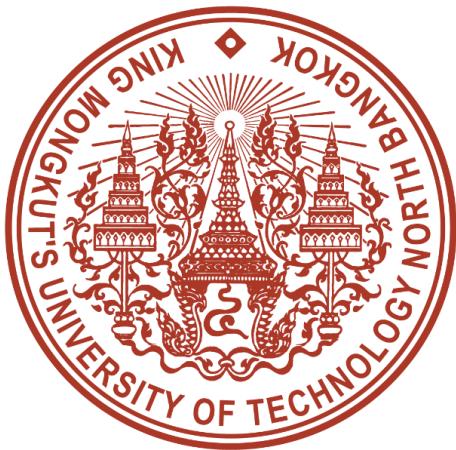


Software Requirements Specification



ระบบเช่ารถออนไลน์

Clubcars Rental

จัดทำโดย

6504062636233 นางสาวพีรยา ใจเที่ยง

6604062616126 นางสาวภัทรนันท์ ปันสุขสวัสดิ์

6604062636119 นางสาวชัชชญา ฉายวัฒนา

6604062636127 นางสาวชีติยาบัตร วรรัตนานาชัย

6604062636135 นางสาวฐานันดา ศิริพละ

เสนอ

ผู้ช่วยศาสตราจารย์ สมิติ ประสมพันธ์

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา 040613306 วิศวกรรมซอฟแวร์
ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิศวกรรมศาสตร์ประยุกต์
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีการศึกษา 2/2568

Table of Content

	Page
1. Introduction	
1.1 ព័ត៌មាននិងការស្នើសុំរបៀប	
1.2 Stakeholder	
1.3 Software Scope	
1.4 Software purpose	
2. Specification Requirements	
2.1 user Requirements	
2.2 System Requirements	
2.3 Non-functional Requirements	
2.4 Domain Requirements	
3. System Architecture	
4. Deployment Diagram	
5. User Interface Prototype	
6. ແຜນរាយឃុំសគេស (Use - Case Diagram)	
7. ແຜນរាយគម្ពាល់(Class Diagram)	
8. ແຜນរាយកិច្ចការណ៍ (Activity Diagram)	
9. ER Diagram	
10. User Acceptance Test	
11. Process Model	
12. Risk Management	
13. Version Control	
14. QA (Quality Assurance)	

Introduction

1.1 ที่มาและความสำคัญของระบบ

ระบบเช่ารถออนไลน์พัฒนาขึ้นเพื่อเพิ่มประสิทธิภาพในการบริหารธุรกิจรถเช่า ตอบสนองความต้องการของลูกค้าในยุคดิจิทัล ลดข้อผิดพลาดจากการเดินที่ใช้ออกสารและโทรศัพท์ โดยใช้เทคโนโลยีเพื่อจัดการการจอง ตรวจสอบสถานะรถ ระบบยังพัฒนาให้ปลอดภัย รองรับการชำระเงินออนไลน์ ลดต้นทุน เพิ่มรายได้ และสร้างความไว้เปรียบในการแข่งขันในตลาดรถเช่าที่เติบโตอย่างรวดเร็ว

1.2 Stakeholder

ในระบบเช่ารถออนไลน์ ผู้มีส่วนได้ส่วนเสียหลักประกอบด้วย:

- ลูกค้า (Customers) – ผู้ใช้บริการที่ต้องการเช่ารถผ่านระบบออนไลน์ คาดหวังความสะดวก รวดเร็ว และความปลอดภัยในการจองและชำระเงิน
- เจ้าของธุรกิจ (Owners) – ผู้บริหารจัดการรถเช่า ควบคุมรายการจอง ตรวจสอบสถานะรถ และบริหารต้นทุน
- พนักงาน (Staff) – ผู้ดูแลการดำเนินงาน เช่น การยืนยันการจอง ตรวจสอบเอกสาร และจัดเตรียมรถให้พร้อมใช้งาน
- ผู้พัฒนาและดูแลระบบ (Developers/System Administrators) – ทีมงานที่ออกแบบ พัฒนา และบำรุงรักษาระบบให้มีประสิทธิภาพและความปลอดภัย
- หน่วยงานกำกับดูแล (Regulatory Authorities) – องค์กรที่เกี่ยวข้องกับกฎหมาย เช่น กรมขนส่งหรือหน่วยงานด้านภาษี ที่กำกับดูแลการดำเนินธุรกิจรถเช่าให้เป็นไปตามข้อกำหนด
- พันธมิตรทางธุรกิจ (Business Partners) – เช่น บริษัทประกันภัย หรือผู้ให้บริการชำระเงินออนไลน์ ที่มีบทบาทสนับสนุนการให้บริการในระบบ

1.3 Software Scope

ระบบเช่ารถออนไลน์ ขอบเขตของซอฟต์แวร์จะระบุถึงฟังก์ชันที่ระบบจะดำเนินการและคุณสมบัติที่ระบบต้องมี เพื่อให้การบริการเช่ารถออนไลน์มีประสิทธิภาพและตอบสนองต่อความต้องการของผู้ใช้ได้อย่างครบถ้วน ขอบเขตของซอฟต์แวร์ในระบบเช่ารถออนไลน์ประกอบด้วย:

- ระบบการจองรถ
 - ช่วยให้ลูกค้าสามารถเลือกและจองรถออนไลน์ได้ตลอด 24 ชั่วโมง
 - รองรับการค้นหารถตามประเภท, ยี่ห้อ, ราคา, และสถานที่

- ระบบตรวจสอบความพร้อมของรถและป้องกันการจองซ้ำซ้อน

2. ระบบการชำระเงินออนไลน์

- รองรับการชำระเงินผ่าน QR code

3. ระบบการจัดการรถ

- ช่วยผู้ให้บริการจัดการรถทั้งหมดในระบบ เช่น เพิ่ม, ลบ ข้อมูลรถ และตรวจสอบสถานะของรถ (ว่าง/เช่าอยู่)

4. ระบบการจัดการบัญชีผู้ใช้งาน

- ลูกค้าสามารถสร้างและจัดการบัญชีของตนเอง
- ระบบเก็บข้อมูลผู้ใช้ เช่น ประวัติการเช่า, การชำระเงิน

5. ระบบการยืนยันและจัดการเอกสาร

- ลูกค้าสามารถอัปโหลดเอกสารที่จำเป็น เช่น ข้อมูลประจำตัว ชื่อจริง นามสกุล
- ระบบจะตรวจสอบเอกสารและยืนยันความถูกต้องก่อนการจองรถ

6. ระบบการจัดการการจอง

- ระบบช่วยในการยืนยันการจองตามคำขอของลูกค้า

1.4 Software purpose

ระบบเช่ารถออนไลน์มีวัตถุประสงค์หลักในการพัฒนาแพลตฟอร์มที่ช่วยให้ลูกค้าสามารถจองรถได้สะดวก รวดเร็ว และปลอดภัย โดยไม่ต้องผ่านกระบวนการห้ามซื้อซ้ำแบบเดิม ระบบนี้ช่วยเพิ่มความสะดวกในการจองรถ พร้อมทั้งปรับปรุงประสิทธิภาพการจัดการรถโดยให้มีความแม่นยำและทันสมัย

อีกทั้งยังเน้นความปลอดภัยในการจัดเก็บข้อมูลผู้ใช้และธุรกรรมการเงินออนไลน์ รวมถึงการลดข้อผิดพลาดในการจองผ่านระบบอัตโนมัติที่ช่วยอัปเดตสถานะการจอง ระบบยังมุ่งเน้นการสร้างประสบการณ์ที่ดีสำหรับลูกค้าผ่านการสนับสนุนที่สะดวกและทันเวลา

System Architecture

Client	User Admin Staff
Access	Web browser (Google Chrome, Firefox, Microsoft Edge) Mobile Responsive Design
Service	Car Search & Reservation System Rental Management Payment & Billing System User Account Management Admin Management
Security	JSONWEB-TOKEN Authentication (JWT) Role-based Access Control (RBAC)
Database	Car Rental Management Database (MySQL)

User Interface Prototype

การออกแบบเบื้องต้นของหน้าจอหรืออินเทอร์เฟซที่ผู้ใช้จะโต้ตอบกับระบบในระหว่างการใช้งาน โดยมีวัตถุประสงค์เพื่อแสดงให้เห็นถึงการจัดวางองค์ประกอบต่าง ๆ และการทำงานของระบบในรูปแบบที่สามารถทดลองใช้งานได้จริงในระดับเบื้องต้น

1. Signin

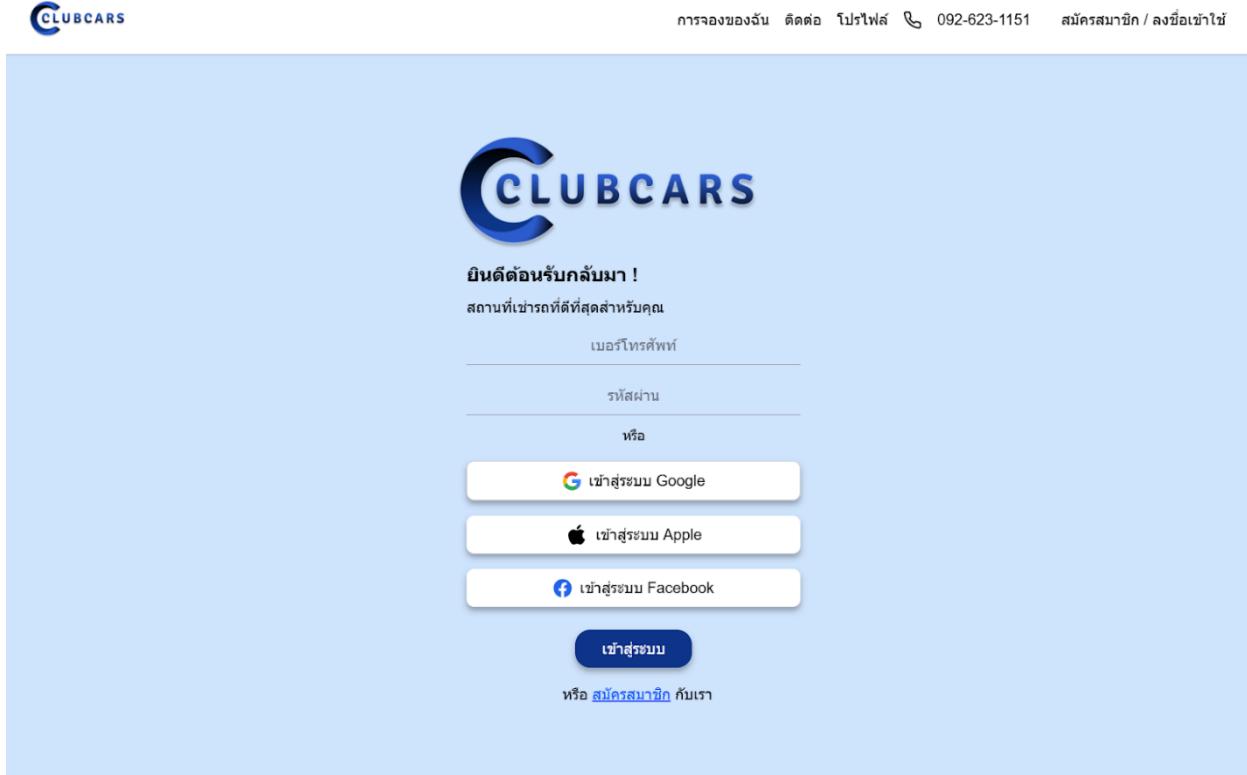
เป็นหน้าสมัครเข้าใช้งานเพื่อเข้าสู่ระบบโดยต้องกรอกข้อมูล firstName lastName Username Phone และ Password ในแต่ละช่อง

The screenshot shows a sign-in form for 'CLUBCARS'. At the top right, there are links for 'การจองของฉัน' (My Bookings), 'ติดต่อ' (Contact), 'โปรไฟล์' (Profile) with a phone icon, '092-623-1151', and 'สมัครสมาชิก / ลงทะเบียนเข้าใช้' (Sign up / Register). The main form has a light blue header with the 'CLUBCARS' logo. Below it, the text 'สมัครสมาชิก' (Sign up) is centered. The form contains five input fields: 'ชื่อ' (Name), 'นามสกุล' (Last name), 'ชื่อผู้ใช้' (Username), 'เมืองที่อาศัยอยู่' (City), and 'รหัสผ่าน' (Password). A large blue button labeled 'สมัครสมาชิก' (Sign up) is at the bottom. Below the button, a link 'มีบัญชีอยู่แล้ว? เข้าสู่ระบบ' (Already have an account? Log in) is shown.

ภาพที่ 1 หน้าจอสมัครเข้าใช้งาน

2. Login

เป็นหน้าจอเข้าสู่ระบบของผู้ใช้งาน ซึ่งผู้ใช้งานจะต้องกรอกชื่อผู้ใช้งาน (Username) และรหัสผ่าน (Password) เพื่อทำการเข้าสู่ระบบ และสามารถเข้าสู่ระบบด้วยบัญชีของ Google



ກາພທີ່ 2 ມີຫຼາຍລັດຖະບານ

3. Home

หน้าแรกหรือหน้า Home ของเว็บไซต์เป็นหน้าหลักที่ผู้ใช้งานจะพบเมื่อเข้าสู่เว็บไซต์ โดยมีการนำเสนอข้อมูลสำคัญและพิเจอร์หลักที่ผู้ใช้สามารถเข้าถึงได้ง่าย และสะดวก

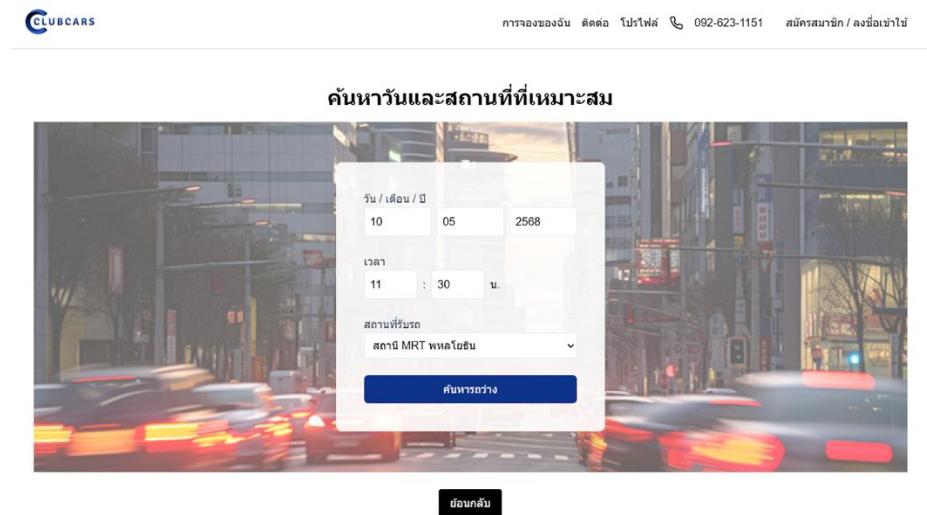
The screenshot shows the ClubCars website homepage. At the top, there's a large banner with a scenic road and a person's arm reaching out from a car window. The text "คันหารถเช่ากับเราได้" (Get a car rental with us) is overlaid. Below the banner, there's a promotional section for "FLASH SALE" with images of cars and people driving. The main menu includes links like "จองรถ" (Book a car), "เช่ารถตามคาด" (Rent by estimate), and "เช่ารถง่ายๆ" (Easy car rental). The footer contains contact information: 092-623-1151, ลูกค้าสมาชิก / ลงรีวิวเจ้าใจ.

The screenshot shows the ClubCars website homepage. At the top, there's a large banner with a scenic road and a person's arm reaching out from a car window. The text "คันหารถเช่ากับเราได้" (Get a car rental with us) is overlaid. Below the banner, there's a promotional section for "FLASH SALE" with images of cars and people driving. The main menu includes links like "จองรถ" (Book a car), "เช่ารถตามคาด" (Rent by estimate), and "เช่ารถง่ายๆ" (Easy car rental). The footer contains contact information: 092-623-1151, ลูกค้าสมาชิก / ลงรีวิวเจ้าใจ.

ภาพที่ 3 หน้าแรก

4. SearchDate

เป็นหน้าที่ให้ผู้ใช้เลือกวันและเวลาที่ต้องการทำการจองรถยนต์ โดยสามารถกรอกข้อมูลต่าง ๆ ได้ เช่น วัน/เดือน/ปี เวลา และสถานที่รับรถเช่า



ภาพที่ 4 ค้นหาร้านและสถานที่รับรถ

5. SearchCars

เป็นหน้าผลลัพธ์การค้นหารถยนต์ที่ตรงกับวันที่และสถานที่ที่ผู้ใช้งานได้เลือกไว้ในหน้า SearchDate โดยจะทำการแสดงผลข้อมูลของรถยนต์ที่สามารถจองได้ในช่วงเวลาันนี้

CLUBCARS

การจองของฉัน ติดต่อ โทร. 092-623-1151 สัมภาษณ์เช่าใช้

จุดรับ-คืนรถ
สนามบินดอนเมือง

วัน-เวลาข้อมูล
29/03/2025 11:30 น.

วัน-เวลาคืนรถ
29/03/2025 11:30 น.

ค้นหารถเช่า

ด้วยการค้นหา : รถว่างทั้งหมด

รถว่างตามเงื่อนไข

รถเก่ง

สถานที่รับรถ
สนามบินดอนเมือง

สถานที่คืนรถ
สนามบินดอนเมือง

ระยะเวลาเช่า (วัน)
ระบุจำนวนวัน

ค้นหา

Toyota Yaris ATIV 2023

รถเก่ง 4 ประตู เกียร์อัตโนมัติ

พื้นที่บริการกับขั้น 1

ประเภท: รถเก่ง (Sedan)

จำนวนที่นั่ง: 4-5 ที่นั่ง

1,500 บาท / วัน

รายละเอียดรถเช่า

Toyota Yaris ATIV 2023

รถเก่ง 4 ประตู เกียร์อัตโนมัติ

พื้นที่บริการกับขั้น 1

ภาพที่ 5 รถยนต์ที่ว่างให้เช่า

6. Rentdetail

เป็นหน้าที่แสดงรายละเอียดของรถยนต์ที่ผู้ใช้งานเลือกจากหน้า SearchCars

ซึ่งในหน้านี้ผู้ใช้จะได้รับข้อมูลเกี่ยวกับรถยนต์ที่สามารถช่วยในการตัดสินใจจองรถ เช่น ชื่อรุ่นรถ ประเภทของรถ จำนวนที่นั่ง ราคาเช่า

The screenshot shows the ClubCars website interface for a car rental listing. At the top, there's a logo for CLUBCARS and some navigation text in Thai. The main title is "รายละเอียดรถเช่า" (Rental Car Details) for a "Mercedes-AMG C 43 Coupe 2". Below the title, there are three images: the interior of the car, the exterior side view, and the interior front seats. To the left, there's a sidebar with icons and text for "ประวัติรถ รถเก่ง" (Car History), "ที่นี่เป็น 4 ที่นั่ง" (This is a 4-seater), "กรุงเทพฯ 2 ระยะ" (Bangkok 2-stage), "นราธิวาส 2 ระยะ" (Narathiwat 2-stage), "เกียร์ ออโต้" (Automatic Gear), and "เชื้อเพลิง เบนซิน" (Fuel Type: Gasoline). On the right, there's a detailed breakdown of the rental costs: "รายละเอียดการจอง" (Booking Details) showing "ราคารถต่อวัน 4,444 x 1 วัน" (Daily rate 4,444 x 1 day), "จำนวนวัน: 1" (Number of days: 1), "ค่าเช่ารถ 1 วัน" (Daily rental fee: B4,444), "ค่ารับ - ค่าส่ง" (Pick-up - Delivery fee: B200), "ค่าส่งรถ 100, ค่าวันรถ 100" (Delivery fee 100, Daily car fee 100), "ค่าน้ำดื่ม" (Drinking water fee), "ค่าน้ำดื่มในวันรับรถ (ได้ศึกวันเดินรถ)" (Drinking water on the day of collection (calculated by day)), and "ราคารถทั้งหมด: B5,000" (Total car cost: B5,000). A note at the bottom says "ราคารถทั้งหมด: B5,000" (Total car cost: B5,000). At the bottom left, it says "ราคารถทั้งหมด: B4,644" (Total car cost: B4,644), and at the bottom right, there's a blue button labeled "เช่ารถคันนี้" (Rent this car).

ภาพที่ 6 รายละเอียดรถเช่า

7. Rental

เป็นหน้าที่ผู้ใช้งานสามารถดูรายละเอียดการจองรถที่ได้ทำการจองไปแล้ว หรือการจองที่กำลังดำเนินการอยู่ ซึ่งช่วยให้ผู้ใช้งานสามารถตรวจสอบสถานะการจองรถเช่าได้ และจัดการข้อมูลต่าง ๆ เกี่ยวกับการเช่ารถ

CLUBCARS

การจองของฉัน

Toyota Camry 2023

หมายเลขอุตสาหกรรม: 21360

วัน-เวลาเดินทาง: 02/04/2025 09:00 น. 3 วัน

สถานะการจอง: ยืนยันการเช่า

สถานะการจอง: ยังไม่ได้รับการยืนยัน

สถานะการจอง: 04/04/2025 15:00 น.

Toyota Corolla 2023

หมายเลขอุตสาหกรรม: 21357

วัน-เวลาเดินทาง: 01/04/2025 10:00 น. 4 วัน

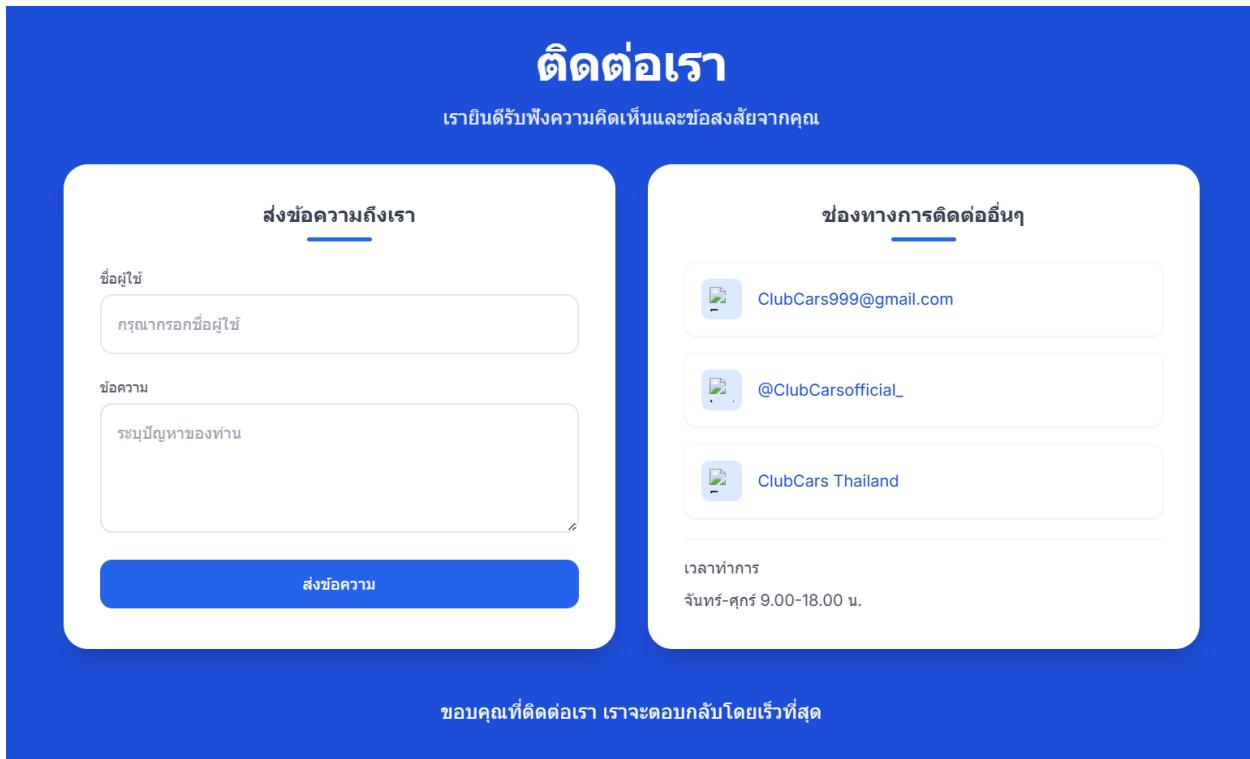
สถานะการจอง: รอการยืนยัน

สถานะการจอง: 04/04/2025 14:00 น.

ภาพที่ 7 ประวัติการจองรถเช่า

8. Contact

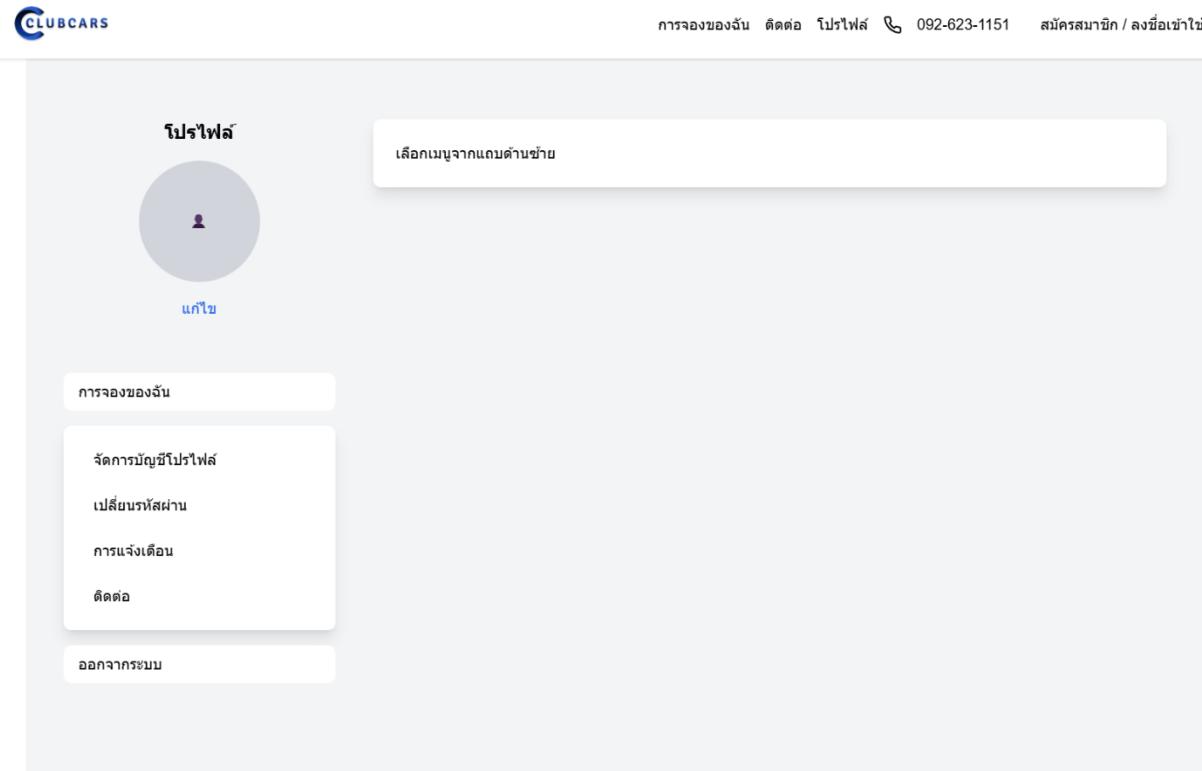
เป็นหน้าที่ผู้ใช้งานสามารถติดต่อหรือสอบถามข้อมูลเพิ่มเติมจากทีมงานหรือฝ่ายบริการลูกค้าได้โดยหน้าติดต่อจะมีหลักหลายช่องทางให้ผู้ใช้เลือกติดต่อ หรือส่งคำถามได้ตามที่ผู้ใช้สะดวก



ภาพที่ 8 ติดต่อ / สอบถาม

9. Profile

เป็นหน้าที่ผู้ใช้งานสามารถเข้าถึงและจัดการข้อมูลส่วนตัวของตัวเองในระบบ เช่น ข้อมูลส่วนตัว การจัดการรหัสผ่าน ประวัติการจองการตั้งค่าการแจ้งเตือน และการออกจากระบบ



Admin

1. Dashboard

เป็นหน้าหลักในการทำงานของ Admin โดยจะมีการนำเสนอฟีเจอร์หลักที่สำคัญ เช่น การจัดการข้อมูลลูกค้า การจัดการข้อมูลรถ และการจัดการข้อมูลพนักงาน

Dashboard

Manage all aspects of the platform efficiently with a modern interface.

[Logout](#)



2. การจัดการพนักงาน

เป็นหน้าที่ Admin สามารถจัดการเกี่ยวกับข้อมูลของพนักงานได้ รวมไปถึงสามารถเพิ่มรายชื่อและเบอร์โทรศัพท์ของพนักงานได้

การจัดการพนักงาน

[เพิ่มพนักงาน](#) [ย้อนกลับ](#)

[ทั้งหมด](#)

ค้นหาจากรหัสพนักงาน, ชื่อ, อีเมล, เมอร์ไพร...

ชื่อ-นามสกุล	เบอร์โทรศัพท์	การจัดการ
jnsfjk lsbfnblk	25662	
ธีร์ ศรี	077659634	
ตั้ม ตาม	08560245	
pp tt	789456123	

[ย้อนกลับ](#) [1](#) [2](#) [3](#) [ถัดไป](#)

3. การเพิ่มพนักงาน

Admin สามารถทำการเพิ่มรายชื่อ นามสกุล และเบอร์โทรศัพท์ของพนักงานใหม่

เพิ่มพนักงานใหม่

ย้อนกลับ

ชื่อ

นามสกุล

เบอร์โทรศัพท์

ยกเลิก

บันทึก

4. การจัดการข้อมูลสถานะรถ

เป็นหน้าที่ admin สามารถเพิ่มข้อมูลรถและสามารถลบรถได้

การจัดการข้อมูลรถ

จัดการข้อมูลรถยนต์ทั้งหมดในระบบ

+ เพิ่มข้อมูลรถใหม่

← ย้อนกลับ

 ค้นหาจากทะเบียนรถ, รุ่น, ยี่ห้อ...

ฟังก์ชัน

พร้อมให้เช่า

กำลังเช่า

ข้อมูลรุ่น

ทะเบียนรถ	รุ่น	ยี่ห้อ	ประเภท	สถานะ	ราคาเช่า	รูปภาพ	การจัดการ
A001	K9	BUBU01	Sedan	พร้อมให้เช่า	100.00	 BUBL	
A002	K009	Wave01	Sedan	พร้อมให้เช่า	120.00	 Wave	
A003	K45	vave3	Pickup	กำลังเช่า	123.00	 vave	

< ย้อนกลับ 1 2 3 สุดไป >

5. การเพิ่มรถ

สามารถเพิ่มรถที่ให้เช่าได้ โดยการกรอกทะเบียนรถ ยี่ห้อรถ รุ่นของรถ ประเภทของรถ ราคาเช่ารายวัน และรูปภาพของรถที่ให้เช่า

เพิ่มรถใหม่

บันทึก

ทะเบียนรถ *

ยี่ห้อ *

รุ่น *

ประเภท *

Sedan

สถานะ *

พร้อมให้เช่า

ราคาเช่า (บาท/วัน) *

รูปภาพรถ *

Choose File No file chosen

ยกเลิก

บันทึก

6. การจัดการข้อมูลลูกค้า

Admin จะสามารถจัดการข้อมูลของลูกค้าได้ และสามารถลบข้อมูลของลูกค้าได้

การจัดการลูกค้า

ย้อนกลับ

ทั้งหมด

ชื่อ-นามสกุล	เบอร์โทร	การจัดการ
pp jj	0881234567	
Staff staff	0877777777	
Ad min	0888888888	
Test ty	0855555555	
Tapanee Siripala	0849875962	
Phattaranun Pansuksawat	0844444444	

ย้อนกลับ 1 2 3 ต่อไป

Staff

1. Dashboard

เป็นหน้าหลักในการทำงานของ Staff โดยจะมีการนำเสนอฟีเจอร์หลักที่สำคัญของ staff เช่น การจัดการเช่ารถ และตารางงานของ staff

แดชบอร์ดพนักงาน

จัดการระบบเช่ารถและตารางงาน

◀ ออกจากระบบ



2. จัดการการเช่ารถ

Staff สามารถจัดการการเช่ารถของลูกค้าได้ โดยสามารถตรวจสอบรายชื่อของลูกค้า การจองและสามารถตรวจสอบได้ว่าลูกค้าได้ทำการจองสำเร็จใหม่

จัดการการเช่ารถ

ระบบจัดการข้อมูลการเช่ารถสำหรับพนักงาน

◀ ย้อนกลับ

รายการเช่ารถ				
ลูกค้า	การจอง	ยอดชำระ	สถานะ	ดำเนินการ
สมชาย ใจดี	#BK001 Toyota Camry วันที่จอง: 2023-06-15	2,500 บาท	รอดตรวจสอบ	✖️ ✓️ ✖️
สunita เก่งมาก	#BK002 Honda Civic วันที่จอง: 2023-06-16	1,800 บาท	ยืนยันแล้ว	✖️
อนุชา สน้ายดี	#BK003 Ford Mustang วันที่จอง: 2023-06-18	3,500 บาท	ปฏิเสธ	✖️

3. ตารางงานของพนักงาน

สามารถตรวจสอบตารางงานแต่ละวันได้ ตรวจสอบได้ว่าพนักงานคนไหนเข้าทำงานวันไหนบ้าง

ตารางงานพนักงาน

ระบบจัดการตารางงานสำหรับพนักงาน

< ย้อนกลับ

◀ เมษายน 2568 ▶

อาทิตย์	จันทร์	อังคาร	พุธ	พฤหัสบดี	ศุกร์	เสาร์
		1	+ 2	+ 3	+ 4	+ 5
6	+ 7	+ 8	+ 9	+ 10	+ 11	+ 12
13	+ 14	+ 15	+ 16	+ 17	+ 18	+ 19
20	+ 21	+ 22	+ 23	+ 24	+ 25	+ 26
27	+ 28	+ 29	+ 30	+		

รายชื่อพนักงาน

ผู้ดูแล	ID
คุณ ต.	ID: 150003 077659634
คุณ ตาม	ID: 150004 08560245
คุณ ต.	ID: 210001 0865478902

Specification Requirements

1. User Requirements

1.1 ระบบจองรถออนไลน์

- ระบบควรรองรับการจองรถตลอด 24 ชั่วโมงทุกวัน
- ผู้ใช้สามารถเลือกวันที่เริ่มต้นและสิ้นสุดการเช่า รวมถึงสถานที่รับและคืนรถ
- ระบบจะแสดงประเภทของรถที่ว่าง เช่น Sedan, SUV, Hatchback

1.2 การชำระเงินออนไลน์

- ระบบรองรับการชำระเงินผ่านการโอนผ่านธนาคารและมีการยืนยันการชำระเงินแบบเรียลไทม์

1.3 การจัดการprofileลูกค้า

- ผู้ใช้สามารถสร้างบัญชีเพื่อบันทึกข้อมูลส่วนตัว (เช่น ข้อมูลการติดต่อ) และประวัติการจอง
- ระบบควรส่งการยืนยันการจองและการแจ้งเตือนสำหรับวันรับและคืนรถผ่านระบบ

1.4 การยืนยันการจองและระบบแจ้งเตือน

- ระบบควรส่งการยืนยันการจองและการชำระเงินผ่านระบบ
- การแจ้งเตือนรถที่จองสำเร็จแสดงรายละเอียด เช่น ประเภทรถ, ระยะเวลาการเช่า, สถานที่รับและคืนรถ, และค่าใช้จ่ายทั้งหมด

1.5 การจัดการหลังการจอง

- ผู้ใช้สามารถยกเลิกการจองได้ภายในระยะเวลาที่กำหนด

1.6 การจัดการโปรโมชั่น

- ระบบควรรองรับการใช้คูปองหรือส่วนลดในระหว่างการจอง

1.7 ความปลอดภัยของข้อมูล

- การควบคุมการเข้าถึงแบบกำหนดบทบาท (Role-Based Access Control) เพื่อให้ลูกค้า, เจ้าหน้าที่, และผู้ดูแลระบบเข้าถึงข้อมูลตามสิทธิ์ของตน

2. System Requirements

2.1 Functional Requirements

ฟังก์ชันสำหรับลูกค้า

- ลูกค้าสามารถค้นหาและจองรถที่ว่างได้
- ลูกค้าสามารถชำระเงินและดูแจ้งเตือนได้
- ลูกค้าสามารถยกเลิกการจองได้ตามนโยบาย
- ลูกค้าจะได้รับการแจ้งเตือนสำหรับการยืนยันการจอง การยกเลิกการจอง

ฟังก์ชันการจอง

- ยืนยัน, ยกเลิกการจอง
- ตรวจสอบสถานะการจอง
- เชื่อมโยงข้อมูลการชำระเงินกับการจอง

ฟังก์ชันสำหรับผู้ดูแลระบบ (Admin)

- เพิ่มรถในระบบ
- จัดการการชำระเงินของลูกค้า
- จัดการบัญชีลูกค้า เพิ่มบัญชี ลบบัญชี
- จัดการและกำหนดราคาค่าเช่า
- จัดการรถ เพิ่มรถ, ลบรถ

ฟังก์ชันการชำระเงิน

- ประมวลผลการชำระเงินอย่างปลอดภัย
- ตรวจสอบสถานะการชำระเงิน

ฟังก์ชันสำหรับเจ้าหน้าที่(Staff)

- จัดการความพร้อมใช้งานของรถ
- ตรวจสอบสภาพรถในขั้นตอนการรับและคืนรถ
- สร้างรายงานการปฏิบัติงานประจำวัน
- สามารถดูตารางงานของตนเองได้

ฟังก์ชันการแจ้งเตือน

- ส่งการแจ้งเตือนสำหรับการจอง, การชำระเงิน
- อัปเดตสถานะการแจ้งเตือนเมื่อผู้ใช้ดำเนินการแล้ว

2.2 Non-Functional Requirements

ประสิทธิภาพ

- ระบบต้องสามารถรับผู้ใช้งานหลายรายพร้อมกันโดยไม่เกิดความล่าช้าในการค้นหา, การจอง, หรือการชำระเงิน
- การดึงข้อมูลจากฐานข้อมูลควรได้รับการปรับแต่งให้มีเวลาตอบสนองรวดเร็วโดยเฉพาะในช่วงที่มีการใช้งานสูง

การขยายตัว

- ระบบควรสามารถขยายตัวเพื่อรับจำนวนผู้ใช้, รถ, และการจองที่เพิ่มขึ้นเมื่อธุรกิจเติบโต

ความน่าเชื่อถือ

- ระบบต้องมีอัตราการพร้อมใช้งาน (Uptime) อย่างน้อย 99.5% และให้บริการจอง, ชำระเงิน, และการแจ้งเตือนอย่างต่อเนื่อง
- ข้อมูลระหว่างการจองและการชำระเงินต้องมีความสอดคล้องกันเสมอ

ความปลอดภัย

- และการกำหนดสิทธิ์ตามบทบาท
- มีการเข้ารหัสผ่านก่อนเก็บเข้าฐานข้อมูล

การใช้งาน

- อินเทอร์เฟซควรใช้งานง่าย พร้อมข้อความแสดงข้อผิดพลาดที่ชัดเจน
- รองรับการใช้งานทั้งบนคอมพิวเตอร์และอุปกรณ์เคลื่อนที่ เพื่อประสบการณ์การใช้งานที่ต่อเนื่อง

การบำรุงรักษา

- ระบบควรได้รับการออกแบบให้สามารถอัปเดตและเพิ่มฟีเจอร์ใหม่ได้ง่ายโดยไม่กระทบต่อการทำงานของพังก์ชันเดิม
- ควรมีการสำรองข้อมูลและการตรวจสอบระบบอย่างสม่ำเสมอ เพื่อให้มั่นใจว่าระบบทำงานได้อย่างต่อเนื่อง

2.3 Domian requirement

1. ข้อกำหนดด้านธุรกิจ

1.1 การบริหารจัดการรถ

- ระบบต้องรองรับการบันทึกข้อมูลรถเช่า เช่น:
 - หมายเลขทะเบียนรถ
 - ประเภทรถ
 - รุ่นและยี่ห้อ
 - สถานะการใช้งาน
- รถแต่ละคันต้องเชื่อมโยงกับสถานที่จัดเก็บ

1.2 การกำหนดราคาค่าเช่า

- ระบบต้องรองรับการตั้งราคาค่าเช่าตาม:
 - ประเภท
 - ระยะเวลาเช่า (รายวัน/รายสัปดาห์)

1.3 การจัดการสาขาและพื้นที่

- ระบบต้องรองรับหลายสาขา โดยสามารถกำหนด:
 - สถานที่รับรถและคืนรถตามกำหนด

1.4 การจัดการลูกค้า

- ต้องมีการตรวจสอบข้อมูลลูกค้า เช่น:
 - ชื่อจริง นามสกุล
 - การชำระเงิน

2. ข้อกำหนดด้านกฎหมาย

2.1 กฎหมายเกี่ยวกับการเช่ารถ

- ระบบมีข้อมูลสัญญาการเช่า เช่น:
 - ข้อมูลลูกค้า (ชื่อ, ที่อยู่, เบอร์โทรศัพท์)
 - ข้อมูลรถเช่า (ทะเบียน, รุ่น, หมายเลขเครื่องยนต์)

2.2 ความเป็นส่วนตัวของข้อมูล (Privacy Law)

- ระบบต้องปฏิบัติตามข้อกำหนดด้านความปลอดภัยของข้อมูล เช่น:
 - พระราชบัญญัติคุ้มครองข้อมูลส่วนบุคคล (PDPA) หรือ GDPR
 - สำหรับลูกค้าในต่างประเทศ
 - ข้อมูลส่วนบุคคล
 - ข้อมูลการชำระเงิน ต้องได้รับการป้องกัน

2.3 กฎหมายการชำระเงิน

- ระบบต้องรองรับการตรวจสอบและเก็บหลักฐานการชำระเงินตามมาตรฐานของธนาคารและหน่วยงานการเงิน
- ต้องเก็บข้อมูลการชำระเงินอย่างปลอดภัย

3. ข้อกำหนดด้านการดำเนินงาน

3.1 การตรวจสอบก่อนและหลังการเช่า

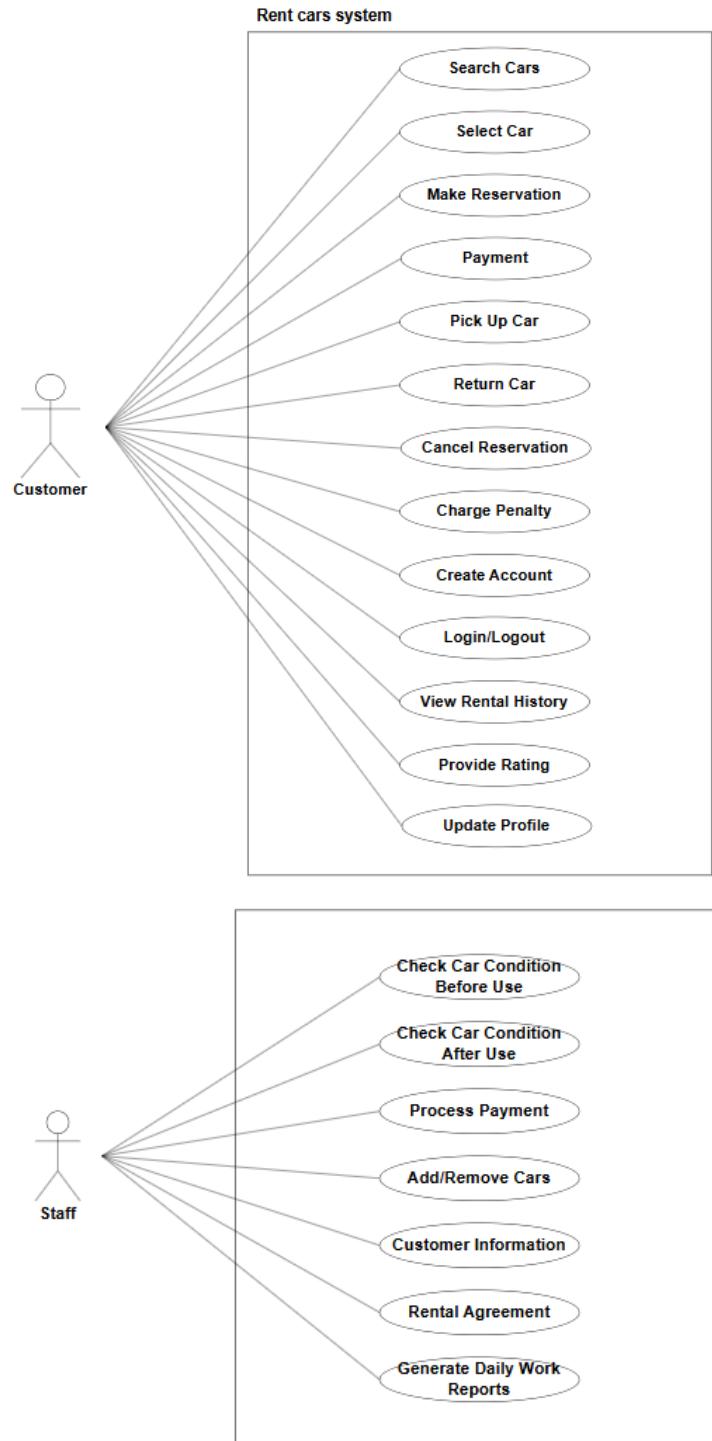
- ระบบต้องรองรับการบันทึกสถานะรถ:
 - สภาพรถก่อนการเช่า (เช่น มีรอยขีดข่วน, ระดับน้ำมัน)
 - สภาพรถหลังการคืนรถ

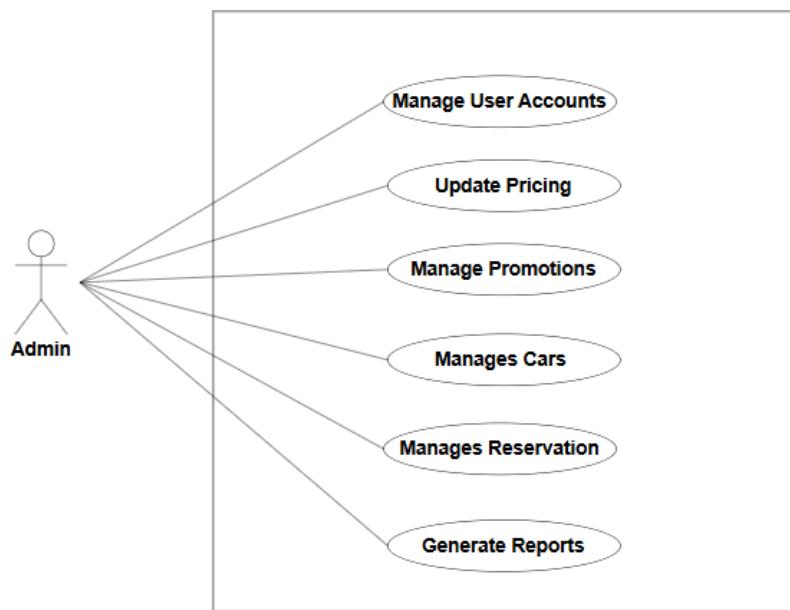
4. ข้อกำหนดด้านธุรกรรมและการรายงาน

4.1 การบันทึกธุรกรรม

- ระบบต้องจัดเก็บประวัติการจองและการชำระเงินทั้งหมด
- ต้องสามารถตรวจสอบการเช่าทั้งหมดของลูกค้าแต่ละรายได้

Use - case Diagram

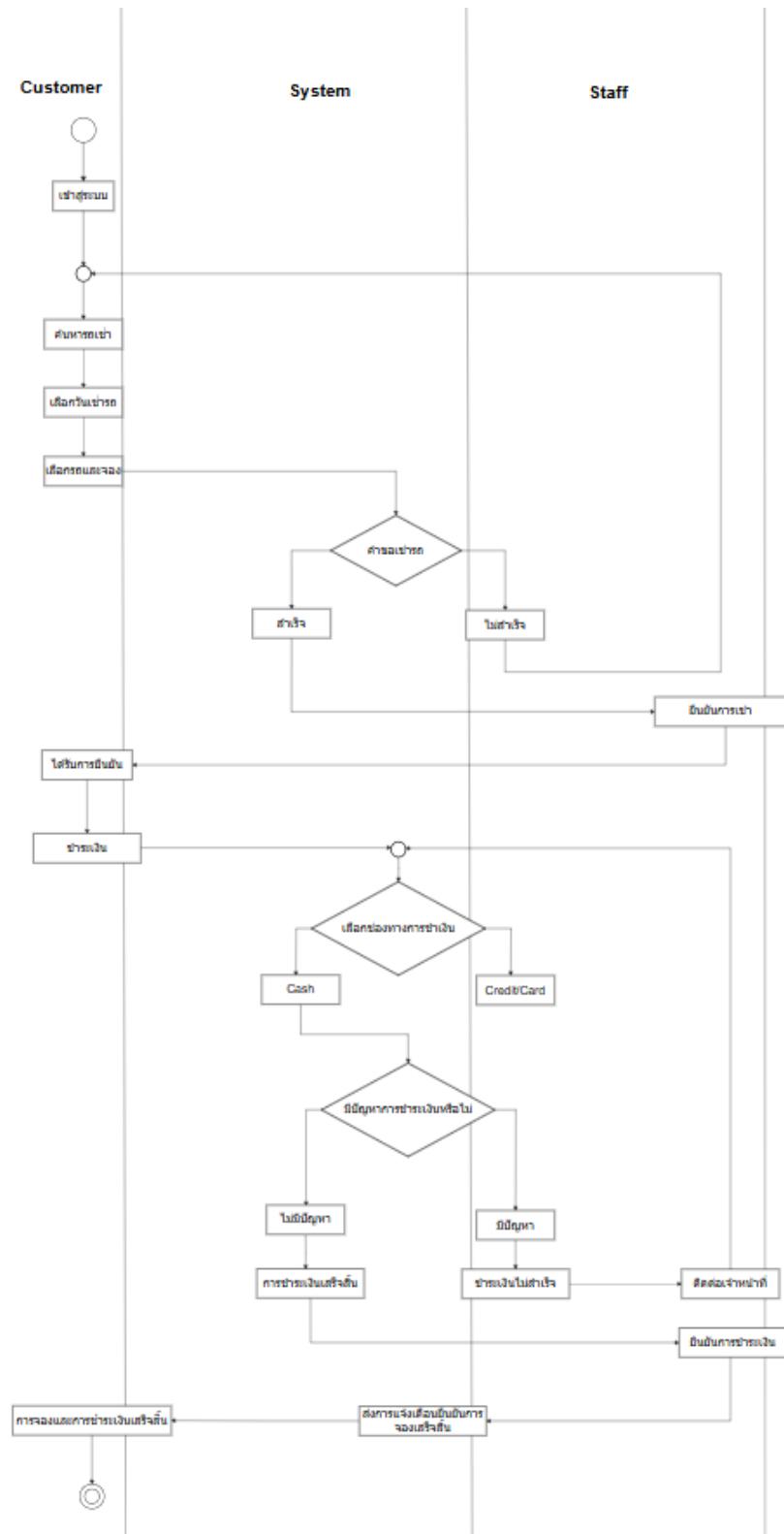




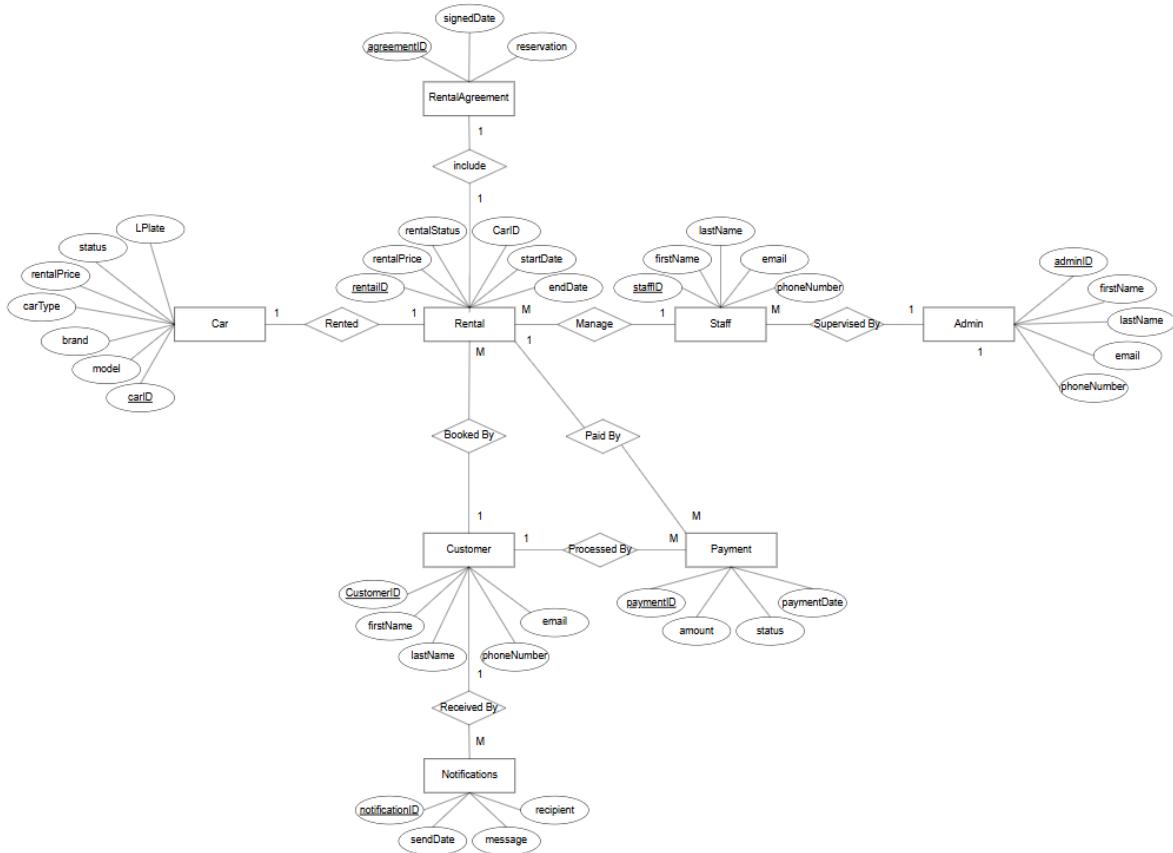
Class Diagram



Activity Diagram



ER diagram



User Acceptance Test (UAT)

วัตถุประสงค์

การทดสอบนี้มีเป้าหมายเพื่อตรวจสอบว่า ระบบเข้ารูปทำงานได้ตรงตามข้อกำหนดของผู้ใช้งานและสามารถดำเนินการได้อย่างถูกต้องก่อนเปิดใช้งานจริง

ขอบเขตของการทดสอบ

- การลงทะเบียนและเข้าสู่ระบบ
- การค้นหาและจองรถ
- การชำระเงินและการทำธุรกรรม
- การจัดการการจอง
- การคืนรถและการคำนวณค่าใช้จ่าย
- การจัดการผู้ดูแลระบบและการจัดการรถและพนักงาน
- ความปลอดภัยและการรักษาข้อมูล
- ประสิทธิภาพของระบบ
- การใช้งานบนอุปกรณ์ต่างๆ

เงื่อนไขการทดสอบ (Acceptance Criteria)

- ระบบต้องแสดงข้อผิดพลาดที่เหมาะสมเมื่อเกิดปัญหา
- ระบบต้องป้องกันข้อผิดพลาดที่ทำให้เกิดช่องโหว่
- ระบบต้องแจ้งเตือนผู้ใช้ในกรณีที่เกิดข้อผิดพลาดระหว่างใช้งาน

Test case 1 : Customer

การลงทะเบียนและเข้าสู่ระบบ

รหัสทดสอบ	UAT-Cac-001	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วนการ ทำงาน	ระบบการเข้าสู่ระบบ/ล็อกอิน/รหัสผ่าน		
ชื่อการทำงาน	การจัดการการเข้าสู่ระบบและสมัครสมาชิกและจัดการรหัสผ่าน		
รหัสหน้าจอ-ชื่อ หน้าจอ	1.Cac1001 - การลงทะเบียนสมาชิกใหม่ 2.Cac1002 - การเข้าสู่ระบบ 3.Cac1003 – การลืมรหัสผ่านและขอ reset รหัสผ่าน		
คำอธิบาย	เป็นในส่วนของการเข้าสู่ระบบและล็อกอินและการขอเปลี่ยนรหัสผ่าน		
เงื่อนไขในการ ทำงาน	ต้องเข้าหน้าระบบเพื่อมาสู่การเข้าสู่ระบบหรือสมัครสมาชิก หรือการกดลืมรหัสผ่าน ในหน้าเข้าสู่ระบบ		
ขั้นตอนการ ทำงาน	1. Cac1002 1. เปิดหน้าลงทะเบียน 2. กรอกข้อมูลที่จำเป็น 3. กดปุ่มสมัครสมาชิก 2.Cac1001 1. เปิดหน้าล็อกอิน 2. กรอกเบอร์โทรศัพท์และรหัสผ่าน 3. กดปุ่มเข้าสู่ระบบ 3.Cac1003 1. กด "ลืมรหัสผ่าน" 2. ตั้งค่ารหัสผ่านใหม่ 3. แจ้งเตือนตั้งรหัสผ่านสำเร็จ		
ผลที่คาดว่าจะ ได้รับ	1.Cac1001 ระบบต้องแจ้งว่าสมัครสำเร็จและสามารถเข้าสู่ระบบได้ 2.Cac1002 ระบบต้องนำผู้ใช้เข้าสู่หน้าหลัก 3.Cac1003 ระบบต้องแจ้งเตือนและให้ผู้ใช้ตั้งรหัสผ่านใหม่สำเร็จ		
การทดสอบการทำงานแบบไม่สมบูรณ์			
การกระทำ	ผลการทำงานที่คาดว่าจะ ได้รับ	ผลการทดสอบ	หมายเหตุ

1.Cac1001 1.1 การกรอก เบอร์โทรศัพท์ หรือ password ที่ไม่ ถูกต้อง 1.2 กรอกข้อมูลที่ไม่ใช่ ตัวเลข	1.Cac1001 ระบบจะขึ้นแจ้งเตือนขึ้นมา และไม่อนุญาตให้เข้าสู่ ระบบ 2.Cac1002 2.1 ระบบจะแจ้งเตือนและ ว่าผิดพลาดที่จุดไหนและไม่ อนุญาตให้ลงทะเบียน สมาชิก	1.Cac1001 ผ่าน 2.Cac1002 ผ่าน
2.Cac1002 2.1การกรอกข้อมูลไม่ ครบหรือผิดหลักของ การเข้าสู่ระบบ	3.Cac1003 3.2 ระบบขึ้นแจ้งเตือนว่า ไม่พบบัญชีผู้ใช้	3.Cac1003 ไม่ผ่าน

**Test case 2 : Customer
ค้นหารถว่างและการจอง**

รหัสทดสอบ	UAT-CB-002	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วน การทำงาน	ระบบการค้นหารถว่างและการจอง		
ชื่อการ ทำงาน	จัดการค้นหารถว่างและการจอง		
รหัสหน้าจอ- ชื่อหน้าจอ	1. CB2001 ค้นหารถโดยตัวกรอง 2. CB2002 การแสดงผลรายละเอียดและราคา 3. CB2003 จองรถ		
คำอธิบาย	ส่วนของการค้นหา/แสดงผลตามลูกค้าต้องการ/การจอง		
เงื่อนไขใน การทำงาน	1.เข้ามาในส่วนของการค้นหารถและใช้ตัวกรองเพื่อค้นหารถ 2.กดดูรายละเอียดของรถที่สนใจ 3.ทำการเลือกรถที่ต้องการและทำการกดจองผ่านระบบ		
ขั้นตอนการ ทำงาน	1. CB2001 1. ป้อนสถานที่และวันที่ 2. ใช้ตัวกรอง 3. กดค้นหา 2.CB2002 1. เลือกรถจากรายการ 2. ดูรายละเอียดและราคาทั้งหมด 3.CB2003 1. เลือกรถที่ต้องการ 2. กรอกข้อมูลผู้เช่า 3. กดยืนยันการจอง		
การทดสอบการทำงานแบบไม่สมบูรณ์			
การกระทำ	ผลการทำงานที่คาดว่าจะ ^{ได้รับ}	ผลการทดสอบ	หมายเหตุ
1.CB2001 ค้นหารถโดยใช้ตัว กรองผิดพลาด 3.CB2003	1.CB2001 ระบบต้องแจ้งเตือนว่าไม่มี รถที่ตรงกับเงื่อนไข 3.CB2003	1.CB2001 ไม่ผ่าน 3.CB2003 ผ่าน	

จองรถแต่ไม่ได้กรอก ข้อมูลผู้ใช้	ระบบต้องแจ้งเตือนให้ กรอกข้อมูลก่อน ดำเนินการ		
------------------------------------	---	--	--

Test case 3 : Customer
การชำระเงินและการทำธุรกรรม

รหัสทดสอบ	UAT-CPM-003	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วนการทำงาน	ระบบการชำระเงินผ่านช่องทางต่าง ๆ และการตรวจสอบการชำระเงิน และการขอคืนเงิน		
ชื่อการทำงาน	การจัดการการชำระเงิน, ตรวจสอบการชำระเงินและการขอคืนเงิน		
รหัสหน้าจอ-ชื่อหน้าจอ	1.CPM3002 - ชำระเงินผ่านธนาคารออนไลน์(QR code) 2.CPM3003 - ตรวจสอบสถานะการชำระเงิน		
คำอธิบาย	เป็นในส่วนของการชำระเงินและคืนเงินรวมถึงการตรวจสอบ		
เงื่อนไขในการทำงาน	ต้องมีการจองรถเกิดขึ้นถึงจะนำไปสู่กระบวนการชำระเงินหรือขอเงินคืน		
ขั้นตอนการทำงาน	1.CPM3002 1.เลือกวิธีการชำระเงิน 2.สแกน QRcode ผ่านระบบ 3.ชำระเงิน 2.CPM3003 1. ไปที่ประวัติการจอง 2. ตรวจสอบสถานะการชำระเงิน 3.CPM3004 1. กดขอยกเลิกการจอง 2. เลือกเหตุผล 3. กดยืนยัน		
ผลที่คาดว่าจะได้รับ	1.CPM3002 ระบบต้องดำเนินการชำระเงินสำเร็จ 2.CPM3003 ระบบต้องแสดงสถานะ "ชำระเงินแล้ว"		
การทดสอบการทำงานแบบไม่สมบูรณ์			
การกระทำ	ผลการทำงานที่คาดว่าจะได้รับ	ผลการทดสอบ	หมายเหตุ

1.CPM3002 เกิดความผิดพลาดใน การแสกนคิวอาร์โค้ด/ สแกนไม่ได้	1.CPM3002 ระบบต้องแจ้งเตือน ข้อผิดพลาดในการชำระเงิน สร้างคิวอาร์โค้ดใหม่	1.CPM3002 ผ่าน 2.CPM3003 ไม่ผ่าน
2.CPM3003 ชำระเงินแล้วแต่ระบบ ไม่อัปเดตสถานะ	2.CPM3003 ระบบต้องแจ้งเตือนปัญหา กับการชำระเงิน	

Test case 4 : Customer

การคืนรถ

รหัสทดสอบ	UAT-Creturn-004	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วนการทำงาน	ระบบการคืนรถสู่บริษัทที่ให้บริการและการคิดค่าปรับหากคืนรถไม่ตรงตามกำหนด		
ชื่อการทำงาน	การจัดการการคืนรถและค่าปรับจากระบบ		
รหัสหน้าจอ-ชื่อหน้าจอ	1.Creturn4001 – คืนรถตรงเวลา		
คำอธิบาย	เป็นในส่วนของการคืนรถตามเวลาที่กำหนด		
เงื่อนไขในการทำงาน	ต้องมีการจองรถเสร็จสิ้นแล้ว ถึงวันเช่าฟุ่มใช้บริการจึงมารับรถมากำหนดและคืนตามกำหนด		
ขั้นตอนการทำงาน	1.Creturn4001 1. นำรถไปคืนที่จุดรับคืน 2. ตรวจสอบสภาพรถ 3. ยืนยันคืนรถ		
ผลที่คาดว่าจะได้รับ	1.Creturn4001 ระบบต้องอัปเดตสถานะการเช่าเป็นคืนแล้ว		

การทดสอบการทำงานแบบไม่สมบูรณ์

การกระทำ	ผลการทำงานที่คาดว่าจะได้รับ	ผลการทดสอบ	หมายเหตุ
1.Creturn4001 คืนรถตรงเวลาแต่ระบบไม่เปลี่ยนสถานะให้เจ้ง กล้ายเป็นคืนล่าช้า	1.Creturn4001 ระบบจะต้องเปลี่ยนสถานะให้หากมีการผิดพลาด	1.Creturn4001 ไม่ผ่าน	

Test case 5 : Customer
ความปลอดภัยและการรักษาข้อมูล

รหัสทดสอบ	UAT-Cse-005	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วนการ ทำงาน	ระบบรักษาความปลอดภัยของผู้ใช้ในระบบ		
ชื่อการทำงาน	การจัดการความปลอดภัย		
รหัสหน้าจอ-ชื่อ หน้าจอ	1.Cse5001 การเข้ารหัสข้อมูลผู้ใช้ 2.Cse5002 ป้องกันการเข้าสู่ระบบด้วยรหัสผิด		
คำอธิบาย	เป็นส่วนของการรักษาความปลอดภัยของ user account		
เงื่อนไขในการ ทำงาน	ต้องมีบัญชีผู้ใช้ของ user ก่อนและมีการใส่รหัสผ่านจาก user		
ขั้นตอนการ ทำงาน	1.Cse5001 1. ตรวจสอบการจัดเก็บข้อมูลส่วนบุคคล 2.Cse5002 1. กรอกรหัสผ่านผิด		
ผลที่คาดว่าจะ ได้รับ	1.Cse5001 • ข้อมูลต้องถูกเข้ารหัสและป้องกันการเข้าถึง 2.Cse5002 • ระบบต้องแจ้งเตือนว่ารหัสผ่าน ผิด		
การทดสอบการทำงานแบบไม่สมบูรณ์			
การกระทำ	ผลการทำงานที่คาดว่าจะ ^{ได้รับ}	ผลการทดสอบ	หมายเหตุ
1.Cse5001 ผู้ใช้สามารถเข้าถึง ^{ข้อมูลของผู้อื่นได้}	1.Cse5001	1.Cse5001 ผ่าน	

	ระบบต้องไม่อนุญาตให้เข้าถึงข้อมูลที่ไม่ใช่ของตนเอง		
--	--	--	--

**Test case 6 : Customer
การใช้งานบนอุปกรณ์ต่าง ๆ**

รหัสทดสอบ	UAT-CDS-006	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วนการทำงาน	ระบบการแสดงผลไปสู่อุปกรณ์ต่าง ๆ		
ชื่อการทำงาน	การแสดงผลหน้าจอของอุปกรณ์ต่าง ๆ		
รหัสหน้าจอ-ชื่อหน้าจอ	1.CDS6001 การใช้งานบนมือถือและ laptop		
คำอธิบาย	เป็นส่วนของการแสดงผลได้อย่างถูกต้องใหม่		
เงื่อนไขในการทำงาน	Website ต้องเสร็จเรียบร้อยและพร้อมให้บริการแล้ว		
ขั้นตอนการทำงาน	1.CDS6001 1. เปิดระบบผ่านมือถือหรือ laptop ต่าง ๆ 2. ทดสอบการใช้งาน		
ผลที่คาดว่าจะได้รับ	1.CDS6001 ระบบต้องแสดงผลถูกต้องและใช้งานง่าย		
การทดสอบการทำงานแบบไม่สมบูรณ์			
การกระทำ	ผลการทำงานที่คาดว่าจะได้รับ	ผลการทดสอบ	หมายเหตุ
1.CDS6001 ระบบไม่สามารถใช้งานบนอุปกรณ์ต่าง ๆ ได้	1.CDS6001 ระบบต้องรองรับการแสดงผลที่เหมาะสม	1.CDS6001 ไม่ผ่าน	

Test Case 7 : Admin
การลงทะเบียนเข้าสู่ระบบ

รหัสทดสอบ	UAT-Adm-007	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วนการทำงาน	ระบบการเข้าสู่ระบบ/ล็อกอิน/รหัสผ่าน		
ชื่อการทำงาน	การจัดการการเข้าสู่ระบบและจัดการรหัสผ่าน		
รหัสหน้าจอ-ชื่อหน้าจอ	1. Adm7001 - การเข้าสู่ระบบ 2. Adm7002 - หน้าแรกหลังจากเข้าสู่ระบบ		
คำอธิบาย	ทดสอบการเข้าสู่ระบบของ Admin โดยการกรอกข้อมูลผู้ใช้และรหัสผ่านที่ถูกต้องและไม่ถูกต้อง เพื่อให้แน่ใจว่าระบบรองรับการทำงานที่ถูกต้อง		
เงื่อนไขในการทำงาน	<ul style="list-style-type: none"> ระบบต้องเสร็จเรียบร้อยและพร้อมให้บริการแล้ว บัญชีผู้ใช้และรหัสผ่านที่ใช้ทดสอบต้องมีอยู่ในระบบ 		
ขั้นตอนการทำงาน	1. Adm7001 - เข้าสู่หน้าเข้าสู่ระบบ(Login) <ul style="list-style-type: none"> กรอกข้อมูลชื่อผู้ใช้ คลิกปุ่ม เข้าสู่ระบบ 2. Adm7002 - ตรวจสอบการแสดงผลหลังจากเข้าสู่ระบบ <ul style="list-style-type: none"> ตรวจสอบว่าเข้าสู่หน้าแดชบอร์ดหรือหน้าแรกของ Admin ได้ถูกต้อง 		
ผลที่คาดว่าจะได้รับ	Adm7002 - ระบบต้องแสดงหน้าแรกหลังจากเข้าสู่ระบบได้ถูกต้อง และสามารถเข้าถึงฟังก์ชันต่าง ๆ ได้		
การทดสอบการทำงานแบบไม่สมบูรณ์			
การกระทำ	ผลการทำงานที่คาดว่าจะได้รับ	ผลการทดสอบ	หมายเหตุ

<p>1. Adm7001 กรอกชื่อผู้ใช้หรือรหัสผ่านที่ไม่ถูกต้อง</p> <p>2. Adm7002 กรอกข้อมูลไม่ครบ</p>	<p>1. Adm7001 ระบบจะแจ้งเตือนข้อผิดพลาด ว่าชื่อผู้ใช้หรือรหัสผ่านไม่ถูกต้อง</p> <p>2. Adm7002 ระบบจะแจ้งเตือนว่า ข้อมูลไม่ครบถ้วนหรือไม่ถูกต้อง และไม่อนุญาตให้เข้าสู่ระบบ</p>	<p>1. Adm7001 ผ่าน</p> <p>2. Adm7002 ผ่าน</p>
--	--	---

Test Case 8 : Admin
การจัดการ Staff Management

รหัสทดสอบ	UAT-Adm-008	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วนการทำงาน	ระบบการจัดการพนักงาน		
ชื่อการทำงาน	การจัดการพนักงาน (Staff Management)		
รหัสหน้าจอ-ชื่อหน้าจอ	1. Adm8001 - การเพิ่มพนักงาน 2. Adm8003 - การลบพนักงาน		
คำอธิบาย	ทดสอบการจัดการพนักงานในระบบของ Admin รวมถึงการเพิ่มและลบข้อมูลพนักงาน		
เงื่อนไขในการทำงาน	<ul style="list-style-type: none"> ระบบต้องสามารถเข้าถึงฟังก์ชันการจัดการพนักงานได้ ข้อมูลพนักงานที่เพิ่มหรือแก้ไขต้องถูกต้องและสอดคล้องกับข้อกำหนดของระบบ 		
ขั้นตอนการทำงาน	<ol style="list-style-type: none"> Adm8001 - เพิ่มพนักงานใหม่ <ul style="list-style-type: none"> กรอกข้อมูลพนักงาน (ชื่อ, ตำแหน่ง, แผนก) กดปุ่ม เพิ่มพนักงาน Adm8003 - ลบพนักงาน <ul style="list-style-type: none"> ค้นหาพนักงานที่ต้องการลบ กดปุ่ม ลบพนักงาน 		
ผลที่คาดว่าจะได้รับ	<ul style="list-style-type: none"> Adm8001 - ระบบต้องเพิ่มพนักงานใหม่ได้อย่างถูกต้อง Adm8003 - ระบบต้องสามารถลบพนักงานได้ 		
การทดสอบการทำงานแบบไม่สมบูรณ์			
การกระทำ	ผลการทำงานที่คาดว่าจะได้รับ	ผลการทดสอบ	หมายเหตุ

<p>1.Adm8001 กรอกข้อมูลไม่ครบถ้วน</p> <p>2. Adm8003 ลบพนักงานที่ยังมีข้อมูลที่เกี่ยวข้องกับระบบอื่น</p>	<p>1.Adm8001 ระบบจะแจ้งเตือนหากข้อมูลไม่ครบถ้วน</p> <p>2.Adm8003 ระบบจะแจ้งเตือนลบข้อมูลเรียบร้อยแล้ว</p>	<p>1. Adm8001 ผ่าน</p> <p>3. Adm8003 ผ่าน</p>	
---	---	---	--

Test Case 9 : Admin
การจัดการ User Management

รหัสทดสอบ	UAT-Adm-0009	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วนการ ทำงาน	ระบบการจัดการผู้ใช้งาน		
ชื่อการทำงาน	การจัดการผู้ใช้งาน (User Management)		
รหัสหน้าจอ- ชื่อหน้าจอ	1. Adm9001 - การแก้ไขข้อมูลผู้ใช้ 2. Adm9002 - การลบผู้ใช้		
คำอธิบาย	ทดสอบการจัดการข้อมูลผู้ใช้งานในระบบของ Admin รวมถึงการแก้ไข และลบข้อมูลผู้ใช้งาน		
เงื่อนไขในการ ทำงาน	<ul style="list-style-type: none"> ข้อมูลผู้ใช้ที่แก้ไขหรือจะลบต้องมีอยู่ในระบบ ระบบต้องสามารถควบคุมการเข้าถึงข้อมูลผู้ใช้ได้อย่างถูกต้อง 		
ขั้นตอนการทำงาน	1. Adm9001 - แก้ไขข้อมูลผู้ใช้ <ul style="list-style-type: none"> ค้นหาผู้ใช้ที่ต้องการแก้ไข แก้ไขข้อมูลที่จำเป็น กดปุ่ม บันทึกการแก้ไข 2. Adm9002 - ลบผู้ใช้ <ul style="list-style-type: none"> ค้นหาผู้ใช้ที่ต้องการลบ กดปุ่ม ลบผู้ใช้ 		
ผลที่คาดว่าจะได้รับ	<ul style="list-style-type: none"> Adm1001 - ระบบต้องแสดงรายการการชำระเงินที่ลูกค้าได้แนบสิปและรอการตรวจสอบ Adm1002 - ระบบต้องอนุมัติการชำระเงินให้หลังจากที่ Admin ตรวจสอบสิปการชำระเงินอย่างถูกต้อง (การอนุมัติการชำระเงินเป็นการตัดสินใจของ Admin เท่านั้น) 		
การทดสอบการทำงานแบบไม่สมบูรณ์			

การกระทำ	ผลการทำงานที่คาดว่าจะได้รับ	ผลการทดสอบ	หมายเหตุ
<p>1. Adm9001 แก้ไขข้อมูลผู้ใช้เมื่อกดต้อง</p> <p>2. Adm9002 ลบผู้ใช้ที่ยังมีการใช้งานในระบบ</p>	<p>1. Adm9001 ระบบจะแจ้งเตือนหากข้อมูลไม่ครบถ้วน</p> <p>2. Adm9002 ระบบจะไม่อนุญาตให้ลบผู้ใช้ที่ยังมีข้อมูลหรือการเชื่อมโยงกับระบบอื่น</p>	<p>1. Adm9001 ไม่ผ่าน</p> <p>2. Adm9002 ไม่ผ่าน</p>	

Test Case 10 : Admin
การตรวจสอบการชำระเงิน (Payment Verification)

รหัสทดสอบ	UAT-Adm-0010	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วนการทำงาน	ระบบการตรวจสอบการชำระเงิน		
ชื่อการทำงาน	การตรวจสอบการชำระเงิน (Payment Verification)		
รหัสหน้าจอ-ชื่อหน้าจอ	1. Adm1001 - การตรวจสอบการชำระเงินที่รอดำเนินการ 2. Adm1002 - การอนุมัติการชำระเงิน		
คำอธิบาย	ทดสอบการตรวจสอบและอนุมัติการชำระเงินในระบบ โดยลูกค้าจะแนบสลิปการชำระเงินและ Admin จะต้องตรวจสอบข้อมูลจากสลิปก่อนที่จะอนุมัติการชำระเงิน		
เงื่อนไขในการทำงาน	<ul style="list-style-type: none"> ลูกค้าต้องได้ทำการอัปโหลดสลิปการชำระเงินแล้ว ระบบต้องแสดงสลิปการชำระเงินในหน้าการตรวจสอบของ Admin Admin ต้องตรวจสอบข้อมูลในสลิปและยืนยันการอนุมัติการชำระเงินได้ 		
ขั้นตอนการทำงาน	1. Adm1001 - ตรวจสอบการชำระเงินที่รอดำเนินการ <ul style="list-style-type: none"> ตรวจสอบรายการการชำระเงินที่ลูกค้าแนบสลิปและการอนุมัติ คลิกที่รายการการชำระเงินเพื่อดูรายละเอียด ตรวจสอบข้อมูลในสลิปการชำระเงินที่ลูกค้าแนบ 2. Adm1002 - อนุมัติการชำระเงิน <ul style="list-style-type: none"> กดปุ่ม ดูสลิป เพื่อตรวจสอบสลิปที่ลูกค้าแนบ ตรวจสอบข้อมูลที่ปรากฏในสลิป (เช่น จำนวนเงิน, วันที่, หมายเลขอการชำระเงิน) หากข้อมูลตรงและถูกต้อง กดปุ่ม อนุมัติการชำระเงิน ระบบต้องบันทึกสถานะการชำระเงินเป็น อนุมัติ และแจ้งผู้ใช้ว่าการชำระเงินได้รับการอนุมัติแล้ว 		

ผลที่คาดว่าจะได้รับ	<ul style="list-style-type: none"> Adm1001 - ระบบต้องแสดงรายการการชำระเงินที่ลูกค้าได้แนบสลิปและรอการตรวจสอบ Adm1002 - ระบบต้องอนุมัติการชำระเงินได้หลังจากที่ Admin ตรวจสอบสลิปการชำระเงินอย่างถูกต้อง (การอนุมัติการชำระเงินเป็นการตัดสินใจของ Admin เท่านั้น) 		
การทดสอบการทำงานแบบไม่สมบูรณ์			
การกระทำ	ผลการทำงานที่คาดว่าจะได้รับ	ผลการทดสอบ	หมายเหตุ
1. Adm1001 ไม่มีการแนบสลิปการชำระเงินจากลูกค้า	1. Adm1001 ระบบจะไม่แสดงสลิปการชำระเงินที่ลูกค้าแนบมา (ไม่มีรูปสลิป)	1. Adm1001 ไม่ผ่าน 2. Adm1002 ไม่ผ่าน	

Test Case 11 : Staff

ระบบเข้าสู่ระบบพนักงาน (Staff Login)

รหัสทดสอบ	UAT-St-0011	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วนการทำงาน	ระบบการเข้าสู่ระบบ/ล็อกอิน/รหัสผ่าน		
ชื่อการทำงาน	การจัดการการเข้าสู่ระบบและจัดการรหัสผ่าน		
รหัสหน้าจอ-ชื่อหน้าจอ	1. St1101 - การสู่ระบบ 2. St1102 - หน้าแรกหลังจากเข้าสู่ระบบ		
คำอธิบาย	<ul style="list-style-type: none"> ● ทดสอบการเข้าสู่ระบบของ Staff โดยการอกรหัสผ่านที่ถูกต้องและไม่ถูกต้อง ● ตรวจสอบว่าระบบรองรับการทำงานที่ถูกต้องและสามารถแสดงผลได้อย่างเหมาะสม 		
เงื่อนไขในการทำงาน	<ul style="list-style-type: none"> ● ระบบต้องเสร็จเรียบร้อยและพร้อมให้บริการแล้ว ● บัญชีผู้ใช้และรหัสผ่านที่ใช้ทดสอบต้องมีอยู่ในระบบ 		
ขั้นตอนการทำงาน	<ul style="list-style-type: none"> ● St1101 - เข้าสู่หน้าเข้าสู่ระบบ <ul style="list-style-type: none"> ○ เปิดหน้า St1001 - การเข้าสู่ระบบ ○ กรอกข้อมูลเบอร์โทรศัพท์และรหัสผ่านที่ถูกต้อง ○ คลิกปุ่ม เข้าสู่ระบบ ● St1102 - ตรวจสอบการแสดงผลหลังจากเข้าสู่ระบบ <ul style="list-style-type: none"> ○ ตรวจสอบว่าระบบนำผู้ใช้ไปยังหน้าแดชบอร์ดหลักของ Staff (St1002) ○ ตรวจสอบว่าสามารถเข้าถึงฟังก์ชันต่าง ๆ ที่อยู่ในหน้าหลักของ Staff ได้อย่างถูกต้อง ○ ตรวจสอบว่า UI/UX แสดงผลได้ถูกต้องในทุกอุปกรณ์ที่รองรับ (มือถือ, แล็ปท็อป) 		

ผลที่คาดว่าจะได้รับ	<p>St1102</p> <ul style="list-style-type: none"> • ระบบต้องแสดงหน้าแรกหลังจากเข้าสู่ระบบได้ถูกต้อง • ผู้ใช้สามารถเข้าถึงฟังก์ชันต่าง ๆ ในหน้าแดชบอร์ดของ Staff ได้ตามสิทธิ์ที่กำหนด • ระบบรองรับการแสดงผลที่เหมาะสมบนอุปกรณ์ทุกประเภท 		
การทดสอบการทำงานแบบไม่สมบูรณ์			
การกระทำ	ผลการทำงานที่คาดว่าจะได้รับ	ผลการทดสอบ	หมายเหตุ
<p>1. St1101 - กรอกชื่อผู้ใช้หรือรหัสผ่านที่ไม่ถูกต้อง</p> <p>2. St1102 - กรอกข้อมูลไม่ครบหรือผิดรูปแบบ (เช่น ไม่มีอีเมล หรือรหัสผ่านผิดรูปแบบ)</p>	<p>1. St1101 - ระบบต้องแจ้งเตือนข้อผิดพลาดว่า "ชื่อผู้ใช้หรือรหัสผ่านไม่ถูกต้อง"</p> <p>2. St1102 - ระบบต้องแจ้งเตือนว่า "ข้อมูลไม่ครบถ้วนหรือไม่ถูกต้อง" และไม่อนุญาตให้เข้าสู่ระบบ</p>	<p>1. St1101 ผ่าน</p> <p>2. St1102 ผ่าน</p>	

Test Case 12 : Staff

การจัดการการเช่ารถ

รหัสทดสอบ	UAT-St-0012	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วนการทำงาน	ระบบการจัดการการเช่ารถ		
ชื่อการทำงาน	การจัดการการเช่ารถ		
รหัสหน้าจอ-ชื่อหน้าจอ	1. St1201 - รายการคำสั่งเช่ารถ 2. St1102 - รายละเอียดคำสั่งเช่ารถ		
คำอธิบาย	<ul style="list-style-type: none"> ทดสอบกระบวนการที่ Staff สามารถดูคำสั่งเช่ารถที่ได้รับจาก Admin ได้ ตรวจสอบว่าข้อมูลการเช่ารถถูกต้องและแสดงผลอย่างเหมาะสม 		
เงื่อนไขในการทำงาน	<ul style="list-style-type: none"> ต้องมีรายการคำสั่งเช่ารถที่ถูกสร้างขึ้นจากลูกค้าแล้ว และได้รับการอนุมัติจาก Admin ระบบต้องสามารถโหลดและแสดงรายการคำสั่งเช่ารถได้ 		
ขั้นตอนการทำงาน	<p>St1201 - เปิดหน้ารายการคำสั่งเช่ารถ</p> <ul style="list-style-type: none"> เปิดหน้า St2001 - รายการคำสั่งเช่ารถ ตรวจสอบว่ามีรายการเช่ารถแสดงขึ้นมาหรือไม่ <p>St2002 - ตรวจสอบรายละเอียดคำสั่งเช่ารถ</p> <ul style="list-style-type: none"> คลิกที่รายการเช่ารถที่ต้องการตรวจสอบ ตรวจสอบข้อมูลการเช่ารถ เช่น รายละเอียดผู้เช่า, วันเวลาเช่า, สถานะการชำระเงิน Staff สามารถดูข้อมูลได้เท่านั้น ไม่มีสิทธิ์แก้ไขคำสั่งเช่ารถ 		

ผลที่คาดว่าจะได้รับ	<ul style="list-style-type: none"> รายการคำสั่งเข้ารถแสดงขึ้นมาถูกต้อง Staff สามารถดูข้อมูลคำสั่งเข้ารถได้ แต่ไม่มีสิทธิ์แก้ไขคำสั่ง ข้อมูลที่แสดงต้องถูกต้องและตรงกับที่ Admin อนุมัติ 		
การทดสอบการทำงานแบบไม่สมบูรณ์			
การกระทำ	ผลการทำงานที่คาดว่าจะได้รับ	ผลการทดสอบ	หมายเหตุ
1. St2001 - "ไม่มีรายการเข้ารถแสดง ขึ้นมา" 2. St2002 - "ข้อมูลคำสั่งเข้ารถแสดง ผลผิดพลาด"	1. St2001 - ระบบต้องแจ้งเตือนว่า "ไม่มีรายการคำสั่งเข้ารถในขณะนี้" 2. St2002 - ระบบต้องแจ้งเตือนข้อผิดพลาดเกี่ยวกับ ข้อมูลที่ไม่ถูกต้อง	1. St1201 ไม่ผ่าน 2. St1202 ไม่ผ่าน	

Test Case 13 : Staff
ระบบเข้าสู่ระบบพนักงาน (Staff Login)

รหัสทดสอบ	UAT-St-0013	วันที่ทดสอบ	00/00/2568
ระบบ/ส่วนการทำงาน	ระบบการจัดตารางงาน		
ชื่อการทำงาน	การดูตารางงานและรายชื่อพนักงานทั้งหมด		
รหัสหน้าจอ-ชื่อหน้าจอ	1. St1301 - ปฏิทินตารางงาน 2. St1102 - รายชื่อพนักงานทั้งหมด		
คำอธิบาย	<ul style="list-style-type: none"> ทดสอบว่าระบบสามารถแสดงตารางงานของพนักงานแต่ละคนได้อย่างถูกต้อง ตรวจสอบว่าระบบสามารถแสดงรายชื่อพนักงานทั้งหมดได้ 		
เงื่อนไขในการทำงาน	<ul style="list-style-type: none"> ต้องมีตารางงานพนักงานที่ถูกสร้างขึ้นจาก Admin และ ระบบต้องสามารถโหลดและแสดงตารางงานพนักงานได้อย่างถูกต้อง 		
ขั้นตอนการทำงาน	<p>St1301 - เปิดหน้าตารางงานพนักงาน</p> <ul style="list-style-type: none"> เปิดหน้า St1301 - ตารางพนักงาน ตรวจสอบว่ามีปฏิทินที่แสดงตารางงานของพนักงานหรือไม่ ตรวจสอบว่ารายชื่อพนักงานและวันที่ทำงานแสดงผลอย่างถูกต้อง <p>St1302 - ตรวจสอบรายละเอียดตารางงานพนักงาน</p> <ul style="list-style-type: none"> คลิกที่วันหรือพนักงานที่ต้องการตรวจสอบ ตรวจสอบรายละเอียดของพนักงาน เช่น ชื่อ, ภาระ, ตำแหน่งงาน Staff สามารถดูข้อมูลได้เท่านั้น ไม่มีสิทธิ์แก้ไขตารางงาน 		
ผลที่คาดว่าจะได้รับ	<ul style="list-style-type: none"> ตารางงานของพนักงานแสดงขึ้นมาถูกต้อง Staff สามารถดูข้อมูลตารางงานพนักงานได้ แต่ไม่มีสิทธิ์แก้ไขข้อมูล 		

	<ul style="list-style-type: none"> ข้อมูลที่แสดงต้องถูกต้องและตรงกับที่ Admin กำหนดไว้ 		
การทดสอบการทำงานแบบไม่สมบูรณ์			
การกระทำ	ผลการทำงานที่คาดว่าจะได้รับ	ผลการทดสอบ	หมายเหตุ
1. St1301 - "ไม่มีตารางงานแสดง ขึ้นมา" 2. St1302 - "ข้อมูลตารางงานแสดง ผลผิดพลาด"	1. St1301 - ระบบต้องแจ้งเตือนว่า "ไม่มีตารางงานในขณะนี้" 2. St1302 - ระบบต้องแจ้งเตือนข้อผิดพลาดเกี่ย วกับข้อมูลที่ไม่ถูกต้อง	1. St1301 ไม่ผ่าน 2. St1302 ไม่ผ่าน	

Agile Models

Agile Model สำหรับระบบเช่ารถ (Car Rental System)

Agile Software Development เป็นแนวทางที่ช่วยให้การพัฒนาระบบที่มีความยืดหยุ่น รวดเร็ว และปรับปรุงต่อเนื่องได้ โดยเฉพาะกับระบบที่ต้องตอบสนองต่อผู้ใช้งานจริง เช่น ระบบเช่ารถ ที่ต้องครอบคลุมทั้งผู้ใช้งาน (Customer) และผู้ดูแลระบบ (Admin) และพนักงาน (Staff)

ในการพัฒนาระบบนี้ เราใช้ Scrum Framework ซึ่งช่วยให้ทีมสามารถแบ่งงานเป็นรอบการทำงาน (Sprint) เพื่อส่งมอบระบบที่ใช้งานได้จริงในแต่ละระยะ

ขั้นตอนการดำเนินงานตาม Agile Model

1. เก็บรวบรวมความต้องการ (Requirement Gathering)

ใช้แนวทาง Epic และ User Story เพื่อแยกความต้องการของระบบเป็นเรื่องย่อย ๆ

ตัวอย่าง:

- Epic 1: การใช้งานของลูกค้า
 - User Story 1: ลูกค้าสามารถลงทะเบียนและเข้าสู่ระบบได้
 - User Story 2: ลูกค้าสามารถค้นหารถเช่าตามประเภทหรือราคา
 - User Story 3: ลูกค้าสามารถจองรถและชำระเงินออนไลน์
 - User Story 4: ลูกค้าตรวจสอบสถานะการจองและประวัติการเช่า
- Epic 2: การจัดการของ Admin
 - User Story 1: Admin สามารถเพิ่ม/ลบข้อมูลรถ
 - User Story 2: Admin จัดการบัญชีผู้ใช้งาน

2. การวางแผน Sprint (Sprint Planning)

- แบ่งการทำงานเป็นรอบ Sprint (2-4 สัปดาห์/รอบ)
- จัดลำดับงานใน Product Backlog และเลือกงานสำคัญมาทำในแต่ละ Sprint

Sprint Backlog: (Sprint 1)

เน้นพัฒนา Frontend ก่อน เพื่อกำหนดรูปแบบการใช้งาน

- ออกแบบหน้าจอ UI สำหรับ:
 - หน้าลงทะเบียน
 - หน้าล็อกอิน
 - หน้าแสดงรายการรถ
- สร้าง Prototype/Wireframe เพื่อให้ทีมเข้าใจตรงกัน
- ทดสอบ Flow การใช้งานผ่าน mock UI

3. การออกแบบสถาปัตยกรรมระบบ (Agile Architecture Design)

- ใช้ Incremental Design คือ ออกแบบและพัฒนาไปพร้อมกัน
- พัฒนาระบบแบบ Modular Monolith หรือ Microservices (ขึ้นอยู่กับขนาดโปรเจกต์)
- เอกสารประกอบ:
 - Class Diagram: Customer, Car, Booking, Payment
 - Sequence Diagram: ขั้นตอนการจองและชำระเงิน
 - Wireframe UI: ใช้สำหรับออกแบบหน้าจอผู้ใช้

4. การพัฒนาและทดสอบ (Development & Testing)

ดำเนินการพัฒนา (Frontend-First Approach)

1. พัฒนา UI ของฟีเจอร์สำคัญก่อน เช่น ลงทะเบียน, ค้นหารถ, หน้าจอง
2. สร้าง mock data/JSON เพื่อเชื่อมกับ frontend ชั่วคราว
3. พัฒนา Backend ตาม API ที่ frontend กำหนด
4. เชื่อม Frontend - Backend เมื่อ API พร้อม

การทดสอบประกอบด้วย:

- Unit Testing: เช่น ตรวจสอบพังก์ชันคำนวณค่าเช่า
- Integration Testing: ทดสอบ API ระหว่าง frontend กับ backend

- UI Testing: ตรวจสอบการใช้งานผ่านหน้าเว็บ/มือถือจริง

5. ส่งมอบและปรับปรุงต่อเนื่อง (Delivery & Feedback)

- ใช้ CI/CD Pipeline (เช่น GitHub Actions, GitLab CI/CD) เพื่อ Deploy ระบบ
- รับ Feedback จากผู้ใช้งานและทีม QA เพื่อแก้ไขหรือเพิ่มฟีเจอร์ใน Sprint ต่อไป

Risk Management

Risk	Risk Description	Affect	Probability	Effect	Strategy
โปรแกรมทำงานไม่ตรงตามความต้องการ	ซอฟต์แวร์ที่พัฒนาไม่เป็นไปตามข้อกำหนดหรือไม่ตอบโจทย์ผู้ใช้	Project , Product	ปานกลาง	สูง	- ควรเก็บ Requirement ให้ชัดเจนก่อนพัฒนา - ทดสอบระบบร่วมกับผู้ใช้ (User Testing)
รถมีปัญหาทางเทคนิค	รถเสียระหว่างการใช้งาน เช่น เครื่องยนต์ขัดข้อง	Product	ปานกลาง	ปานกลาง	- บำรุงรักษารถเป็นประจำ - ให้บริการซ่อมเหลืออุปกรณ์
ผู้พัฒนาขาดทักษะทางเทคนิค	นักพัฒนาไม่มีความชำนาญเพียงพอในเทคโนโลยีที่ใช้พัฒนา	Product, Project	ปานกลาง	สูง	- เลือกใช้เทคโนโลยีที่เหมาะสมกับทักษะของทีม
ปัญหาด้านประสิทธิภาพของระบบ	ระบบอาจประมวลผลช้าหรือทำงานผิดพลาดเมื่อมีผู้ใช้งานจำนวนมาก	Product	ปานกลาง	สูง	- ปรับปรุงโครงสร้างฐานข้อมูลและโค้ดใหม่ประสิทธิภาพ
ขาดการสื่อสารระหว่างทีมพัฒนาและผู้ใช้งาน	ขาดการสื่อสารที่ดีระหว่างทีมพัฒนาและผู้ใช้งานเกี่ยวกับความต้องการ	Project	ปานกลาง	สูง	- สื่อสารกับผู้ใช้งานอย่างสม่ำเสมอ
เวลาไม่เพียงพอ	งานของวิชาอื่น يؤะ หรือผู้พัฒนาว่างไม่ตรงกัน	Project	สูง	สูง	- กำหนดเวลา - ทำ scope งานที่สำคัญก่อน

Version Control

1. มีการใช้ Git จัดการ version สามารถดึงโค้ดอันเก่ามาได้
2. Author – สมาชิกในกลุ่มสามารถเข้าถึง Github “ได้ประกอบไปด้วย

1 Project Management

2 Programmer

3 Tester

4 System Analysis

5 UX/UI design

3. มีการใช้ Github repository เป็นที่เก็บโค้ด

4. มีการ commit code เข้าไปอย่างสม่ำเสมอ

QA (Quality Assurance)

No.	Name	Type	Test Case	Effect	Status
1.	Register / Login	Functional	1. สามารถสมัครสมาชิกได้ด้วยอีเมลและเบอร์โทรศัพท์	ช่วยให้ผู้ใช้ใหม่เริ่มต้นใช้งานได้สะดวก	ผ่าน
			2. สามารถเข้าสู่ระบบได้เมื่อกรอกข้อมูลถูกต้อง	เพิ่มความน่าเชื่อถือของระบบ	ผ่าน
			3. ระบบแจ้งเตือนเมื่อใส่รหัสผิด 3 ครั้ง	ป้องกันการโจมตีแบบ brute-force	ไม่ผ่าน
			4. ระบบตรวจสอบความถูกต้องของ Textfield ก่อนกด SUBMIT	ลดโอกาสเกิดข้อผิดพลาดจากการป้อนข้อมูล	ผ่าน
2.	Car Search	Functional	1. สามารถค้นหารถตามวันที่และสถานที่ได้	เพิ่มความสะดวกและลดเวลาการค้นหา	ไม่ผ่าน
			2. สามารถกรองผลลัพธ์ตามประเภทรถ	ช่วยให้ผู้ใช้ได้รถตามความต้องการมากขึ้น	ไม่ผ่าน
			3. ระบบแสดงผลรถและราคางานที่เลือก	ข้อมูลชัดเจน ช่วยตัดสินใจง่ายขึ้น	ไม่ผ่าน
			4. ระบบแจ้งเมื่อไม่มีรถว่างในช่วงเวลาที่นัด	ป้องกันการจองซ้ำ ลดความสับสน	ไม่ผ่าน
3.	Booking Process	Functional	1. เลือกรถแล้วสามารถดำเนินการจองได้	เป็นพังก์ชันหลักของระบบ	ผ่าน
			2. ระบบบันทึกข้อมูลการจองในฐานข้อมูล	เก็บหลักฐานเพื่อจัดการหลังบ้าน	ไม่ผ่าน
			3. ผู้ใช้ได้รับการยืนยันการจองผ่านระบบ	เพิ่มความมั่นใจให้ผู้ใช้	ผ่าน
			4. ระบบคำนวณราคาถูกต้อง	ป้องกันข้อผิดพลาดด้านการเงิน	ผ่าน
4.	Payment Gateway	Functional	1. ระบบเชื่อมต่อกับผู้ให้บริการชำระเงินได้	ชำระเงินได้จริง ปลอดภัย	ผ่าน
			2. แสดงสถานะสำเร็จ/ล้มเหลวของการจ่ายเงิน	ลดความสับสนเมื่อการชำระเงินไม่ผ่าน	ผ่าน

			3. ระบบไม่ให้ชำระเงินข้าหาก รายการสำเร็จแล้ว	ป้องกันการหักเงินข้า	ไม่ผ่าน
5.	Booking History	Functional	1. ผู้ใช้สามารถดูรายการจอง ย้อนหลังได้	ตรวจสอบประวัติการใช้งาน ได้ง่าย	ผ่าน
			2. รายการแสดงครบถ้วน (วันที่, รถ, ราคา, สถานะ)	ผู้ใช้มั่นใจในข้อมูลการจอง ของตนเอง	ผ่าน
6.	Notification	Functional	1. ระบบแจ้งเตือนเมื่อยืนยัน การจอง	เพิ่มความมั่นใจให้ user	ผ่าน
			2. แจ้งเตือนเมื่อชำระเงินเสร็จ สิ้น	เพิ่มความมั่นใจให้ user	ผ่าน
7.	Performance	Non- Functional	1. ระบบรองรับผู้ใช้จำนวนมาก มากพร้อมกันได้	ระบบไม่ล่มช่วงมีคนใช้งาน สูง	ไม่ผ่าน
8.	Security	Non- Functional	1. ป้องกัน SQL Injection และเข้ารหัสข้อมูลผู้ใช้	ลดความเสี่ยงข้อมูลรั่วไหล	ผ่าน
9.	Usability	Non- Functional	1. หน้าเว็บใช้งานง่าย	เพิ่มประสบการณ์ใช้งาน ของผู้ใช้	ผ่าน



คู่มือระบบ (อธิบาย Code) Frontend & Backend
ระบบเช่ารถ(ClubCars)

จัดทำโดย

- 6504062636233 นางสาวพิริยา ใจเที่ยง
6604062616126 นางสาวกัทรนันท์ บันสุขสวัสดิ์
6604062636119 นางสาวชัชชญา ฉายวัฒนะ
6604062636127 นางสาวชีตียาโยต วรรณนาชัย
6604062636135 นางสาวฐานปณี ศิริพละ

เสนอ

ผู้ช่วยศาสตราจารย์ สฤทธิ์ ประสมพันธ์

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา 040613306 วิศวกรรมซอฟแวร์
ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

Frontend

1.Login

1.1 ตัวแปร State และ Router

```
const Login = () => {
  const [phone, setPhone] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");
  const [loading, setLoading] = useState(false);
  const [isClient, setIsClient] = useState(false);

  const router = useRouter(); // ใช้ router เพื่อนำทาง
```

หน้าที่: เก็บข้อมูลและสถานะต่าง ๆ ของฟอร์ม เช่น:

- phone, password: เก็บค่าที่ผู้ใช้งานกรอก
- error: เก็บข้อความแจ้งเตือนหรือข้อผิดพลาด
- loading: ใช้แสดงว่าระบบกำลังประมวลผล
- isClient: ใช้เช็คว่ารันอยู่ฝั่ง client เพื่อหลีกเลี่ยงปัญหา SSR (Server Side Rendering)

1.2 useEffect สำหรับเช็ค Client Side

```
useEffect(() => {
  setIsClient(true);
}, []);
```

หน้าที่: เมื่อ component โหลดเสร็จในฝั่ง client และ จะตั้งค่า isClient เป็น true เพื่อให้ render UI ได้

1.3 handle Submit – พังก์ชันเมื่อกด “เข้าสู่ระบบ”

```
// เมื่อพิมพ์เสร็จสิ้น handleSubmit [กดปุ่ม async
const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setError("");

  if (!phone.trim() || !password.trim()) {
    setError("กรุณากรอกเบอร์โทรศัพท์และรหัสผ่านให้ครบถ้วน");
    return;
  }

  setLoading(true);

  // ส่งข้อมูล Login ไปยัง API
  const res = await fetch("/api/login", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ phone, password }),
  });

  const data = await res.json();

  if (res.status === 200) {
    // เก็บ login token
    localStorage.setItem("token", data.token);

    alert(data.message);
    if (data.role === "admin") {
      router.push("/admin/");
    } else{
      router.push("/");
    }
  } else {
    // คำแนะนำเพิ่มเติม
    setError(data.message);
  }

  setLoading(false);
};
```

หน้าที่:

1. ยกเลิกการ reload หน้าเว็บด้วย e.preventDefault()
2. เช็คว่าเบอร์โทรและรหัสผ่านไม่ว่าง
3. ส่งข้อมูลไปที่ API /api/login แบบ POST
4. รอรับค่ากลับจาก server
5. ถ้า login ผ่าน:
 - เก็บ token ลงใน localStorage
 - แสดงข้อความ และพาไปยังหน้า / หรือ /admin/ ตาม role
6. ถ้า login ไม่ผ่าน:
 - แสดงข้อความ error

1.4 handleSocialLogin – ล็อกอินด้วยบัญชีอื่น

```
const handleSocialLogin = (provider: string) => {
  alert(`ปิดหน้าต่างล็อกอิน ${provider}`);
  // ที่นี่สามารถ OAuth หรือ Firebase Authentication เพื่อเข้าสู่ระบบจริง
};

if (!isClient) {
  return null; // ไม่ได้เรียกเดอร์ค่อนไฟเพเนนด้านความปลอดภัยในฝั่งไคลเอนต์
}
```

หน้าที่:

1. พังก์ชันนี้แค่แสดง alert บอกว่าเราจะเข้าสู่ระบบผ่าน Google / Apple / Facebook
2. ยังไม่ได้เชื่อมกับระบบ OAuth จริง (แค่เตรียมไว้)

1.5 Return & UI – ส่วนแสดงผล

```
return (
  <div className="full-screen-background">
    <div className="rounded-xl p-4 max-w-sm w-full text-center mt-20">
      <div className="text-left">
        
        <h2 className="text-xl font-bold mb-2">บันทึกนักเรียน !</h2>
        <p className="text-500 mb-2">สถานที่ใช้งานที่ต้องการเข้าชม</p>
      </div>

      <input
        type="tel"
        className="text-center w-full p-2 border-b border-gray-400 placeholder-gray-500 bg-transparent mb-3 focus:outline-none focus:ring-2 focus:ring-blue-500"
        placeholder="เบอร์โทรศัพท์"
        value={phone}
        onChange={(e) => setPhone(e.target.value)}
        required
      />
      <input
        type="password"
        className="text-center w-full p-2 border-b border-gray-400 placeholder-gray-500 bg-transparent mb-3 focus:outline-none focus:ring-2 focus:ring-blue-500"
        placeholder="รหัสผ่าน"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
        required
      />
      {error && <div className="text-red-500 text-sm">{error}</div>}

      <p className="text-black-250 text-sm mb-4">กด</p>
      <button onClick={() => handleSocialLogin("Google")}>w-full bg-white hover:bg-gray-200 text-black py-2 rounded-lg btn-shadow mb-4 flex items-center justify-center gap-2 border border-gray-300>
        |  บันทึก Google
      </button>

      <button onClick={() => handleSocialLogin("Apple")}>w-full bg-white hover:bg-gray-200 text-black py-2 rounded-lg mb-4 btn-shadow flex items-center justify-center gap-2>
        | <a href="#" className="text-black-500 h-6 w-6" /> บันทึก Apple
      </button>

      <button onClick={() => handleSocialLogin("Facebook")}>w-full bg-white hover:bg-gray-200 text-black py-2 rounded-lg mb-6 btn-shadow flex items-center justify-center gap-2>
        | <a href="#" className="text-blue-600 h-5 w-5" /> บันทึก Facebook
      </button>

      <button onClick={handleSubmit}>block mx-auto w-[35%] bg-[#0D3489] hover:bg-[#092C5D] btn-shadow text-white py-2 rounded-2xl mb-4 text-center>
        | บันทึก
      </button>

      <p className="mt-4 text-black ml-2">
        <a href="#">ลืมรหัสผ่าน</a>
        <a href="/sign_in" className="text-[#004FFA] underline">ล็อกอิน</a>
        <a href="#">ลืมรหัสผ่าน</a>
      </p>
    </div>
  </div>
```

ประกอบด้วย:

- โลโก้ + ข้อความต้อนรับ
- Input เบอร์โทรศัพท์ & รหัสผ่าน
- Error message ถ้ามี

- ปุ่ม Social Login (Google / Apple / Facebook)
- ปุ่มเข้าสู่ระบบหลัก
- ลิงก์สมัครสมาชิก

ตกแต่งด้วย Tailwind CSS: เช่น rounded-xl, p-4, text-center, hover:bg-gray-200, ฯลฯ เพื่อให้หน้าเว็บดูดี ใช้งานง่าย

2.Signin

2.1 Imports – การนำเข้าโมดูลที่ใช้

```
"use client";

import { useState } from "react";
import { useRouter } from "next/navigation";
import Link from "next/link";
import './globals.css';
import { mysqlPool } from "@/utils/db";
```

- use client บอกว่า component นี้ทำงานผ่าน client (React)
- useState สำหรับจัดการข้อมูลฟอร์มแบบเรียลไทม์
- useRouter ใช้เปลี่ยนหน้า (เช่น ไปหน้า login หลังสมัครเสร็จ)
- Link ใช้ลิงก์แบบไม่โหลดหน้าใหม่
- globals.css โหลดสไต์ล์ทั่ว ๆ ไป
- mysqlPool เมื่อมีอนเดรียมไว้แต่ ยังไม่ได้ใช้ใน component นี้

2.2 ตัวแปร State – จัดการข้อมูลที่ผู้ใช้กรอก

```
const signin = () => {
  const [firstName, setFirstName] = useState("");
  const [lastName, setLastName] = useState("");
  const [username, setUsername] = useState("");
  const [phone, setPhone] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");
  const [loading, setLoading] = useState(false);
  const router = useRouter();
```

หน้าที่ เก็บข้อมูลแต่ละช่องของฟอร์มที่ผู้ใช้กรอก

- รวมถึง error message และสถานะ loading ระหว่างส่งข้อมูลไป backend

2.3 handleSubmit – พังค์ชั้นจัดการเมื่อกดปุ่มสมัครสมาชิก

```
const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setError("");

  if (!firstName || !lastName || !username || !phone || !password) {
    setError("⚠️ กรุณากรอกข้อมูลให้ครบถ้วน");
    return;
  }

  setLoading(true);

  //setTimeout(() => {
  //  sessionStorage.setItem("user", JSON.stringify({ firstName, lastName, username, phone }));
  //  setLoading(false);
  //  alert("บัญชีสมัครเรียบร้อย!");
  //  router.push("/signin");
  //}, 1500);

  try {
    const response = await fetch("/api/signin", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        firstName,
        lastName,
        username,
        phone,
        password,
      }),
    });

    const data = await response.json();

    if (response.status === 200) {
      alert(data.message); // แสดงข้อความสำหรับสมาชิกสำเร็จ
      router.push("/signin"); // ไปยังหน้าหลัก
    } else {
      setError(data.message); // แสดงข้อความผิดพลาด
    }
  } catch (err) {
    setError("⚠️ เกิดข้อผิดพลาดในการสมัครสมาชิก");
    console.error("Error during signup:", err);
  } finally {
    setLoading(false);
  }
};
```

หน้าที่ :

- ป้องกันการ reload หน้า (e.preventDefault())
- เช็คว่าผู้ใช้กรอกข้อมูลครบหรือยัง → ถ้าไม่ครบแสดง error
- ถ้าครบ:
 - ส่งข้อมูลไปยัง API /api/signin แบบ POST โดยมีข้อมูลเป็น JSON
 - รอรับผลลัพธ์จาก server

- ถ้าสำเร็จ → แสดง alert และส่งผู้ใช้ไปหน้า login (/signin)
 - ถ้าไม่สำเร็จ → แสดง error จากฝั่ง server
4. มีการจัดการ error และแสดงในหน้าเว็บ

2.4 return – UI ส่วนหน้าเว็บ

```

return (
  <div className="full-screen-background">
    <div className="max-w-md w-full bg-white p-6 rounded-lg shadow-lg mt-20">
      <div className="flex justify-center mb-4">
        
      </div>

      <h1 className="text-2xl font-medium mb-4 text-center text-gray-700 font-inter">ເລືດສາມາດ</h1>

      <form onSubmit={handleSubmit} className="space-y-4">
        <div>
          <label className="block text-sm font-medium text-gray-600">ຊື່ນາມ</label>
          <input
            type="text"
            className="w-full px-4 py-2 border rounded-md focus:ring focus:ring-blue-300"
            value={firstName}
            onChange={(e) => setFirstName(e.target.value)}
            required
          />
        </div>

        <div>
          <label className="block text-sm font-medium text-gray-600">ນາມສະກຳ</label>
          <input
            type="text"
            className="w-full px-4 py-2 border rounded-md focus:ring focus:ring-blue-300"
            value={lastName}
            onChange={(e) => setLastName(e.target.value)}
            required
          />
        </div>

        <div>
          <label className="block text-sm font-medium text-gray-600">ຊື່ຜົນປະວັນ</label>
          <input
            type="text"
            className="w-full px-4 py-2 border rounded-md focus:ring focus:ring-blue-300"
            value={username}
            onChange={(e) => setUsername(e.target.value)}
            required
          />
        </div>

        <div>
          <label className="block text-sm font-medium text-gray-600">ເບີໂທີ່ພໍາຕົວ</label>
          <input
            type="tel"
            className="w-full px-4 py-2 border rounded-md focus:ring focus:ring-blue-300"
            value={phone}
            onChange={(e) => setPhone(e.target.value)}
            required
          />
        </div>

        <div>
          <label className="block text-sm font-medium text-gray-600">ລິຫລວງ</label>
          <input
            type="password"
            className="w-full px-4 py-2 border rounded-md focus:ring focus:ring-blue-300"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            required
          />
        </div>
      </form>

      {error && <div className="text-red-500 text-sm">{error}</div>}

      <button
        type="submit"
        className="w-1/2 mx-auto py-2 bg-[#003489] text-white rounded-md hover:bg-[#002C5D] transition flex justify-center items-center"
        disabled={loading}
      >
        {loading ? "ເລືດສາມາດສໍາເລັດ..." : "ເລືດສາມາດ"}
      </button>
    </div>
  );
};

export default signin;

```

หน้าที่ :

- โลโก้แสดงด้านบน
- หัวข้อ "สมัครสมาชิก"
- ฟอร์มประกอบด้วย:
 - ช่องกรอกชื่อ, นามสกุล, ชื่อผู้ใช้, เบอร์โทร, รหัสผ่าน
 - แสดงข้อความ error ถ้ามี
- ปุ่มสมัครสมาชิก (มีสถานะ loading)
- ลิงก์ไปหน้าเข้าสู่ระบบ หากมีบัญชีแล้ว

ตกแต่งด้วย Tailwind CSS เช่น rounded-md, shadow-lg, hover:bg-[#092C5D] ฯลฯ

3.Rental

3.1 ตัวแปร State ต่าง ๆ

```
const Contact: React.FC = () => [
  const [username, setUsername] = useState("");
  const [message, setMessage] = useState("");
  const [isSubmitting, setIsSubmitting] = useState(false);
  const [successMessage, setSuccessMessage] = useState("");
  const [errorMessage, setErrorMessage] = useState("");
  const router = useRouter();
```

หน้าที่ :

- username, message: ค่าที่ผู้ใช้กรอกเข้ามาในแบบฟอร์ม
- isSubmitting: ใช้ควบคุมสถานะระหว่างส่งข้อความ (ป้องกันกดซ้ำ)
- successMessage, errorMessage: แสดงข้อความผลลัพธ์หลังจากส่ง

3.2 พังก์ชัน handleSubmit

```
const handleSubmit = (e: React.FormEvent) => {
  e.preventDefault();
  setIsSubmitting(true);

  setTimeout(() => {
    setIsSubmitting(false);
    setSuccessMessage("บันทึกข้อมูลสำเร็จ");
    setUsername("");
    setMessage("");
  }, 1500);
};
```

หน้าที่

1. ยกเลิกการ reload หน้า
2. ตั้งค่าสถานะ isSubmitting ให้เป็น true
3. ใช้ setTimeout จำลองการส่งข้อความ (delay 1.5 วิ)
4. เมื่อส่งเสร็จ:

- เคลียร์ช่องกรอก
- แสดงข้อความ “ข้อความของคุณถูกส่งเรียบร้อยแล้ว!”

หมายเหตุ: ยังไม่มีการเชื่อมต่อกับ backend จริง แค่จำลองส่งข้อความเท่านั้น

3.3 Contact Form – แบบฟอร์มส่งข้อความ

```
<form onSubmit={handleSubmit} className="flex flex-col space-y-6">
  <input
    type="text"
    placeholder="Username"
    className="border-2 border-gray-300 p-4 rounded-xl w-full focus:outline-none focus:ring-2 focus:ring-blue-600"
    value={username}
    onChange={(e) => setUsername(e.target.value)}
    required
  />
  <textarea
    placeholder="ເພີ້ມຂໍ້ຕົວອາຫານ"
    className="border-2 border-gray-300 p-4 rounded-xl h-32 w-full focus:outline-none focus:ring-2 focus:ring-blue-600"
    value={message}
    onChange={(e) => setMessage(e.target.value)}
    required
  ></textarea>
  <button
    type="submit"
    disabled={isSubmitting}
    className="bg-blue-600 text-white py-3 rounded-xl hover:bg-blue-800 disabled:bg-blue-400 transition-colors duration-200"
  >
    {isSubmitting ? "ກຳລັງ..." : "ສອບຕົວການ"}
  </button>
</form>

{successMessage && (
  <div className="mt-4 text-green-600 text-center">{successMessage}</div>
)}
{errorMessage && (
  <div className="mt-4 text-red-600 text-center">{errorMessage}</div>
)}
<a href="https://instagram.com/ClubCarsofficial_"
  className="flex items-center justify-start bg-white p-3 rounded-md hover:bg-gray-100 transition-all duration-200"
  target="_blank"
  rel="noopener noreferrer"
>
  <Image
    src="/images/instagram-icon.png"
    alt="Instagram Icon"
    width={24}
    height={24}
  />
  <span className="text-blue-700 text-xl ml-3">ClubCarsofficial_</span>
</a>
<a
  href="https://facebook.com/ClubCarsThailand"
  className="flex items-center justify-start bg-white p-3 rounded-md hover:bg-gray-100 transition-all duration-200"
  target="_blank"
  rel="noopener noreferrer"
>
```

ໜ້າທີ່

- ແສດງໄວຄອນ + ຂໍ້ມູນຄົດຕ່ອແບບລິງກີ
- ມີ Email, Instagram, Facebook (ລິງກີເປີດແຫັບໃໝ່)
- ລິງກີ Facebook ທຳກັນຫລາຍອັນ (ອາຈຕ້ອງລົບໃໝ່ເທົ່ານີ້)

4. Rental

4.1 Interface Rental & ข้อมูลจำลอง

```
interface Rental {
  id: string;
  carName: string;
  location: string;
  pickupDate: string;
  returnDate: string;
  duration: string;
  status: string;
  imageUrl: string;
}

const rentals: Rental[] = [
```

หน้าที่

- interface Rental: กำหนดโครงสร้างของข้อมูลการเช่ารถ เช่น carName, location, pickupDate, status ฯลฯ
- rentals: ข้อมูลการจองแบบ mock (จำลอง) 3 รายการ ที่จะนำมาแสดงในหน้าเว็บ

4.2 ພຶກສັນ Component Rental

```

const Rental = () => {
  const router = useRouter();

  return (
    <div className="mt-20">
      <h1 className="text-blue-900 text-3xl font-bold mb-6">ກ່າວຄະດີອຳນວຍ</h1>
      <div className="space-y-6">
        {rentals.map(rental) => (
          <div key={rental.id} className="bg-[#F2F2F2] rounded-lg py-1 px-8 flex flex-col items-center">
            <div className="flex w-full items-center">
              {/* Column 1 */}
              <div className="w-full sm:w-1/3 flex flex-col items-start space-y-3 relative">
                <h2 className="text-xl font-bold text-gray-900 text-start p-3 rounded-lg">
                  {rental.carName}
                </h2>
                <div className="w-72 sm:w-72 h-72 max-w-full">
                  <Image
                    src={rental.imageUrl}
                    alt={rental.carName}
                    width={300}
                    height={300}
                  />
                </div>
              </div>
            </div>

            {/* Column 2: Booking Details */}
            <div className="w-full sm:w-1/3 px-11 flex flex-col">
              <p className="text-gray-500 text-l font-medium py-1">
                ດັວກທີ່ຈະມີ
              </p>
              <span className="text-black text-xl font-semibold py-1">{rental.id}</span>

              <div className="mt-4 py-6">
                <div className="flex space-x-2 mt-1">
                  <Image src="/location.png" alt="Location Icon" width={20} height={20} />
                  <p className="text-gray-500 text-sm font-medium">ເຖິງນິກົມ</p>
                </div>
                <p className="text-black text-xl font-semibold py-2">{rental.location}</p>
              </div>

              {/* ໂດຍບໍ່ເຫັນໄດ້ */}
              <button className="mt-5 text-gray-500 text-sm font-semibold underline">
                ໂດຍບໍ່ເຫັນໄດ້
              </button>
            </div>

            {/* Column 3: Rental Dates & Status */}
            <div className="w-full sm:w-1/3 flex flex-col items-center rounded-lg">
              /* Pickup Date */
              <div className="flex flex-col items-center py-3">
                <div className="flex items-center space-x-2">
                  <Image src="/calendar.png" alt="Calendar Icon" width={18} height={18} />
                  <p className="text-gray-500 text-xs font-medium">ກົດຕົວວັນທີ</p>
                </div>
                <p className="text-black text-xs font-semibold font-medium mt-1">{rental.pickupDate}</p>
              </div>

              /* Duration */
              <div className="bg-white px-3 py-1 rounded-full text-gray-700 text-xs">
                {rental.duration}
              </div>
            </div>
          </div>
        )}
      </div>
    </div>
  );
}

```

ໜ້າທີ່

- ວິນລຸ່ມແສດງກາຈອງທີ່ໜົດໃນ rentals
- ຈັດ layout ແບບ 3 ຄອລົມນີ້:
 1. ຂໍ້ມູນຮົດ + ຮູປກາພ
 2. ຮາຍລະເອີ້ດກາຈອງ ເຊັ່ນ ພມາຍເລີຂ + ຈຸດຮັບຮັດ
 3. ວັນ-ເວລາ + ສຕານະກາຈອງ + ປຸ່ມດູຮາຍລະເອີ້ດ

4.3 การวนลูปแสดงข้อมูลการจองแต่ละรายการ

```
{rentals.map((rental) => (
  <div key={rental.id} className="bg-[#F2F2F2] rounded-lg py-1 px-8 flex flex-col items-center" >
    <div className="flex w-full items-center">
```

ภายในแต่ละรายการ แบ่งเป็น 3 คอลัมน์หลัก:

4.3.1 Column 1: ข้อมูลรถ + รูปภาพ

```
<h2 className="text-xl font-bold text-gray-900 text-start p-3 rounded-lg">
  {rental.carName}
</h2>
<div className="w-72 sm:w-72 h-72 max-w-full">
  <Image
    src={rental.imageUrl}
    alt={rental.carName}
    width={300}
    height={300}
  />
</div>
</div>
```

- แสดงชื่อรถที่จอง เช่น Toyota Camry
- แสดงรูปภาพของรถจากไฟล์ใน /public

4.3.2 Column 2: รายละเอียดการจอง

```
/* Column 2: Booking Details */
<div className="w-full sm:w-1/3 px-11 flex flex-col">
  <p className="text-gray-500 text-1 font-medium py-1">
    หมายเลขอกรถเช่า </p>
  <span className="text-black text-xl font-semibold py-1">{rental.id}</span>

  <div className="mt-4 py-6">
    <div className="flex space-x-2 mt-1">
      <Image src="/location.png" alt="Location Icon" width={20} height={20} />
      <p className="text-gray-500 text-sm font-medium">จ.เชียงใหม่</p>
    </div>
    <p className="text-black text-xl font-semibold py-2">{rental.location}</p>
  </div>

  /* ปุ่มยกเลิกการจอง */
  <button className="mt-5 text-gray-500 text-sm font-semibold underline">
    ยกเลิกการเช่า
  </button>
</div>
```

- แสดงหมายเลขอกรถเช่า และสถานที่รับ-คืนรถ
- มีปุ่ม “ยกเลิกการเช่า”

4.3.3 Column 3: วัน-เวลา + สถานะ + ปั๊ม

- แสดงวันที่รับ/คืนรถ
 - แสดงสถานะการจอง: ยืนยันการเช่า, รอการยืนยัน, ยกเลิกการเช่า (โดยใช้สีตามสถานะ)
 - ปุ่ม “ดูรายละเอียด” → พาไปหน้ารายละเอียด /rental/[id]

4.4 ปุ่ม “ดรายลั๊กอี้ด”

```
onClick={() => router.push(`/rental/${rental.id}`)})
```

เมื่อผู้ใช้กด จะนำทางไปยังหน้าแสดงรายละเอียดเฉพาะของการจองนั้น เช่น /rental/21360

5.Rental

5.1 ข้อมูลการจอง

```
interface Rental {  
    id: string;  
    carName: string;  
    location: string;  
    pickupDate: string;  
    returnDate: string;  
    duration: string;  
    status: string;  
    imageUrl: string;  
}
```

```
const rentals: Rental[] = [
```

- เก็บข้อมูลการจองรถเข้าในรูปแบบของอาร์เรย์ (rentals)
- แต่ละรายการมี id, ชื่อรถ, สถานที่, วันที่รับ-คืน, สถานะ และรูปภาพ

5.2 โครงสร้างของ UI หลัก (Rental Component)

```
const Rental = () => {
  const router = useRouter();

  return (
    <div className="mt-20">
      <h1 className="text-blue-900 text-3xl font-bold mb-6">การจองของฉัน</h1>
      <div className="space-y-6">
        {rentals.map((rental) => (
          <div key={rental.id} className="bg-[#F2F2F2] rounded-lg py-1 px-8 flex flex-col items-center" >
            <div className="flex w-full items-center">

              {/* column 1 */}
              <div className="w-full sm:w-1/3 flex flex-col items-start space-y-3 relative">
                <h2 className="text-xl font-bold text-gray-900 text-start p-3 rounded-lg">
                  {rental.carName}
                </h2>
                <div className="w-72 sm:w-72 h-72 max-w-full">
                  <Image
                    src={rental.imageUrl}>
                </div>
              </div>
            </div>
          </div>
        ))
      </div>
    </div>
  );
}
```

- ใช้ useRouter จาก Next.js เพื่อใช้ push navigation ไปยังหน้ารายละเอียดการจอง
- วนลูป (map) ผ่าน rentals และแสดงผลข้อมูลแต่ละรายการ

5.3 คอลัมน์ที่ 1: ข้อมูลรถ

```
/* Column 1 */
<div className="w-full sm:w-1/3 flex flex-col items-start space-y-3 relative">
  <h2 className="text-xl font-bold text-gray-900 text-start p-3 rounded-lg">
    {rental.carName}
  </h2>
  <div className="w-72 sm:w-72 h-72 max-w-full">
```

- แสดง ชื่อรถ และ รูปภาพรถ

5.4 คอลัมน์ที่ 2: รายละเอียดการจอง

```
/* Column 2: Booking Details */


หมายเลขบัญชี </p>
{rental.id}</span>



จังหวัด - ศูนย์



{rental.location}</p>



ปุ่มยกเลิกการจอง </button>
ยกเลิกการจอง </button>


```

- แสดง หมายเลขการเช่า

5.5 คอลัมน์ที่ 3: วันที่รับ-คืน และสถานะ

```
/* Column 3: Rental Dates & Status */


/* Pickup Date */

วันที่รับรถ



{rental.pickupDate}</p>



/* Duration */

{rental.duration}



/* Return Date */

วันที่คืนรถ



{rental.returnDate}</p>


```

- แสดง วันรับ-คืนรถ
- เปลี่ยนสีข้อความของ สถานะการจอง ตามเงื่อนไข

6. Contact

```
const Contact: React.FC = () => {
  const [username, setUsername] = useState("");
  const [message, setMessage] = useState("");
  const [isSubmitting, setIsSubmitting] = useState(false);
  const [successMessage, setSuccessMessage] = useState("");
  const [errorMessage, setErrorMessage] = useState("");
  const router = useRouter();
```

- username และ setUsername: ใช้ในการเก็บและอัปเดตค่าชื่อผู้ใช้ที่กรอก
- message และ setMessage: ใช้ในการเก็บและอัปเดตข้อความที่ผู้ใช้ส่ง
- isSubmitting และ setIsSubmitting: ใช้ในการตรวจสอบสถานะการส่งข้อมูล (กำลังส่งหรือไม่)
- successMessage และ setSuccessMessage: เก็บข้อความที่จะแสดงเมื่อการส่งข้อความสำเร็จ
- errorMessage และ setErrorMessage:
 เก็บข้อความที่จะแสดงเมื่อเกิดข้อผิดพลาดในการส่งข้อความ
- router: ใช้สำหรับการนำทางหน้าในแอปพลิเคชัน Next.js (ในโค้ดนี้ไม่ได้ใช้จริง ๆ)

```
const handleSubmit = (e: React.FormEvent) => {
  e.preventDefault();
  setIsSubmitting(true);

  setTimeout(() => {
    setIsSubmitting(false);
    setSuccessMessage("ข้อความของคุณถูกส่งเรียบร้อยแล้ว!");
    setUsername("");
    setMessage("");
  }, 1500);
};
```

- handleSubmit: พังก์ชันนี้จะทำงานเมื่อผู้ใช้กดปุ่ม "ส่งข้อความ"
- e.preventDefault(): ป้องกันการรีเฟรชของหน้าหรือการกระทำการของฟอร์มตามปกติ
- setIsSubmitting(true): ตั้งค่าสถานะเป็นกำลังส่ง
- setTimeout: หลังจาก 1.5 วินาที (จำลองการส่งข้อมูล), จะตั้งค่าสถานะการส่งเป็นเสร็จสิ้น
 และแสดงข้อความสำเร็จ พร้อมล้างข้อมูลในฟอร์ม

```
<form onSubmit={handleSubmit} className="flex flex-col space-y-1">
  <div className="space-y-1">
    <label htmlFor="username" className="text-sm font-medium text-gray-600">
      ຊື່ຜູ້ໃຊ້
    </label>
    <input
      id="username"
      type="text"
      placeholder="ນາມສະກຳລັກທີ່"
      className="border-2 border-gray-200 p-3 md:p-4 rounded-xl w-full focus:outline-none focus:border-blue-600 focus:ring-1 focus:ring-blue-600 transition-all"
      value={username}
      onChange={(e) => setUsername(e.target.value)}
      required
    />
  </div>

  <div className="space-y-1">
    <label htmlFor="message" className="text-sm font-medium text-gray-600">
      ດຳວັດ
    </label>
    <textarea
      id="message"
      placeholder="ສະບັບພາກອອກທຶນ"
      className="border-2 border-gray-200 p-3 md:p-4 rounded-xl h-32 w-full focus:outline-none focus:border-blue-600 focus:ring-1 focus:ring-blue-600 transition-all"
      value={message}
      onChange={(e) => setMessage(e.target.value)}
      required
    ></textarea>
  </div>

  <button
    type="submit"
    disabled={isSubmitting}
    className="bg-blue-600 text-white py-3 px-6 rounded-xl hover:bg-blue-700 disabled:bg-blue-400 transition-all duration-300 font-medium flex items-center justify-center"
  >
    {isSubmitting ? (
      <>
        <svg className="animate-spin -ml-2 h-4 w-4 text-white" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">
          <circle className="opacity-25" cx="12" cy="12" r="10" stroke="currentColor" strokeWidth="4"/>
          <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8v0c5.373 0 0 5.373 0 12h4zm2 5.291a7.962 7.962 0 014 12h0c3 0 3.135 5.824 3 7.938l-2.647z"/>
        </svg>
        ດຳເນັ້ນ...
      </>
    ) : "ສົ່ງສໍາຄັນ"}
  </button>
</form>
```

- ฟอร์มนี้มีช่องกรอกชื่อผู้ใช้และข้อความ โดยใช้ input และ textarea
 - ปุ่มส่งจะถูกปิดใช้งาน (disabled) เมื่อกำลังส่งข้อมูล และแสดงข้อความ "กำลังส่ง..."

```
{successMessage && (
  <div className="p-3 bg-green-100 text-green-700 rounded-lg text-center animate-fade-in">
    |   {successMessage}
  </div>
)
{errorMessage && (
  <div className="p-3 bg-red-100 text-red-700 rounded-lg text-center animate-fade-in">
    |   {errorMessage}
  </div>
)
</div>
```

- หากมีข้อความสำเร็จหรือข้อผิดพลาด จะถูกแสดงในกล่องข้อความที่มีสีที่เหมาะสม

```
<div className="space-y-4">
  <a href="mailto:ClubCars999@gmail.com"
    className="flex items-center bg-white p-4 rounded-xl shadow-sm hover:shadow-md transition-all duration-300 border border-gray-100 hover:border-blue-200">
    >
      <div className="bg-blue-100 p-2 rounded-lg">
        <Image
          src="/images/email-icon.png"
          alt="Email Icon"
          width={24}
          height={24}
          className="w-6 h-6"
        />
      </div>
      <span className="text-blue-700 font-medium ml-4">ClubCars999@gmail.com</span>
    </a>
```

- ช่องทางการติดต่ออื่นๆ เช่น อีเมล, Instagram, และ Facebook
ที่สามารถคลิกเพื่อไปยังแอปพลิเคชันหรืออีเมล
- แสดงเวลาทำการของธุรกิจ

```
<div className="pt-4 border-t border-gray-100">
  <h3 className="font-medium text-gray-600 mb-2">เวลาทำการ</h3>
  <p className="text-gray-600">จันทร์-ศุกร์ 9.00-18.00 น.</p>
</div>
```

7.Rental Agreement

7.1 รายการเงื่อนไขและข้อตกลง (sections)

```
const sections = [
  {
    title: "ข้อกำหนดและเงื่อนไข",
    icon: <FaCar className="text-blue-500" />,
    items: [
      "ผู้เช่าต้องมีใบขับขี่ที่ถูกต้อง",
      "ห้ามสูบบุหรี่หรือดื่มแอลกอฮอล์ในรถ",
      "หากคืนรถล้าหลังมีค่าปรับ"
    ]
  },
  {
    title: "คุณสมบัติผู้เช่า",
    icon: <FaIdCard className="text-green-500" />,
    items: [
      "อายุขึ้นต้น: ผู้เช่าต้องมีอายุอย่างน้อย 21 ปี",
      "ใบขับขี่: ผู้เช่าต้องมีใบขับขี่ที่ถูกต้องและมีอายุมากกว่า 1 ปี",
      "เอกสารยืนยันตัวตน: เช่น บัตรประชาชนหรือพาสปอร์ตที่ยังไม่หมดอายุ"
    ]
  },
  {
    title: "ประวัติภัยและความซื่นตรง",
    icon: <FaShieldAlt className="text-purple-500" />,
    items: [
      "ผู้เช่าต้องทำประวัติภัยดี (อาจเป็นประวัติพื้นฐานหรือแบบครอบคลุม)",
      "ระบุความคุ้มครองในกรณีเกิดอุบัติเหตุ ความเสียหาย หรือการโจรมรุนแรง"
    ]
  },
  {
    title: "เงินมัดจำและค่าใช้จ่าย",
    icon: <FaMoneyBillWave className="text-yellow-500" />,
    items: [
      "เงินมัดจำ: ผู้เช่าต้องชำระเงินมัดจำก่อนรับรถ",
      "เงือนไขการคืนเงินมัดจำ: เงินมัดจำจะคืนให้เมื่อรถถูกส่งคืนในสภาพดีและไม่มีความเสียหาย",
      "ค่าเช่า: ระบุค่าเช่าตามประเภทของรถและระยะเวลา"
    ]
  }
]
```

- ใช้ Array of Objects เพื่อเก็บหัวข้อเงื่อนไข
- แต่ละหัวข้อมี title (ชื่อหัวข้อ), icon (ไอคอน), items (รายการเงื่อนไข)
- สามารถเพิ่ม-ลดหัวข้อ ได้ง่าย

7.2 พังก์ชันเปลี่ยนหน้าเมื่อกดปุ่มยืนยัน

```
const handleSubmit = () => {
  if (isAccepted) {
    router.push("/paymentSuccess");
  }
};
```

- ตรวจสอบว่า checkbox ถูกติ๊ก หรือไม่
- ถ้าผู้ใช้กดยอมรับ จะพาไปหน้า /paymentSuccess

7.3 การแสดงแต่ละเงื่อนไข (map)

```
/* Agreement Sections */
<div className="space-y-8">
  {sections.map((section, index) => (
    <div key={index} className="border-b border-gray-200 pb-6 last:border-0">
      <div className="flex items-center mb-4">
        <div className="mr-3 text-xl">
          </div>
        <h2 className="text-xl font-semibold text-gray-800">{section.title}</h2>
      </div>
      <ul className="space-y-3 ml-10">
        {section.items.map((item, itemIndex) => (
          <li key={itemIndex} className="flex items-start">
            <span className="inline-block w-2 h-2 bg-blue-500 rounded-full mt-2 mr-3"></span>
            <span className="text-gray-700">{item}</span>
          </li>
        )))
      </ul>
    </div>
  )));
</div>
```

- วนลูป (map) เพื่อแสดง หัวข้อและเงื่อนไข แต่ละข้อ
- ใช้ Bullet Point (span วงกลมเล็ก) แทนจุดแบบธรรมดा

7.4 ปุ่ม Checkbox ยอมรับเงื่อนไข

```
/* Acceptance Section */
<div className="mt-10 p-6 bg-gray-50 rounded-lg">
  <div className="flex items-start mb-6">
    <input
      type="checkbox"
      id="agreement-checkbox"
      checked={isAccepted}
      onChange={() => setIsAccepted(!isAccepted)}
      className="mt-1 w-5 h-5 text-blue-600 rounded focus:ring-blue-500"
    />
    <label htmlFor="agreement-checkbox" className="ml-3 block text-gray-700">
      <span className="font-medium">ฉันยอมรับเงื่อนไขการใช้งานทั้งหมดที่ระบุไว้ข้างต้น</span>
      <p className="text-sm text-gray-500 mt-1">โดยการคลิกยืนยัน ฉันรับทราบว่าสิ่งใดๆ ก็ตามและเข้าใจข้อตกลงทั้งหมดแล้ว</p>
    </label>
  </div>
</div>
```

- ถ้าผู้ใช้ติํกที่ Checkbox → isAccepted จะเป็น true

8. SearchDate

8.1 การกำหนดค่าเริ่มต้นและ State

```
export default function BookingPage() {
  const [day, setDay] = useState("10");
  const [month, setMonth] = useState("05");
  const [year, setYear] = useState("2568");
  const [hour, setHour] = useState("11");
  const [minute, setMinute] = useState("30");
  const [pickupLocation, setPickupLocation] = useState("MRT_Pahonyothin");

  const formattedDate = `${year}-${month}-${day}T${hour}:${minute}`;
}
```

- ใช้ useState เพื่อเก็บข้อมูล วันที่ เวลา และสถานที่รับรถ
- ค่าตั้งต้นของปีเป็น 2568 (พ.ศ.) ตามปฏิทินไทย
- คำนวนค่าวันที่เป็น formattedDate เพื่อส่งไปที่ URL

8.2 เลือกวันที่ เวลา และสถานที่รับรถ

```
/* เลือกวัน */


<label>วัน / เดือน / ปี</label>
  <div>
    <input type="text" value={day} onChange={(e) => setDay(e.target.value)} />
    <input type="text" value={month} onChange={(e) => setMonth(e.target.value)} />
    <input type="text" value={year} onChange={(e) => setYear(e.target.value)} />
  </div>



/* เลือกเวลา */


<label>เวลา</label>
  <div>
    <input type="text" value={hour} onChange={(e) => setHour(e.target.value)} />
    <span>:</span>
    <input type="text" value={minute} onChange={(e) => setMinute(e.target.value)} />
    <span>:</span>
  </div>



/* เลือกสถานที่ */


<label>สถานที่รับรถ</label>
  <select>
    <option value="MRT_Pahonyothin">สถานที่ MRT พหลโยธิน</option>
    <option value="DMK">สถานที่เดลิมอนด์</option>
    <option value="BKK">สถานที่กรุงเทพฯ</option>
  </select>


```

9. SearchDate

9.1 State และการจัดการข้อมูล

```

const [pickupDate, setPickupDate] = useState(formatDate(initialPickupDate));
const [pickupLocation, setPickupLocation] = useState(initialPickupLocation);
const [returnDate, setReturnDate] = useState("");
const [cars, setCars] = useState<any[]>([]);
const [loading, setLoading] = useState(true);

```

- เก็บค่า วันที่รับรถ, สถานที่รับรถ, วันที่คืนรถ, รายการรถทั้งหมด
- ใช้ loading เพื่อตรวจสอบสถานะการโหลดข้อมูล

9.2 การดึงข้อมูลรถจาก API

```

useEffect(() => {
  console.log("pickupDate from URL:", initialPickupDate);
  console.log("pickupLocation from URL:", initialPickupLocation);

  setPickupDate(formatDate(initialPickupDate));
  setPickupLocation(initialPickupLocation);

  const fetchCars = async () => {
    try {
      const res = await fetch("/api/getcar");
      const data = await res.json();
      if (res.ok) {
        setCars(data);
      } else {
        console.error("Error fetching cars data:", data.message);
      }
      setLoading(false);
    } catch (error) {
      console.error("Error fetching cars data:", error);
      setLoading(false);
    }
  };
  fetchCars();
}, [initialPickupDate, initialPickupLocation]);

```

- โหลดข้อมูลรถ จาก API /api/getcar และอัปเดตรายการ cars
- ตรวจสอบ initialPickupDate, initialPickupLocation ถ้าเปลี่ยนค่าให้โหลดข้อมูลใหม่

9.3 การค้นหารถตามตัวกรอง

```
const handleSearch = async () => {
  setLoading(true);

  const body = {
    minPrice: minPrice ? Number(minPrice) : 1,
    maxPrice: maxPrice ? Number(maxPrice) : 9999999,
    carType,
    pickupLocation,
    returnLocation,
    rentalDays: rentalDays ? Number(rentalDays) : 1,
  };

  try {
    const response = await fetch("/api/SearchCars", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(body),
    });
  }

  const data = await response.json();

  if (response.ok) {
    setCars(data);
  } else {
    console.error("Error fetching cars:", data.message);
  }
} catch (error) {
  console.error("Error fetching cars:", error);
}

setLoading(false);
};
```

- เมื่อกดปุ่ม "ค้นหา" จะส่งค่าตัวกรองไปที่ /api/SearchCars (เป็น POST Request)
- หากสำเร็จจะอัปเดตรายการ cars ด้วยผลลัพธ์ใหม่

9.4 การเลือกและจองรถ

```
const handleDetails = (selectedCar: any) => {
  setBooking({
    pickupDate: pickupDate || "",
    pickupLocation: pickupLocation || "",
    returnDate: returnDate || "",
    selectedCar,
  });

  // [อ่านคู่รุ่นเพื่อให้ setBooking ทำงานก่อนเปลี่ยนหน้า]
  setTimeout(() => {
    router.push(`/rentdetail`);
  }, 300);
};

if (loading) return <div>[กำลังโหลด]...</div>;
```

- ใช้ Context API (useBooking) เพื่อเก็บข้อมูลรถที่เลือก
- หลังจากอัปเดต setBooking ให้ redirect ไปที่ /rentdetail

10 Profile

10.1 การแจ้งเตือน (NotificationSettings)

```
export function NotificationSettings() {
  const [notifications, setNotifications] = useState<Notification[]>([]);
  const [expandedNotification, setExpandedNotification] = useState<number | null>(null);
```

- โหลดข้อมูลแจ้งเตือนจาก API /api/getNotifications
- ตรวจสอบข้อมูลที่ได้จาก API
- แสดงรายการแจ้งเตือน และให้กดเพื่อขยายดูรายละเอียด

10.2 พังค์ชั่นจัดการโปรไฟล์หลัก (Pro)

```
export default function Pro() {
  const [activeTab, setActiveTab] = useState<string | null>(null);
  const [userName, setUserName] = useState(""); // เริ่มต้นเป็น string ว่าง
  const [phoneNumber, setPhoneNumber] = useState(""); // เริ่มต้นเป็น string ว่าง
  const [oldPassword, setOldPassword] = useState(""); // รหัสผ่านเก่า
  const [newPassword, setNewPassword] = useState(""); // รหัสผ่านใหม่
  const router = useRouter();
```

- ใช้ useState จัดการสถานะของ แท็บที่เลือก
- ใช้ useEffect โหลด userName จาก localStorage
- มีเมนูด้านซ้ายให้เลือกเปลี่ยนข้อมูลต่าง ๆ

10.3 เมนูจัดการบัญชี (menuItems)

```
const menuItems = [
  { id: "profile", label: "จัดการบัญชีโปรไฟล์" },
  { id: "password", label: "เปลี่ยนรหัสผ่าน" },
  { id: "notification", label: "การแจ้งเตือน" },
  { id: "contact", label: "ติดต่อ" },
];
interface DataToSend {
  userName?: string;
  phoneNumber?: string;
}
```

- เก็บ ตัวเลือกเมนู สำหรับกดเลือกแท็บ
- ใช้ id เพื่อกำหนดค่าให้ activeTab

10.4 ดึงข้อมูลและอัปเดตโปรไฟล์ (handleSubmit)

```
const handleSubmit = async () => {
  const token = localStorage.getItem("token");
  const customerID = sessionStorage.getItem("CustomerID"); // ดึง CustomerID จาก sessionStorage
```

- ดึง token จาก localStorage และ customerID จาก sessionStorage
- ตรวจสอบว่าผู้ใช้กรอกข้อมูลหรือไม่
- ส่ง API /api/manageUsername เพื่ออัปเดตโปรไฟล์

10.5 เปลี่ยนรหัสผ่าน (handleSubmitPasswordChange)

```
const handleSubmitPasswordChange = async () => {
  const token = localStorage.getItem("token");
  const customerID = sessionStorage.getItem("CustomerID"); // ดึง CustomerID จาก sessionStorage
  console.log("CustomerID:", customerID);

  if (!customerID) {
    alert("กรุณาเข้าสู่ระบบก่อน");
    return;
  }
```

- ตรวจสอบ customerID และ token
- ส่ง oldPassword และ newPassword ไปที่ API /api/changePW

10.6 ออกจากระบบ (handleLogout)

```
//  
const handleLogout = async () => {  
  const name = sessionStorage.getItem("name");  
  if (!name) {  
    return;  
  }  
  if (typeof window !== 'undefined') {  
    const token = localStorage.getItem("token");  
    sessionStorage.removeItem("CustomerID");  
    sessionStorage.removeItem("name");  
  
    try {  
      const response = await fetch("/api/logout", {  
        method: "POST",  
        headers: {  
          "Authorization": `Bearer ${token}`,  
        },  
      });  
  
      const result = await response.json();  
      alert(result.message); // ออกจากระบบสำเร็จ  
    } catch (error) {  
      console.error(error);  
    }  
  }  
};
```

- ลบ CustomerID และ name ออกจาก sessionStorage
- ลบ token ออกจาก localStorage
- ส่ง API /api/logout แล้วเปลี่ยนหน้าไปยังหน้าหลัก

11. Checkout

11.1 ฟอร์มกรอกข้อมูลผู้เช่า

```
const handleFileChange = (e: React.ChangeEvent<HTMLInputElement>) => {  
  const selectedFile = e.target.files?.[0];  
  if (selectedFile) {  
    setFile(URL.createObjectURL(selectedFile));  
  }  
};
```

- ใช้ useState จัดเก็บข้อมูลผู้เช่า
- มีฟิล์ฟ์ firstName, lastName, email, phone, address, additionalInfo

11.2 สรุปข้อมูลการเช่ารถ และอัปโหลดสลิป

11.2.1 ปุ่มลบไฟล์

```
const handleRemoveFile = () => {  
  setFile(null);  
};
```

12. Content

12.1 พังก์ชันส่งฟอร์ม

```
const handleSubmit = (e: React.FormEvent) => {
  e.preventDefault();
  setIsSubmitting(true);

  setTimeout(() => [
    setIsSubmitting(false),
    setSuccessMessage("บันทึกสำเร็จ"),
    setUsername(""),
    setMessage("")
  ], 1500);
};
```

- ป้องกันการรีเฟรชหน้า → e.preventDefault();
- แสดงสถานะ "กำลังส่ง..." → setIsSubmitting(true);
- จำลองการส่งฟอร์ม (ใช้ setTimeout 1.5 วินาที)
- ตั้งค่า isSubmitting กลับเป็น false
- แสดงข้อความสำเร็จ
- รีเซ็ตค่าฟอร์ม

12.2 แสดงข้อความแจ้งเตือน

```
{successMessage && (
  <div className="p-3 bg-green-100 text-green-700 rounded-lg text-center animate-fade-in">
    {successMessage}
  </div>
)
{errorMessage && (
  <div className="p-3 bg-red-100 text-red-700 rounded-lg text-center animate-fade-in">
    {errorMessage}
  </div>
)
}</div>
```

- ถ้ามีข้อความสำเร็จ (successMessage) → แสดงพื้นหลังสีเขียว
- ถ้ามีข้อความผิดพลาด (errorMessage) → แสดงพื้นหลังสีแดง

Backend

1.Signin

1.1 การนำเข้าที่จำเป็น

```
import { NextRequest, NextResponse } from "next/server";
import { mysqlPool } from "@/utils/db";
import bcrypt from "bcryptjs";
```

หน้าที่ของแต่ละตัว:

- NextRequest, NextResponse: ใช้กับ Next.js API route แบบใหม่ (App Router)
- mysqlPool: เป็น connection pool ที่ใช้เชื่อมกับฐานข้อมูล MySQL
- bcrypt: ใช้สำหรับ เข้ารหัสรหัสผ่าน (hash password) ก่อนบันทึกลงฐานข้อมูล

1.2 พิ้งก์ชัน POST() – รับคำขอสมัครสมาชิก

```
export async function POST(req: NextRequest) { // กำหนดประเภทของ req
  console.log("📌 API signin hit");
```

หน้าที่

เป็นพิ้งก์ชันที่ทำงานเมื่อมี HTTP POST มาที่ /api/signin
รับ object req ซึ่งเป็นข้อมูลของ request จากฝั่ง client

1.3 อ่านข้อมูลจาก body ของ request

```
try {
  const body = await req.json(); // อ่าน JSON request body
  const { firstName, lastName, username, phone, password } = body;
```

หน้าที่

- แปลง JSON body ที่ client ส่งเข้ามาให้เป็น object
- ดึงข้อมูลจากฟอร์มที่ผู้ใช้กรอก: ชื่อ, นามสกุล, ชื่อผู้ใช้, เบอร์โทร, รหัสผ่าน

1.4 ตรวจสอบว่ากรอกข้อมูลครบหรือไม่

```
// ตรวจสอบข้อมูลที่กรอก
if (!firstName || !lastName || !username || !phone || !password) {
    return NextResponse.json({ message: "⚠️ กรุณากลับเข้ามูลให้ครบถ้วน!", { status: 400 } });
}
```

ถ้ามีช่องว่างเปล่า → ส่ง response กลับทันทีพร้อมสถานะ 400 Bad Request

1.5 เข้ารหัสผ่านด้วย bcrypt

```
}
```

// เข้ารหัสผ่านก่อนเก็บในฐานข้อมูล
const hashedPassword = await bcrypt.hash(password, 10); // 10 คือจำนวน salt rounds

- ใช้ bcrypt เข้ารหัสผ่านก่อนเก็บลงฐานข้อมูล
- 10 คือจำนวนรอบในการ salt (รอบที่ใช้ทำให้รหัสปลดภัยชี้น)

1.6 สร้าง SQL Query และบันทึกลงฐานข้อมูล

```
const query = `
    INSERT INTO Customer (firstName, lastName, userName, phoneNumber, password)
    VALUES (?, ?, ?, ?, ?)
`;
```



```
await mysqlPool.promise().query(query, [firstName, lastName, username, phone, hashedPassword]);
```

- ใช้คำสั่ง INSERT เพื่อบันทึกข้อมูลลงในตาราง Customer
- ใช้ ? เพื่อป้องกัน SQL Injection (prepared statement)

1.7 ส่ง response กลับไปยัง client

```
return NextResponse.json({ message: "✅ สมัครสมาชิกสำเร็จ!", { status: 200 }));
```

ถ้าบันทึกข้อมูลได้สำเร็จ → ตอบกลับว่า “สมัครสมาชิกสำเร็จ” พร้อม status 200 OK

1.8 กรณีเกิด Error

```
// app/api/login.js
try {
  catch (err) {
    console.error("⚠️ Error inserting customer data:", err);
    return NextResponse.json({ message: "❌ ล้มเหลวในการสมัครสมาชิก" }, { status: 500 });
  }
}
```

- ถ้ามี error เช่น เชื่อมต่อฐานข้อมูลไม่ได้ → แสดง log บน console
- ส่งกลับ status 500 Internal Server Error พร้อมข้อความ

2. Login

2.1 การ import โมดูล

```
// app/api/login.js
import { NextRequest, NextResponse } from "next/server";
import { mysqlPool } from "@/utils/db";
import bcrypt from "bcryptjs"; // ใช้ bcryptjs และ bcrypt
const jwt = require('jsonwebtoken'); // ใช้ jwt สำหรับสร้าง Token
import { config } from "@/config"; // ตั้ง secretkey
```

- NextRequest, NextResponse: ใช้ใน API route แบบใหม่ (Next.js App Router)
- mysqlPool: ใช้เชื่อมตอกับฐานข้อมูล MySQL
- bcryptjs: ใช้ตรวจสอบความถูกต้องของรหัสผ่าน (เปรียบเทียบกับรหัสที่ถูกเข้ารหัส)
- jsonwebtoken: ใช้สร้าง JWT token เพื่อยืนยันตัวตนของผู้ใช้
- config: ใช้โหลดค่า JWT_SECRET ที่เก็บไว้แยกต่างหาก

2.2 ตั้งค่า Secret Key สำหรับ JWT

```
const secret = config.JWT_SECRET; // ตั้งค่า secret key สำหรับ JWT
```

2.3 พังก์ชัน POST() — Main Function

```
export async function POST(req: NextRequest) {
```

2.4 อ่านข้อมูลจาก body และตรวจสอบความถูกต้อง

```
// อ่าน JSON request body
const body = await req.json();
const { phone, password } = body;

// ตรวจสอบข้อมูลที่กรอก
if (!phone || !password) {
  return NextResponse.json({ message: "⚠️ กรุณากรอกข้อมูลให้ครบถ้วน" }, { status: 400 });
}
```

- อ่านค่าที่ผู้ใช้กรอกจาก body: เบอร์โทรศัพท์ และ รหัสผ่าน
- ถ้าไม่ครบ ส่ง error 400

2.5 ดึงข้อมูลผู้ใช้จากฐานข้อมูล

```
// ดึงข้อมูลผู้ใช้จากฐานข้อมูล
const query = `SELECT * FROM Customer WHERE phoneNumber = ?`;
```



```
const [rows] = await mysqlPool.promise().query(query, [phone]);
```

ใช้คำสั่ง SQL เพื่อหาແղວໃນตาราง Customer ที่มีเบอร์โทรศัพท์ที่ผู้ใช้กรอก

2.6 ตรวจสอบว่ามีผู้ใช้หรือไม่

```
// เช็คจำนวนแถว
if (rows.length === 0) {
  return NextResponse.json({ message: "✗ เบอร์โทรศัพท์นี้ไม่มีในระบบ" }, { status: 404 });
}
```

ถ้าไม่มีข้อมูลผู้ใช้งาน → ตอบกลับว่า “ไม่พบผู้ใช้”

2.7 ตรวจสอบรหัสผ่านด้วย bcrypt

```
// ตรวจสอบรหัสผ่าน
const user = rows[0];
const isPasswordValid = await bcrypt.compare(password, user.password);

if (!isPasswordValid) {
  return NextResponse.json({ message: "✗ รหัสผ่านไม่ถูกต้อง" }, { status: 401 });
}
```

- เปรียบเทียบรหัสผ่านที่ผู้ใช้กรอกกับรหัสที่ถูกเข้ารหัสใน DB
- ถ้าไม่ตรงกัน → ตอบกลับว่า “รหัสผ่านไม่ถูกต้อง”

2.8 ตรวจสอบ admin และสร้าง token

```
if (user.phoneNumber === '0888888888') {
  // ถ้าเป็น admin
  const token = jwt.sign(
    { role: "admin", phone: user.phoneNumber },
    secret, // คีย์ที่ใช้สร้าง JWT
    { expiresIn: "1h" }
  );
  return NextResponse.json({ message: "✓ เข้าสู่ระบบสำเร็จ (admin)", token, role: "admin" }, { status: 200 });
}
```

- ถ้าหมายเลขโทรศัพท์ตรงกับ admin → สร้าง JWT สำหรับ admin

- เพิ่ม role เป็น "admin" ใน payload ของ token
- ส่งกลับ token + message

2.9 ถ้าเป็นผู้ใช้ทั่วไป

```
// สร้าง JWT Token
const token = jwt.sign(
  { id: user.CustomerID, phone: user.phoneNumber, role: "user" ,firstName:user.firstName,lastName:user.lastName,userName:user.userName},
  secret,
  { expiresIn: "1h" } // ตั้งค่าเวลาหมดอายุของ Token
);
```

- สร้าง token แบบมีข้อมูลผู้ใช้ใส่ไว้ใน payload
- Token นี้ใช้ยืนยันตนบนฝั่ง client ในหน้าอื่น ๆ
- หมดอายุใน 1 ชั่วโมง

2.10 ตอบกลับเมื่อเข้าสู่ระบบสำเร็จ

```
// ส่งการตอบกลับเมื่อเข้าสู่ระบบสำเร็จ พร้อม JWT Token
return NextResponse.json({ message: "✅ เข้าสู่ระบบสำเร็จ", token }, { status: 200 });
```

2.11 กรณีเกิดข้อผิดพลาด

```
} catch (err) {
  console.error("⚠️ Error during login:", err);
  return NextResponse.json({ message: "❌ ล้มเหลวในการเข้าสู่ระบบ" }, { status: 500 });
}
```

ถ้ามีปัญหาเช่น ฐานข้อมูลล่ม หรือรหัสไม่ปัญหา → ตอบกลับ error 500

3.Addcars

3.1 พังก์ชัน POST(req) (พังก์ชันหลักของ API) และ อ่านค่าจาก FormData

```

    < export async function POST(req) {
    <   try {
    <     const formData = await req.formData();

    // ตรวจสอบข้อมูลที่รับมา
    <     const LPlate = formData.get("LPlate");
    <     const model = formData.get("model");
    <     const brand = formData.get("brand");
    <     const carType = formData.get("carType");
    <     const status = formData.get("status");
    <     const rentalPrice = formData.get("rentalPrice");
    <     const carImg = formData.get("carImg");

    // ตรวจสอบข้อมูลที่รับมา
    <     if (!LPlate || !model || !brand || !carType || !rentalPrice || !status) {
    <       return NextResponse.json(
    <         { message: "⚠️ กรุณากรอกข้อมูลให้ครบถ้วน" },
    <         { status: 400 }
    <       );
    <     }
  
```

- ใช้สำหรับรับคำขอ POST จาก Frontend
- รับข้อมูลจาก FormData เพื่อเพิ่มข้อมูลรถ

3.2 ตรวจสอบว่าข้อมูลครบถ้วนหรือไม่

```

const carImg = formData.get( carImg );

// ตรวจสอบข้อมูลที่รับมา
if (!LPlate || !model || !brand || !carType || !rentalPrice || !status) {
  return NextResponse.json(
    { message: "⚠️ กรุณากรอกข้อมูลให้ครบถ้วน" },
    { status: 400 }
  );
}
  
```

เช็คว่าข้อมูลสำคัญครบหรือไม่ ถ้าไม่ครบจะส่ง 400 Bad Request

3.3 คำสั่ง SQL สำหรับบันทึกข้อมูล

```

// สั่งคำสั่ง SQL
const query = `
    INSERT INTO Car (
        LPlate,
        model,
        brand,
        carType,
        rentalPrice,
        status,
        carImg
    ) VALUES (?, ?, ?, ?, ?, ?, ?)
`;

```

3.4 ส่ง Response กลับไปหา Frontend (กรณีสำเร็จ)

```

return NextResponse.json(
    {
        success: true,
        message: "เพิ่มข้อมูลรถเรียบร้อยแล้ว",
        data: { LPlate, model, brand }
    },
    { status: 200 }
);

```

ถ้าบันทึกสำเร็จ จะส่ง JSON กลับไป

3.5 ส่ง Response กลับไปหา Frontend (กรณีผิดพลาด)

```

return NextResponse.json(
    {
        success: false,
        message: errorMessage,
        error: err.message
    },
    { status: 500 }
);

```

4.Addstaff

4.1 พังก์ชัน GET() – ดึงข้อมูลพนักงาน

```

    export async function GET() {
      try {
        const query = `SELECT staffID, firstName, lastName, phoneNumber FROM Staff`;
        const [rows] = await mysqlPool.promise().query(query);

        console.log("Data from DB: ", rows); // Debug

        return NextResponse.json(rows, { status: 200 });
      } catch (err) {
        console.error("ERROR: ", err.message, err.stack);
        return NextResponse.json({ message: `Internal Error: ${err.message}` }, { status: 500 });
      }
    }
  
```

- เพื่อดึงข้อมูลพนักงานทั้งหมดจากฐานข้อมูล MySQL
- ใช้ mysqlPool.promise().query(query) → เพื่อใช้ Promise-based API ของ MySQL
- ถ้าได้ผลลัพธ์ → ส่ง rows (ข้อมูลทั้งหมดที่ดึงมา) กลับไปในรูปแบบ JSON
- ถ้ามีข้อผิดพลาด (catch block) → ส่ง JSON Response ที่มี error message กลับไป (HTTP Status 500)

4.2 พังก์ชัน POST(req) – เพิ่มข้อมูลพนักงาน

```

export async function POST(req) {
  try {
    const { firstName, lastName, phoneNumber } = await req.json();
    const query = `INSERT INTO Staff (firstName, lastName, phoneNumber) VALUES (?, ?, ?)`;
    const [result] = await mysqlPool.promise().query(query, [firstName, lastName, phoneNumber]);

    return NextResponse.json({ message: "Thêm thành công!", staffID: result.insertId }, { status: 201 });
  } catch (err) {
    console.error("ERROR: ", err.message, err.stack);
    return NextResponse.json({ message: `Internal Error: ${err.message}` }, { status: 500 });
  }
}
  
```

- อ่านข้อมูลที่ส่งมาจาก Request Body
- ใช้ SQL Query ในการ INSERT ข้อมูล
- ? → ใช้ Prepared Statement เพื่อป้องกัน SQL Injection
- mysqlPool.promise().query(query, [firstName, lastName, phoneNumber]) → ทำให้ Query เป็น Promise-based API
- result.insertId → คือ staffID ที่เพิ่งถูกเพิ่มเข้าฐานข้อมูล
- กรณีเกิดข้อผิดพลาด (catch block)

5.Cars

5.1 พังก์ชัน GET() – ดึงข้อมูลรถที่พร้อมให้เช่า

```

export async function GET() {
  try {
    const query = "SELECT * FROM Car WHERE status = 'พร้อมให้เช่า'";
    const [rows] = await mysqlPool.promise().query(query);

    return NextResponse.json(rows, { status: 200 });
  } catch (error) {
    console.error("Error fetching cars:", error);
  }

  return NextResponse.json({ message: "Failed to fetch data" }, { status: 500 });
}

```

5.1.1 สร้าง SQL Query เพื่อดึงข้อมูลรถที่ "พร้อมให้เช่า"

5.1.2 ใช้ mysqlPool.promise().query(query)

5.1.3 ส่งข้อมูลกลับเป็น JSON Response

5.1.4 จัดการข้อผิดพลาด (catch block)

6.ChangePW

6.1 พังก์ชัน PUT(req) – อัปเดตรหัสผ่าน

```

export async function PUT(req: NextRequest) {
  try {
    console.log("⭐ API password hit");

    // รับค่า oldPassword, newPassword และ customerID จาก request body
    const { oldPassword, newPassword, customerID } = await req.json();
  }
}

```

รับข้อมูลจาก req.json()

- oldPassword → รหัสผ่านเก่าที่ผู้ใช้ป้อน
- newPassword → รหัสผ่านใหม่ที่ต้องการเปลี่ยน
- customerID → ไอดีของลูกค้าที่ต้องการเปลี่ยนรหัส

6.2 ตรวจสอบ Token Authentication

```
// เช็คว่า Token มีค่าหรือไม่
const token = req.headers.get("Authorization")?.split(" ")[1];

// เช็คว่า Token มีค่าหรือไม่
if (!token) {
  console.error("⚠️ No token provided");
  return NextResponse.json({ error: "กรุณาเข้าสู่ระบบ" }, { status: 401 });
}
```

- ดึง JWT Token จาก Header (Authorization: Bearer <token>)
- ถ้าไม่มี Token → ส่งสถานะ 401 Unauthorized

6.3 ดึงรหัสผ่านเดิมจากฐานข้อมูล

```
// ตรวจสอบรหัสผ่านเดิม
const getUserQuery = `SELECT password FROM Customer WHERE CustomerID = ?`;
const [user]: any = await mysqlPool.promise().query(getUserQuery, [customerID]);

if (user.length === 0) {
  return NextResponse.json({ error: "ไม่พบผู้ใช้" }, { status: 404 });
}
```

- ใช้ SELECT password FROM Customer WHERE CustomerID = ? เพื่อดึงรหัสผ่านของลูกค้าจากฐานข้อมูล
- ถ้าไม่พบข้อมูลในฐานข้อมูล → ส่ง HTTP 404 Not Found

6.4 เข้ารหัสผ่านใหม่ และอัปเดตในฐานข้อมูล

```
const updatePasswordQuery = `UPDATE Customer SET password = ? WHERE CustomerID = ?`;
await mysqlPool.promise().query(updatePasswordQuery, [hashedNewPassword, customerID]);

return NextResponse.json({ message: "เปลี่ยนรหัสผ่านสำเร็จ" });

} catch (error) {
  console.error("⚠️ Error in manageUsername:", error);
  return NextResponse.json({ error: "Internal Server Error" }, { status: 500 });
}
```

- เข้ารหัส (bcrypt.hash) รหัสผ่านใหม่
- ใช้ SQL UPDATE เพื่ออัปเดตรหัสผ่านใหม่ของลูกค้าในฐานข้อมูล
- ส่ง HTTP 200 OK พร้อมข้อความ "เปลี่ยนรหัสผ่านสำเร็จ"

7.deletecar

```
import { NextResponse } from "next/server";
import { mysqlPool } from "@/utils/db";
```

- NextResponse: เป็นเครื่องมือของ Next.js ที่ใช้สำหรับส่ง JSON response กลับไปที่คลาวน์
- mysqlPool: เป็นตัวเชื่อมต่อฐานข้อมูล MySQL ที่ใช้ connection pooling เพื่อให้การเชื่อมต่อเร็วขึ้นและประหยัดทรัพยากร

7.1 กำหนดฟังก์ชัน DELETE(req)

```
export async function DELETE(req) {
```

- ฟังก์ชันนี้ใช้สำหรับจัดการ HTTP DELETE request ที่ส่งมาจากคลาวน์
- เป็นฟังก์ชันแบบ asynchronous เพราะต้องรอการทำงานกับฐานข้อมูล

7.2 รับข้อมูลจาก request

```
const { LPlate } = await req.json();
```

- ดึงค่าข้อมูลที่ส่งมาจากการ request body (ในรูปแบบ JSON)
- คาดหวังให้มีค่าตัวแปร LPlate (ป้ายทะเบียนรถ)

7.3 ตรวจสอบว่าผู้ใช้งานได้กรอกป้ายทะเบียนมากหรือไม่

```
if (!LPlate) {
  return NextResponse.json({ message: "⚠️ กรุณากรอกป้ายทะเบียนรถ" }, { status: 400 });
```

- ถ้าผู้ใช้ไม่กรอก LPlate จะส่ง response กลับไปพร้อมสถานะ 400 (Bad Request)
- ⚠️ หมายความว่า: คลาวน์ต้องส่งข้อมูลให้ถูกต้องก่อน

7.4 กำหนดคำสั่ง SQL สำหรับลบข้อมูล

```
const query = "DELETE FROM Car WHERE LPlate = ?";
```

- คำสั่ง SQL DELETE ใช้สำหรับลบรายการฐานข้อมูล
- ? เป็น placeholder สำหรับค่าที่จะใส่ลงไป (เพื่อป้องกัน SQL Injection)

7.5 รันคำสั่ง SQL และรับผลลัพธ์

```
const [result] = await mysqlPool.promise().query(query, [LPlate]);
```

- ใช้ mysqlPool.promise().query() เพื่อให้รองรับ async/await
- [result] คือผลลัพธ์จากการรันคำสั่ง SQL
- result.affectedRows คือจำนวนแถวที่ถูกลบออกจากฐานข้อมูล

7.6 ตรวจสอบว่ามีข้อมูลให้ลบหรือไม่

```
if (result.affectedRows === 0) {
  return NextResponse.json({ message: "ไม่มีรถที่ต้องการลบ" },
  { status: 404 });
}
```

- ถ้า affectedRows === 0 หมายความว่า ไม่มีรถที่ตรงกับป้ายทะเบียนที่ส่งมา
- ส่ง response พร้อม สถานะ 404 (Not Found)

7.7 ส่ง response เมื่อลบสำเร็จ

```
return NextResponse.json({ message: "ลบข้อมูลรถเรียบร้อยแล้ว" },
{ status: 200 });
```

- ถ้ารถถูกลบสำเร็จ จะส่ง response พร้อม สถานะ 200 (OK)
- แจ้งให้ผู้ใช้ทราบว่า ลบข้อมูลเรียบร้อยแล้ว

7.8 จัดการข้อผิดพลาด (Error Handling)

```
} catch (err) {
  console.error("⚠️ ERROR: ", err.message, err.stack);
  return NextResponse.json({ message:
    `การเชื่อมต่อเซิร์ฟเวอร์ล้มเหลว: ${err.message}` }, { status:
    500 });
}
```

- ถ้ามีข้อผิดพลาด เช่น
 - ปัญหาการเชื่อมต่อฐานข้อมูล

- คำสั่ง SQL ผิดพลาด
- จะบันทึก error ลง console และส่ง response พร้อมสถานะ 500 (Internal Server Error)

8. deletecustomer

```
import { NextResponse } from "next/server";
import { mysqlPool } from "@/utils/db"; // เชื่อมต่อฐานข้อมูล MySQL
```

- NextResponse: ใช้สำหรับสร้างและส่ง JSON response กลับไปยังไคลเอนต์
- mysqlPool: เป็นตัวเชื่อมต่อฐานข้อมูล MySQL โดยใช้ connection pool เพื่อประยุกต์ทรัพยากร

8.1 กำหนดฟังก์ชัน DELETE(req)

```
export async function DELETE(req) {
```

- ฟังก์ชันนี้ถูกเรียกใช้เมื่อมี HTTP DELETE request มาจากเซิร์ฟเวอร์
- เป็นฟังก์ชันแบบ async เพราะต้องรอการทำงานกับฐานข้อมูล

8.2 ดึงค่าจาก request body

```
// ดึงข้อมูลจาก body ของคำขอ
const { customerID } = await req.json(); // เปลี่ยนจาก CustomerID  
เป็น customerID
```

- รับค่า customerID ที่ส่งมาจากไคลเอนต์
- ใช้ await req.json() เพื่อแปลงข้อมูล JSON ที่ส่งมาเป็น JavaScript Object

8.3 ตรวจสอบว่าผู้ใช้กรอก customerID มาหรือไม่

```
if (!customerID) {
  return NextResponse.json({ message: "⚠️ กรุณาระบุ customerID" }, { status: 400 });
```

- ถ้า ไม่มี customerID ที่ส่งมา → ส่ง response แจ้งเตือน 400 (Bad Request)
- ป้องกันข้อผิดพลาดที่เกิดจากการส่งข้อมูลไม่ครบ

8.4 เขียนคำสั่ง SQL สำหรับลบข้อมูล

```
// เขียนคำสั่ง SQL เพื่อลบข้อมูลจากฐานข้อมูล
const query = `DELETE FROM Customer WHERE CustomerID = ?`;
```

- ใช้คำสั่ง SQL DELETE เพื่อลบข้อมูลจากตาราง Customer
- ? เป็น placeholder สำหรับการป้องกัน SQL Injection

8.5 รันคำสั่ง SQL และรับผลลัพธ์

```
// ทำการ query และลบข้อมูล
const [result] = await mysqlPool.promise().query(query,
[customerID]);
```

- ใช้ mysqlPool.promise().query() เพื่อรอรับ async/await
- [result] คือผลลัพธ์จากการรัน SQL

8.6 ตรวจสอบว่ามีแถวที่ถูกลบหรือไม่

```
// ตรวจสอบว่าไม่มีแถวที่ถูกลบ (หมายความว่าไม่พบ customerID)
if (result.affectedRows === 0) {
  return NextResponse.json({ message:
    "ไม่พบผู้ใช้ที่ต้องการลบ" }, { status: 404 });
}
```

- ถ้า affectedRows === 0 หมายความว่า ไม่มีผู้ใช้ที่มี customerID ตรงกับที่ส่งมา
- ส่ง response แจ้งเตือน 404 (Not Found)

8.7 ส่ง response เมื่อลบสำเร็จ

```
// ส่งข้อความเมื่อการลบสำเร็จ
return NextResponse.json({ message: "ลบผู้ใช้เรียบร้อยแล้ว" }, {
  status: 200 });
```

- ถ้าการลบสำเร็จ ส่ง response พร้อม 200 (OK)

8.8 จัดการข้อผิดพลาด (Error Handling)

```

} catch (err) {
  console.error("⚠️ ERROR: ", err.message, err.stack);
  return NextResponse.json({ message: `ล้มเหลว: ${err.message}` }, { status: 500 });
}

```

- ถ้าเกิดข้อผิดพลาด เช่น
 - เชื่อมต่อฐานข้อมูลไม่สำเร็จ
 - คำสั่ง SQL ผิดพลาด
- จะแสดง error ใน console และส่ง response 500 (Internal Server Error)

9.deletestaff

```

import { NextResponse } from "next/server";
import { mysqlPool } from "@/utils/db";

```

- NextResponse: ใช้สำหรับสร้างและส่ง JSON response กลับไปที่client เอนเตอร์
- mysqlPool: เป็นตัวเชื่อมต่อ MySQL Database โดยใช้ connection pooling เพื่อประหยัดทรัพยากร

9.1 กำหนดฟังก์ชัน DELETE(req)

```

export async function DELETE(req) {

```

- ฟังก์ชันนี้ใช้จัดการ HTTP DELETE request
- ใช้ async เพราะต้องมีการเชื่อมต่อฐานข้อมูลที่ใช้เวลา

9.2 ดึงค่า staffID จาก request body

```

const { staffID } = await req.json();

```

- ใช้ await req.json() เพื่อแปลง JSON request body เป็น JavaScript Object
- ดึงค่า staffID ที่ส่งมาจาก client เอนเตอร์

9.3 ตรวจสอบว่าผู้ใช้ส่งค่า staffID มาหรือไม่

```

if (!staffID) {
  return NextResponse.json({ message: "⚠️ กรุณาระบุ staffID" },
  { status: 400 });
}

```

- ถ้า ไม่มี staffID → ส่ง response แจ้งเตือน 400 (Bad Request)

-  ป้องกันข้อผิดพลาดที่เกิดจากการส่งข้อมูลไม่ครบ

9.4 เขียนคำสั่ง SQL สำหรับลบข้อมูล

```
const query = `DELETE FROM Staff WHERE staffID = ?`;
```

- ใช้ SQL DELETE เพื่อลบพนักงานจากตาราง Staff
- ? เป็น placeholder เพื่อป้องกัน SQL Injection

9.5 รันคำสั่ง SQL และรับผลลัพธ์

```
const [result] = await mysqlPool.promise().query(query,
[staffID]);
```

- ใช้ mysqlPool.promise().query() เพื่อรับ response ของ async/await
- [result] คือผลลัพธ์ของคำสั่ง SQL

9.6 ตรวจสอบว่ามีข้อมูลที่ถูกลบหรือไม่

```
if (result.affectedRows === 0) {
  return NextResponse.json({ message:
    "ไม่พบพนักงานที่ต้องการลบ" }, { status: 404 });
}
```

- ถ้า affectedRows === 0 → ไม่มีพนักงานที่มี staffID ตรงกับที่ส่งมา
- ส่ง response แจ้งว่า 404 (Not Found)

9.7 ส่ง response เมื่อลบสำเร็จ

```
return NextResponse.json({ message: "ลบพนักงานเรียบร้อยแล้ว" }, {
  status: 200 });
```

- ถ้าลบข้อมูลสำเร็จ → ส่ง response พร้อม 200 (OK)

9.8 จัดการข้อผิดพลาด (Error Handling)

```

} catch (err) {
    console.error("⚠️ ERROR: ", err.message, err.stack);
    return NextResponse.json({ message: `ล้มเหลว: ${err.message}` },
        { status: 500 });
}

```

- ถ้ามีข้อผิดพลาด เช่น
 - เชื่อมต่อฐานข้อมูลไม่สำเร็จ
 - คำสั่ง SQL ผิดพลาด
- บันทึก error ลง console และส่ง response 500 (Internal Server Error)

10.getcar

```

import { NextResponse } from "next/server";
import { mysqlPool } from "@/utils/db";

```

- NextResponse: ใช้สำหรับสร้างและส่ง JSON response กลับไปที่ клиент
- mysqlPool: ใช้สำหรับเชื่อมต่อฐานข้อมูล MySQL

10.1 กำหนดพังก์ชัน GET(request)

```

export async function GET(request) {

```

- พังก์ชันนี้จะถูกเรียกเมื่อมี HTTP GET request manyang API
- ใช้ async เพราะต้องรอการทำงานของฐานข้อมูล

10.2 ดึงค่าพารามิเตอร์จาก URL

```

const { searchParams } = new URL(request.url);

const available = searchParams.get("available");
const minPrice = searchParams.get("minPrice");
const maxPrice = searchParams.get("maxPrice");
const searchTerm = searchParams.get("search");
const sortBy = searchParams.get("sortBy") || "rentalPrice";

```

ใช้ searchParams.get() เพื่อดึงค่าพารามิเตอร์จาก URL

ตัวอย่างพารามิเตอร์ที่สามารถใช้ได้

- available=true → คันหารถที่พร้อมให้เช่า
- minPrice=1000 → คันหารถที่มีราคาเช่าขั้นต่ำ 1,000
- maxPrice=5000 → คันหารถที่มีราคาเช่าสูงสุด 5,000
- search=Toyota → คันหารถที่มี brand หรือ model เป็น "Toyota"
- sortBy=brand → เรียงลำดับตาม brand

ถ้า sortBy ไม่มีค่า จะใช้ "rentalPrice" เป็นค่าเริ่มต้น

10.3 กำหนด URL ของรูปภาพ

```
const baseImageUrl = process.env.BASE_IMAGE_URL;
```

- BASE_IMAGE_URL เป็น environment variable ที่เก็บ URL หลักของรูปภาพ

10.4 สร้างคำสั่ง SQL Query

```
let query = "SELECT * FROM Car WHERE 1=1";
const params = [];
```

- ใช้ "WHERE 1=1" เพื่อให้สามารถเพิ่มเงื่อนไขได้ง่ายขึ้น
params เป็นอาร์เรย์ที่ใช้เก็บค่าพารามิเตอร์สำหรับป้องกัน SQL Injection

10.5 เพิ่มเงื่อนไขในการกรองข้อมูล

```

    if (available) {
      query += " AND status = ?";
      params.push(available);
    }
    if (minPrice) {
      query += " AND rentalPrice >= ?";
      params.push(parseFloat(minPrice));
    }
    if (maxPrice) {
      query += " AND rentalPrice <= ?";
      params.push(parseFloat(maxPrice));
    }
    if (searchTerm) {
      query += " AND (model LIKE ? OR brand LIKE ?)";
      params.push(`%${searchTerm}%`, `%${searchTerm}%`);
    }
  }

```

- available → กรองเฉพาะรถที่มีสถานะตรงกับค่า available
- minPrice & maxPrice → กรองรถตามช่วงราคา
- searchTerm → ค้นหารถจาก model หรือ brand ที่มีคำค้นหา

10.6 ตรวจสอบค่าที่ใช้เรียงลำดับ (sortBy)

```

const validSortColumns = ['rentalPrice', 'brand', 'model',
  'carType', 'status'];
const safeSortBy = validSortColumns.includes(sortBy) ? sortBy :
  'rentalPrice';
query += ` ORDER BY ${mysqlPool.escapeId(safeSortBy)} ASC`;

```

- เช็คว่า sortBy อยู่ในรายการที่กำหนด หรือไม่ (validSortColumns)
- ถ้าไม่ใช่ → ใช้ค่า "rentalPrice" เป็นค่าเริ่มต้น
- ใช้ mysqlPool.escapeId() เพื่อป้องกัน SQL Injection

10.7 รันคำสั่ง SQL และรับข้อมูลรถ

```
const [cars] = await mysqlPool.promise().query(query, params);
```

- ใช้ mysqlPool.promise().query() เพื่อรับ async/await
- [cars] คือ รายการรถทั้งหมด ที่ได้จากฐานข้อมูล

10.8 ปรับแต่ง URL รูปภาพของรถแต่ละคัน

```
const carsWithImageUrls = cars.map((car) => {
  let carImgUrl = null;
  if (car.carImg) {
    carImgUrl = car.carImg.startsWith('http')
      ? car.carImg
      : `${baseUrl}/${car.carImg.replace(/^\/+/, '')}`;
  }

  return {
    ...car,
    carImgUrl,
    // เก็บค่าเดิมไว้เพื่อความเข้ากันได้
    carImg: car.carImg
  };
});
```

- ตรวจสอบว่ามีรูปภาพ (carImg) หรือไม่
- ถ้า URL รูปภาพไม่ได้ชี้นั่นด้วย http → ให้ต่อ กับ baseUrl
- สร้าง carImgUrl ใหม่เพื่อเก็บ URL ที่ถูกต้อง

10.9 ส่งข้อมูลกลับไปยังโคลเลนต์

```
return NextResponse.json(carsWithImageUrls);
```

- ส่งข้อมูลรถ (รวมถึง URL รูปภาพที่แก้ไขแล้ว) กลับไป

10.10 จัดการข้อผิดพลาด (Error Handling)

```

    } catch (err) {
      console.error("🚨 Error fetching cars:", {
        message: err.message,
        stack: err.stack,
        url: request.url
      });

      return NextResponse.json(
        {
          message: "เกิดข้อผิดพลาดในการดึงข้อมูลรถ",
          error: err.message
        },
        { status: 500 }
      );
    }
  }
}

```

- ถ้าเกิดข้อผิดพลาด (เช่น ฐานข้อมูลล่ม) → บันทึก error ลง console
- ส่ง response 500 (Internal Server Error) กลับไป

11.getcustomer

11.1 นำเข้าโมดูลที่จำเป็น

```

import { NextResponse } from "next/server";
import { mysqlPool } from "@/utils/db"; // เชื่อมต่อฐานข้อมูล MySQL

```

- NextResponse → ใช้สำหรับสร้างและส่ง JSON response กลับไปที่โคลเอนต์
- mysqlPool → ใช้สำหรับเชื่อมต่อกับฐานข้อมูล MySQL

11.2 สร้างฟังก์ชัน GET() เพื่อดึงข้อมูลลูกค้า

```

export async function GET() {

```

- ฟังก์ชันนี้จะทำงานเมื่อมี HTTP GET request มาที่ API
- ใช้ async เพราะต้องรอผลลัพธ์จากฐานข้อมูล

11.3 สร้างคำสั่ง SQL Query

```
const query = `SELECT CustomerID, firstName, lastName,  
phoneNumber FROM Customer`;
```

- คำสั่ง SQL SELECT ใช้ดึงข้อมูลเฉพาะบางคอลัมน์จากตาราง Customer
- ข้อมูลที่ดึงมา
 - CustomerID → ไอดีลูกค้า
 - firstName → ชื่อลูกค้า
 - lastName → นามสกุล
 - phoneNumber → เบอร์โทรศัพท์

11.4 รันคำสั่ง SQL และรับข้อมูลลูกค้า

```
const [rows] = await mysqlPool.promise().query(query);
```

- ใช้ mysqlPool.promise().query() เพื่อรอรับ async/await
- rows คือ รายการลูกค้าทั้งหมด ที่ได้จากการดึงข้อมูล

11.5 แสดงข้อมูลที่ดึงมา (Debugging)

```
console.log("📢 Data from DB: ", rows); // Debug
```

- แสดงข้อมูลที่ดึงมาจากฐานข้อมูลใน console
- ใช้สำหรับ Debugging

11.6 ส่งข้อมูลกลับไปยังไคลเอนต์

```
return NextResponse.json(rows, { status: 200 });
```

- ส่ง JSON response ที่มีรายการลูกค้ากลับไปพร้อม HTTP Status 200 (OK)

11.7 จัดการข้อผิดพลาด (Error Handling)

```
        } catch (err) {
          console.error("🚨 ERROR: ", err.message, err.stack);
          return NextResponse.json({ message: "✗ ล้มเหลว: ${err.message}" },
            { status: 500 });
        }
      }
```

- ถ้าเกิดข้อผิดพลาด (เช่น ฐานข้อมูลล้ม)
 - แสดง error ใน console
 - ส่ง response 500 (Internal Server Error) พร้อมข้อความ ✗ ล้มเหลว

13.getNotifications

13.1 นำเข้าโมดูลที่จำเป็น

```
import { mysqlPool } from "@/utils/db";
```

- mysqlPool → ใช้สำหรับเชื่อมต่อกับฐานข้อมูล MySQL

13.2 กำหนดฟังก์ชัน handler() สำหรับจัดการ API

```
export default async function handler(req, res) {
```

- ฟังก์ชันนี้เป็น API Route Handler สำหรับจัดการคำขอ (request) และตอบกลับ (response)
- ใช้ async เพราะต้องรอผลลัพธ์จากฐานข้อมูล

13.3 ตรวจสอบประเภทของ HTTP Request

```
if (req.method !== "GET") {
  return res.status(405).json({ message: "Method Not Allowed" });
}
```

- ถ้า request ไม่ใช่ GET → ส่ง HTTP 405 (Method Not Allowed)
- ป้องกันการใช้ POST, PUT, DELETE บน API นี้

13.4 สร้างคำสั่ง SQL Query

```
const query = "SELECT * FROM Notifications";
```

- ดึงข้อมูลทั้งหมดจากตาราง Notifications

13.5 รันคำสั่ง SQL และรับข้อมูลแจ้งเตือน

```
const [rows] = await mysqlPool.promise().query(query);
```

- ใช้ mysqlPool.promise().query() เพื่อรอรับ async/await
- rows คือ รายการการแจ้งเตือนทั้งหมด ที่ได้จากฐานข้อมูล

13.6 ส่งข้อมูลกลับไปยังclient เอนเดอร์

```
return res.status(200).json(rows);
```

- ส่ง JSON response ที่มีรายการแจ้งเตือนกลับไปพร้อม HTTP Status 200 (OK)

13.7 จัดการข้อผิดพลาด (Error Handling)

```
} catch (err) {
  console.error("🚨 ERROR:", err.message, err.stack);
  return res.status(500).json({ message: `ล้มเหลว: ${err.message}` });
}
```

- ถ้าเกิดข้อผิดพลาด (เช่น ฐานข้อมูลล่ม)
 - แสดง error ใน console
 - ส่ง response 500 (Internal Server Error) พร้อมข้อความ ล้มเหลว

14.getstaff

14.1 นำเข้าโมดูลที่จำเป็น

```
import { NextResponse } from "next/server";
import { mysqlPool } from "@/utils/db"; // เชื่อมต่อฐานข้อมูล MySQL
```

- `NextResponse` → ใช้สำหรับสร้างและส่ง JSON response กลับไปยังไคลเอนต์
- `mysqlPool` → ใช้สำหรับเชื่อมต่อกับฐานข้อมูล MySQL

14.2 สร้างฟังก์ชัน GET() สำหรับดึงข้อมูลพนักงาน

```
export async function GET() {
```

- ฟังก์ชันนี้เป็น API Handler ที่ทำงานเมื่อมี HTTP GET request

14.3 เขียนคำสั่ง SQL Query

```
const query = `SELECT staffID, firstName, lastName, phoneNumber
FROM Staff`;
```

- คำสั่ง SQL ใช้ SELECT เพื่อดึงข้อมูลจากตาราง Staff
- ดึงเฉพาะข้อมูลที่จำเป็น ได้แก่
 - `staffID` → รหัสพนักงาน
 - `firstName` → ชื่อ
 - `lastName` → นามสกุล
 - `phoneNumber` → เบอร์โทรศัพท์

14.4 รันคำสั่ง SQL และดึงข้อมูลพนักงาน

```
const [rows] = await mysqlPool.promise().query(query);
```

- ใช้ `mysqlPool.promise().query()` เพื่อรอรับ async/await
- `rows` คือ รายการพนักงานทั้งหมด ที่ได้จากฐานข้อมูล

14.5 แสดงข้อมูลที่ดึงมา (Debugging)

```
console.log("🔔 Data from DB: ", rows); // Debug
```

- แสดงข้อมูลพนักงานที่ดึงมาจากฐานข้อมูลใน console

- ใช้สำหรับ Debugging

14.6 ส่งข้อมูลกลับไปยังclient เอนเตอร์

```
return NextResponse.json(rows, { status: 200 });
```

- ส่ง JSON response ที่มีรายการพนักงานกลับไปพร้อม HTTP Status 200 (OK)

14.7 จัดการข้อผิดพลาด (Error Handling)

```
} catch (err) {
  console.error("🚨 ERROR: ", err.message, err.stack);
  return NextResponse.json({ message: "✗ ล้มเหลว: ${err.message}" },
    { status: 500 });
}
```

- ถ้าเกิดข้อผิดพลาด (เช่น ฐานข้อมูลล่ม)
 - แสดง error ใน console
 - ส่ง response 500 (Internal Server Error) พร้อมข้อความ ✗ ล้มเหลว

15.getToken

15.1 นำเข้าโมดูลที่จำเป็น

```
import { NextRequest, NextResponse } from "next/server";
import jwt, { JwtPayload } from "jsonwebtoken";
import { config } from "@/config";
```

- NextRequest → ใช้สำหรับจัดการ request ใน Next.js API
- NextResponse → ใช้สำหรับสร้าง response ใน Next.js API
- jsonwebtoken (jwt) → ใช้สำหรับตรวจสอบ (verify) และถอดรหัส (decode) JWT Token
- JwtPayload → ใช้เพื่อบรรบุประเภทของข้อมูลที่อยู่ใน Token

- config → ใช้สำหรับเข้าถึงค่าคอนฟิก เช่น JWT_SECRET ที่ใช้สำหรับตรวจสอบ Token

15.2 พังก์ชัน getToken(req)

```
export function getToken(req: NextRequest) {
```

- พังก์ชันนี้รับ request (req) เป็นพารามิเตอร์
- ใช้สำหรับ ดึง JWT Token จาก Header และตรวจสอบความถูกต้อง

15.3 ดึง JWT Token จาก Header

```
const token = req.headers.get("authorization")?.split(" ")[1];
```

- อ่านค่า Authorization จาก Headers
- split(" ") → แยกค่าออกเป็น 2 ส่วน เช่น "Bearer <token>" และเลือก <token>

15.4 ตรวจสอบว่า Token มีค่าหรือไม่

```
if (!token) {
  return { error: NextResponse.json({ error: "Unauthorized" }), status: 401 };
}
```

- ถ้าไม่มี Token → ส่ง response 401 Unauthorized

15.5 ตรวจสอบความถูกต้องของ Token

```
try {
  const decoded = jwt.verify(token, config.JWT_SECRET) as JwtPayload;
  return { decoded };
```

- ใช้ jwt.verify(token, secretKey) เพื่อตรวจสอบว่ามีการเปลี่ยนแปลงค่า Token หรือไม่
- ถ้าผ่านการตรวจสอบ → คืนค่า decoded payload ของ Token

15.6 กรณี Token ไม่ถูกต้อง

```
| } catch (error) {  
| | return { error: NextResponse.json({ error: "Invalid token" }),  
| | status: 403 } };  
| }
```

- ถ้า Token หมดอายุ หรือ ไม่ถูกต้อง → ส่ง response 403 Forbidden

16.logout

16.1 นำเข้า NextResponse

```
import { NextResponse } from "next/server";
```

- NextResponse → ใช้สำหรับสร้าง HTTP Response ใน API ของ Next.js

16.2 สร้างฟังก์ชัน POST() สำหรับ Logout

```
export async function POST() {  
  try {
```

- ฟังก์ชันนี้ใช้ HTTP POST Method → สำหรับ ออกจากระบบ (Logout)
- ไม่มีพารามิเตอร์ เพราะไม่ต้องรับข้อมูลจากผู้ใช้

16.3 แสดงข้อความใน Console เพื่อ Debug

```
  console.log("🔴 API logout hit");
```

- แสดงข้อความ "API logout hit" → เพื่อให้รู้ว่า API ถูกเรียกใช้งาน

16.4 สร้าง Response สำหรับ Logout

```
response.headers.set("Set-Cookie", "token=; HttpOnly; Path=/; Max-Age=0");
```

- ส่ง JSON Response กลับไปที่ Client
- Status Code: 200 (OK)
- ข้อความแจ้งว่า "ออกจากระบบสำเร็จ"

16.5 ลบ Token ออกจาก Cookie

```
response.headers.set("Set-Cookie", "token=; HttpOnly; Path=/; Max-Age=0");
```

- ตั้งค่า Cookie "token" ให้เป็น ค่าว่าง ("")
- HttpOnly → ป้องกันการเข้าถึง Cookie จาก JavaScript
- Path=/ → ทำให้ Cookie ใช้ได้กับทุกหน้าเว็บ
- Max-Age=0 → ทำให้ Cookie หมดอายุทันที (ลบออก)

16.6 แสดงข้อความใน Console เมื่อ Token ถูกลบ

```
console.log("🚀 Token ถูกลบออกแล้ว!");
```

- แจ้งว่า Token ถูกลบเรียบร้อย

16.7 คืนค่า Response

```
return response;
```

- คืนค่า Response ไปที่ Client

16.8 จัดการข้อผิดพลาด

```
    } catch (error) {
      console.error("🚨 Error during logout:", error);
      return NextResponse.json({ message: "❌ ล้มเหลวในการออกจากระบบ" }, {
        status: 500
      });
    }
  }
```

- ถ้ามีข้อผิดพลาด → แสดงข้อความ "ล้มเหลวในการออกจากระบบ"
- ส่ง Status Code: 500 (Internal Server Error)

17.manageUsername

17.1 นำเข้าโมดูลที่จำเป็น

```
import { NextRequest, NextResponse } from "next/server";
const jwt = require("jsonwebtoken");
import { mysqlPool } from "@/utils/db";
import { config } from "@/config"; // ตั้ง secret key
const SECRET_KEY = config.JWT_SECRET;
import { cookies } from "next/headers";
```

- NextRequest และ NextResponse: ใช้สำหรับจัดการคำขอและการตอบกลับใน API ของ Next.js.
- jwt: ใช้สำหรับการตรวจสอบและยืนยัน JWT token.
- mysqlPool: ใช้เชื่อมต่อกับฐานข้อมูล MySQL.
- config: ใช้ตั้ง JWT_SECRET จากไฟล์การตั้งค่า (config) สำหรับการตรวจสอบ Token.

17.2 พังก์ชัน PUT สำหรับการอัปเดตโปรไฟล์

```
export async function PUT(req: NextRequest) {
```

- พังก์ชันนี้จะทำงานเมื่อมีการเรียก API ด้วย HTTP PUT method
ซึ่งจะใช้สำหรับการอัปเดตข้อมูลโปรไฟล์ของผู้ใช้.

17.3 รับข้อมูลจาก Header และ Body

```
const token = req.headers.get("authorization")?.split(" ")[1];
const { userName, phoneNumber, customerID } = await req.json();
```

- token: ดึง JWT token จาก header ที่มี Authorization: Bearer <token>.
- userName, phoneNumber, customerID: รับค่าจาก body ของคำขอ API ซึ่งจะใช้ในการอัปเดตข้อมูลโปรไฟล์.

17.4 ตรวจสอบว่า Token มีอยู่หรือไม่

```
if (!token) {
  console.error("⚠️ No token provided");
  return NextResponse.json({ error: "กรุณาเข้าสู่ระบบ" }, { status: 401 });
}
```

- ถ้าไม่มี Token จะตอบกลับ 401 Unauthorized พร้อมข้อความ "กรุณาเข้าสู่ระบบ".

17.5 ตรวจสอบว่าได้ส่งข้อมูลที่จำเป็นหรือไม่

```
if (!userName && !phoneNumber) {
  return NextResponse.json({ error: "โปรดใส่ username หรือเบอร์โทรศัพท์" }, { status: 400 });
}
```

- ถ้าไม่มีข้อมูล userName หรือ phoneNumber ที่จะอัปเดต จะตอบกลับ 400 Bad Request พร้อมข้อความ "โปรดใส่ username หรือเบอร์โทรศัพท์".

17.6 ตรวจสอบข้อมูลที่อัปเดตว่าไม่ซ้ำกันในฐานข้อมูล

```
let errors: string[] = [];
```

- สร้าง ตัวแปร errors เพื่อเก็บข้อผิดพลาดที่เกิดขึ้นระหว่างการตรวจสอบและอัปเดตข้อมูล.

17.6.1 ตรวจสอบ userName

```
// เช็ค userName ข้า ถ้ามีการส่งมา
if (userName) {
    const checkUserQuery = `SELECT CustomerID FROM Customer WHERE
    userName = ? AND CustomerID != ?`;
    const [existingUser]: any = await mysqlPool.promise().query(
        checkUserQuery, [userName, customerID]);

    if (existingUser.length > 0) {
        errors.push("มีคนใช้ชื่อนี้แล้ว");
    } else {
        // อัปเดต userName
        const updateUserQuery = `UPDATE Customer SET userName = ? WHERE
        CustomerID = ?`;
        await mysqlPool.promise().query(updateUserQuery, [userName,
        customerID]);
    }
}
```

- ถ้ามีการส่ง userName มาใหม่:
 - เช็คว่ามีผู้ใช้ที่ใช้ชื่อเดียวกันแล้วหรือไม่ในฐานข้อมูล.
 - ถ้ามีชื่อผู้ใช้ซ้ำ จะเก็บข้อผิดพลาดใน errors.
 - ถ้าไม่มีชื่อซ้ำ จะทำการ อัปเดต ชื่อผู้ใช้.

17.6.2 ตรวจสอบ phoneNumber

```
// เช็ค phoneNumber ช้า ถ้ามีการส่งมา
if (phoneNumber) {
    const checkphoneQuery = `SELECT CustomerID FROM Customer WHERE
    phoneNumber = ? AND CustomerID != ?`;
    const [existingPhone]: any = await mysqlPool.promise().query
    (checkphoneQuery, [phoneNumber, customerID]);

    if (existingPhone.length > 0) {
        errors.push("เบอร์นี้ในระบบแล้ว");
    } else if (!/^\d+$/.test(phoneNumber)) {
        errors.push("กรุณากรอกตัวเลข");
    } else {
        // อัปเดต phoneNumber
        const updatePhoneQuery = `UPDATE Customer SET phoneNumber = ?
        WHERE CustomerID = ?`;
        await mysqlPool.promise().query(updatePhoneQuery, [phoneNumber,
        customerID]);
    }
}
```

- ถ้ามีการส่ง phoneNumber มาใหม่:
 - เช็คว่ามีเบอร์โทรศัพท์ซ้ำกับผู้ใช้คนอื่นหรือไม่.
 - ถ้าเบอร์ช้า หรือไม่ใช่ตัวเลข จะเก็บข้อผิดพลาดใน errors.
 - ถ้าเบอร์ไม่ซ้ำและถูกต้อง จะทำการ อัปเดต เบอร์โทรศัพท์.

17.7 ส่งข้อมูลกลับเมื่อมีข้อผิดพลาด

```
// ถ้ามีข้อผิดพลาด ให้ส่งกลับทั้งหมด
if (errors.length > 0) {
    return NextResponse.json({ error: errors }, { status: 400 });
}
```

- ถ้าไม่มีข้อผิดพลาดจากการตรวจสอบ userName หรือ phoneNumber จะส่ง 400 Bad Request พร้อมกับข้อผิดพลาดที่เกิดขึ้น.

17.8 ส่งข้อความสำเร็จเมื่ออัปเดตข้อมูล

```
return NextResponse.json({ message: "อัพเดตโปรไฟล์สำเร็จ ✅" });
```

- ถ้าไม่มีข้อผิดพลาด จะตอบกลับ 200 OK พร้อมข้อความ "อัพเดตโปรไฟล์สำเร็จ".

17.9 จัดการข้อผิดพลาด

```
} catch (error) {
  console.error(error);
  return NextResponse.json({ error: "Internal Server Error" }, {
    status: 500 });
}
```

- ถ้ามีข้อผิดพลาดในระหว่างการทำงานของ API เช่นการเชื่อมต่อฐานข้อมูล หรือข้อผิดพลาดอื่น ๆ จะส่ง 500 Internal Server Error พร้อมข้อความ "Internal Server Error".

18.NTadd

18.1 นำเข้าโมดูลที่จำเป็น

```
import { NextRequest, NextResponse } from "next/server";
import { mysqlPool } from "@/utils/db";
import { getToken } from "@/app/api/getToken/route";
```

- NextRequest และ NextResponse: ใช้ในการจัดการคำขอและการตอบกลับใน Next.js API
- mysqlPool: ใช้เชื่อมต่อกับฐานข้อมูล MySQL และทำการ query.
- getToken: พังค์ชันที่ใช้ในการดึงและตรวจสอบ JWT token ที่ถูกส่งมาจากคำขอ (request).

18.2 พังค์ชัน POST สำหรับการเพิ่ม Notification

```
export async function POST(req: NextRequest) {
```

- พังค์ชันนี้จะทำงานเมื่อมีการส่งคำขอ POST ไปยัง API ซึ่งทำหน้าที่ในการเพิ่ม Notification ลงในฐานข้อมูล.

18.3 ดึง CustomerID จาก Token

```
const { decoded, error } = getToken(req);
if (error) return error;
```

- getToken: พังค์ชันนี้ใช้ในการดึง JWT token จาก header ของคำขอและทำการตรวจสอบความถูกต้องของ token.
 - decoded: ข้อมูลที่ถูก decode จาก token เช่น CustomerID ที่ใช้ในการระบุผู้ใช้.
 - error: หากไม่สามารถตรวจสอบหรือ decode token ได้ จะมีการส่ง ข้อผิดพลาด กลับ.

18.4 ตรวจสอบว่า Token ถูกต้องหรือไม่

```
const customerID = decoded?.CustomerID;
if (!customerID) return NextResponse.json({ error: "Invalid token" },
{ status: 403 });
```

- ถ้า CustomerID ไม่ถูกต้อง (หรือไม่มีใน decoded), จะส่ง 403 Forbidden กลับไปพร้อมกับข้อความ "Invalid token".

18.5 รับข้อมูลจาก Request Body

```
// ดึงข้อมูลจาก request body
const { message, recipient } = await req.json();
```

- รับข้อมูลจาก request body โดยที่ต้องมีการส่งค่า message และ recipient มาในคำขอ ซึ่งใช้สำหรับเนื้อหาของการแจ้งเตือนและผู้รับ.

18.6 ตรวจสอบว่ามีข้อมูลครบถ้วนหรือไม่

```
if (!message || !recipient) {  
    return NextResponse.json({ error: "Missing required fields" }, {  
        status: 400  
    })  
}
```

- ถ้าไม่มีข้อมูล message หรือ recipient ในคำขอ จะส่ง 400 Bad Request กลับไปพร้อมข้อความ "Missing required fields".

18.7 จัดการวันที่และเวลา

```
// วันที่ปัจจุบัน  
const sendDate = new Date().toISOString().slice(0, 19).replace("T", "  
");
```

- ดึง วันที่และเวลา ปัจจุบันในรูปแบบ ISO (YYYY-MM-DD HH:MM:SS) โดยใช้ toISOString และตัดเวลา T ออก เพื่อให้เหมาะสมกับการจัดเก็บในฐานข้อมูล.

18.8 สร้าง SQL Query สำหรับเพิ่ม Notification

```
// สร้าง Query สำหรับเพิ่ม Notification  
const insertQuery = `  
    INSERT INTO Notifications (sendDate, message, recipient,  
    CustomerID)  
    VALUES (?, ?, ?, ?)  
`;
```

- สร้าง SQL Query เพื่อเพิ่มการแจ้งเตือน (Notification) ลงในตาราง Notifications:
 - sendDate: วันที่และเวลาที่ส่ง.
 - message: ข้อความการแจ้งเตือน.
 - recipient: ผู้รับการแจ้งเตือน.
 - CustomerID: ไอดีของผู้ใช้ที่ส่งการแจ้งเตือน.

18.9 Execute Query เพื่อเพิ่มข้อมูล

```
// Execute Query
await mysqlPool.query(insertQuery, [sendDate, message, recipient,
customerID]);
```

- ใช้ mysqlPool.query ในการรัน SQL Query เพื่อเพิ่มข้อมูลการแจ้งเตือนลงในฐานข้อมูล.

18.10 ส่งคำตอบกลับเมื่อสำเร็จ

```
return NextResponse.json({ success: true, message: "Notification
added successfully" }, { status: 201 });
```

- ถ้าการเพิ่มข้อมูลเสร็จสมบูรณ์ จะส่ง 201 Created พร้อมข้อความ "Notification added successfully" กลับไปที่ client.

18.11 จัดการข้อผิดพลาด

```
} catch (error) {
  console.error("Error adding notification:", error);
  return NextResponse.json({ error: "Internal Server Error" }, {
    status: 500 });
}
```

- ถ้ามีข้อผิดพลาดในระหว่างการทำงานของฟังก์ชัน เช่น การเชื่อมต่อฐานข้อมูลผิดพลาด หรือข้อมูลที่ได้รับไม่ถูกต้อง จะส่ง 500 Internal Server Error พร้อมข้อความ "Internal Server Error".

19.NTget

19.1 นำเข้าโมดูลที่จำเป็น

```
import { NextRequest, NextResponse } from "next/server";
import { mysqlPool } from "@/utils/db";
import { getToken } from "@/app/api/getToken/route"; //  
ใช้เพื่อดึงข้อมูล token
```

- NextRequest และ NextResponse: ใช้ในการจัดการคำขอและการตอบกลับใน Next.js API.
- mysqlPool: เชื่อมต่อ กับฐานข้อมูล MySQL เพื่อรันคำสั่ง SQL.
- getToken: พังค์ชันนี้ใช้ในการดึงและตรวจสอบ JWT token ที่ถูกส่งมาจากการคำขอ (request).

19.2 พังค์ชัน GET สำหรับดึงข้อมูลการแจ้งเตือน

```
export async function GET(req: NextRequest) {
```

- พังค์ชันนี้จะทำงานเมื่อมีการส่งคำขอ GET ไปยัง API
ซึ่งทำหน้าที่ดึงข้อมูลการแจ้งเตือนจากฐานข้อมูล.

19.3 ดึงข้อมูลจาก Token

```
// ดึงข้อมูลจาก token
const { decoded, error } = getToken(req);
if (error) return error;
```

- ใช้ getToken ในการดึงข้อมูลจาก JWT token ที่ถูกส่งมาใน request header.
 - decoded: ข้อมูลที่ได้รับจากการ decode token ซึ่งจะมีข้อมูลเช่น role และ CustomerID.
 - error: หากไม่สามารถดึงหรือ decode token ได้ จะมีข้อผิดพลาดและส่งกลับ.

19.4 ตรวจสอบข้อมูลบทบาท (Role) และ CustomerID

```
const role = decoded?.role; // รับข้อมูล role (admin หรือ user)
const customerID = decoded?.CustomerID; // รับข้อมูล CustomerID จาก token
```

- role: รับข้อมูลบทบาทของผู้ใช้ (เช่น admin หรือ user).
- customerID: รับข้อมูล CustomerID ของผู้ใช้จาก token เพื่อใช้ในการกรองข้อมูล.

19.5 กำหนดเงื่อนไขการกรองข้อมูลตามบทบาท

```
// กำหนดเงื่อนไขการกรองข้อมูลตามบทบาท
let whereClause = "";
if (role === "admin") {
    // แคดมินสามารถดูข้อมูลทั้งหมด
    whereClause = "";
} else if (role === "user") {
    // ลูกค้าดูข้อมูลของตัวเอง
    whereClause = `WHERE n.CustomerID = ${customerID}`;
} else {
    return NextResponse.json({ error: "Unauthorized" }, { status: 403 });
}
```

- admin: ถ้าผู้ใช้มีบทบาทเป็น admin จะไม่มีกำหนดเงื่อนไข (สามารถดูข้อมูลทั้งหมด).
- user: ถ้าผู้ใช้มีบทบาทเป็น user จะสามารถดูข้อมูลเฉพาะของตัวเองเท่านั้นโดยใช้ CustomerID ของผู้ใช้.
- ถ้าบทบาทไม่ตรงกับ admin หรือ user จะส่ง 403 Unauthorized กลับไป.

19.6 สร้าง SQL Query สำหรับดึงข้อมูล

```
// สร้างคำสั่ง SQL สำหรับดึงข้อมูล Notification, Payment, และ Rental
const selectQuery = `
SELECT
    n.notificationID,
    n.message,
    n.sendDate,
    n.recipient,
    n.CustomerID,
    p.paymentID,
    p.amount,
    p.receipt,
    r.rentalID,
    r.rentalStatus,
    r.rentalPrice,
    r.startDate,
    r.endDate,
    r.Location
FROM Notifications n
LEFT JOIN Payment p ON n.paymentID = p.paymentID
LEFT JOIN Rental r ON r.rentalID = p.rentalID
${whereClause} -- เงื่อนไขกรองตาม role หรือ CustomerID
ORDER BY n.sendDate DESC -- เรียงตามวันที่ส่ง Notification`
```

- SQL Query นี้ใช้ JOIN 3 ตาราง:
 - Notifications (การแจ้งเตือน)
 - Payment (การชำระเงิน)

- Rental (การเช่ารายการ)
- ข้อมูลที่ดึงออกมามาคือ:
 - n.notificationID, n.message, n.sendDate, n.recipient, n.CustomerID: ข้อมูลการแจ้งเตือน.
 - p.paymentID, p.amount, p.receipt: ข้อมูลการชำระเงิน.
 - r.rentalID, r.rentalStatus, r.rentalPrice, r.startDate, r.endDate, r.Location: ข้อมูลการเช่า.
- whereClause: เงื่อนไขกรองข้อมูลที่ได้จากบทบาท (role).
- ORDER BY n.sendDate DESC: เรียงข้อมูลการแจ้งเตือนตามวันที่ส่ง (ล่าสุดไปก่อนสุด).

19.7 ดึงข้อมูลจากฐานข้อมูล

```
// ดึงข้อมูลจากฐานข้อมูล
const [notifications] = await mysqlPool.promise().query(selectQuery);
```

- รันคำสั่ง SQL และดึงข้อมูลการแจ้งเตือนจากฐานข้อมูล.

19.8 ส่งผลลัพธ์กลับไปยัง Client

```
// ส่งผลลัพธ์กลับไปยัง Client
return NextResponse.json({ success: true, data: notifications }, {
  status: 200 });
```

- ส่ง 200 OK กลับไปยัง client พร้อมกับข้อมูล notifications ที่ดึงมา.

19.9 จัดการข้อผิดพลาด

```
} catch (error) {
  // ถ้ามีข้อผิดพลาดเกิดขึ้น
  console.error("Error fetching notifications:", error);
  return NextResponse.json({ error: "Internal Server Error" }, {
    status: 500 });
}
```

- ถ้ามีข้อผิดพลาดเกิดขึ้น (เช่น การเชื่อมต่อฐานข้อมูลผิดพลาด), จะส่ง 500 Internal Server Error กลับไป.

20.RTdetailadd

20.1 นำเข้าโมดูลที่จำเป็น

```
import { NextResponse } from "next/server";
import { mysqlPool } from "@/utils/db";
```

- NextResponse: ใช้ในการตอบกลับข้อมูลไปยัง client ในรูปแบบ JSON พร้อมกับสถานะ (status) ที่กำหนด.
- mysqlPool: ใช้ในการเชื่อมต่อกับฐานข้อมูล MySQL (ผ่าน Pool เพื่อจัดการการเชื่อมต่อหลายครั้งได้ดีขึ้น).

20.2 พัฒนา POST สำหรับเพิ่มข้อมูลการเช่า

```
export async function POST(request) {
```

- พัฒนานี้จะทำงานเมื่อมีการส่งคำขอ POST ไปยัง API เพื่อเพิ่มข้อมูลการเช่าใหม่ในฐานข้อมูล.

20.3 ดึงข้อมูลจาก Request Body

```
const {
  rentalStatus,
  rentalPrice,
  startDate,
  endDate,
  Location,
  Days,
  CarID,
  staffID
} = await request.json();
```

- ส่วนมาดังนี้จะประกอบไปด้วย:
 - rentalStatus: สถานะการเช่า

- rentalPrice: ราคากำไรเช่า
- startDate: วันที่เริ่มเช่า
- endDate: วันที่สิ้นสุดการเช่า
- Location: สถานที่ที่รถเช่า
- Days: จำนวนวันที่เช่า (ถ้าไม่ได้คำนวณจากวันที่เริ่ม-สิ้นสุด)
- CarID: รหัสรถที่เช่า
- staffID: รหัสพนักงานที่ทำการเช่า

20.4 ตรวจสอบข้อมูลที่จำเป็น

```
// Validate required fields
if (!rentalStatus || !rentalPrice || !startDate || !endDate ||
!Location || !Days || !CarID || !staffID) {
  return NextResponse.json(
    { error: "กรุณากรอกข้อมูลให้ครบถ้วน" },
    { status: 400 }
  );
}
```

- ตรวจสอบว่า ข้อมูลทั้งหมดที่จำเป็น ได้ถูกส่งมาในคำขอหรือไม่ ถ้าไม่ครบถ้วนจะตอบกลับด้วย 400 Bad Request พร้อมข้อความผิดพลาดว่า "กรุณากรอกข้อมูลให้ครบถ้วน".

20.5 คำนวณจำนวนวันในการเช่า (ถ้ายังไม่ระบุในคำขอ)

```
// Calculate days if needed
const start = new Date(startDate);
const end = new Date(endDate);
const calculatedDays = Math.ceil((end - start) / (1000 * 60 * 60
* 24));
```

- แปลงวันที่เริ่มต้น (startDate) และวันที่สิ้นสุด (endDate) ให้เป็น Date object.
- คำนวณจำนวนวันในการเช่าโดยการลบวันที่สิ้นสุดจากวันที่เริ่มต้น และแปลงเป็นจำนวนวัน (calculatedDays).

20.6 เชื่อมตอกับฐานข้อมูล

```
const connection = await mysqlPool.getConnection();
```

- เชื่อมต่อกับฐานข้อมูล MySQL ผ่าน connection pool เพื่อให้การเชื่อมต่อมีประสิทธิภาพมากขึ้น.

20.7 แทรกรหัสการเช่าในฐานข้อมูล

```
try {
    const [result] = await connection.query(
        `INSERT INTO Rental
        (rentalStatus, rentalPrice, startDate, endDate, Location, Days,
        CarID, staffID)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)`,
        [rentalStatus, rentalPrice, startDate, endDate, Location,
        calculatedDays || Days, CarID, staffID]
    );
}
```

- ใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลการเช่าในตาราง Rental.
- ข้อมูลที่แทรกรวม:
 - rentalStatus: สถานะการเช่า
 - rentalPrice: ราคาการเช่า
 - startDate: วันที่เริ่มต้น
 - endDate: วันที่สิ้นสุด
 - Location: สถานที่เช่า
 - Days: จำนวนวันในการเช่า (คำนวนหรือจากคำขอ)
 - CarID: รหัสรถที่เช่า
 - staffID: รหัสพนักงาน
- ?: ใช้ในการป้องกัน SQL injection โดยการแทนค่าจากตัวแปรที่มาจากการขอ.

20.8 ตอบกลับข้อมูลสำเร็จ

```
return NextResponse.json({
    message: "เพิ่มข้อมูลการเช่าสำเร็จ",
    rentalID: result.insertId
}, { status: 201 });
```

- ถ้าการแทรกรหัสการเช่าสำเร็จ จะตอบกลับด้วย 201 Created และข้อมูลที่สำคัญ:
 - message: ข้อความบอกว่า "เพิ่มข้อมูลการเช่าสำเร็จ".

- rentalID: รหัสการเช่าที่ถูกสร้างใหม่ (จาก result.insertId).

20.9 ปิดการเชื่อมต่อฐานข้อมูล

```

    } finally {
      connection.release();
    }
  }
}

```

- ปล่อยการเชื่อมต่อฐานข้อมูลเมื่อทำงานเสร็จ เพื่อให้ connection สามารถนำกลับมาใช้ใหม่ได้.

20.10 จัดการข้อผิดพลาด

```

} catch (error) {
  console.error("[เกิดข้อผิดพลาดในการเพิ่มข้อมูลการเช่า]", error);
  return NextResponse.json(
    { error: "[เกิดข้อผิดพลาดในการเพิ่มข้อมูลการเช่า]" },
    { status: 500 }
  );
}

```

- ถ้ามีข้อผิดพลาดเกิดขึ้นในกระบวนการใดๆ (เช่น การเชื่อมต่อฐานข้อมูลหรือการแทรกข้อมูล) จะส่งกลับ 500 Internal Server Error พร้อมข้อความแสดงข้อผิดพลาด.

21. SearchCars

21.1 นำเข้าโมดูลที่จำเป็น

```

import { NextRequest, NextResponse } from "next/server";
import { mysqlPool } from "@/utils/db";

```

- NextRequest และ NextResponse: ใช้สำหรับรับคำขอและตอบกลับคำขอ HTTP ใน Next.js
- mysqlPool: ใช้ในการเชื่อมตอกับฐานข้อมูล MySQL เพื่อให้สามารถดำเนินการกับฐานข้อมูลได้

21.2 กำหนด Interface สำหรับ Car

```
interface Car {  
    carID: number;  
    LPplate: string;  
    model: string;  
    brand: string;  
    carType: string;  
    rentalPrice: number;  
    status: string;  
    CarImg: string;  
}
```

- Car: กำหนดโครงสร้างข้อมูลสำหรับรถที่มีในฐานข้อมูล โดยประกอบด้วย:
 - carID: รหัสของรถ
 - LPplate: ป้ายทะเบียนรถ
 - model: รุ่นของรถ
 - brand: ยี่ห้อของรถ
 - carType: ประเภทของรถ
 - rentalPrice: ราคาค่าเช่าต่อวัน
 - status: สถานะการเช่าของรถ (เช่น "available" หรือ "rented")
 - CarImg: ลิงค์หรือ URL ของภาพรถ

21.3 พังก์ชัน POST สำหรับค้นหาข้อมูลรถ

```
export async function POST(req: NextRequest) {  
    const data = req  
    console.log("API Received Body:", data);  
}
```

- พังก์ชันนี้จะทำงานเมื่อมีการส่งคำขอ POST ไปยัง API เพื่อค้นหาข้อมูลรถที่พร้อมให้เช่า

21.4 ดึงข้อมูลจาก Request Body

```
const requestBody = await req.json();  
console.log("API Received Body:", requestBody);
```

- ดึงข้อมูลจาก request body โดยใช้ req.json() ซึ่งข้อมูลจะถูกส่งมาในรูปแบบ JSON
- ข้อมูลที่ดึงออกมาแล้วจะแสดงใน console เพื่อการตรวจสอบ

21.5 ดึงค่าเงื่อนไขการค้นหาจาก Request Body

```
const { minPrice, maxPrice, carType, pickupLocation, returnLocation, rentalDays } = requestBody;
```

- ดึงข้อมูลการค้นหาจาก request body ได้แก่:
 - minPrice: ราคาต่ำสุดของรถที่ต้องการค้นหา
 - maxPrice: ราคากลางสุดของรถ
 - carType: ประเภทของรถที่ต้องการค้นหา
 - pickupLocation: สถานที่รับรถ
 - returnLocation: สถานที่คืนรถ
 - rentalDays: จำนวนวันที่เช่า

21.6 เตรียม SQL Query สำหรับการค้นหารถ

```
let query = "SELECT carID, LPlate, model, brand, carType, rentalPrice, status, CarImg FROM Car WHERE status = 'available'";
```

- เริ่มต้นด้วยการเขียนคำสั่ง SQL ที่เลือกข้อมูลจากตาราง Car โดยค้นหารถที่มีสถานะเป็น "available" (รถที่พร้อมให้เช่า).

21.7 การเพิ่มเงื่อนไขการค้นหา (กรองข้อมูล)

```
if (typeof minPrice === "number" && typeof maxPrice === "number") {
  query += " AND rentalPrice BETWEEN ? AND ?";
  queryParams.push(minPrice, maxPrice);
}
```

- ถ้า minPrice และ maxPrice เป็นตัวเลข จะเพิ่มเงื่อนไขใน SQL เพื่อกรองรถที่มีราคาอยู่ในช่วงที่กำหนด.

```
if (typeof carType === "string") {
  query += " AND carType = ?";
  queryParams.push(carType);
}
```

- ถ้ามีการระบุ carType เป็นประเภทของรถที่ต้องการ คำสั่ง SQL จะเพิ่มเงื่อนไขกรองรถตามประเภทที่ระบุ.

```

if (pickupLocation) {
  query += " AND pickupLocation = ?";
  queryParams.push(pickupLocation);
}

if (returnLocation) {
  query += " AND returnLocation = ?";
  queryParams.push(returnLocation);
}

if (rentalDays && !isNaN(rentalDays)) {
  query += " AND rentalDays >= ?";
  queryParams.push(rentalDays);
}

```

- หาก pickupLocation, returnLocation, หรือ rentalDays ถูกระบุมาในคำขอ จะเพิ่มเงื่อนไขการกรองตามสถานที่รับรถ, สถานที่คืนรถ, หรือจำนวนวันที่เช่าตามลำดับ.

21.8 แสดงคำสั่ง SQL และพารามิเตอร์

```
console.log("Final SQL Query:", query, queryParams);
```

- แสดงคำสั่ง SQL ที่ได้สร้างขึ้นพร้อมกับค่าพารามิเตอร์ที่ใช้ในการกรองข้อมูล เพื่อการตรวจสอบว่า query ที่ส่งไปยังฐานข้อมูลถูกต้องหรือไม่.

21.9 การดึงข้อมูลจากฐานข้อมูล

```
const [rows] = await mysqlPool.promise().query(query, queryParams) as [any[], any];
```

- ส่งคำสั่ง SQL ที่สร้างขึ้นไปยังฐานข้อมูล MySQL โดยใช้ mysqlPool.promise().query() เพื่อดึงข้อมูลที่ตรงกับเงื่อนไขที่ตั้งไว้.
- rows จะเก็บผลลัพธ์ของ query ที่ดึงมาในรูปแบบอาร์เรย์.

21.10 ตอบกลับข้อมูลผลลัพธ์

```
return NextResponse.json(rows, { status: 200 });
```

- ส่งผลลัพธ์ที่ได้จากฐานข้อมูลกลับไปยัง client ในรูปแบบ JSON พร้อมสถานะ 200 OK.

21.11 จัดการข้อผิดพลาด

```
} catch (error) {
  console.error("✖ API Error:", error);
  return NextResponse.json({ message: "ล้มเหลวในการดึงข้อมูลรถ" }, {
    status: 500
})
```

- ถ้ามีข้อผิดพลาดเกิดขึ้นในระหว่างการดำเนินการ (เช่น การเชื่อมตอกับฐานข้อมูลหรือการคิวรีข้อมูล) จะมีการจับข้อผิดพลาดและส่งกลับ 500 Internal Server Error พร้อมข้อความว่า "ล้มเหลวในการดึงข้อมูลรถ".

Backend Admin

1. User Management

1.1 ໂຫລດຂໍ້ມູນລູກຄ້າຈາກ API

```
useEffect(() => {
  const fetchCustomers = async () => {
    try {
      const res = await fetch("/api/getcustomer");
      const data = await res.json();
      setCustomers(data);
    } catch (error) {
      console.error("Error fetching customers:", error);
    }
  };
  fetchCustomers();
}, []);
```

- ເນື່ອ component ໂຫລດ ຈະດຶງຂໍ້ມູນລູກຄ້າທີ່ຮັບມາຈາກ API /api/getcustomer
- ເກີບໄວ້ໃນ state customers

1.2 ດັນຫາລູກຄ້າ

```
const filteredCustomers = (customers || []).filter(customer => {
  return customer.CustomerID?.toString().includes(searchTerm) ||
    customer.firstName?.includes(searchTerm) ||
    customer.lastName?.includes(searchTerm) ||
    customer.phoneNumber?.includes(searchTerm);
});
```

ກຽບສູ່ໃຊ້ໃຫຍ່ໃນສອງ search ເພື່ອ ຊື້ໆ ຂໍ້ອຳນວຍ

1.3 ໂບລູກຄ້າ

```
const handleDelete = async (customerID: number) => {
  try {
    // ສັງເກົນ DELETE ໄປ API
    const response = await fetch(`api/deletecustomer`, {
      method: "DELETE",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({ customerID }), // ລືມ customerID ໃນ JSON
    });

    // ຄວາມສອນການດອນກລົບວ່າເປົ້າ JSON
    const data = await response.json();

    if (response.ok) {
      alert(data.message);
      setCustomers((prevCustomers) => prevCustomers.filter((customer) => customer.CustomerID !== customerID));
    }
  } catch (error) {
    console.error("Error deleting customer:", error);
  }
};
```

- ส่ง DELETE ไปยัง /api/deletecustomer
- ลบลูกค้านั้นออกจาก state เพื่ออัปเดตหน้าจอทันที

1.4 ไปหน้าแก้ไขข้อมูลลูกค้า

```
// [RENDERED]
const goToEditCustomer = (customerId: string) => {
  router.push(`admin/customer-management/editcustomer?id=${customerId}`);
};

return (
  <div className="pt-20 min-h-screen p-6">
    {/* Header */}
    <div className="flex flex-col md:flex-row justify-between items-start md:items-center mb-6">
      <h1 className="text-2xl md:text-3xl font-bold text-gray-800 mb-4 md:mb-0">การจัดการลูกค้า</h1>
      <div className="flex flex-col md:flex-row gap-4 w-full md:w-auto">
        <Link href="/admin">
          <button className="bg-blue-600 hover:bg-blue-700 text-white px-4 py-2 rounded-lg transition-colors">กลับหน้าหลัก</button>
        </Link>
      </div>
    </div>
  </div>
```

ใช้ router.push เพื่อเปลี่ยนหน้าไปยังฟอร์มแก้ไขลูกค้า

2.CarManagement

2.1 โหลดข้อมูลรถทั้งหมด

```
useEffect(() => {
  const fetchCars = async () => {
    try {
      const res = await fetch("/api/getcar");
      const data = await res.json();
      setCars(data);
    } catch (error) {
      console.error("Error fetching cars:", error);
    }
  };
  fetchCars();
}, []);
```

- ใช้ useEffect ในการเรียก API /api/getcar ทุกครั้งที่มีการ渲染หน้าจอ

- เก็บไว้ใน state cars

2.2 ค้นหาและกรองสถานะ

```
// กรองรถตามสถานะและค้นหา
const filteredCars = cars.filter((car) => {
  const matchesSearch =
    car.LPlate.includes(searchTerm) ||
    car.model.includes(searchTerm) ||
    car.brand.includes(searchTerm);
  const matchesTab = activeTab === "all" || car.status === activeTab;
  return matchesSearch && matchesTab;
});
```

- return (
- ค้นหาจาก ทะเบียน, รุ่น, ยี่ห้อ
 - กรองตามสถานะ: ทั้งหมด / พร้อมให้เช่า / กำลังเช่า / ซ่อมบำรุง

2.3 ลบข้อมูลรถ

```
const handleDelete = async (LPlate: string) => {
  try {
    // เรียก API [เพื่อลบรถคัน] LPlate
    const response = await fetch(`^/api/deletecar`, {
      method: "DELETE",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({ LPlate }), // ส่ง LPlate ของรถไปใน request body
    });
  }
};
```

- เรียก API ลบรถตามเลขทะเบียน (LPlate)
- อัปเดต state cars ทันทีหลังลบ

2.4 ไปหน้าแก้ไขรถ

```
<button
  onClick={() => router.push(`^/admin/car-management/edit?LPlate=${car.LPlate}`)}
```

พาผู้ใช้ไปหน้าแก้ไขข้อมูลรถ พร้อมแนบทะเบียนรถไปใน query

2.5 เพิ่มรถใหม่

```
</Link>
<Link href="/admin/car-management/add">
```

ปุ่ม “เพิ่มข้อมูลรถใหม่” → ไปหน้าฟอร์มเพิ่มรถ

2.add Cars (component cars management)

2.1 เก็บข้อมูลรถจากแบบฟอร์ม

```
const [LPlate, setLPlate] = useState("");
const [model, setModel] = useState("");
const [brand, setBrand] = useState("");
const [carType, setCarType] = useState("Sedan");
const [status, setStatus] = useState("พร้อมให้เช่า");
const [rentalPrice, setRentalPrice] = useState("") // แปลงเป็น empty string
```

ใช้ useState เก็บค่าทะเบียน, รุ่น, ยี่ห้อ, ประเภท, สถานะ, และราคาเช่าของรถ

2.2 ส่งข้อมูลไปยัง API /api/addcar

```
const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();

  const car = {
    LPlate,
    model,
    brand,
    carType,
    status,
    rentalPrice: Number(rentalPrice), // แปลงเป็น number ที่นี่
  };

  try {
    const res = await fetch("/api/addcar", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(car),
    });
  }
};
```

- เมื่อกด “บันทึก” → จะส่งข้อมูลที่กรอกไปยัง API
- ถ้า res.ok สำเร็จ → redirect ไปหน้า /admin/car-management

2.3 ลิงก์ย้อนกลับ / ยกเลิก

```
<Link href="/admin/car-management">
  <button className="px-6 py-2 bg-blue-600 text-white rounded-lg shadow-md hover:bg-blue-700">ย้อนกลับ</button>
</Link>
```

ถ้า res.ok สำเร็จ → redirect ไปหน้า /admin/car-management

ปุ่ม “บันทึก” และ “ยกเลิก” พาผู้ใช้กลับไปยังหน้ารายการรถ

2.4 Dropdown + Validation

- carType และ status ใช้ <select> เพื่อจำกัดค่าที่เลือกได้
- rentalPrice ใช้ <input type="number" /> และแปลงเป็น Number() ก่อนส่ง

3.Staff management

3.1 โหลดข้อมูลสถิติ (useEffect)

```
useEffect(() => {
  const fetchData = async () => {
    try {
      const res1 = await fetch("/api/user-management");
      const res2 = await fetch("/api/car-management");
      const res3 = await fetch("/api/staff-management");
      const res4 = await fetch("/api/stats");
      const res5 = await fetch("/api/visits");
      const res6 = await fetch("/api/bookings");
```

- โหลดข้อมูลจากหลาย API เช่น: ผู้ใช้, รถ, บุคลากร, การจอง, สถิติ
- เก็บข้อมูลไว้ใน state ต่าง ๆ

3.2 ปุ่มออกจากระบบ

```
const handleLogout = () => {
  // ลบ token ออกจาก sessionStorage
  sessionStorage.removeItem("adminToken");
  // กลับไปหน้าแรก
  router.push("/");
```

- ลบ token ออกจาก sessionStorage
- กลับไปหน้าแรกหลัง logout

3.3 กล่องแสดงจำนวนรถ / ผู้ใช้ (mock)

```
<p>{carCount}</p>
<button onClick={() => setCarCount(carCount + 1)}>เพิ่มรถ</button>
```

- ยังเป็น mock button (เพิ่ม/ลดตัวเลขด้วยปุ่มกด)
- ใช้ useState แสดงจำนวน รถ และ ผู้ใช้ แบบจำลอง

3.4 ค้นหาสถิติตามช่วงเวลา

```
const handleSearch = async () => {
  const response = await fetch(`api/stats?filter=${filter}&start=${startDate}&end=${endDate}`);
  const data = await response.json();
  setStats(data);
};
```

ยังไม่ได้แสดง input UI แต่ function เตรียมพร้อมสำหรับค้นหาข้อมูลรายงานตามช่วงเวลา

3.5 Card ลิงก์ไปหน้าจัดการต่าง ๆ

```
/* Dashboard Cards */


<Card href="/admin/user-management" icon={FaUsers} title="User Management" bgColor="bg-yellow-500 hover:bg-yellow-600" />
  <Card href="/admin/car-management" icon={FaCar} title="Car Management" bgColor="bg-green-500 hover:bg-green-600" />
  <Card href="/admin/staff-management" icon={FaUsers} title="Staff Management" bgColor="bg-red-500 hover:bg-yellow-600" />
  <Card href="/admin/payment-verification" icon={FaMoneyBillWave} title="Payment Verification" bgColor="bg-purple-500 hover:bg-purple-600" />


```

ลิงก์ไปหน้าจัดการต่าง ๆ เช่น:

- ผู้ใช้
- รถ
- พนักงาน
- ตรวจสอบการชำระเงิน

4 AddUser (Component ของ UserManagement)

4.1 โหลดข้อมูลลูกค้าจาก API

```
useEffect(() => {
  const fetchCustomers = async () => {
    try {
      const res = await fetch("api/getcustomer");
      const data = await res.json();
      setCustomers(data);
    } catch (error) {
      console.error("Error fetching customers:", error);
    }
  };
  fetchCustomers();
}, []);
```

- ใช้ useEffect ดึงข้อมูลลูกค้าจาก backend ทันทีเมื่อ component โหลด
- เก็บข้อมูลไว้ใน customers state

4.2 ค้นหาลูกค้าตามชื่อ เบอร์ หรือรหัส

```
filteredCustomers.map((customer) =>
  <div key={customer.CustomerID} className="grid grid-cols-4 p-4 border-b hover:bg-gray-50">
    /* หัวกระทง */
    <div className="col-span-2 truncate">
      {customer.firstName} {customer.lastName}
    </div>

    /* หมายเลข */
    <div className="col-span-1 truncate">{customer.phoneNumber}</div>
    <div className="flex gap-2">

      <button
        className="text-blue-600 hover:text-blue-800 p-1"
        title="แก้ไข"
      >
        <FiEdit size={18} />
      </button>

      <button
        onClick={() => handleDelete(customer.CustomerID)}
        className="text-red-600 hover:text-red-800 p-1"
        title="ลบ"
      >
        <FiTrash2 size={18} />
      </button>
    </div>
  </div>
)
● ดูรายละเอียดเพิ่มเติมของ component, บล็อก, ฟังก์ชัน, ฯลฯ ที่ใช้
```

- แสดงเฉพาะข้อมูลที่ตรงกับสิ่งที่ผู้ใช้พิมพ์

4.3 ลบลูกค้าผ่าน API

```
const handleDelete = async (customerID: number) => {
  try {
    // ส่งคำขอ DELETE ไปยัง API
    const response = await fetch(`api/deletecustomer`, {
      method: "DELETE",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({ customerID }), // นำ customerID ไป JSON
    });

    // ตรวจสอบการตอบกลับมาเป็น JSON
    const data = await response.json();

    if (response.ok) {
      alert(data.message);
      setCustomers((prevCustomers) => prevCustomers.filter((customer) => customer.CustomerID !== customerID));
      return true;
    } else {
      alert(data.message || "ไม่สามารถลบข้อมูลลูกค้าได้");
      return false;
    }
  } catch (error) {
    console.error("Error deleting customer: ", error);
    alert("เกิดข้อผิดพลาดในการลบข้อมูล");
    return false;
  }
};
```

- ส่งคำขอ DELETE ไปยัง /api/deletecustomer

- ลบลูกค้าออกจากหน้าจอทันทีหลังลบสำเร็จ

4.4 ไปหน้าแก้ไขลูกค้า

```
// ไปหน้าแก้ไขลูกค้า
const goToEditCustomer = (customerId: string) => {
  router.push(`/admin/customer-management/editcustomer?id=${customerId}`);
};
```

- ลบลูกค้าออกจากหน้าจอทันทีหลังลบสำเร็จ

5.Staff management

5.1 โหลดข้อมูลพนักงานจาก API

```
useEffect(() => {
  const fetchStaffs = async () => {
    try {
      const res = await fetch("/api/getstaff");
      const data = await res.json();
      setStaffs(data);
    } catch (error) {
      console.error("Error fetching staffs:", error);
    }
  };
  fetchStaffs();
}, []);
```

- เมื่อ component โหลด → ดึงข้อมูลพนักงานทั้งหมดจาก API
- เก็บไว้ใน staffs state

5.2 ค้นหาพนักงานตามคำค้น

```
const filteredStaffs = (staffs || []).filter(staff => {
  return staff.staffID?.toString().includes(searchTerm) ||
    staff.firstName?.includes(searchTerm) ||
    staff.lastName?.includes(searchTerm) ||
    staff.phoneNumber?.includes(searchTerm);
});
```

รองรับการค้นหาด้วย: รหัสพนักงาน, ชื่อ, เบอร์โทร

5.3 ลบพนักงานผ่าน API

```
const handleDelete = async (staffID: number) => {
  try {
    const response = await fetch(`api/deletestaff`, {
      method: "DELETE",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({staffID}),
    });
    const data = await response.json();
    if (response.ok) {
      alert(data.message);
      setStaffs((prevStaffs) => prevStaffs.filter((staff) => staff.staffID !== staffID));
      return true;
    } else {
      alert(data.message);
      return false;
    }
  } catch (error) {
    console.error("⚠️ Error deleting car: ", error);
    alert("เกิดข้อผิดพลาดในการลบข้อมูล");
    return false;
  }
};
```

▪ หมายเหตุ: เหตุการณ์

- อัปเดต state ให้ลับออกจาก UI ทันที

5.4 เพิ่มพนักงานใหม่

```
// [เพิ่มพนักงาน]
const goToAddStaff = () => {
  router.push("/admin/staff-management/addstaff");
};
```

ปุ่ม “เพิ่มพนักงาน” พาไปยังหน้าฟอร์มเพิ่มข้อมูลพนักงาน

5.5 ไปหน้าแก้ไขพนักงาน (function มี)

```
// [ไปหน้าแก้ไขพนักงาน]
const goToEditStaff = (staffId: string) => {
  router.push(`admin/staff-management/editstaff?id=${staffId}`);
};
```

6.Add staff (component ของ StaffManagement)

6.1 เก็บข้อมูลจากแบบฟอร์มด้วย useState

```
const [firstName, setFirstName] = useState("");
const [lastName, setLastName] = useState("");
const [phoneNumber, setPhoneNumber] = useState("");
const router = useRouter();
```

ใช้ useState เพื่อเก็บข้อมูลที่ผู้ใช้งานกรอกในช่องชื่อ-นามสกุล และเบอร์โทรศัพท์

6.2 ส่งข้อมูลไปยัง API /api/addstaff

```
try {
  const res = await fetch("/api/addstaff", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(staff),
  });

  const result = await res.json();

  if (res.ok) {
    alert("เพิ่มพนักงานสำเร็จ!");
    router.push("/admin/staff-management");
  } else {
    alert("เกิดข้อผิดพลาด: " + result.message);
  }
} catch (error) {
  alert("เกิดข้อผิดพลาดในการเชื่อมต่อกับเซิร์ฟเวอร์");
}
```

- ส่งคำขอ POST ไปยัง backend เพื่อเพิ่มพนักงานใหม่
- ข้อมูลจะถูกส่งในรูปแบบ JSON

6.3 ตรวจสอบผลลัพธ์ + Redirect

```
if (res.ok) {
  alert("เพิ่มพนักงานสำเร็จ!");
  router.push("/admin/staff-management");
} else {
  alert("เกิดข้อผิดพลาด: " + result.message)
}
} catch (error) {
  alert("เกิดข้อผิดพลาดในการเชื่อมต่อกับเซิร์ฟเวอร์");
}
```

ถ้าเพิ่มสำเร็จ → แสดงข้อความ + ย้ายกลับหน้ารายการพนักงาน

6.4 แจ้งข้อผิดพลาด

```
|   }
| } catch (error) {
|   alert("เกิดข้อผิดพลาดในการเชื่อมต่อกับเซิร์ฟเวอร์");
|
|};
```

ถ้า server ตอบกลับ error หรือเกิดปัญหา → แจ้งผู้ใช้ทันที

Backend Staff

1 manage_rental

1.1 การตั้งค่าเริ่มต้นของคอมโพเนนต์

```
const ManageRental = () => {
  const router = useRouter();
  const [rentals] = useState<Rental>[]>([
    {
      id: "BK001",
      customerName: "สมชาย ใจดี",
      carModel: "Toyota Camry",
      bookingDate: "2023-06-15",
      totalPrice: 2500,
      status: "รอตรวจสอบ"
    },
    {
      id: "BK002",
      customerName: "ศรีดา กองมาก",
      carModel: "Honda Civic",
      bookingDate: "2023-06-16",
      totalPrice: 1800,
      status: "อนุมัติแล้ว"
    },
    {
      id: "BK003",
      customerName: "อุษ娜 สนับดี",
      carModel: "Ford Mustang",
      bookingDate: "2023-06-18",
      totalPrice: 3500,
      status: "ปฏิเสธ"
    },
  ]);
  const [searchTerm, setSearchTerm] = useState("");
  const [activeStatus, setActiveStatus] = useState("ทั้งหมด");
```

- rentals → เก็บข้อมูลการเช่ารถ (จำลองไว้ใน useState)
- searchTerm → เก็บค่าการค้นหาจาก input
- activeStatus → เก็บสถานะที่ถูกเลือก (ใช้กรองข้อมูล)

1.2 พังก์ชันกรองข้อมูล

```
const filteredRentals = rentals.filter(rental => {
  const matchesSearch = rental.customerName.includes(searchTerm) || rental.id.includes(searchTerm);
  const matchesStatus = activeStatus === "ทั้งหมด" || rental.status === activeStatus;
  return matchesSearch && matchesStatus;
});
```

- กรองการค้นหา → ถ้า searchTerm ตรงกับ customerName หรือ id
- กรองตามสถานะ → ถ้าสถานะตรงกับ activeStatus หรือเลือก "ทั้งหมด"

1.3 ตารางแสดงข้อมูลการเช่ารถ

```
/* Rental Table */


<div className="overflow-x-auto">
    <table className="min-w-full divide-y divide-gray-200">
      <thead className="bg-gray-50">
        <tr>
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">ລູກຄ້າ</th>
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">ກາງຈອງ</th>
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">ມັດປາກະນຸ</th>
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">ສຕານະ</th>
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">ທຳເນີນກາງ</th>
        </tr>
      </thead>


```

- แสดงข้อมูล ລູກຄ້າ, ກາງຈອງ, ຮາຄາທີ່ໜ້າຮະ, ແລະສຕານະ
- ໃຊ້ toLocaleString() ທຳໄໝແສດງຕົວເລຂຽກແບບອ່ານ່າຍ

1.4 ປຸ່ມດຳເນີນການ

```
</td>
<td className="px-6 py-4 whitespace-norewrap text-sm font-medium">
  <button className="text-blue-600 hover:text-blue-900 mr-3">
    <FaEye />
  </button>
  {rental.status === "ຮອດຈາກສອບ" && (
    <>
      <button className="text-green-600 hover:text-green-900 mr-3">
        <FaCheck />
      </button>
      <button className="text-red-600 hover:text-red-900">
        <FaTimes />
      </button>
    </>
  )}
</td>
</tr>
```

- ປຸ່ມ ດູຮາຍລະເວີຍດ (FaEye)
- ດຳສຕານະຄື່ອງ "ຮອດຈາກສອບ" ຈະແສດງປຸ່ມ ອຸນໍມືດີ (FaCheck) ແລະປັບປຸງເສັ້ນ (FaTimes)

2. Staff

2.1 ຄອມໂພහෙනත් Card (ການດັ່ງເນື້ອ)

```
const Card = ({ href, icon, title, bgColor }: { href: string; icon: React.ElementType; title: string; bgColor: string }) => (
  <Link href={href} className={`${`p-5 rounded-xl shadow-md hover:shadow-lg transition-all flex items-center gap-4 ${bgColor}`}`}>
    <Icon className="text-white text-3xl" />
    <span className="text-lg font-medium text-white">{title}</span>
  </Link>
);
```

- คอมโพเนนต์นี้สร้าง การ์ดลิงก์ “ไปยังหน้าอื่น”
- รับ props ได้แก่:
- href → URL ที่จะไป
- icon → ไอคอนที่แสดง
- title → ข้อความ
- bgColor → สีพื้นหลัง
- ใช้ Link เพื่อเปลี่ยนหน้าแบบ ไม่ต้องโหลดใหม่

2.2 คอมโพเนนต์ StaffDashboard

```
const StaffDashboard = () => [
  const router = useRouter();
  const handleLogout = () => {
    sessionStorage.removeItem("staffToken");
    router.push("/");
  };
]
```

- useRouter → ใช้เปลี่ยนเส้นทาง
- handleLogout → เมื่อกดปุ่มออกจากระบบ:
 1. ลบ Token ของพนักงานใน sessionStorage
 2. เปลี่ยนหน้าไปยัง หน้าหลัก (/)

3 Schedule (staff)

3.1 การจัดการสถานะและโหลดข้อมูล

```
const StaffSchedule = () => {
  const router = useRouter();
  const [currentDate, setCurrentDate] = useState(new Date());
  const [quickAddData, setQuickAddData] = useState<{ date: string, show: boolean }>({ date: "", show: false });
  const [selectedStaffId, setSelectedStaffId] = useState("");
  const [schedules, setSchedules] = useState<ScheduleItem[]>([]);
  const [staffList, setStaffList] = useState<StaffMember[]>([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState("");
```

3.1.1 การดึงข้อมูลพนักงาน: โค้ดที่ใช้ดึงข้อมูลจาก API:

```
// Fetch staff list from API
useEffect(() => {
  const fetchStaff = async () => {
    try {
      const response = await fetch('/api/getstaff');
      if (!response.ok) {
        throw new Error('Failed to fetch staff');
      }
      const data = await response.json();
      setStaffList(data);
      setLoading(false);
    } catch (error) {
      console.error("Failed to fetch staff:", error);
      setError("Failed to load staff data");
      setLoading(false);
    }
  };
  fetchStaff();
}, []);
```

```

const year = currentDate.getFullYear();
const month = currentDate.getMonth();
const daysInMonth = new Date(year, month + 1, 0).getDate();
const firstDayOfMonth = new Date(year, month, 1).getDay();

const handlePrevMonth = () => {
    setCurrentDate(new Date(year, month - 1, 1));
};

const handleNextMonth = () => {
    setCurrentDate(new Date(year, month + 1, 1));
};

```

ฟังก์ชันเหล่านี้จะปรับค่าของ currentDate เพื่อให้ปฏิทินแสดงเดือนก่อนหน้า หรือเดือนถัดไปเมื่อผู้ใช้คลิกปุ่มที่เกี่ยวข้อง.

3.3 พังก์ชันการเพิ่มงาน (Quick Add)

```

const handleAddClick = (dateStr: string, e: React.MouseEvent) => {
    e.stopPropagation();
    setQuickAddData({ date: dateStr, show: true });
    setSelectedStaffId("");
};

```

3.3.1 พังก์ชันสำหรับการเพิ่มตารางงาน

```

const handleAddSchedule = () => {
    if (selectedStaffId) {
        const selectedStaffMember = staffList.find(staff => staff.staffID === selectedStaffId);
        if (selectedStaffMember) {
            const newSchedule: ScheduleItem = {
                id: Date.now().toString(),
                staffName: `${selectedStaffMember.firstName} ${selectedStaffMember.lastName}`,
                staffId: selectedStaffMember.staffID,
                date: quickAddData.date
            };
            setSchedules([...schedules, newSchedule]);
        }
        setQuickAddData({ date: "", show: false });
    }
};

```

3.4 การลบตารางงาน

```

const handleDeleteSchedule = (id: string) => {
    setSchedules(schedules.filter(item => item.id !== id));
};

```

3.5 Modal สำหรับการเพิ่มงาน

```
/* Quick Add Modal */
{quickAddData.show && (
  <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center p-4 z-50">
    <div className="bg-white rounded-lg shadow-xl w-full max-w-sm p-6">
      <h3 className="text-lg font-semibold mb-4 text-gray-800">
        ພັນຍົການເລຳຮັບເວັບໃໝ່ (new Date(quickAddData.date).toLocaleDateString('th-TH'))
      </h3>
      <div className="mb-4">
        <label className="block text-sm font-medium text-gray-700 mb-1">ເລືອກພັກງານ</label>
        <select
          className="w-full p-2 border rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500 outline-none transition-all"
          value={selectedStaffId}
          onChange={(e) => setSelectedStaffId(e.target.value)}
          autoFocus
        >
          <option value="">-- ຖຸມເລືອກພັກງານ --</option>
          {staffList.map(staff => (
            <option
              key={staff.staffID}
              value={staff.staffID}
              disabled={schedules.some(s =>
                s.staffID === staff.staffID && s.date === quickAddData.date
              ))
            >
              {staff.firstName} {staff.lastName}
              {schedules.some(s =>
                s.staffID === staff.staffID && s.date === quickAddData.date
              ) && ' (ຈຳນວດ)'}
            </option>
          ))
        </select>
      </div>
    </div>
  </div>
)}
```