1.ความสามารถในการทดสอบ APIs หลากหลายรูปแบบ: Postman ช่วยให้ผู้ใช้สามารถทดสอบ APIs ได้ในรูปแบบต่างๆ เช่น HTTP, REST, SOAP, GraphQL และอื่นๆ ทำให้ผู้ใช้สามารถทดสอบ APIs ของตนหรือ APIs จากบริการอื่นๆ ได้อย่างหลากหลายและยืดหยุ่น.

2.การจัดการและเก็บสคริปต์การทดสอบ: Postman ช่วยให้ผู้ใช้สามารถเขียนและจัดการสคริปต์การทดสอบ API ได้อย่างสะดวก รวมถึงการจัดการข้อมูลทดสอบ การจัดการการรับส่งข้อมูลระหว่าง request และ response และการสร้างรูปแบบการทดสอบที่เป็นไปตามความต้องการของผู้ใช้.

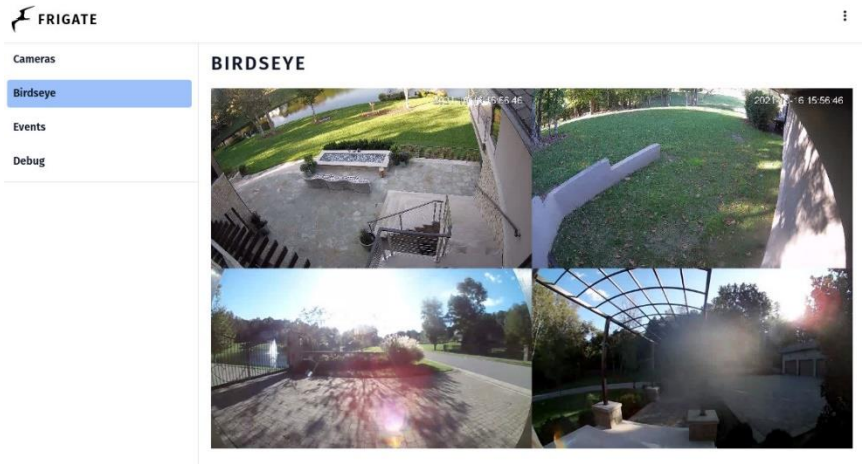
3.การแบ่งปันและการทำงานร่วมกันในทีม: Postman มีความสามารถในการแบ่งปันสคริปต์การทดสอบ ข้อมูลการทดสอบ และการตั้งค่า APIs ซึ่งทำให้ทีมสามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ โดยไม่จำเป็นต้องแชร์โค้ดหรือข้อมูลข้างเคียง.

4.ความสามารถในการจัดการเรียกใช้ APIs ในระยะยาว: Postman ช่วยให้ผู้ใช้สามารถจัดการกับ APIs ในระยะยาวได้ง่ายขึ้น ด้วยการเก็บรวบรวมประวัติการใช้งาน APIs การตรวจสอบความถูกต้องของ APIs และการจัดการความสามารถในการทดสอบที่ได้ผ่านการพัฒนาอย่างมีประสิทธิภาพ.

5.ระบบความปลอดภัย: Postman มีระบบความปลอดภัยที่ช่วยให้ผู้ใช้สามารถจัดการและควบคุมการเข้าถึง APIs และข้อมูลการทดสอบได้อย่างมีความปลอดภัย.

6.ทฤษฎีเกี่ยวกับ FrigateNVR

"Frigate NVR" เป็นคำย่อที่ใช้ในวงการเทคโนโลยีและระบบการรักษาความปลอดภัย เพื่ออธิบายระบบการบันทึกวิดีโอ (Video Recording System) ที่ใช้ในการจัดเก็บภาพจากกล้องวงจรปิด (CCTV) หรือกล้องรักษาความปลอดภัยอื่น ๆ โดยที่คำว่า "NVR" หมายถึง "Network Video Recorder" ซึ่งเป็นอุปกรณ์ที่ใช้ในการบันทึกและจัดเก็บวิดีโอจากกล้องระยะไกลผ่านระบบเครือข่าย (Network). ส่วนคำว่า "Frigate" อาจเป็นชื่อของแบรนด์หรือรุ่นของ NVR ที่เฉพาะเจาะจงมากกว่านี้ โดยบางบริษัทหรือผู้ผลิตอาจใช้ชื่อสินค้าเพื่อแสดงถึงคุณสมบัติหรือความสามารถเฉพาะที่พวกเขามี อย่างไรก็ตาม ในกรณีนี้ "Frigate NVR" อาจหมายถึง NVR ที่มีคุณสมบัติหรือเทคโนโลยีที่เน้นไปที่การรักษาความปลอดภัยในระดับสูงหรือมีความยืดหยุ่นในการปรับแต่งตามความต้องการของผู้ใช้งานโดยเฉพาะ



ภาพที่ 3.6 Frigate NVR

ความสามารถและคุณสมบัติ

1.คุณภาพวิดีโอสูง: รองรับบันทึกวิดีโอคุณภาพสูงและ 4K.

2.ความปลอดภัย: มีระบบเข้ารหัสข้อมูลและการจัดการผู้ใช้งานเพื่อความปลอดภัย.

3.ยืดหยุ่นในการใช้งาน: ง่ายต่อการติดตั้งและใช้งาน ลดความซับซ้อนในการบริหารจัดการ.

4.ประหยัดเวลาและค่าใช้จ่าย: ช่วยลดค่าใช้จ่ายในระยะยาวด้วยคุณสมบัติการบำรุงรักษาและปรับปรุงที่น้อยลง.

5.รองรับมาตรฐานและกล้อง IP: สามารถใช้กับกล้องที่มีมาตรฐานและกล้อง IP ได้.

6.การเชื่อมต่อระบบอื่น ๆ: สามารถเชื่อมต่อกับระบบอื่น เช่น ระบบการแจ้งเตือนหรือระบบควบคุมการเข้าถึงได้.

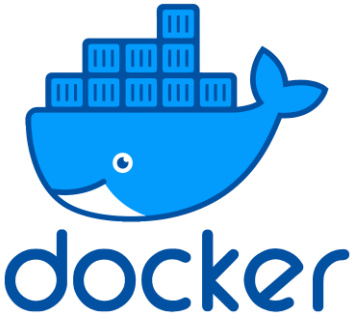
7. ทฤษฎีเกี่ยวกับ Gin framework

Gin framework เป็นเฟรมเวิร์ก (framework) สำหรับพัฒนาเว็บแอปพลิเคชัน (web applications) และเว็บเซอร์วิส (web services) ด้วยภาษา Go (หรือ Golang) ซึ่งเป็นภาษาโปรแกรมมิ่งที่พัฒนาขึ้นโดย Google โดย Gin framework มีเป้าหมายที่จะทำให้การพัฒนาเว็บแอปพลิเคชันด้วยภาษา Go เป็นไปอย่างรวดเร็วและง่ายดายมากขึ้น โดยมีการออกแบบให้มีประสิทธิภาพสูง การใช้งานง่าย และมีความยืดหยุ่นในการปรับแต่งตามความต้องการของผู้ใช้งานแต่ละคน นอกจากนี้ Gin framework ยังมีความสามารถในการจัดการ HTTP requests, routing, middleware, การสร้าง API, และฟีเจอร์อื่น ๆ อีกมากมายที่จำเป็นสำหรับการพัฒนาเว็บแอปพลิเคชันและเว็บเซอร์วิสใน Go

ภาพที่ 3.7 Gin framework ภาษา Go

8. ทฤษฎีเกี่ยวกับ Docker

Docker เป็นเครื่องมือแบบ open-source และแพลตฟอร์มซอฟต์แวร์ที่ช่วยให้คุณสามารถสร้าง ทดสอบ และติดตั้งแอปพลิเคชันได้อย่างรวดเร็วและมีประสิทธิภาพ ด้วยการบรรจุซอฟต์แวร์ลงในหน่วยมาตรฐานที่เรียกว่าคอนเทนเนอร์ (Container) ที่มีทุกสิ่งที่ต้องการในการเรียกใช้งานแอปพลิเคชัน รวมถึงไลบรารี เครื่องมือสำหรับระบบ โค้ด และการตั้งเวลารัน



ภาพที่ 3.8 โปรแกรม Docker

การใช้ Docker ช่วยให้คุณสามารถติดตั้งและปรับขนาดแอปพลิเคชันได้ให้เหมาะกับทุกสภาพแวดล้อม (Deploy) และมั่นใจได้ว่าโค้ดของคุณจะเรียกใช้ได้อย่างรวดเร็ว ด้วยความสามารถในการรองรับการติดตั้งและใช้งานบนระบบปฏิบัติการที่หลากหลาย เช่น Linux, Windows, และ MAC

ด้วยคุณสมบัติและประโยชน์ที่มาพร้อมกับ Docker ทำให้มันเป็นเครื่องมือที่ได้รับความนิยมอย่างแพร่หลายในวงกว้างและเป็นส่วนสำคัญของโลกของการพัฒนาซอฟต์แวร์ ช่วยเพิ่มความสามารถในการจัดการและปรับปรุงระบบอย่างมีประสิทธิภาพ ให้กับนักพัฒนาและองค์กรทั่วไปได้อย่างมีประสิทธิภาพและเป็นระบบ

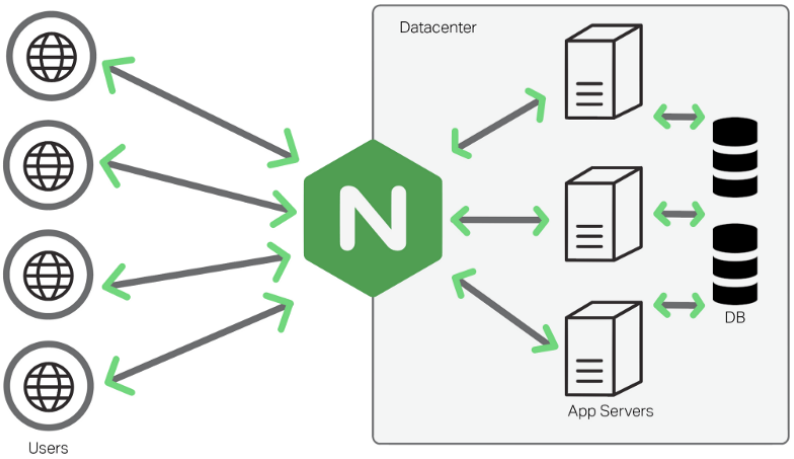
9. ทฤษฎีเกี่ยวกับ Jenkins

Jenkins เป็นเครื่องมือที่ใช้สำหรับการทำ Continuous Integration (CI) และ Continuous Deployment (CD) ซึ่งเป็นกระบวนการในการพัฒนาซอฟต์แวร์ที่ช่วยให้ทีมพัฒนาซอฟต์แวร์สามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ โดย Jenkins ช่วยในการสร้างบิลด์ (build), ทดสอบ (test), และส่งมอบ (deploy) ซอฟต์แวร์โดยอัตโนมัติ ผ่านกระบวนการอัตโนมัติหลายขั้นตอนที่กำหนดไว้ล่วงหน้า ซึ่งทำให้การพัฒนาและการประสานงานระหว่างทีมมีประสิทธิภาพมากขึ้น โดย Jenkins เป็นโปรแกรมที่เปิดซอร์ส (open-source) และมีการพัฒนาต่อยอดโดยชุมชนของนักพัฒนาทั่วโลก ทำให้มีความยืดหยุ่นสูงในการปรับแต่งและการใช้งานได้ในหลากหลายสถานการณ์ของโครงการและองค์กรต่าง ๆ ทั้งในรูปแบบการติดตั้งบนเซิร์ฟเวอร์ส่วนตัว หรือในรูปแบบการใช้บริการคลาวด์ผ่าน Jenkins CI/CD บนระบบคลาวด์อื่น ๆ เช่น AWS, Google Cloud Platform, หรือ Microsoft Azure ได้อย่างสะดวกสบาย

ภาพที่ 3.9 Jenkins

10. ทฤษฎีเกี่ยวกับ Nginx

Nginx เป็นเว็บเซิร์ฟเวอร์ที่เป็นโครงการเปิดอินโกเนียส (open-source) ซึ่งถูกออกแบบมาเพื่อให้บริการเว็บไซต์และแอปพลิเคชันออนไลน์อย่างมีประสิทธิภาพ โดยส่วนใหญ่นิยมใช้เป็นเว็บเซิร์ฟเวอร์ที่ให้บริการและจัดการ HTTP, HTTPS, SMTP, POP3, และ IMAP รวมถึงการทำงานเป็นโปรกิจซี (reverse proxy) และโปรกิจซีโปรเซส (load balancer) อีกด้วย Nginx เป็นทางเลือกที่นิยมสำหรับการจัดการการจัดเก็บและส่งสิ่งของต่างๆ ในเว็บแอปพลิเคชันออนไลน์ในระดับองค์กรหรือระดับขององค์กรขนาดใหญ่ ด้วยความเสถียร และประสิทธิภาพที่สูง โดยมีการใช้งานกับเว็บไซต์ที่มีการใช้งานสูงโดยส่วนใหญ่อย่างต่อเนื่อง ตัวอย่างเช่นเว็บไซต์ข่าว บริการเกมออนไลน์ แอปพลิเคชันการแพร่กระจายเนื้อหา (CDN) และอื่นๆ อีกมากมาย โดย Nginx เป็นโปรแกรมที่ได้รับความนิยมมากในช่วงหลายปีที่ผ่านมา เนื่องจากความเสถียร และประสิทธิภาพที่ดีในการจัดการการร้องขอ (request) และการทำงานในสภาพแวดล้อมที่มีการเพิ่มเติมของโอกาสของการใช้งานสูง และมีฟีเจอร์การปรับเปลี่ยนและตั้งค่าที่ยืดหยุ่นได้ในสถานการณ์ที่ต่างกัน



ภาพที่ 3.10 NginX

11. ทฤษฎีเกี่ยวกับ Gitlab

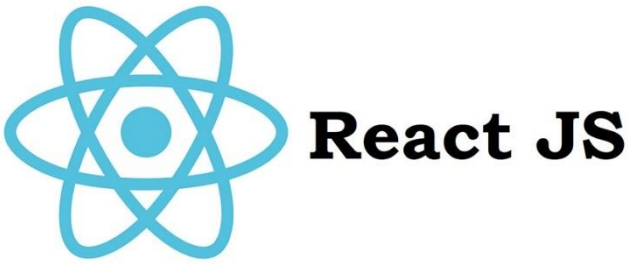
GitLab เป็นแพลตฟอร์มการพัฒนาซอฟต์แวร์ที่ใช้ระบบ Git ในการจัดการรหัสซอร์สของโครงการซอฟต์แวร์ต่าง ๆ โดย GitLab มีความคล้ายกับ GitHub ซึ่งเป็นระบบ Git Hosting ที่แพร่หลายที่สุดในโลก แต่ GitLab นอกจากจะมีคุณสมบัติเสมือน GitHub แล้ว ยังมีคุณสมบัติอื่น ๆ ที่ช่วยให้การจัดการโครงการซอฟต์แวร์และการพัฒนาเป็นเรื่องที่สะดวกสบายมากยิ่งขึ้น เช่น การจัดการ CI/CD (Continuous Integration/Continuous Deployment), ระบบปัญหา (Issue

tracking), การจัดการงาน (Project management), การเขียนคำแนะนำ (Wiki), และอื่น ๆ อีกมากมาย นอกจากนี้ GitLab ยังมีเวอร์ชันที่สามารถติดตั้งได้บนเซิร์ฟเวอร์ของคุณเองซึ่งช่วยให้คุณควบคุมข้อมูลของโครงการได้ด้วยตัวเองและมีความเป็นส่วนตัวสูงขึ้นในกรณีที่คุณต้องการความปลอดภัยและความควบคุมที่สูงขึ้นในการพัฒนาซอฟต์แวร์ของคุณ รวมทั้งมี GitLab.com ซึ่งเป็นเวอร์ชันที่ให้บริการแบบเว็บฟรีและแบบเสียค่าใช้จ่ายสำหรับการจัดการโครงการซอฟต์แวร์ของคุณผ่านอินเทอร์เน็ต

ภาพที่ 3.11 Gitlab

12. ทฤษฎีเกี่ยวกับ ReactJS

React เป็นไลบรารี JavaScript ที่ใช้สำหรับการพัฒนาแอปพลิเคชันเว็บ โดย React มุ่งเน้นไปที่การสร้างอินเตอร์เฟซผู้ใช้ (UI) ที่มีประสิทธิภาพสูง โดยมีความยืดหยุ่นและสามารถใช้งานได้ง่าย ซึ่งเป็นที่นิยมในการพัฒนาเว็บแอปพลิเคชันที่มีการปรับปรุงและการสร้างส่วนของหน้าเว็บอย่างต่อเนื่อง



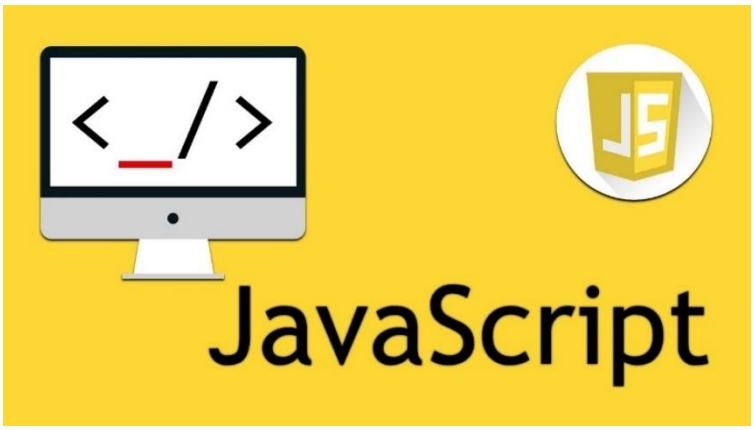
ภาพที่ 3.12 ReactJS

React ใช้งานร่วมกับ JavaScript และ JSX (JavaScript XML) เพื่อสร้าง Component ซึ่งเป็นส่วนประกอบของ UI แต่ละอัน แต่ละ Component จะมีสถานะของตัวเองและสามารถถูกนำมาประกอบกันเป็นโครงสร้างของ UI ที่สมบูรณ์ได้

React มีคุณสมบัติที่มีประสิทธิภาพเช่น Virtual DOM ที่ช่วยเพิ่มประสิทธิภาพในการแสดงผล UI โดยลดการเข้าถึง DOM จริงของเว็บไซต์ React ยังมีระบบการจัดการสถานะ (state) และอื่น ๆ ที่ช่วยให้การจัดการข้อมูลและการอัพเดต UI เป็นเรื่องง่าย นอกจากนี้ React ยังมีชุดเครื่องมือที่หลากหลายเช่น React Router สำหรับการจัดการเส้นทางของแอปพลิเคชัน React และ Redux สำหรับการจัดการสถานะของแอปพลิเคชันใหญ่ ๆ อีกด้วย

13. ทฤษฎีเกี่ยวกับ JavaScript

JavaScript ถูกสร้างขึ้นโดย Brendan Eich ในปี 1995 ขณะที่เขาทำงานที่ Netscape Communications Corporation ในตอนแรกมันถูกเรียกว่า "LiveScript" แต่ต่อมาได้เปลี่ยนชื่อเป็น "JavaScript" เพื่อให้สอดคล้องกับความนิยมของ Java ในช่วงเวลานั้น มีมาตรฐานเชิงอุตสาหกรรมที่เรียกว่า ECMAScript ซึ่งเป็นส่วนใหญ่ของภาษา JavaScript ที่ถูกจับจัดเก็บไว้ในเอกสารมาตรฐานของ ECMA International ในปัจจุบันเวอร์ชันล่าสุดคือ ECMAScript 2021



ภาพที่ 3.13 JavaScript

ภาษา JavaScript เป็นภาษาโปรแกรมมิ่งที่ใช้สำหรับพัฒนาเว็บและแอปพลิเคชันที่ทำงานบนเว็บเบราว์เซอร์ นอกจากนี้ยังสามารถใช้งานได้ในส่วนของเซิร์ฟเวอร์และแอปพลิเคชันทั่วไปด้วย เป็นภาษาที่เป็นมาตรฐานในการพัฒนาเว็บและมีความสามารถในการปรับแต่งและประมวลผลเนื้อหาบนหน้าเว็บได้อย่างหลากหลาย นอกจากนี้ JavaScript เป็นภาษาที่รองรับการทำงานแบบอินเตอร์

แอคทีฟ (Interactive) ซึ่งทำให้เว็บไซต์มีประสิทธิภาพและประสบการณ์ของผู้ใช้ที่ดีขึ้นได้ โดยปกติ JavaScript ใช้งานร่วมกับ HTML และ CSS เพื่อสร้างเว็บไซต์และแอปพลิเคชันที่มีประสิทธิภาพและประสบการณ์ผู้ใช้ที่ดีในเว็บบราวเซอร์ต่างๆ นอกจากนี้ยังมีการใช้ JavaScript ในการพัฒนาแอปพลิเคชันมือถือ (Mobile Apps) และแอปพลิเคชันด้านเซิร์ฟเวอร์ด้วย JavaScript Frameworks ต่างๆ เช่น Node.js ซึ่งเป็นเฟรมเวิร์คที่ทำให้ JavaScript สามารถใช้งานได้ในเซิร์ฟเวอร์เช่นเดียวกับการพัฒนาแอปพลิเคชันแบบ Client-Side ได้ด้วย นับเป็นหนึ่งในภาษาโปรแกรมมิ่งที่ได้รับความนิยมและใช้งานกันอย่างแพร่หลายในปัจจุบัน ด้วยความสามารถในการใช้งานทั้งในด้านของเว็บและแอปพลิเคชัน รวมถึงความยืดหยุ่นในการปรับแต่งและประมวลผลข้อมูลต่างๆ ทำให้ JavaScript เป็นภาษาที่น่าสนใจและนับถือในการพัฒนาโปรแกรมมิ่งทั้งในมิติต่างๆ ของแพลตฟอร์มเว็บและแอปพลิเคชันทั่วๆ ไป

14. ทฤษฎีเกี่ยวกับ CSS

CSS ย่อมาจาก Cascading Style Sheet มักเรียกโดยย่อว่า "สไตล์ชีต" คือภาษาที่ใช้เป็นส่วนของการจัดรูปแบบการแสดงผลเอกสาร HTML โดยที่ CSS กำหนดกฎเกณฑ์ในการระบุรูปแบบ (หรือ "Style") ของเนื้อหาในเอกสาร อันได้แก่ สีของข้อความ สีพื้นหลัง ประเภทตัวอักษร และการจัดวางข้อความ ซึ่งการกำหนดรูปแบบ หรือ Style นี้ใช้หลักการของการแยกเนื้อหาเอกสาร HTML ออกจากคำสั่งที่ใช้ในการจัดรูปแบบการแสดงผล กำหนดให้รูปแบบของการแสดงผลเอกสาร ไม่ขึ้นอยู่กับเนื้อหาของเอกสาร เพื่อให้ง่ายต่อการจัดรูปแบบการแสดงผลลัพธ์ของเอกสาร HTML โดยเฉพาะในกรณีที่มีการเปลี่ยนแปลงเนื้อหาเอกสารบ่อยครั้ง หรือต้องการควบคุมให้รูปแบบการแสดงผลเอกสาร HTML มีลักษณะของความสม่ำเสมอทั่วกันทุกหน้าเอกสารภายในเว็บไซต์เดียวกัน โดยกฎเกณฑ์ในการกำหนดรูปแบบ (Style) เอกสาร HTML ถูกเพิ่มเข้ามาครั้งแรกใน HTML 4.0 เมื่อปีพ.ศ. 2539 ในรูปแบบของ CSS level 1 Recommendations ที่กำหนดโดย องค์กร World Wide Web Consortium หรือ W3C

ภาพที่ 3.14 CSS

ประโยชน์ของ CSS

1.CSS มีคุณสมบัติมากกว่า tag ของ html เช่น การกำหนดกรอบให้ข้อความ รวมทั้งสีรูปแบบของข้อความที่กล่าวมาแล้ว

2.CSS นั้นกำหนดที่ต้นของไฟล์ html หรือตำแหน่งอื่น ๆ ก็ได้ และสามารถมีผล กับเอกสารทั้งหมด หมายถึงกำหนด ครั้งเดียวจุดเดียวก็มีผลกับการแสดงผลทั้งหมด ทำให้เวลาแก้ไขหรือปรับปรุงทำได้สะดวก ไม่ต้องไล่ตามแก้ tag ต่างๆ ทั่วทั้งเอกสาร

3.CSS สามารถกำหนดแยกไว้ต่างหากจาก ไฟล์เอกสาร html และสามารถนำมาใช้ร่วม กับเอกสารหลายไฟล์ได้ การแก้ไขก็แก้เพียง จุดเดียวก็มีผลกับเอกสารทั้งหมด

CSS กับ HTML / XHTML นั้นทำหน้าที่คนละอย่างกัน โดย HTML / XHTML จะทำหน้าที่ในการวางโครงร่างเอกสารอย่างเป็นรูปแบบ ถูกต้อง เข้าใจง่าย ไม่เกี่ยวข้องกับการแสดงผล ส่วน CSS จะทำหน้าที่ในการตกแต่งเอกสารให้สวยงาม เรียกได้ว่า HTML /XHTML คือส่วน coding ส่วน CSS คือส่วน design

15. ทฤษฎีเกี่ยวกับ HTML

HTML (Hypertext Markup Language) เป็นภาษาหลักที่ใช้ในการสร้างไฟล์เว็บเพจ โดยมีแนวคิดมาจากการสร้างเอกสารไฮเปอร์เท็กซ์ (Hypertext Document) ซึ่งพัฒนาขึ้นมาจากภาษา SGML (Standard Generalized Markup Language) โดย Tim Berners-Lee เป็นภาษามาตรฐานที่ใช้พัฒนาเอกสารในรูปแบบของเว็บเพจเพื่อเผยแพร่บนระบบเครือข่ายอินเทอร์เน็ต โครงสร้างการเขียนของ HTML อาศัยตัวกำกับที่เรียกว่าแท็ก (Tag) เพื่อควบคุมการแสดงผลของข้อความ, รูปภาพ, หรือวัตถุอื่น ๆ และเอกสาร HTML นั้นสามารถใช้งานได้ผ่านโปรแกรมเว็บบราวเซอร์ เช่น Mozilla Firefox, Opera, Netscape Navigator, และ Internet Explorer เป็นต้น

ภาพที่ 3.15 โครงสร้างภาษา HTML

ในปัจจุบัน HTML เป็นมาตรฐานหนึ่งของ ISO ซึ่งจัดการโดย World Wide Web Consortium (W3C) ซึ่งผลักดันรูปแบบของ HTML แบบใหม่ที่เรียกว่า XHTML ที่เป็นลักษณะของโครงสร้าง XML ที่มีหลักเกณฑ์ในการกำหนดโครงสร้างของโปรแกรมที่มีรูปแบบที่มาตรฐานกว่า HTML รุ่น 4.01 ที่ใช้งานอยู่ในปัจจุบัน ในขณะเดียวกัน HTML รุ่น 5 ยังคงอยู่ในระหว่างการพิจารณาในการใช้งาน

โครงสร้างพื้นฐานของ HTML ประกอบไปด้วยส่วนของคำสั่ง 2 ส่วนคือส่วนที่เป็นส่วนหัว (Head) และส่วนที่เป็นเนื้อหา (Body) โดยมีรูปแบบคำสั่งดังภาพที่แสดงไว้ในเอกสารแนบ

16. Figma

Figma เป็นเครื่องมือออกแบบกราฟิกแบบเว็บเบาะแสดงการทำงานแบบ collaborative (ร่วมมือกัน) ที่ทำงานบนเว็บบราวเซอร์ ซึ่งมีความสามารถในการสร้างส่วนต่างๆ ของการออกแบบเว็บ เช่น การออกแบบโครงสร้างของเว็บไซต์ (Wireframe) การออกแบบแบบต่างๆ การสร้างพื้นหลัง (Backgrounds) และอื่นๆ อีกมากมาย โดย Figma มีความยืดหยุ่นในการใช้งานและสามารถทำงานร่วมกันได้ในเวลาเดียวกันโดยไม่จำเป็นต้องใช้เครื่องมือหรือซอฟต์แวร์เพิ่มเติม

ทำให้เป็นทางเลือกที่ยอดเยี่ยมสำหรับทีมที่ทำงานร่วมกันหรือผู้ที่ต้องการให้การติดตามและเข้าถึงงานออกแบบได้ง่ายและสะดวก เราสามารถใช้ Figma ได้ในการออกแบบโดยไม่ว่าจะเป็นผ่านคอมพิวเตอร์หรืออุปกรณ์พกพาที่มีเว็บเบราว์เซอร์ได้ทุกที่ทุกเวลา และสามารถแชร์งานออกแบบหรือเชิญชวนผู้อื่นเข้าร่วมงานได้โดยง่ายดายผ่านลิงก์ที่สร้างขึ้นจาก Figma ได้อย่างรวดเร็วและสะดวก

นอกจากนี้ Figma ยังมีฟีเจอร์การแสดงผล (prototyping) ซึ่งช่วยให้สามารถสร้างพร็อตตี้ประสบการณ์การใช้งานของเว็บไซต์หรือแอพพลิเคชั่นได้ง่ายและรวดเร็ว

ภาพที่ 3.16 โปรแกรม Figma

นอกจากนี้ยังมีการสนับสนุนการทำงานแบบออฟไลน์ (offline mode) ซึ่งช่วยให้สามารถทำงานได้ทุกที่ทุกเวลาแม้ในสถานการณ์ที่ไม่มีการเชื่อมต่ออินเทอร์เน็ตด้วย สรุปคือ Figma เป็นเครื่องมือออกแบบกราฟิกที่มีความสามารถในการทำงานร่วมกันและออกแบบได้ง่ายและสะดวกบนเว็บบราวเซอร์ ทำให้เป็นทางเลือกที่น่าสนใจสำหรับทีมที่ทำงานร่วมกันหรือผู้ที่ต้องการให้การติดตามและเข้าถึงงานออกแบบได้ง่ายและสะดวก

17. Google Cloud Server

Google Cloud Server (เซิร์ฟเวอร์บนคลาวด์ของ Google) คือบริการคลาวด์คอมพิวติ้ง (cloud computing) ที่ให้บริการโฮสติ้งและการคำนวณผ่านพื้นที่คลาวด์ของ Google ซึ่งประกอบไปด้วยซอฟต์แวร์และอุปกรณ์ที่ใช้เป็นเทคโนโลยีของ Google เพื่อให้บริการแก่ผู้ใช้งาน

ภาพที่ 3.17 Google Cloud

โดย Google Cloud Server มีความสามารถที่หลากหลาย เช่น ให้บริการเว็บไซต์และแอปพลิเคชันออนไลน์ การเก็บข้อมูล การทำคลัสเตอร์ (cluster) การพัฒนาและการทดสอบโปรแกรม และอื่น ๆ อีกมากมาย ผู้ใช้สามารถเช่าบริการนี้ตามความต้องการของพวกเขาโดยจ่ายเงินตามการใช้งานหรือรายเดือนตามแผนที่เลือกใช้งาน

18. Portainer

Portainer เป็นเครื่องมือที่ใช้สำหรับจัดการและควบคุม Docker containers อย่างง่ายดายผ่านทางเว็บเบราว์เซอร์ โดยไม่จำเป็นต้องใช้คำสั่ง command line interface (CLI) ของ Docker โดยตรง มันช่วยให้ผู้ใช้สามารถดูสถานะของ containers, images, volumes และ networks ได้ง่ายดาย รวมถึงสามารถสร้าง, แก้ไข, และลบ containers และอื่น ๆ ได้โดยไม่ต้องใช้คำสั่ง CLI นอกจากนี้ยังมีฟีเจอร์ต่าง ๆ อีกมากมาย เช่น การจัดการกับ Docker swarm, การเซ็ตอัพ LDAP authentication, และ monitoring ของ containers ซึ่ง Portainer ได้รับความนิยมในการใช้งาน Docker แบบที่ใช้งานไม่มีคำสั่ง CLI และให้การจัดการที่เรียบง่ายและสะดวกสบายมากขึ้น

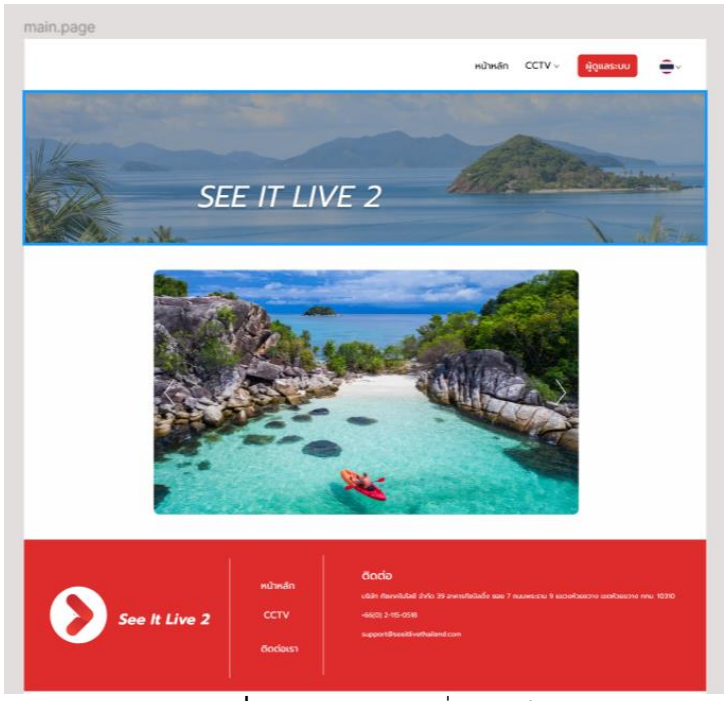
ภาพที่ 3.18 Portainer

วิธีดำเนินการวิจัย

1. **การวิเคราะห์ (System Analysis)**
   - **1.1 การวิเคราะห์ความต้องระบบ**
   - **1.2 การวางแผนในการทำงาน**
2. **การออกแบบ**

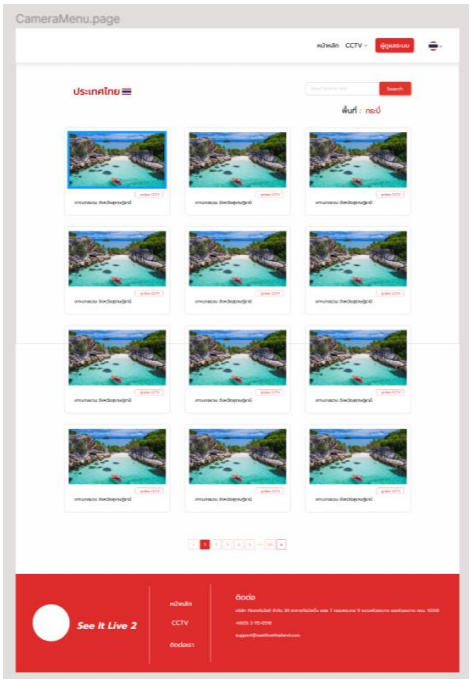
2.1 การออกแบบหน้าเว็บไซต์

2.1.1 หน้าหลัก



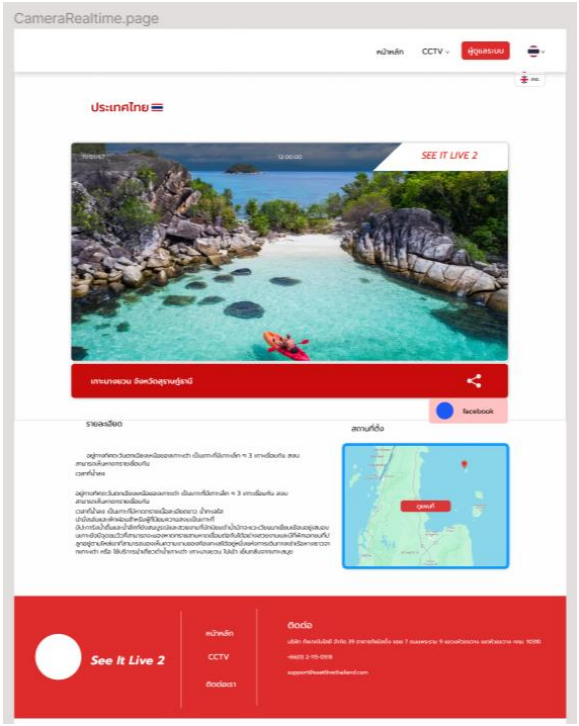
ภาพที่ 3.19 หน้าหลักเมื่อเปิดเว็บไซต์

2.1.2 หน้าเมนูกล้อง



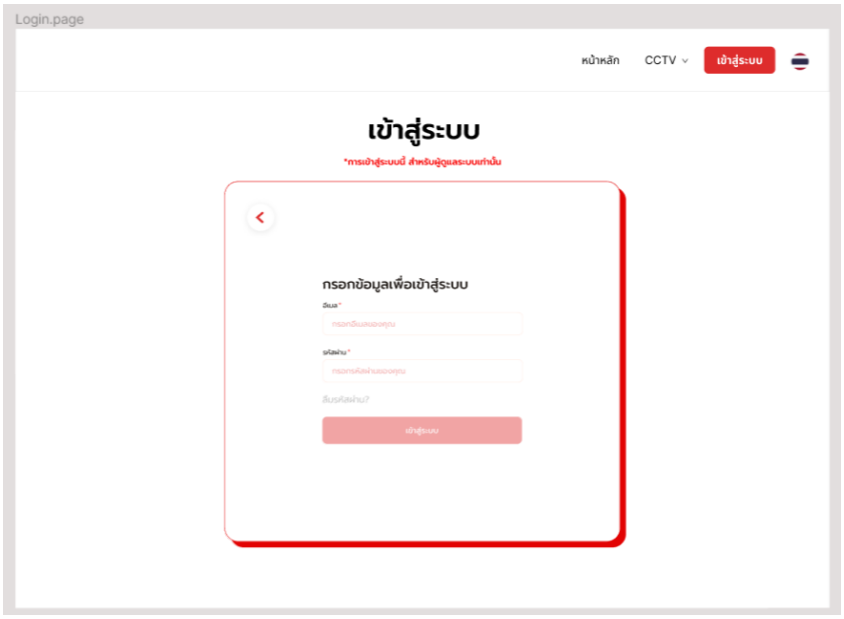
ภาพที่ 3.20 หน้าเมนูกล้อง

2.1.3 หน้าเมื่อกดดูกล้อง



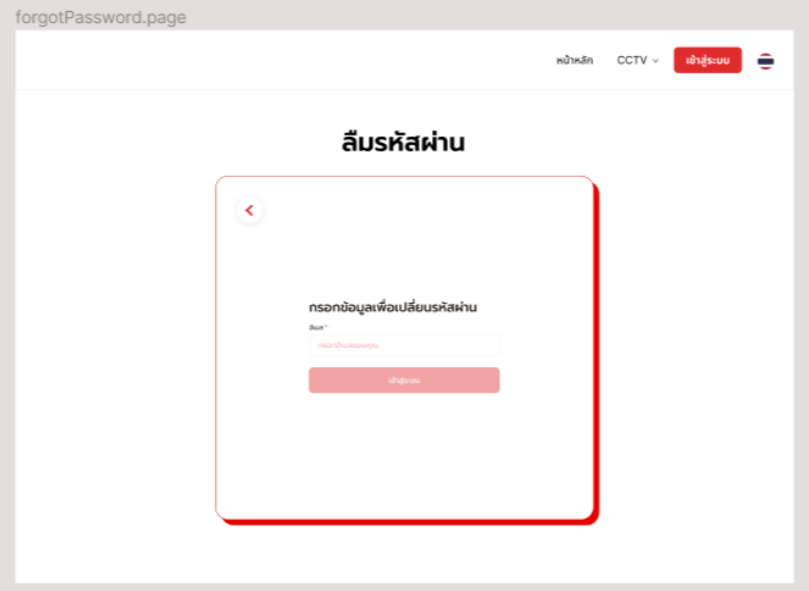
ภาพที่ 3.21 หน้าข้อมูลและการแสดงผลกล้องวงจรปิด

2.1.4 หน้าเข้าสู่ระบบ



ภาพที่ 3.22 หน้าเข้าสู่ระบบ

2.1.5 หน้าลืมรหัสผ่าน

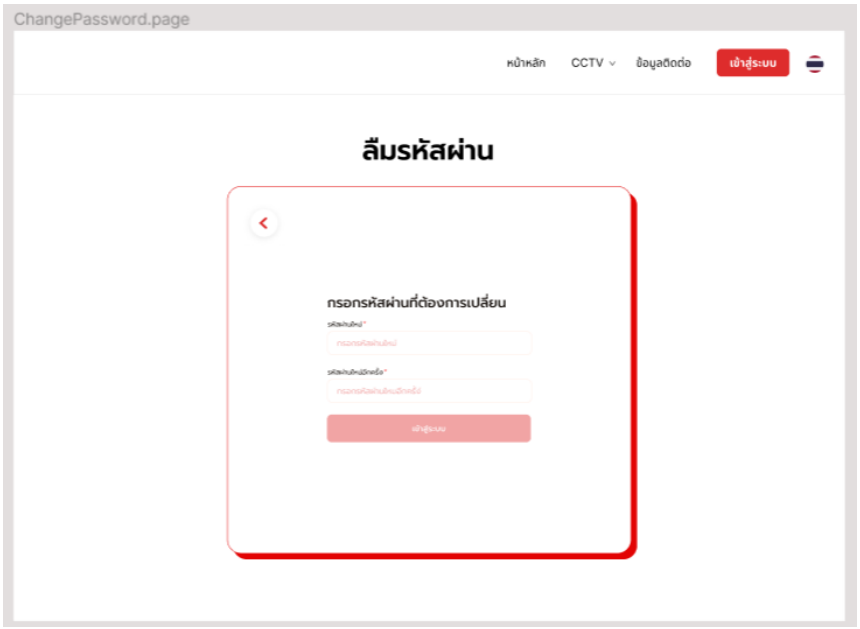


ภาพที่ 3.23 หน้ากรอกอีเมลเพื่อขอเปลี่ยนรหัสผ่าน

2.1.6 หน้ากรอกOtpเมื่อกดลืมรหัสผ่าน

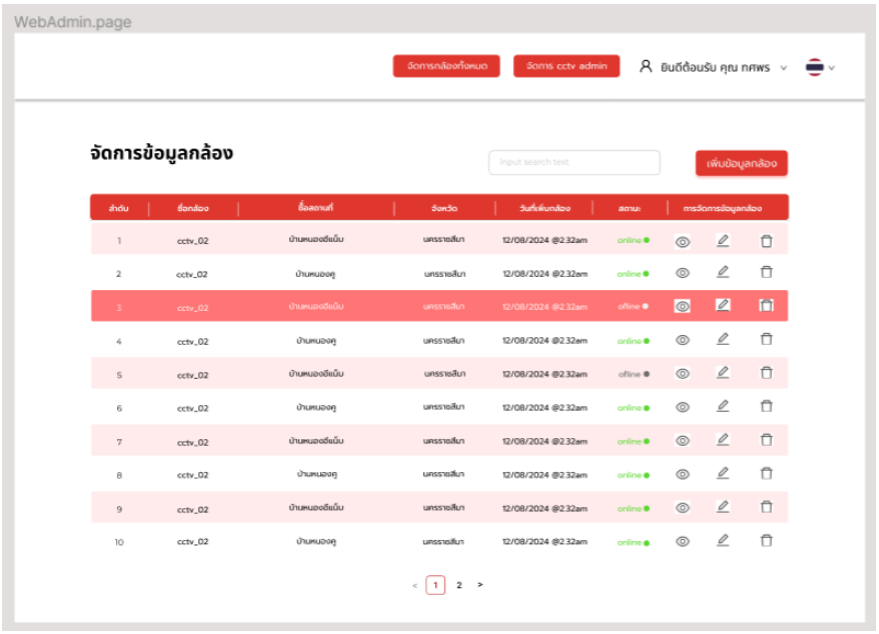
ภาพที่ 3.24 หน้ากรอกรหัส otp

2.1.7 หน้าเปลี่ยนรหัสผ่านเมื่อกดลืมรหัสผ่าน



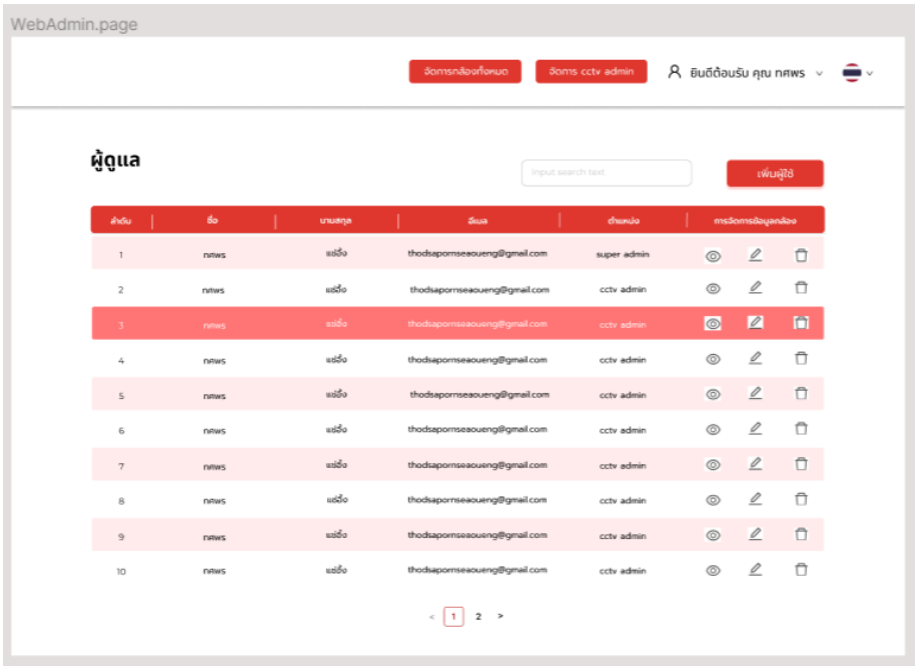
ภาพที่ 3.25 หน้าเปลี่ยนรหัสผ่านเมื่อกดลืมรหัส

2.1.8 หน้าการจัดการกล้องหลังบ้านเมื่อผู้ดูแลระบบล็อกอิน



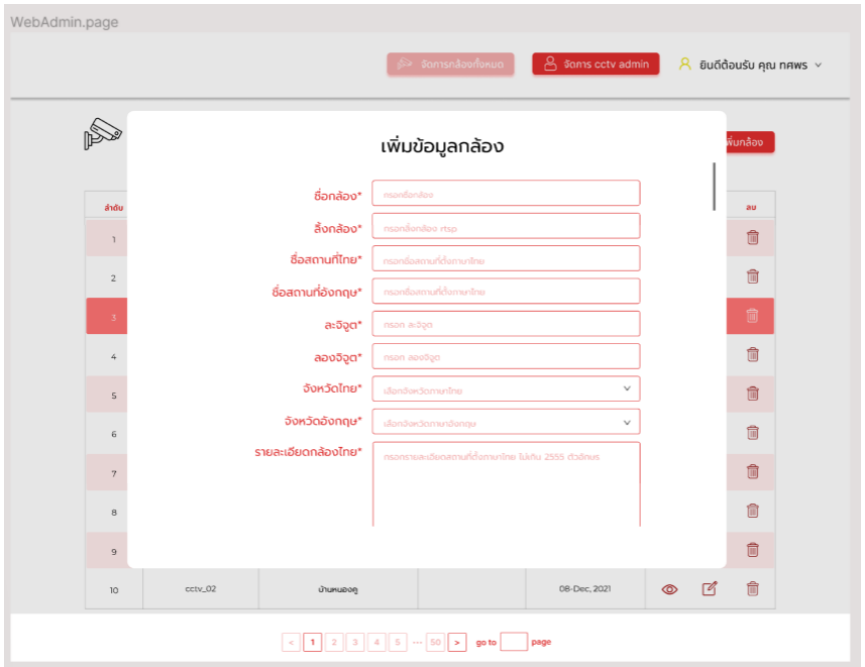
ภาพที่ 3.26 หน้าการจัดการข้อมูลกล้อง

2.1.9 หน้าการจัดการผู้ดูแลหลังบ้านสำหรับผู้ดูแลสูงสุด



ภาพที่ 3.27 หน้าการจัดการผู้ดูแลระบบ

2.1.10 หน้าการเพิ่มข้อมูลกล้อง

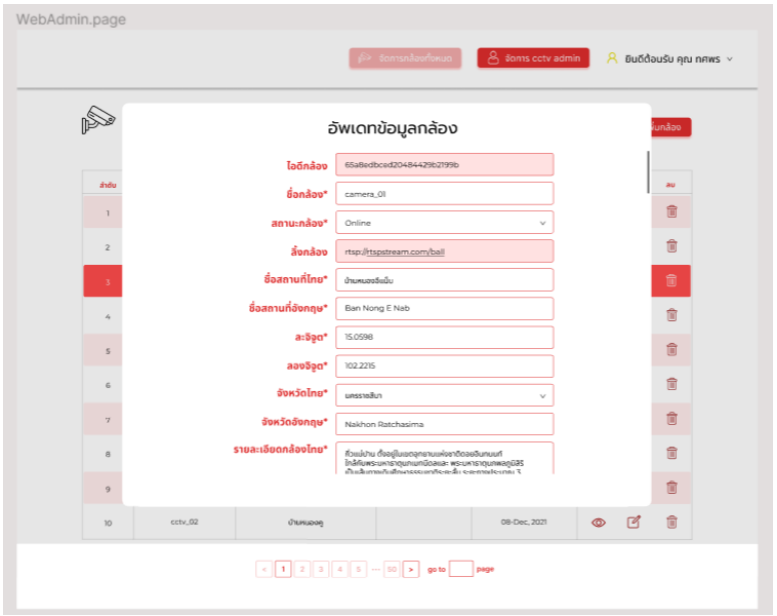


ภาพที่ 3.28 หน้าเพิ่มกล้องเข้าสู่ระบบ

2.1.11 หน้าการดูข้อมูลกล้อง

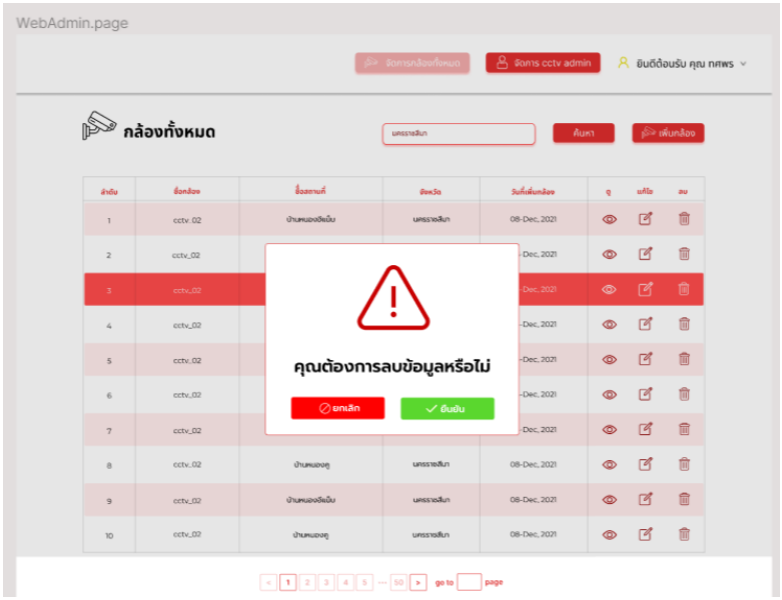
ภาพที่ 3.29 หน้าดูข้อมูลกล้อง

2.1.12 หน้าการอัพเดทข้อมูลกล้อง



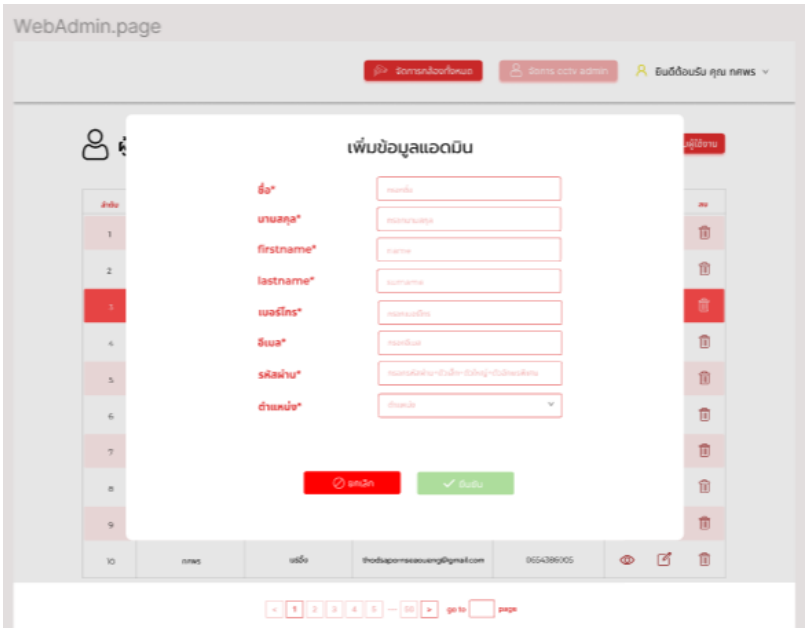
ภาพที่ 3.30 หน้าอัพเดทข้อมูลกล้อง

2.1.13 หน้าการลบข้อมูลกล้อง



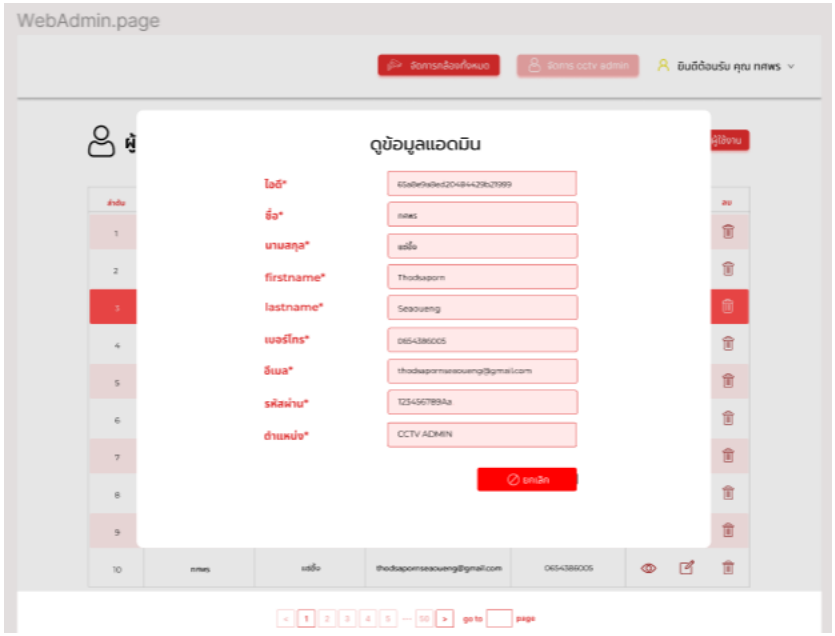
ภาพที่ 3.31 หน้าลบกล้อง

2.1.14 หน้าการเพิ่มข้อมูลผู้ดูแล



ภาพที่ 3.32 หน้าเพิ่มข้อมูลผู้ดูแลระบบ

2.1.15 หน้าการดูข้อมูลผู้ดูแล

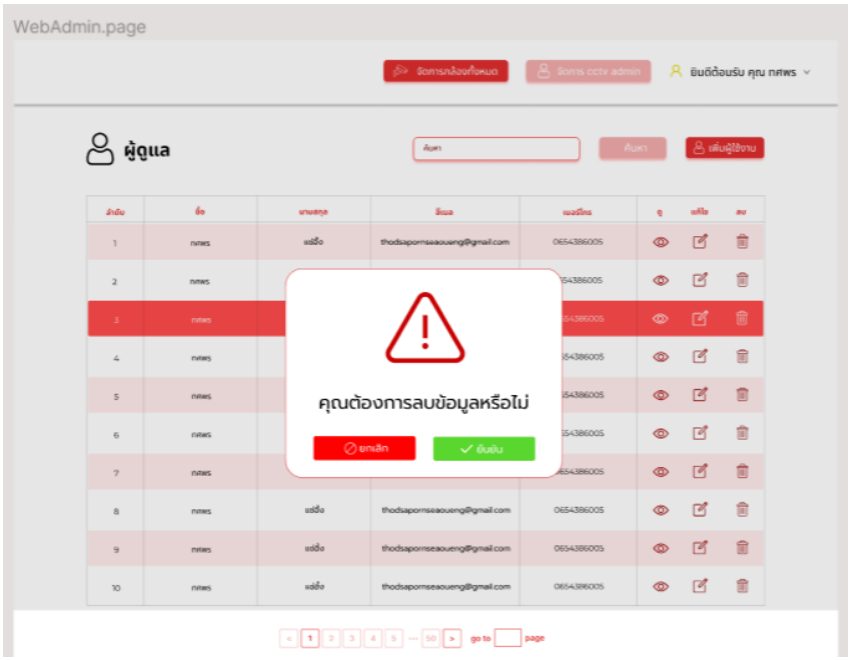


ภาพที่ 3.33 ดูข้อมูลผู้ดูแลระบบ

2.1.16 หน้าการอัพเดทข้อมูลผู้ดูแล

ภาพที่ 3.34 หน้าอัพเดทข้อมูลผู้ดูแลระบบ

2.1.16 หน้าการลบข้อมูลผู้ดูแล



ภาพที่ 3.35 หน้าลบข้อมูลผู้ดูแลระบบ

2.2 การออก Data Dictionary

ตารางที่ 1 CCTV ADMIN

Collection Name (cctv admin)

รายการ(ไทย)	Field	ประเภทข้อมูล	จำนวน	request	หมายเหตุ
ไอดี	_id	string	255	Yes	
ชื่อ	first_name	string	255	Yes	แสดงชื่อผู้เข้าใช้งาน
นามสกุล	last_name	string	255	Yes	
อีเมล	email	string	255	Yes	
รหัสผ่าน	password	string	100	Yes	พิมพ์ใหญ่+เล็กใหญ่
เบอร์โทร	telephone	string	10	Yes	
ตำแหน่ง	role	string	100	Yes	cctv_admin , super admin
ยืนยันตัวตน	verify	string	20		

ตารางที่ 2 Authentication

Collection Name (Authentication.token)

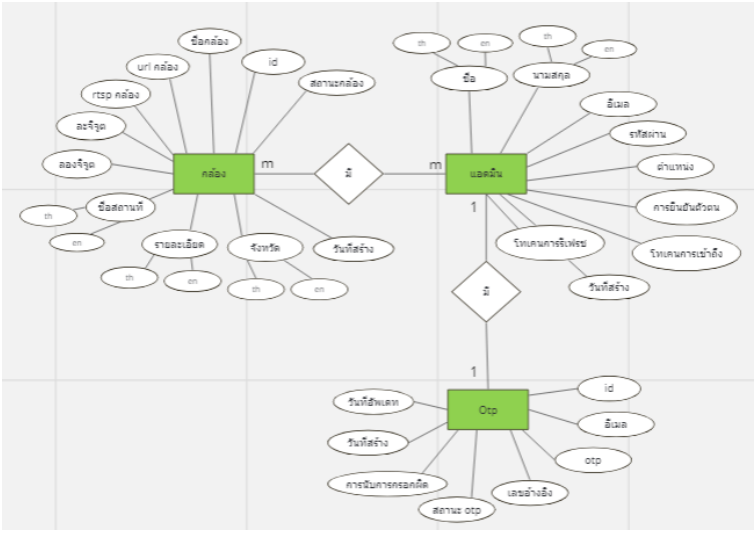
รายการ(ไทย)	Field	ประเภทข้อมูล	จำนวน	request	หมายเหตุ
ไอดี	_id	string	100	Yes	
ไอดีผู้ใช้	user_id	string	100	Yes	
โทเค้นเข้าถึง	access_token	string	100	Yes	
โทเค้นรีเฟรช	refresh_token	string	100	Yes	
เวลาหมดอายุ	expired	int	100	Yes	

ตารางที่ 3 Camera

Collection Name (Camera)

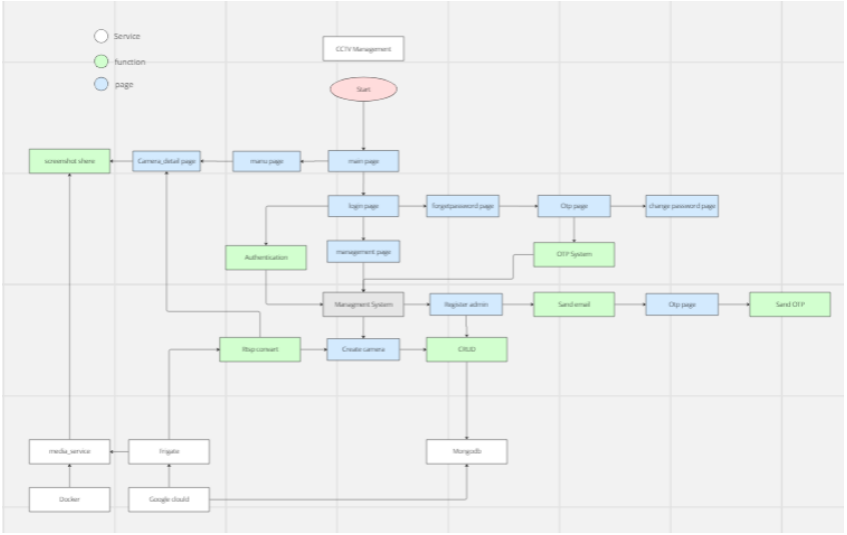
รายการ(ไทย)	Field	ประเภทข้อมูล	จำนวน	request	หมายเหตุ
ไอดีกล้อง	_id	string	255	Yes	
ชื่อกล้อง	cctv_name	string	255	Yes	
ลิ้งกล้อง	rtsp	string	255	Yes	
รูปปกกล้อง	camera_img	string	255	Yes	.png(cctvadminเพิ่มรูปเอง)
รูปสกรีนช็อตแชร์	screenshot	string	255	Yes	.png
สถานะ	status	string	255	Yes	online , ofline
วันที่เพิ่มกล้อง	created_at	string	255		unit time stamp
วันที่อัพเดท	update_at	string	255		unit time stamp
ชื่อสถานที่	place_name	object			
พิกัด	coordinates	object			
จังหวัด	province	object			dropdown
รายละเอียดกล้อง	detail	object			

2.3 การออก ER Diagram



ภาพที่ 3.36 ER Diagram

2.4 การออก Work flow



ภาพที่ 3.37 Work flow การทำงานทั้งหมด

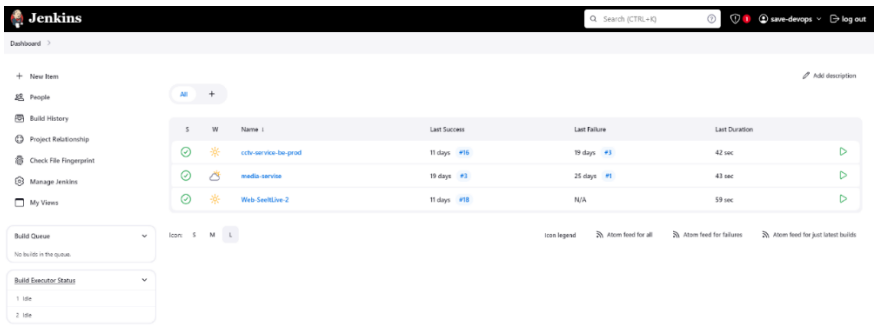
2.5 การออก Flow การทำงานระบบ OTp

ภาพที่ 3.37 Flow การทำงานของ Otp ทั้งหมด

3. การใช้งานและการให้บริการ (Deployment and Operation)

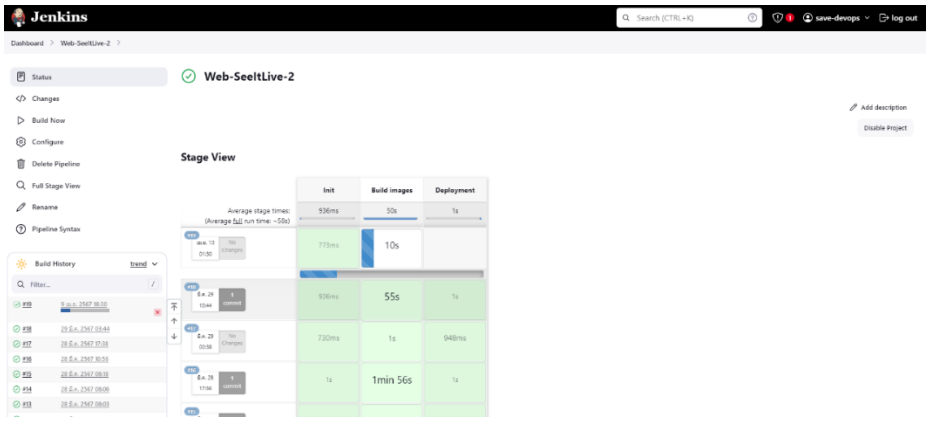
3.1 ออกบริการระบบเว็บไซต์ด้วยการ Deploy หervice

3.1.1 หน้าต่าง Jenkins เพื่อ Deploy service



ภาพที่ 3.38 หน้าเว็บ Jenkins

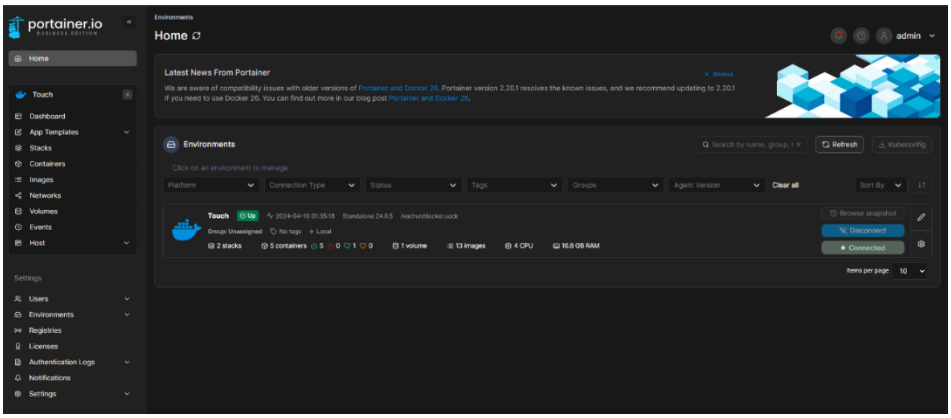
3.1.2 หน้าต่าง Jenkins เพื่อ Deploy service



ภาพที่ 3.39 ตัวอย่างหน้าต่าง Deploy Web ของหน้าบ้าน

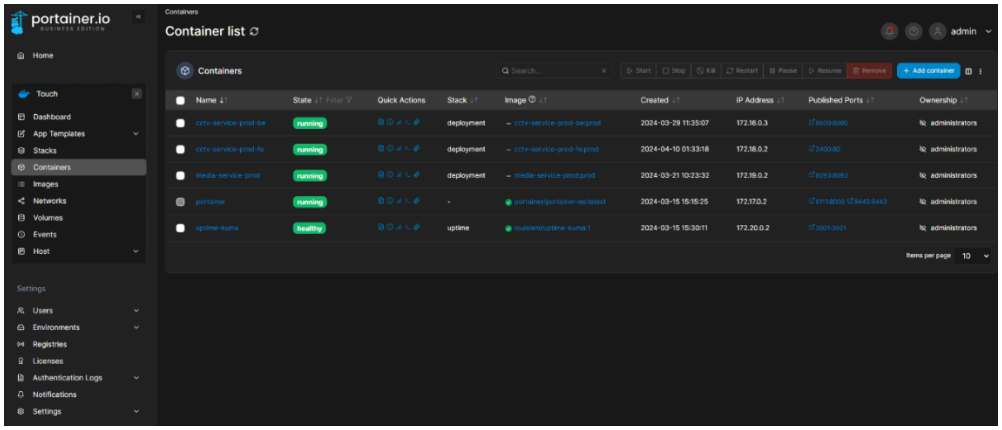
3.2. การให้บริการระบบ Portainer

3.2.1 หน้าแรกของ Portainer เมื่อ longin แล้ว



ภาพที่ 3.40 หน้าเว็บ Portainer

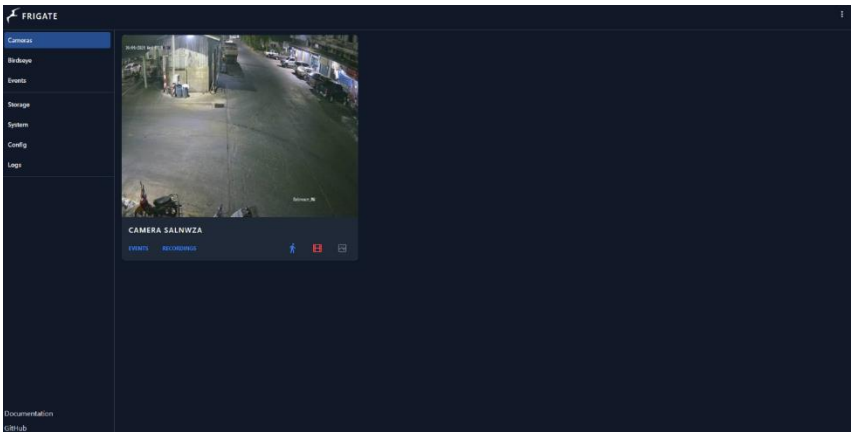
3.2.2 หน้าต่างของ Portainer เมื่อเข้าดู Container



ภาพที่ 3.41 หน้าเว็บ Portainer ดู Container list

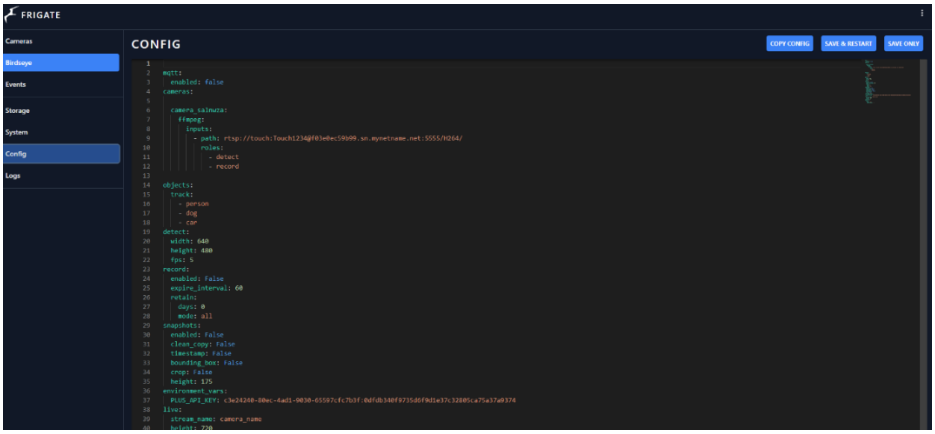
3.2. การให้บริการของระบบ Frigate

3.2.1 หน้าแรกของ Frigate



ภาพที่ 3.42 หน้าแรกของ Frigate

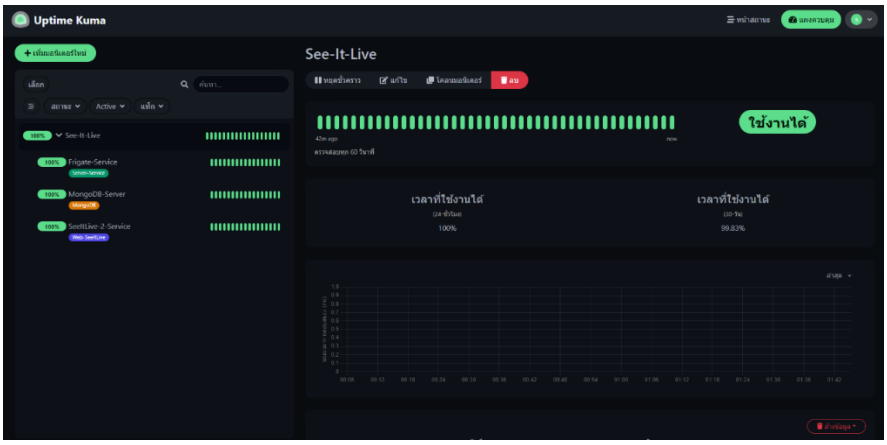
3.2.2 หน้า config ของ Frigate เพื่อเพิ่ม ลบ แก้ไขกล้อง



ภาพที่ 3.42 หน้า config ของ Frigate

3.2. การให้บริการของระบบ Uptime Kuma

3.2.1 หน้าแรกของ Uptime Kuma



ภาพที่ 3.43 หน้าแรกของ Uptime Kuma

3.2.2 หน้าแสดงข้อมูลของ Uptime Kuma

ภาพที่ 3.44 หน้าแสดงข้อมูลของ Uptime Kuma

สรุปผลและข้อเสนอแนะ

1. สรุปผล

จากการทำโครงงานครั้งนี้เว็บไซต์ สามารถทำงานได้ตามฟังก์ชั่นที่ได้กำหนดไว้ ไม่ว่าจะเป็นการจัดการกล้อง และ ผู้ดูแล ระบบ Otp เมื่อสร้างบัญชีผู้ดูแลและการลืมรหัส การนำกล้องวงจรปิดนำมาแสดงผลบนเว็บ การออกแบบหน้าเว็บ รับรองระบบ 2 ภาษา ผู้ใช้งานจะแบ่งทั้งหมด 3 ระดับ คือ 1. ผู้ใช้งานทั่วไป 2. ผู้ดูแลระบบ 3.ผู้ดูแลระบบสูงสุด โดยเก็บข้อมูลในฐานข้อมูล Mongodb โดย

ยังมีการแสดงผลบางส่วนที่ยังทำงานได้ไม่เต็มประสิทธิภาพ เช่น การแสดงผลภาพกล้องวงจรปิดที่มีความไม่ลื่นไหล

2. ข้อเสนอแนะในการทำวิจัย

การทำระบบจัดการกล้องวงจรปิดบนเว็บอาจมีการแสดงผลภาพของกล้องวงจรปิดที่ยังไม่ลื่นไหล เนื่องจากทรัพยากรในการติดตั้ง Service Frigate ที่ใช้ในการประมวลผลกล้องวงจรปิด ไม่เพียงพอ ในการพัฒนาระบบ ได้ใช้เครื่องมือที่ไม่เคยใช้ ทำให้ศึกษาเพื่อใช้งานใช้ระยะเวลานานและการพัฒนาระบบจริงเอกสารการใช้งานของเครื่องมือมีน้อยจึงทำให้ใช้เวลาในการพัฒนายาวนานขึ้น

บทที่ 4

สรุปผลการปฏิบัติงานและข้อเสนอแนะ

จากการปฏิบัติงานใน บริษัท ทัชเทคโนโลยี จำกัด ได้รับความรู้ต่างๆ ที่เป็นประสบการณ์ต่อไปในอนาคต ได้เรียนรู้การวิเคราะห์ความต้องการระบบ การออกแบบฐานข้อมูล การเรียนรู้การเขียนโปรแกรมด้วยภาษา Golang การออกแบบ Gui หน้าเว็บไซต์ การออกแบบ Flow การทำงาน การออกแบบ Timeline กำหนดงานที่ได้รับหมอบหมาย และวันที่ต้องส่งหมอบงาน การใช้งานฐานข้อมูล Mongodb สามารถสรุปได้ดังนี้

จากการทำโครงงานครั้งนี้เว็บไซต์ สามารถทำงานได้ตามฟังชั่นที่ได้กำหนดไว้ ไม่ว่าจะเป็นการจัดการกล้อง และ ผู้ดูแล ระบบ Otp 6 หลักเมื่อสร้างบัญชีผู้ดูแลสำหรับการยืนยันตัวตนโดยจะส่งรหัส otp ไปยังอีเมลผู้สมัครและการลืมรหัสสำหรับการเปลี่ยนรหัส การนำกล้องวงจรปิดนำมาแสดงผลบนเว็บ การออกแบบหน้าเว็บ รับรองระบบ 2 ภาษาโดยการเก็บข้อมูลกล้อง 2 ภาษา สำหรับการเปลี่ยนโดยข้อมูลที่มาจาก Api ในระบบจะแบ่งผู้ใช้งานจะแบ่งทั้งหมด 3 ระดับ คือ 1. ผู้ใช้งานทั่วไป สามารถเข้าดูกล้องวงจรปิดบนเว็บได้อย่างเดียว 2. ผู้ดูแลระบบ สามารถเข้าถึงข้อมูลกล้องวงจรปิด สามารถจัดการกล้องวงจรปิดได้ 3. ผู้ดูแลระบบสูงสุด สามารถเข้าถึงข้อมูลกล้องวงจรปิด สามารถจัดการกล้องวงจรปิดและจัดการข้อมูลผู้ดูแลจะเก็บข้อมูลทั้งหมดในฐานข้อมูลกล้องวงจรปิดและข้อมูลผู้ดูแลระบบด้วยฐานข้อมูล Mongodb บน Server โดยดึงข้อมูลไปยังเว็บไซต์เพื่อนำไปใช้กับระบบหลังบ้าน โดยนำข้อมูลกล้องทำเป็น Api แยกเป็นเส้นในแต่ละกล้อง จากนั้นนำมารวมกับ Api เส้นกล้องที่แปลงแล้วโดยการแสดงผลภาพกล้องวงจรปิดใช้ Service ชื่อว่า Frigate โดยนำกล้องวงจรปิดมาแปลงเป็บลิ้ง Url และแปลงเป็น Api เพื่อในไปใช้บนเว็บ โดยในการแสดงผลจะแสดงผลเป็นภาพ Realtime แต่ยังมีการแสดงผลบางส่วนที่ยังทำงานได้ไม่เต็มประสิทธิภาพเนื่องจากทรัพยากรที่ใช้ในการประมวลผลไม่เพียงพอในการประมวลผลกล้องหลายตัว

สรุปผลการปฏิบัติงาน

1. ด้านคุณธรรมจริยธรรมในการปฏิบัติงาน

1.1 มีความซื่อสัตว์ต่อหน้าที่และงานที่ได้รับหมอบหมายปฏิบัติงานด้วยความจริงใจ และไม่คดโกงหรือหลอกลวงผู้อื่น จึงจะได้ความไว้วางใจจากผู้ร่วมงาน

1.2 มีความเสียสละ ในการทำงานร่วมกับผู้อื่น เห็นแก่ประโยชน์ส่วนรวมมากกว่าประโยชน์ส่วนตน ไม่เห็นแก่ตัว

1.3 มีความยุติธรรมในการทำงานไม่ลำเอียงหรือยึดถือสิ่งใดสิ่งหนึ่ง

1.4 มีความขยันและอดทนในการทำงาน มุ่งมั่นต่องานที่ได้รับหมอบหมาย เพื่อให้งานบรรลุเป้าหมาย

1.5 มีความรับผิดชอบในการทำงาน ต้องมีความรับผิดชอบต่องานที่ได้รับหมอบหมาย

1.6 มีความตรงต่อเวลา ไม่มาทำงานสายและส่งงานที่ได้รับหมอบหมายตามกำหนด

2. ด้านการเรียนรู้การทำงานในสถานประกอบการ

2.1 การบริหารการจัดการในเรื่องส่วนตัวต่างๆ มีขั้นตอน จัดสรรเวลาอย่างมีระบบติดต่อสื่อสารกันระหว่างหน่วยงานภานใน

2.2 ได้เพิ่มพูนทักษะต่างๆในทุกตำแหน่ง

2.3 การจัดการเอกสารสำหรับคนในทีม

2.4 มารยาททางสังคมในการทำงานในสถานประกอบการ

2.5 แนวทางในการแก้ไขปัญหาต่างๆ ในสถานประกอบการซึ่งมีอายุแตกต่างๆกัน

2.6 การตรงต่อเวลา เรื่องนัดการประชุม เรื่องเข้าทำงาน และการส่งมอบงานที่ได้รับหมอบหมาย

2.7 การรับมือและแก้ไขปัญหาเฉพาะหน้า

2.8 ได้เรียนรู้ถึงสภาพการทำงาน สังคม และวัฒณธรรมจากสถานที่ประกอบการจริง

2.9 ได้เรียนรู้เครื่องมือต่างๆที่หน่วยงานนำมาใช้ในการทำงานจริง

2.10 ได้เรียนรู้การทำงานร่วมกับผู้อื่น และเพิ่มทักษะการเรียนรู้ละการทำงานในองค์กร

2.11 การค้นคว้าและการศึกษาเรียนรู้เพิ่มเติมนอกเหนือในเวลางาน

3. ด้านการใช้สติปัญญาแก้ปัญหาในการทำงาน

3.1 ได้เรียนรู้และปฏิบัติงานจริงและรวมถึงการทำงานขององค์กร

3.2 การเรียนรู้การแก้ไขปัญหาในสถานการจริงเมื่อพบข้อผิดพลาด

4. ด้านการทำงานร่วมกันในองค์กร

4.1 ได้ทำความรู้จักกับพนักงานหรือบุคคลที่เกี่ยวข้องภายในหน่วยงานในแต่ละตำแหน่ง

4.2 ได้เรียนรู้ถึงระบบการวางแผนการทำงาน การอยุ่ในสังคม

5. ด้านการใช้เครื่องมือ อุปกรณ์ ในการทำงาน

5.1 ได้เรียนรู้เครื่องมือในการจัดการฐานข้อมูล Mongodb

5.2 ได้เรียนรู้ภาษา Golang เป็นเครื่องมือในการทำงาน

5.3 ได้เรียนรู้ใช้งาน React สำหรับการเขียนหน้าเว็บไซต์

ประโยชน์ที่ได้รับจากการปฏิบัติงาน

1. ประโยชน์ต่อตนเอง

1.1 ประสบการณ์วิชาชีพตามสาขาวิชาที่เรียนเพิ่มเติมจากการเรียนรู้ในมหาวิทยาลัย

1.2 ได้รับประสบการณ์การความรู้ใหม่ที่เพิ่มขึ้น จากการเรียนในสถานศึกษา ที่มีความจำเป็นในการทำงานในอนาคต

1.3 ได้รับประสบการณ์การทำงานร่วมกับผู้อื่น และมีหน้าที่รับผิดชอบตามที่ได้รับหมอบหมาย

1.4 เกิดทักษะการสื่อสารข้อมูล (Communication Skill)

1.5 การค้นหาตัวเอง ค้นหาทักษะที่ถนัด สำหรับการเลือกสายอาชีพในอนาคต

1.6 เป็นบัณฑิตที่มีศักยภาพในการทำงานมากขึ้นและมีโอกาศได้รับการเสนองานก่อนสำเร็จการศึกษา

2. ประโยชน์ต่อสถานประกอบการ

2.1 เพื่อสร้างความสัมพันธ์ที่ดีระหว่างมหาวิทยาลัยกับสถานประกอบการ

2.2 ระบบสหกิจศึกษาเป็นวิธีการช่วยคัดเลือกให้นักศึกษามีโอกาศในการเป็นพนักงานประจำองค์กรในอนาคต

3. ประโยชน์ต่อมหาวิทยาลัย

3.1 เพื่อได้รับข้อมูลย้อนกลับมาปรับปรุงหลักสูตรและการเรียนการสอนในอนาคต

3.2 เพื่อให้เกิดความร่วมมือทางวิชาการและสร้างความสัมพันธ์กับสถานประกอบการ

3.3 เป็นการเพิ่มศักยภาพของอาจารย์และเพิ่มประสบการณ์ในภาคปฏิบัติและสามารถนำปัญหาที่เกิดขึ้นมาประยุกต์ พัฒนา กับการเรียนการสอนภายในห้องเรียนได้

ข้อเสนอแนะ

1. ข้อเสนอแนะต่อนักศึกษาที่จะออกปฏิบัติงานในภาคการศึกษาต่อไป

1.1 ศึกษาหน่วยงานหรือสถานประกอบการณ์ที่ต้องการให้พร้อม และเตรียมความพร้อมของตนเองในการปฏิบัติงาน

2. ข้อเสนอแนะต่อสถานประกอบการณ์

(ไม่มี)

3. ข้อเสนอแนะต่ออาจารย์นิเทศ

(ไม่มี)

4. ข้อเสนอแนะต่อมหาวิทยาลัย

4.1 การให้เวลาในการติดต่อกับสถานประกอบการที่ต้องการยื่นสมัครน้อยเกินไปทำให้นักศึกษาไม่ได้สถานประกอบการที่ต้องการ

4.2 การแจ้งเกี่ยวกับเอกสารที่ต้องส่งหลังจากฝึกสหกิจศึกษาจบไม่ชัดเจน

5. ข้อเสนอแนะอื่นๆ

5.1 ทักษะที่ได้จากในมหาลัยยังไม่มากพอในการทำงานในสถานประกอบการจริง

5.2 เรื่องความมั่นใจในตัวเอง เมื่อเผชิญหน้าในการทำงานจริง

บรรณานุกรม

Api Spacification Document (2022) [ออนไลน์]
แหล่งที่มา : https://www.etda.or.th/getattachment/46715917-c119-4b19-b916-809259de19ef/API-Specification-Document.aspx

MongoDB. (2020). "บทนำสู่ MongoDB." เอกสาร MongoDB. [ออนไลน์]. สืบค้นจาก: https://docs.mongodb.com/manual/introduction/.

Golang. (2019). "Go ที่มีประสิทธิภาพ." ภาษาโปรแกรม Go. [ออนไลน์]. สืบค้นจาก: https://golang.org/doc/effective_go.html.

Visual Studio Code. (2021). "อินเตอร์เฟซผู้ใช้ของ Visual Studio Code." [ออนไลน์]. https://code.visualstudio.com/docs/getstarted/userinterface.

RESTful API. (2020). "บทแนะนำ REST API." บทแนะนำ RESTful API. [ออนไลน์]. สืบค้นจาก: https://www.restapitutorial.com/.

Postman. (2018). "เอกสาร Postman." Postman. [ออนไลน์]. สืบค้นจาก: https://learning.postman.com/docs/getting-started/introduction/.

FrigateNVR. (2021). "เริ่มต้นใช้งาน FrigateNVR." เอกสาร FrigateNVR. [ออนไลน์]. สืบค้นจาก: https://docs.frigate.video/en/latest/.

Gin framework. (2019). "เอกสาร Gin." Gin. [ออนไลน์]. สืบค้นจาก: https://pkg.go.dev/github.com/gin-gonic/gin.

Docker. (2020). "เอกสาร Docker." Docker. [ออนไลน์]. สืบค้นจาก: https://docs.docker.com/.

Jenkins. (2020). "เอกสารผู้ใช้ Jenkins." Jenkins. [ออนไลน์]. สืบค้นจาก: https://www.jenkins.io/doc/.

Nginx. (2021). "คู่มือเบื้องต้น." Nginx. [ออนไลน์]. สืบค้นจาก: https://nginx.org/en/docs/beginners_guide.html.

Gitlab. (2021). "เอกสาร Gitlab." Gitlab. [ออนไลน์]. สืบค้นจาก: https://docs.gitlab.com/.

ReactJS. (2020). "เอกสาร React." React. [ออนไลน์]. สืบค้นจาก: https://reactjs.org/docs/getting-started.html.

JavaScript. (2019). "เอกสาร JavaScript." MDN Web Docs. [ออนไลน์]. สืบค้นจาก: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

CSS. (2020). "เอกสาร CSS." MDN Web Docs. [ออนไลน์]. สืบค้นจาก: https://developer.mozilla.org/en-US/docs/Web/CSS.

HTML. (2021). "เอกสาร HTML." MDN Web Docs. [ออนไลน์]. สืบค้นจาก: https://developer.mozilla.org/en-US/docs/Web/HTML.

Figma. (2021). "ศูนย์ความช่วยเหลือ Figma." Figma. [ออนไลน์]. สืบค้นจาก: https://help.figma.com/.

Google Cloud Server. (2020). "เอกสาร Google Cloud." Google Cloud. [ออนไลน์]. สืบค้นจาก: https://cloud.google.com/docs.

Portainer. (2019). "เอกสาร Portainer." Portainer. [ออนไลน์]. สืบค้นจาก: https://documentation.portainer.io/.

ประวัติผู้จัดทำ

ชื่อ-นามสกุล ณัฐชรีณ ไปยะโพธิ์ศรี
สาขาวิชา วิทยาการคอมพิวเตอร์
คณะ วิทยาศาสตร์และเทคโนโลยี
ประวัติการศึกษา ระดับมัธยมศึกษาตอนต้น โรงเรียนหนองกี่พิทยาคม
ระดับมัธยมศึกษาตอนปลาย โรงเรียนหนองกี่พิทยาคม
ระดับปริญญาตรี มหาวิทยาลัยราชภัฏนครราชสีมา
สถานที่ติดต่อ 53/1หมู่10 ต.ดอนอะราง อ.หนองกี่ จ.บุรีรัมย์ 31210
โทรศัพท์ 093-4862155
อีเมล natchiii4869@gmail.com

ชื่อ-นามสกุล นายทศพร แซ่อึ้ง
สาขาวิชา เทคโนโลยีสารสนเทศ
คณะ วิทยาศาสตร์และเทคโนโลยี
ประวัติการศึกษา ระดับมัธยมศึกษาตอนต้น โรงเรียนกลางดงบุณณวิทยา
ระดับมัธยมศึกษาตอนปลาย โรงเรียนขามทะเลสอวิทยา
ระดับปริญญาตรี มหาวิทยาลัยราชภัฏนครราชสีมา
สถานที่ติดต่อ 323 หมู่ 2 ต.ขามทะเลสอ อ.ขามทะเลสอ จ.นครราชสีมา
โทรศัพท์ 065-4386005
อีเมล gin25443@gmail.com

ชื่อ-นามสกุล	ธนกร ทองคล้าย	
สาขาวิชา	เทคโนโลยีสารสนเทศ	
คณะ	วิทยาศาสตร์และเทคโนโลยี	
ประวัติการศึกษา	ระดับประถมศึกษา	โรงเรียนเมืองนครราชสีมา
	ระดับมัธยมศึกษาตอนต้น	โรงเรียนสุรธรรมพิทักษ์
	ระดับมัธยมศึกษาตอนปลาย	โรงเรียนสุรธรรมพิทักษ์
	ระดับปริญญาตรี	มหาวิทยาลัยราชภัฏนครราชสีมา
สถานที่ติดต่อ	567 หมู่4 ตำบลโพธิ์กลาง จังหวัดนครราชสีมา 30000	
โทรศัพท์	087-2420683	
อีเมล	asas81976@gmail.com	

ชื่อ-นามสกุล	ปรวรรธ จำปาพันธุ์	
สาขาวิชา	เทคโนโลยีสารสนเทศ	
คณะ	วิทยาศาสตร์และเทคโนโลยี	
ประวัติการศึกษา	ระดับประถมศึกษา	โรงเรียนอนุบาลนางรอง บุรีรัมย์
	ระดับมัธยมศึกษาตอนต้น	โรงเรียนนางรอง บุรีรัมย์
	ระดับมัธยมศึกษาตอนปลาย	โรงเรียนนางรอง บุรีรัมย์
	ระดับปริญญาตรี	มหาวิทยาลัยราชภัฏนครราชสีมา
สถานที่ติดต่อ	102/12 ถ.พายัพทิศ ต.ในเมือง อ.เมือง จ.นครราชสีมา 30000	
โทรศัพท์	093-5513692	
อีเมล	6340208121@nrru.ac.th	