ระบบจัดการการเช่ารถ

Car Rental Management System

นางสาวพัตตรีมา โพธิสัตย์ รหัส 6806022510360 Sec3

นายธนบดี เอี่ยมสะอาด รหัส 6806022510386 Sec3

นายจิรวัฒน์ เชิดสูงเนิน รหัส 6806022510505 Sec3

นายมีสุข ดีเสมอ รหัส 6806022510513 Sec3

โครงการสหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาตรี

สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ปีการศึกษา 2568

คำนำ

การจัดทำโครงงาน “ระบบจัดการการเช่ารถ” ฉบับนี้ เป็นส่วนหนึ่งของรายวิชา Computer Programming ในหลักสูตรปริญญาตรี สาขาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ โดยมีวัตถุประสงค์เพื่อให้นักศึกษาได้ฝึกฝน และประยุกต์ใช้ความรู้ด้านการเขียนโปรแกรมในการพัฒนาระบบที่สามารถใช้งานได้จริง

โครงงานนี้มุ่งเน้นการออกแบบและพัฒนาโปรแกรมด้วยภาษา Python ซึ่งเป็นภาษาที่ใช้ในการเรียนการสอนรายวิชา Computer Programming โดยตรง ทั้งนี้จะช่วยเสริมสร้างทักษะการคิดวิเคราะห์ การออกแบบเชิงระบบ และการแก้ปัญหาทางเทคนิค เพื่อเตรียมความพร้อมให้แก่นักศึกษาในการประกอบวิชาชีพด้านวิศวกรรมสารสนเทศ และเครือข่ายต่อไปในอนาคต

คณะผู้จัดทำหวังเป็นอย่างยิ่งว่า รายงานฉบับนี้จะเป็นประโยชน์ต่อผู้ที่สนใจศึกษา นักเรียน นักศึกษา ตลอดจนผู้ที่ต้องการค้นคว้าหาความรู้เกี่ยวกับการพัฒนาโปรแกรมระบบงาน หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด คณะผู้จัดทำขอน้อมรับไว้ด้วยความยินดี และขออภัยมา ณ ที่นี้

สารบัญ

สารบัญ(ต่อ)

หน้า

สารบัญภาพ

หน้า

สารบัญภาพ(ต่อ)

หน้า

สารบัญภาพ(ต่อ)

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 วัตถุประสงค์ของโครงการ

1.1.1 เพื่อพัฒนาระบบจัดการการเช่ารถให้สามารถทำงานได้อย่างมีประสิทธิภาพ

1.1.2 เพื่อฝึกฝนทักษะการเขียนโปรแกรมด้วยภาษา Python

1.1.3 เพื่อเรียนรู้วิธีการจัดการข้อมูลและไฟล์แบบ Binary File I/O โดยใช้โครงสร้างข้อมูล (struct)

1.1.4 เพื่อเรียนรู้การทำงานร่วมกันเป็นทีมขอบเขตของโครงงาน

1.2 ขอบเขตของโครงการ

1.2.1 ระบบจัดการการเช่ารถ มีฟังก์ชันหลัก ๆ ดังนี้:

1.2.1.1 เพิ่มรถ (Add Car)

1.2.1.2 แก้ไขข้อมูลรถ (Update Car)

1.2.1.3 ลบรถ (Logical Delete Car)

1.2.1.4 ดูข้อมูลรถ (View Cars: All, Active, Deleted, One)

1.2.1.5 เพิ่มลูกค้า (Add Customer)

1.2.1.6 ดูรายชื่อลูกค้า (View All Customers)

1.2.1.7 เพิ่มรายการเช่า (Add Rental)

1.2.1.8 ดูรายการเช่าทั้งหมด (View All Rentals)

1.2.1.9 สร้างข้อมูลตัวอย่าง (Create Sample Data)

1.2.1.10 สร้างรายงาน (Generate Report)

1.2.1.11 เมนูหลักและเมนูย่อยสำหรับดำเนินการต่าง ๆ

1.2.1.12 ออกจากโปรแกรม

1.2.2 ระบบจัดการการเช่ารถ ประกอบด้วย 3 ไฟล์ข้อมูลหลักที่เป็นไฟล์ไบนารี ได้แก่:

1.2.2.1 ไฟล์ข้อมูลรถ cars.dat

1.2.2.2 ไฟล์ข้อมูลลูกค้า customers.dat

1.2.2.3 ไฟล์ข้อมูลการเช่า rentals.dat

1.2.2.4 ไฟล์รายงานที่เป็น Text File ชื่อ report.txt

1.2.3 ระบบจัดการการเช่ารถ มีการจัดเก็บข้อมูลในไฟล์ไบนารีรูปแบบ Fixed-length Record และมีส่วน Header สำหรับจัดเก็บข้อมูลเมตา (Metadata)

1.2.4 ระบบจัดการการเช่ารถ เป็นโปรแกรมที่ทำงานผ่าน Command Line Interface (CLI) ที่มีเมนูให้ผู้ใช้สามารถเลือกดำเนินการได้

1.3 ประโยชน์ที่ได้รับ

1.3.1 พัฒนาระบบที่สามารถใช้ในการจัดการการเช่ารถได้อย่างมีประสิทธิภาพ

1.3.2 พัฒนาทักษะการเขียนโปรแกรมด้วยภาษา Python และการประยุกต์ใช้ Python Standard Library

1.3.3 เรียนรู้การจัดการข้อมูลและไฟล์ในรูปแบบไบนารี (Binary File I/O) และการใช้โมดูล struct ในการจัดเรียงข้อมูล

1.3.4 เรียนรู้การทำงานร่วมกันเป็นทีมในการพัฒนาซอฟต์แวร์

1.4 เครื่องมือที่คาดว่าจะต้องใช้

1.4.1 โปรแกรม Visual Studio Code หรือ Editor อื่น ๆ ที่รองรับภาษา Python

1.4.2 โปรแกรม Microsoft Office สำหรับจัดทำเอกสาร

บทที่ 2
ระบบจัดการการเช่ารถ

2.1 แฟ้มเก็บข้อมูลรถ cars.dat

แฟ้มข้อมูลรถประกอบด้วย 11 ฟิลด์หลัก ซึ่งแต่ละฟิลด์มีรายละเอียดและความสำคัญ ดังนี้

ฟิลด์	ชนิด	ขนาด (bytes)	ตัวอย่าง
car_id	i	4	รหัสรถ
status	i	4	สถานะ Active=1, Deleted=0
is_rented	i	4	การเช่า Rented=1, Available=0
year	i	4	ปีที่ผลิต
daily_rate_thb	f	4	ค่าเช่าต่อวัน (บาท)
odometer_km	i	4	เลขไมล์
license_plate	12s	12	ทะเบียนรถ
brand	12s	12	ยี่ห้อ
model	16s	16	รุ่นรถ
created_at	i	4	timestamp สร้าง record
updated_at	i	4	timestamp แก้ไข record

ตารางที่ 2-1 แฟ้มข้อมูลรถ

2.1.1 car_id รหัสรถ

car_id เป็นรหัสรถที่ใช้ในการระบุรถแต่ละคันไม่ให้ซ้ำกัน ฟิลด์นี้ถูกสร้างขึ้นในรูปแบบตัวเลขจำนวนเต็ม (integer) เช่น 1001, 1002, 1003 เป็นต้น การมีรหัสรถที่ไม่ซ้ำกันช่วยหลีกเลี่ยงความสับสนระหว่างรถหลายคัน และช่วยให้สามารถค้นหา เรียกดู และอ้างอิงข้อมูลรถได้อย่างแม่นยำและรวดเร็ว โดยเฉพาะอย่างยิ่งเมื่อมีข้อมูลรถจำนวนมากในระบบ

2.1.2 status สถานะของรถ

status ใช้เก็บสถานะของระเบียนรถ โดยเป็นตัวเลขจำนวนเต็ม (integer) กำหนดค่า 1 หมายถึงข้อมูลยัง Active และ 0 หมายถึงข้อมูลถูกลบ (Deleted) การใช้สถานะแบบนี้ช่วยให้ระบบสามารถจัดการลบข้อมูลแบบ Logical Delete ได้โดยไม่ต้องลบระเบียนจริงในแฟ้ม ซึ่งมีประโยชน์ต่อ

การบันทึกประวัติและสามารถกู้คืนข้อมูลได้หากจำเป็น

2.1.3 is_rented สถานะการเช่า

is_rented เป็นค่าจำนวนเต็ม (integer) ที่ใช้กำหนดสถานะการเช่าของรถ หากค่าเป็น 1 หมายถึงรถกำลังถูกเช่าอยู่ และถ้าเป็น 0 หมายถึงรถว่างพร้อมให้เช่า ฟิลด์นี้มีความสำคัญในการตรวจสอบการใช้งานรถในปัจจุบัน และป้องกันไม่ให้รถคันเดียวกันถูกจองหรือปล่อยเช่าซ้ำซ้อน

2.1.4 year ปีที่ผลิต

year เก็บข้อมูลปีที่รถถูกผลิตออกมา ใช้ชนิดจำนวนเต็ม (integer) เช่น 2020, 2021 เป็นต้น การมีข้อมูลปีที่ผลิตช่วยให้ผู้ใช้หรือเจ้าหน้าที่ทราบอายุการใช้งานของรถ ซึ่งมีผลต่อราคาค่าเช่า และการบำรุงรักษา

2.1.5 daily_rate_thb ค่าเช่าต่อวัน

daily_rate_thb เป็นตัวเลขชนิดทศนิยม (float) ที่ใช้เก็บราคาค่าเช่ารถต่อวันในหน่วยบาท เช่น 1200.0, 1500.0 การกำหนดค่าเช่าต่อวันเป็นฟิลด์เฉพาะช่วยให้ระบบสามารถคำนวณค่าบริการรวมได้สะดวกเมื่อลูกค้าเช่ารถหลายวัน

2.1.6 odometer_km เลขไมล์

odometer_km ใช้เก็บตัวเลขจำนวนเต็ม (integer) ที่บอกระยะทางการใช้งานของรถ (กิโลเมตร) เช่น 35000, 78000 เป็นต้น ฟิลด์นี้มีความสำคัญต่อการตรวจสอบสภาพรถ การบำรุงรักษาตามระยะ และการประเมินความคุ้มค่าในการปล่อยเช่า

2.1.7 license_plate ทะเบียนรถ

license_plate ใช้เก็บข้อมูลตัวอักษร (string) แบบความยาวคงที่ 12 ไบต์ สำหรับบันทึกหมายเลขทะเบียนรถ เช่น "ABC-1234" การเก็บทะเบียนรถทำให้สามารถระบุตัวตนรถคันจริงได้ตรงกับเอกสารทางราชการ ป้องกันความผิดพลาดในการเช่าและการบันทึกข้อมูล

2.1.8 brand ยี่ห้อรถ

brand เป็นฟิลด์ข้อความ (string) ความยาวคงที่ 12 ไบต์ ใช้เก็บยี่ห้อรถ เช่น "Toyota", "Honda" การมีข้อมูลยี่ห้อช่วยให้ผู้เช่าเลือกได้ตามความต้องการและเป็นข้อมูลสำคัญต่อการวิเคราะห์ความนิยมของรถแต่ละยี่ห้อในระบบ

2.1.9 model รุ่นรถ

model เป็นฟิลด์ข้อความ (string) ความยาวคงที่ 16 ไบต์ ใช้เก็บชื่อรุ่นของรถ เช่น "Vios", "Civic" การมีข้อมูลรุ่นช่วยระบุรายละเอียดรถได้ชัดเจนมากขึ้น เพราะรถแต่ละยี่ห้ออาจมีหลายรุ่นย่อยที่แตกต่างกันทั้งรูปลักษณ์และสมรรถนะ

2.1.10 created_at เวลาที่สร้างระเบียน

created_at ใช้เก็บค่าเวลาในรูปแบบ timestamp (ชนิดจำนวนเต็ม integer) เพื่อบันทึกเวลาที่มีการสร้างข้อมูลรถระเบียนนั้น ๆ ฟิลด์นี้ช่วยให้สามารถตรวจสอบได้ว่าข้อมูลรถถูกเพิ่มเข้าสู่ระบบตั้งแต่เมื่อไร และสามารถใช้ในการจัดทำรายงานหรือสถิติได้

2.1.11 updated_at เวลาที่แก้ไขล่าสุด

updated_at ใช้เก็บค่าเวลาในรูปแบบ timestamp (ชนิดจำนวนเต็ม integer) เช่นเดียวกับ created_at แต่บันทึกเฉพาะเวลาที่มีการแก้ไขหรืออัปเดตข้อมูลล่าสุดของรถ ฟิลด์นี้มีความสำคัญในการติดตามการเปลี่ยนแปลงของข้อมูล ช่วยเพิ่มความถูกต้อง และความน่าเชื่อถือในการจัดการข้อมูล

2.2 แฟ้มเก็บข้อมูลลูกค้า customers.dat

แฟ้มข้อมูลลูกค้าประกอบด้วยฟิลด์หลัก 10 ฟิลด์ ใช้สำหรับบันทึกข้อมูลของผู้เช่ารถ เพื่อให้ระบบสามารถจัดการการเช่าได้อย่างถูกต้องและสะดวก

ฟิลด์	ชนิด	ขนาด (bytes)	ตัวอย่าง
cust_id	i	4	รหัสลูกค้า
status	i	4	สถานะ Active=1, Deleted=0
citizen_id	16s	16	เลขบัตรประชาชน
name	32s	32	ชื่อลูกค้า
surname	32s	32	นามสกุล
phone	16s	16	เบอร์โทรศัพท์
email	32s	32	อีเมล
address	64s	64	ที่อยู่
created_at	i	4	timestamp สร้าง record
updated_at	i	4	timestamp แก้ไข record

ตารางที่ 2-2 แฟ้มข้อมูลลูกค้า

2.2.1 cust_id รหัสลูกค้า

cust_id เป็นตัวเลขจำนวนเต็ม (integer) ใช้ในการระบุเอกลักษณ์ของลูกค้าแต่ละคนไม่ให้ซ้ำกัน เช่น 2001, 2002 เป็นต้น รหัสลูกค้าช่วยให้ค้นหาและเชื่อมโยงข้อมูลการเช่ากับลูกค้าได้อย่างรวดเร็ว โดยไม่สับสนแม้ว่าจะมีชื่อลูกค้าที่เหมือนกัน

2.2.2 status สถานะของลูกค้า

status เป็นค่าจำนวนเต็ม (integer) ใช้กำหนดสถานะของระเบียนลูกค้า ค่า 1 หมายถึงข้อมูลยัง Active และ 0 หมายถึงข้อมูลถูกลบ (Deleted) การใช้ Logical Delete ทำให้ยังคงบันทึกประวัติเดิมได้แม้ข้อมูลไม่ถูกใช้งานแล้ว

2.2.3 citizen_id เลขบัตรประชาชน

citizen_id ใช้เก็บหมายเลขบัตรประชาชน ความยาวคงที่ 16 ไบต์ (string) เช่น "1234567890123" ข้อมูลนี้ช่วยยืนยันตัวตนลูกค้า และป้องกันการสวมรอยหรือการใช้ข้อมูลซ้ำซ้อน

2.2.4 name ชื่อลูกค้า

name เป็นข้อความ (string) ความยาวคงที่ 32 ไบต์ ใช้เก็บชื่อลูกค้า เช่น "สมชาย" การเก็บชื่อจริงช่วยให้ติดต่อและบริการลูกค้าได้ตรงตัวตน และยังเป็นข้อมูลสำคัญในสัญญาเช่ารถ

2.2.5 surname นามสกุล

surname เป็นข้อความ (string) ความยาวคงที่ 32 ไบต์ ใช้เก็บนามสกุลลูกค้า เช่น "ใจดี" เมื่อนำมาใช้คู่กับชื่อจริงจะทำให้ระบุบุคคลได้อย่างถูกต้องและชัดเจน

2.2.6 phone เบอร์โทรศัพท์

phone ใช้เก็บข้อมูลหมายเลขโทรศัพท์ ความยาวคงที่ 16 ไบต์ (string) เช่น "0812345678" ฟิลด์นี้มีความสำคัญสำหรับการติดต่อกับลูกค้า ทั้งในกรณีการจองรถ การยืนยันการเช่า หรือการติดต่อกรณีฉุกเฉิน

2.2.7 email อีเมล

email เป็นข้อความ (string) ที่มีความยาวคงที่ 32 ไบต์ โดยจะใช้ในการเก็บอีเมลลูกค้า เช่น "customer@gmail.com" อีเมลช่วยให้สามารถส่งข้อมูลการจอง ใบเสร็จ หรือการแจ้งเตือนต่าง ๆ ไปยังลูกค้าได้สะดวก

2.2.8 address ที่อยู่

address เป็นข้อความ (string) ความยาวคงที่ 64 ไบต์ ใช้เก็บที่อยู่ของลูกค้า เช่น "123 หมู่ 4 แขวงลาดพร้าว เขตลาดพร้าว กรุงเทพฯ" การบันทึกที่อยู่มีความสำคัญต่อการทำสัญญา การตรวจสอบข้อมูล และการติดต่ออย่างเป็นทางการ

2.2.9 created_at เวลาที่สร้างระเบียน

created_at ใช้เก็บค่าเวลาในรูปแบบ timestamp (integer) เพื่อบันทึกเวลาที่สร้างข้อมูลลูกค้าในระบบ การมีฟิลด์นี้ช่วยให้ทราบว่าลูกค้าคนนี้เริ่มต้นใช้บริการตั้งแต่เมื่อใด และใช้ประกอบการทำรายงาน

2.2.10 updated_at เวลาที่แก้ไขล่าสุด

updated_at เป็น timestamp (integer) เช่นเดียวกับ created_at แต่ใช้เก็บเวลาที่มีการแก้ไขหรืออัปเดตข้อมูลล่าสุดของลูกค้า เพื่อช่วยติดตามความเปลี่ยนแปลงของข้อมูล และทำให้ระบบมีความถูกต้องทันสมัย

2.3 แฟ้มเก็บข้อมูลการเช่า rentals.dat

แฟ้มข้อมูลการเช่าใช้สำหรับเก็บรายละเอียดของการเช่ารถแต่ละครั้ง โดยมีการเชื่อมโยงระหว่าง รถ (cars.dat) และ ลูกค้า (customers.dat) เพื่อให้ระบบสามารถติดตามการใช้งานรถและข้อมูลลูกค้าได้ครบถ้วน

ฟิลด์	ชนิด	ขนาด (bytes	ตัวอย่าง
rental_id	i	4	รหัสการเช่า
status	i	4	สถานะ Active=1, Deleted=0
car_id	I	4	รหัสรถที่เช่า
cust_id	I	4	รหัสลูกค้า
Rent_date	I	4	วันที่เริ่มเช่า (timestamp)
return_date	I	4	วันที่คืนจริง (timestamp)
due_date	I	4	กำหนดวันคืน (timestamp)
total_days	i	4	จำนวนวันที่เช่า
total_amount	f	4	ค่าเช่ารวม (บาท)
created_at	i	4	timestamp สร้าง record
updated_at	i	4	timestamp แก้ไข record

ตารางที่ 2-3 แฟ้มข้อมูลการเช่า

2.3.1 rental_id รหัสการเช่า

rental_id เป็นรหัสไม่ซ้ำกันที่ใช้ระบุธุรกรรมการเช่าแต่ละครั้ง มีชนิดเป็นตัวเลขจำนวนเต็ม (integer) เช่น 3001, 3002 เป็นต้น การใช้รหัสเช่าแยกเฉพาะทำให้สามารถอ้างอิงธุรกรรมได้อย่างรวดเร็ว และช่วยแยกความแตกต่างแม้ว่าจะเป็นลูกค้าคนเดียวกันที่เช่าหลายครั้ง

2.3.2 status สถานะของการเช่า

status เป็นจำนวนเต็ม (integer) ใช้กำหนดสถานะของระเบียนการเช่า ค่า 1 หมายถึงยังใช้งานอยู่ (Active) และ 0 หมายถึงถูกลบ (Deleted) เพื่อช่วยจัดการข้อมูลด้วยวิธี Logical Delete และรักษาประวัติการเช่าเดิม

2.3.3 car_id รหัสรถที่เช่า

car_id เป็นจำนวนเต็ม (integer) ที่อ้างอิงไปยังฟิลด์ car_id ของแฟ้ม cars.dat เพื่อระบุว่าธุรกรรมการเช่านี้ใช้รถคันใด การเชื่อมโยงข้อมูลแบบนี้ช่วยให้ระบบสามารถติดตามประวัติการใช้งานของรถแต่ละคันได้อย่างถูกต้อง

2.3.4 cust_id รหัสลูกค้า

cust_id เป็นจำนวนเต็ม (integer) ที่อ้างอิงไปยังฟิลด์ cust_id ของแฟ้ม customers.dat เพื่อบอกว่าการเช่านี้เป็นของลูกค้าคนใด ช่วยให้สามารถตรวจสอบประวัติการเช่าและข้อมูลลูกค้าได้สะดวก

2.3.5 rent_date วันที่เริ่มเช่า

rent_date ใช้เก็บค่า timestamp (integer) ที่ระบุวันและเวลาที่เริ่มเช่ารถจริง ๆ ฟิลด์นี้มีความสำคัญต่อการคำนวณจำนวนวันเช่า และเป็นหลักฐานยืนยันการเริ่มสัญญาเช่า

2.3.6 return_date วันที่คืนจริง

return_date ใช้เก็บค่า timestamp (integer) เพื่อระบุวันและเวลาที่ลูกค้าคืนรถจริง ฟิลด์นี้ช่วยตรวจสอบความตรงต่อเวลาของการคืนรถ และใช้เปรียบเทียบกับ due_date เพื่อพิจารณาค่าปรับหรือค่าใช้จ่ายเพิ่มเติม

2.3.7 due_date กำหนดวันคืน

due_date เป็น timestamp (integer) ที่กำหนดวันครบกำหนดคืนรถตามสัญญา ช่วยให้สามารถเปรียบเทียบกับ return_date ได้ว่าลูกค้าคืนรถตรงเวลาหรือไม่ และใช้เป็นเงื่อนไขในการคิดค่าปรับกรณีคืนช้า

2.3.8 total_days จำนวนวันที่เช่า

total_days ใช้เก็บจำนวนเต็ม (integer) ของจำนวนวันทั้งหมดที่ลูกค้าเช่ารถ เช่น 3, 7, 10 วัน การเก็บข้อมูลนี้ทำให้สามารถตรวจสอบและยืนยันการคิดค่าเช่ารวมได้อย่างถูกต้อง

2.3.9 total_amount ค่าเช่ารวม

total_amount เป็นตัวเลขทศนิยม (float) ใช้เก็บผลรวมค่าเช่าทั้งหมด (หน่วยบาท) ซึ่งคำนวณจากจำนวนวัน (total_days) × ค่าเช่าต่อวัน (daily_rate_thb) ของรถ ฟิลด์นี้ทำให้สามารถแสดงค่าใช้จ่ายของลูกค้าได้ทันทีโดยไม่ต้องคำนวณซ้ำ

2.3.10 created_at เวลาที่สร้างระเบียน

created_at เป็น timestamp (integer) เก็บเวลาที่สร้างระเบียนการเช่า เพื่อใช้ตรวจสอบว่าสัญญาการเช่าเริ่มถูกบันทึกตั้งแต่เมื่อใด และเป็นประโยชน์ในการทำรายงานสถิติ

2.3.11 updated_at เวลาที่แก้ไขล่าสุด

updated_at เป็น timestamp (integer) ใช้เก็บเวลาที่มีการอัปเดตข้อมูลการเช่าล่าสุด เช่น มีการเปลี่ยนแปลงจำนวนวันหรือการแก้ไขค่าเช่า ช่วยให้การติดตามความถูกต้องของข้อมูลเป็นไปอย่างโปร่งใสและน่าเชื่อถือ

2.4 รายงานสรุป Report.txt

แฟ้ม report.txt เป็นแฟ้มข้อความที่ใช้สำหรับบันทึกข้อมูลการรายงานและสรุปผลการดำเนินงาน เช่น รายงานรถทั้งหมด รายงานลูกค้าทั้งหมด รายงานการเช่ารถ และรายงานสถิติการใช้งาน โดยสามารถเปิดอ่านได้ด้วยโปรแกรม Text Editor ทั่วไป

```
Car Rental Management System - Sample Reports
Generated At: 2025-10-03 22:49:06

=== Report A: Rentals by Customer (sample) ===
+-------------+--------------------------+----------------+------------+------------+------+---------------+
| Customer ID | Customer Name            | Car Rented     | Pick-up    | Return     | Days | Total Charge  |
+-------------+--------------------------+----------------+------------+------------+------+---------------+
| C1001       | Somchai Jaidee           | Toyota Vios    | 2025-09-22 | 2025-09-25 | 3    |       2700.00 |
| C1004       | Ms.Lileana Park          | Isuzu D-Max    | 2025-09-23 | 2025-09-25 | 2    |       3000.00 |
| C1003       | Mr.Anan Srisuk           | Honda Civic    | 2025-09-23 | 2025-09-29 | 6    |       7200.00 |
+-------------+--------------------------+----------------+------------+------------+------+---------------+

=== Report B: Rental Details (sample) ===
+--------+---------+-------------+------------+------------+------+----------+
| RentID | Car ID  | Customer ID | Pick-up    | Return     | Days | Returned |
+--------+---------+-------------+------------+------------+------+----------+
| 1001   | 1001    | C1001       | 2025-09-22 | 2025-09-25 | 3    | No       |
| 1002   | 1002    | C1004       | 2025-09-23 | 2025-09-25 | 2    | No       |
| 1003   | 1003    | C1003       | 2025-09-23 | 2025-09-29 | 6    | No       |
| 1004   | 1004    | C1002       | 2025-09-28 | 2025-09-29 | 1    | No       |
| 1005   | 1005    | C1005       | 2025-10-03 | 2025-10-07 | 4    | No       |
+--------+---------+-------------+------------+------------+------+----------+

=== Report C: Car Summary (Active only) ===
+-------+------------+-----------+-----------+------+------------+---------+
| CarID | Plate      | Brand     | Model     | Year | Rate (THB) | Status  |
+-------+------------+-----------+-----------+------+------------+---------+
| 1001  | ABX-1234   | Toyota    | Vios      | 2019 | 900.00     | Active  |
| 1003  | BCD-5678   | Honda     | Civic     | 2020 | 1200.00    | Active  |
| 1004  | ทพพ-76     | Mazda     | 2         | 2021 | 1000.00    | Active  |
| 1005  | EHI-7890   | Ford      | Ranger    | 2017 | 1300.00    | Active  |
| 1006  | BCD-5682   | Honda     | Civic     | 2018 | 500.00     | Active  |
| 1007  | REGH-5826  | FORT      | Mirasuki  | 2000 | 200.00     | Active  |
+-------+------------+-----------+-----------+------+------------+---------+

Summary:
- Active Cars     : 6
- Currently Rented: 4
- Available Now   : 2
- Rate Min/Max/Avg: 200.00 / 1300.00 / 850.00

Cars by Brand:
- Honda : 2
- Toyota : 1
- Mazda : 1
```

รูปภาพที่ 2-1 ไฟล์ Report

2.4.1 header_text ส่วนหัวรายงาน

เป็นฟิลด์ข้อความ (string 100 bytes) ใช้เก็บข้อความส่วนหัวรายงาน เช่น "Car Rental Management System – Sample Reports" เพื่อแสดงชื่อหรือวัตถุประสงค์ของรายงาน

2.4.2 generated_at วันที่และเวลาที่สร้างรายงาน (YYYY-MM-DD HH:MM:SS)

เป็นฟิลด์ข้อความ (string 25 bytes) ใช้เก็บวันและเวลาที่รายงานถูกสร้างขึ้น เช่น "2025-10-03 22:49:06"

2.4.3 report_sections ส่วนรายงานย่อย

ไฟล์ report.txt ถูกแบ่งเป็น 3 ส่วนหลัก ได้แก่ Report A: Rentals by Customer, Report B: Rental Details และ Report C: Car Summary

2.4.4 section_table_header หัวตาราง

เป็นฟิลด์ข้อความ (string 80 bytes) ใช้เก็บหัวตารางสำหรับแต่ละส่วน เช่น Customer ID, Customer Name, Car Rented, Pick-up, Return, Days, Total Charge

2.4.5 section_records ข้อมูลตาราง

เป็นฟิลด์ข้อความ (string 120 * N bytes, โดย N = จำนวนเรคอร์ด) ใช้เก็บข้อมูลจริงของแต่ละบรรทัด เช่น C1001, Somchai Jaidee, Toyota Vios, 2025-09-22, 2025-09-25, 3, 2700.00

2.4.6 summary_section สรุปข้อมูล

เป็นฟิลด์ข้อความ (string 150 bytes) ใช้เก็บข้อมูลสรุประบบ เช่น Active Cars, Currently Rented, Available Now และ Rate Min/Max/Avg

2.4.7 statistics_section สถิติการใช้งาน

เป็นฟิลด์ข้อความ (string 150 bytes) ใช้เก็บข้อมูลสถิติ เช่น Cars by Brand: Honda 2, Toyota 1, Mazda 1 เพื่อช่วยวิเคราะห์เชิงสถิติ

บทที่ 3
ขั้นตอนการดำเนินงาน

โปรแกรมการใช้งานระบบจัดการการเช่ารถ เป็นระบบจัดการการเช่ารถ จะช่วยในการจัดเก็บและบริหารข้อมูลรถให้ง่ายขึ้น ผู้ใช้สามารถเพิ่ม แก้ไข ลบ (แบบ Logical Delete) และแสดงข้อมูลรถได้สะดวก โดยข้อมูลของรถที่จัดเก็บในระบบจะประกอบไปด้วย Car ID, License Plate, Brand, Model, Year, Daily Rate, Odometer, Status และข้อมูลอื่น ๆ ที่เกี่ยวข้อง เพื่อความสะดวกในการเรียกใช้งาน โปรแกรมยังรองรับการค้นหาหรือดูข้อมูลเฉพาะที่ต้องการ เช่น ดูเฉพาะรถที่ Active หรือรถที่ถูกลบแล้ว

การจัดการข้อมูลรถทั้งหมดทำงานผ่านเมนูหลักของโปรแกรม ผู้ใช้สามารถเลือกเมนู Manage Cars เพื่อเข้าสู่การทำงานเกี่ยวกับข้อมูลรถ ซึ่งจะมีฟังก์ชันให้เลือกใช้งาน ได้แก่ เพิ่มรถใหม่ แก้ไขข้อมูลรถ ลบข้อมูลรถ และแสดงข้อมูลรถ

สำหรับผู้ใช้งานโปรแกรม

3.1 การจัดการข้อมูลรถ (Cars Management)

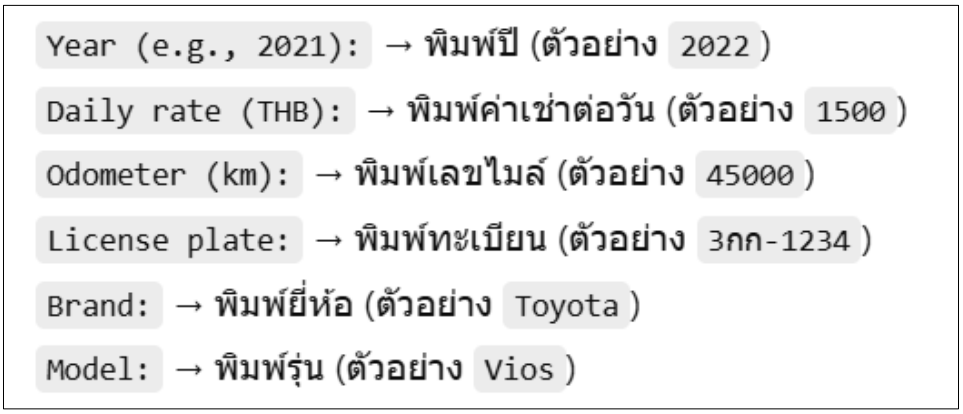
3.1.1 วิธีการเพิ่มข้อมูลรถ Add Car

3.1.1.1 กรอกหมายเลข 1 หลังคำว่า choose: ตามด้วยการคลิกที่ Enter เพื่อเรียกฟังก์ชัน Manage Cars เมื่อเลือก 1 เมนูย่อยของ Manage Cars จะขึ้นมา ประกอบด้วยตัวเลือกหลักดังนี้

```
Main Menu:
1) Add Car
2) Update Car
3) Delete Car (logical)
4) View Cars
   a) View one
   b) View all
   c) View only Active
   d) View only Deleted
5) Generate Report (3 examples -> report.txt)
6) Create Sample Data
7) Customers (view/add)
8) Rentals (view/add)
0) Exit
Choose: 1
```

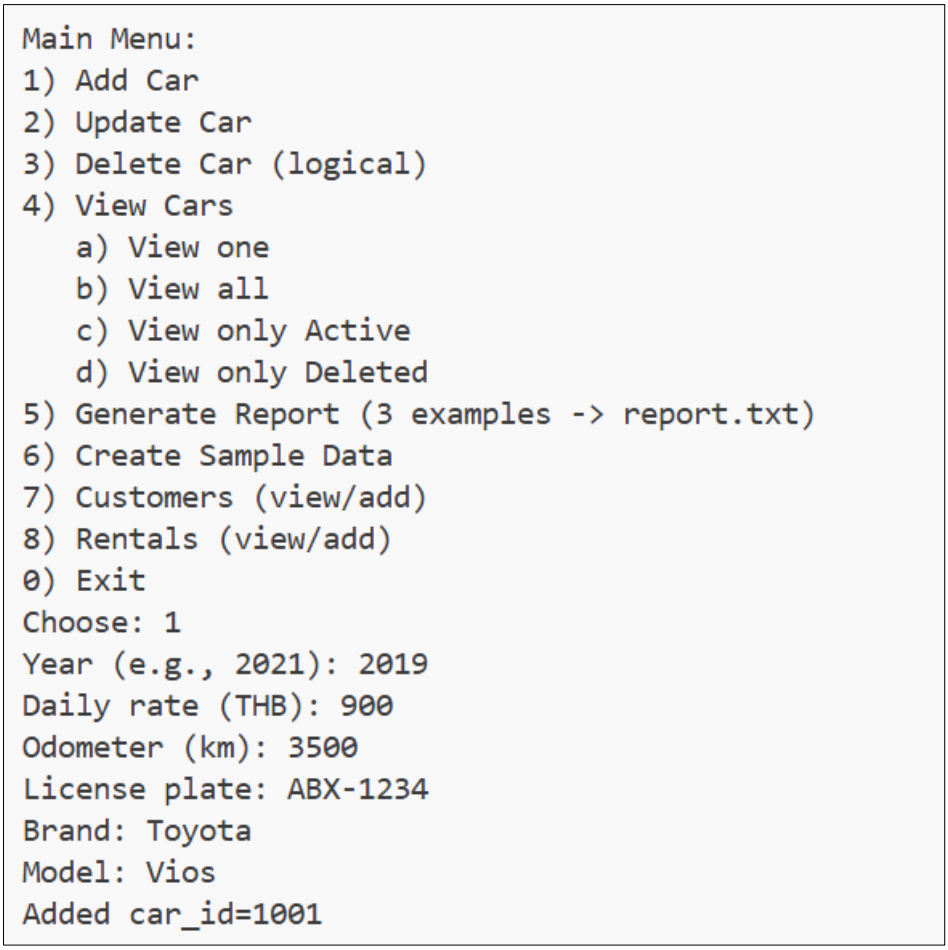
รูปภาพที่ 3-1 ตัวอย่างหน้าโปรแกรมระบบจัดการการเช่ารถ

3.1.1.2 โปรแกรมจะถามข้อมูลทีละฟิลด์ โดยแต่ละฟิลด์จะมีรูปแบบการเก็บข้อมูลดังนี้



รูปภาพที่ 3-2 ตัวอย่างรูปแบบการรับข้อมูลฟิลด์ Add car

3.1.1.3 เมื่อกรอกข้อมูลเสร็จสิ้น ระบบจะเพิ่มเรคอร์ดใหม่ และแสดงข้อความยืนยัน เช่น Added car_id=1001



รูปภาพที่ 3-3 หน้าแสดงข้อความยืนยันการเพิ่มข้อมูลรถ

3.1.2 วิธีการแก้ไขข้อมูลรถ Add Car

กรอกหมายเลข 2 หลังคำว่า choose: ตามด้วย Enter เพื่อไปที่ Update car เพื่อแก้ไขข้อมูลรถ โปรแกรมจะถามทีละฟิลด์อีกครั้ง หากข้อมูลที่ไม่ต้องการแก้ไขให้กด Enter

```
Main Menu:
1) Add Car
2) Update Car
3) Delete Car (logical)
4) View Cars
   a) View one
   b) View all
   c) View only Active
   d) View only Deleted
5) Generate Report (3 examples -> report.txt)
6) Create Sample Data
7) Customers (view/add)
8) Rentals (view/add)
0) Exit
Choose: 2
Enter car_id to update: 1002
Current: {'car_id': 1002, 'status': 0, 'is_rented': 0, 'year': 2020, 'daily_rate
: 'Isuzu', 'model': 'D-Max', 'created_at': 1759504346, 'updated_at': 1759504588}
Year [2020]: 2018
Daily rate [1500.0]:
Odometer [5800]:
License plate [Isuzu]: กท-234
Brand [Isuzu]:
Model [D-Max]:
Updated
```

รูปภาพที่ 3-4 หน้าแสดงการแก้ไขข้อมูลรถ

3.1.3 การลบรายการรถ Delete car

กรอกหมายเลข 3 หลังคำว่า choose: ตามด้วย Enter เพื่อไปที่ Delete car เพื่อแก้ไขข้อมูลรถ โปรแกรมจะถามว่า Enter car_id to delete (logical): ให้ผู้ใช้กรอกหมายเลข Car ID ของรถที่ต้องการลบ

```
Main Menu:
1) Add Car
2) Update Car
3) Delete Car (logical)
4) View Cars
   a) View one
   b) View all
   c) View only Active
   d) View only Deleted
5) Generate Report (3 examples -> report.txt)
6) Create Sample Data
7) Customers (view/add)
8) Rentals (view/add)
0) Exit
Choose: 3
Enter car_id to delete (logical): 1002
Deleted (logical)
```

รูปภาพที่ 3-5 หน้าแสดงการลบข้อมูลรถ

3.1.4 วิธีการดูหน้ารายการข้อมูลรถ View All Cars

กรอกหมายเลข 4 หลังคำว่า choose: ตามด้วย Enter เพื่อไปที่ View cars เพื่อดูข้อมูลรถ โปรแกรมจะถาม sub-menu ย่อยให้เลือก a/b/c/d ดังนี้

```
Main Menu:
1) Add Car
2) Update Car
3) Delete Car (logical)
4) View Cars
   a) View one
   b) View all
   c) View only Active
   d) View only Deleted
5) Generate Report (3 examples -> report.txt)
6) Create Sample Data
7) Customers (view/add)
8) Rentals (view/add)
0) Exit
Choose: 4
```

รูปภาพที่ 3-6 หน้าแสดงการดูข้อมูลรถ

3.1.4.1 View one (a)

ใช้ดูรายละเอียดรถทีละคัน โดยวิธีการใช้งานคือให้พิมพ์ 4 ตามด้วย Enter แล้วพิมพ์ a ลงใน a/b/c/d: หลังจากนั้นโปรแกรมจะถามผู้ใช้ต้องการดูข้อมูลรถเฉพาะรหัสรถอะไร ให้กรอกหมายเลข Car ID ในช่อง Enter car_id to view: ตามตัวอย่างรูป ดังนี้

```
Choose: 4
a/b/c/d: a
Enter car_id to view: 1001
{
  "car_id": 1001,
  "status": 1,
  "is_rented": 1,
  "year": 2019,
  "daily_rate_thb": 900.0,
  "odometer_km": 3500,
  "license_plate": "ABX-1234",
  "brand": "Toyota",
  "model": "Vios",
  "created_at": 1759504243,
  "updated_at": 1759505482
}
```

รูปภาพที่ 3-7 หน้าแสดงการดูข้อมูลรถ View one (a)

3.1.4.2 View one (b)

ใช้ดูรายละเอียดรถทีละคัน โดยวิธีการใช้งานคือให้พิมพ์ 4 ตามด้วย Enter แล้วพิมพ์ b ลงใน a/b/c/d: โปรแกรมจะแสดงข้อมูลรถทุกคันตามตัวอย่างรูป ดังนี้

```
Main Menu:
1) Add Car
2) Update Car
3) Delete Car (logical)
4) View Cars
   a) View one
   b) View all
   c) View only Active
   d) View only Deleted
5) Generate Report (3 examples -> report.txt)
6) Create Sample Data
7) Customers (view/add)
8) Rentals (view/add)
0) Exit
Choose: 4
a/b/c/d: b
+------+------------+-----------+-----------+------+----------+--------+
| ID   | Plate      | Brand     | Model     | Year | Rate     | Status |
+------+------------+-----------+-----------+------+----------+--------+
| 1001 | ABX-1234   | Toyota    | Vios      | 2019 | 900.00   | Active|
| 1002 | กท-234     | Isuzu     | D-Max     | 2018 | 1500.00  | Deleted|
| 1003 | BCD-5678   | Honda     | Civic     | 2020 | 1200.00  | Active|
| 1004 | ขท-763     | Mazda     | 2         | 2021 | 1000.00  | Active|
| 1005 | EHI-7890   | Ford      | Ranger    | 2017 | 1300.00  | Active|
+------+------------+-----------+-----------+------+----------+--------+
```

รูปภาพที่ 3-8 หน้าแสดงการดูข้อมูลรถ View one (b)

3.1.4.3 View one (c)

ใช้ดูรายละเอียดรถทีละคัน โดยวิธีการใช้งานคือให้พิมพ์ 4 ตามด้วย Enter แล้วพิมพ์ c ลงใน a/b/c/d: โปรแกรมจะแสดงข้อมูลรถทุกคันแต่ยกเว้นรถที่ถูกลบออกไปแล้วตามตัวอย่างรูป ดังนี้

```
Main Menu:
1) Add Car
2) Update Car
3) Delete Car (logical)
4) View Cars
   a) View one
   b) View all
   c) View only Active
   d) View only Deleted
5) Generate Report (3 examples -> report.txt)
6) Create Sample Data
7) Customers (view/add)
8) Rentals (view/add)
0) Exit
Choose: 4
a/b/c/d: c
+------+------------+-----------+-----------+------+----------+--------+
| ID   | Plate      | Brand     | Model     | Year | Rate     | Status |
+------+------------+-----------+-----------+------+----------+--------+
| 1001 | ABX-1234   | Toyota    | Vios      | 2019 | 900.00   | Active|
| 1003 | BCD-5678   | Honda     | Civic     | 2020 | 1200.00  | Active|
| 1004 | ขท-763     | Mazda     | 2         | 2021 | 1000.00  | Active|
| 1005 | EHI-7890   | Ford      | Ranger    | 2017 | 1300.00  | Active|
+------+------------+-----------+-----------+------+----------+--------+
```

รูปภาพที่ 3-9 หน้าแสดงการดูข้อมูลรถ View one (c)

3.1.4.4 View one (d)

ใช้ดูรายละเอียดรถทีละคัน โดยวิธีการใช้งานคือให้พิมพ์ 4 ตามด้วย Enter แล้วพิมพ์ c ลงใน a/b/c/d: โปรแกรมจะแสดงข้อมูลเฉพาะรถที่ถูกลบตามตัวอย่างรูป ดังนี้

```
Main Menu:
1) Add Car
2) Update Car
3) Delete Car (logical)
4) View Cars
   a) View one
   b) View all
   c) View only Active
   d) View only Deleted
5) Generate Report (3 examples -> report.txt)
6) Create Sample Data
7) Customers (view/add)
8) Rentals (view/add)
0) Exit
Choose: 4
a/b/c/d: d
+------+------------+-----------+-----------+------+----------+--------+
| ID   | Plate      | Brand     | Model     | Year | Rate     | Status |
+------+------------+-----------+-----------+------+----------+--------+
| 1002 | กท-234     | Isuzu     | D-Max     | 2018 | 1500.00  | Deleted|
+------+------------+-----------+-----------+------+----------+--------+
```

รูปภาพที่ 3-10 หน้าแสดงการดูข้อมูลรถ View one (d)

3.2 การจัดการข้อมูลลูกค้า (Customer Management)

3.2.1 วิธีการเพิ่มข้อมูลลูกค้า Add Customer

กรอกหมายเลข 7 หลังคำว่า choose: ตามด้วย Enter หลังจากนั้นโปรแกรมจะให้เลือกระหว่าง 1 การดูข้อมูลลูกค้า, 2 การเพิ่มข้อมูลลูกค้า, 3 การย้อนกลับไปที่เมนูหลัก ให้เลือก 2 เพื่อเพิ่มข้อมูลลูกค้า แล้วกรอกข้อมูลลูกค้าทีละบรรทัดตามตัวอย่าง ดังนี้

```
Choose: 7
Customers Menu:
1) View all customers
2) Add a customer (interactive)
0) Back
Choice: 2
Name: Somchai Jaidee
Phone: 0944867899
Email: Somchai78@gmail.com
Added customer id 1001
```

รูปภาพที่ 3-11 หน้าการเพิ่มข้อมูลลูกค้า

3.2.2 วิธีการดูหน้ารายการข้อมูลของลูกค้า

กรอกหมายเลข 7 หลังคำว่า choose: ตามด้วย Enter หลังจากนั้นโปรแกรมจะให้เลือกระหว่าง 1 การดูข้อมูลลูกค้า, 2 การเพิ่มข้อมูลลูกค้า, 3 การย้อนกลับไปที่เมนูหลัก ให้เลือก 1 เพื่อดูข้อมูลลูกค้าทั้งหมดตามตัวอย่าง ดังนี้

```
Main Menu:
1) Add Car
2) Update Car
3) Delete Car (logical)
4) View Cars
   a) View one
   b) View all
   c) View only Active
   d) View only Deleted
5) Generate Report (3 examples -> report.txt)
6) Create Sample Data
7) Customers (view/add)
8) Rentals (view/add)
0) Exit
Choose: 7
Customers Menu:
1) View all customers
2) Add a customer (interactive)
0) Back
Choice: 1
+------+--------------------------------+----------------+--------------------+
| ID   | Name                           | Phone          | Email              |
+------+--------------------------------+----------------+--------------------+
| 1001 | Somchai Jaidee                 | 0944867899     | Somchai78@gmail.co |
| 1002 | Ms.Kamoltip Nimnuan            | 0862507095     | Nimnuan95@gmail.co |
| 1003 | Mr.Anan Srisuk                 | 0865417155     | arip450@email.com  |
| 1004 | Ms.Lileana Park                | 0928201656     | rov5v5@gmail.com   |
| 1005 | Mr.John Smith                  | 0689755190     | johnkub55@gmail.co |
| 1006 | Mr. Somchai Jaidee             | 080-000-1000   | user1006@example.c |
| 1007 | Ms. Kamoltip Nimnuan           | 080-000-1001   | user1007@example.c |
| 1008 | Mr. Anan Srisuk                | 080-000-1002   | user1008@example.c |
| 1009 | Ms. Lina Park                  | 080-000-1003   | user1009@example.c |
| 1010 | Mr. John Smith                 | 080-000-1004   | user1010@example.c |
| 1011 | Ms. Anna Lee                   | 080-000-1005   | user1011@example.c |
| 1012 | Mr. David Brown                | 080-000-1006   | user1012@example.c |
| 1013 | Ms. Sara Kim                   | 080-000-1007   | user1013@example.c |
| 1014 | Mr. Tom Jones                  | 080-000-1008   | user1014@example.c |
| 1015 | Ms. Maria Gomez                | 080-000-1009   | user1015@example.c |
+------+--------------------------------+----------------+--------------------+
```

รูปภาพที่ 3-12 หน้าการดูข้อมูลลูกค้า

3.3 จัดการข้อมูลการเช่า (Rentals Management)

3.3.1 เพิ่มรายการเช่า (Add Rental)

กรอกหมายเลข 8 หลังคำว่า choose: ตามด้วย Enter หลังจากนั้นโปรแกรมจะให้เลือกระหว่าง 1 การดูข้อมูลการเช่ารถ, 2 การเพิ่มข้อมูลการเช่ารถ, 3 การย้อนกลับไปที่เมนูหลัก ให้เลือก 2 เพื่อเพิ่มข้อมูลการเช่ารถ แล้วกรอกข้อมูลลูกค้าทีละบรรทัดตามตัวอย่าง ดังนี้

```
Main Menu:
1) Add Car
2) Update Car
3) Delete Car (logical)
4) View Cars
   a) View one
   b) View all
   c) View only Active
   d) View only Deleted
5) Generate Report (3 examples -> report.txt)
6) Create Sample Data
7) Customers (view/add)
8) Rentals (view/add)
0) Exit
Choose: 8
Rentals Menu:
1) View all rentals
2) Add a rental (interactive)
0) Back
Choice: 2
Car ID: 1001
Customer ID: 1001
Pick-up date (YYYY-MM-DD): 2025-09-22
Days: 3
Added rental id 1001
```

รูปภาพที่ 3-13 หน้าการเพิ่มข้อมูลการเช่ารถ

3.3.2 แสดงรายการเช่าทั้งหมด (View All Rentals)

กรอกหมายเลข 8 หลังคำว่า choose: ตามด้วย Enter หลังจากนั้นโปรแกรมจะให้เลือกระหว่าง 1 การดูข้อมูลการเช่ารถ, 2 การเพิ่มข้อมูลการเช่ารถ, 3 การย้อนกลับไปที่เมนูหลัก ให้เลือก 1 เพื่อดูข้อมูลการเช่ารถตามตัวอย่าง ดังนี้

```
Choose: 8
Rentals Menu:
1) View all rentals
2) Add a rental (interactive)
0) Back
Choice: 1
+------+--------+--------+---------------------+------+---------+
| Rent | Car ID | CustID | Pick-up (YYYY-MM-DD) | Days | Returned |
+------+--------+--------+---------------------+------+---------+
| 1001 | 1001   | 1001   | 2025-09-22          | 3    | 0       |
| 1002 | 1002   | 1004   | 2025-09-23          | 2    | 0       |
| 1003 | 1003   | 1003   | 2025-09-23          | 6    | 0       |
| 1004 | 1004   | 1002   | 2025-09-28          | 1    | 0       |
| 1005 | 1005   | 1005   | 2025-10-03          | 4    | 0       |
+------+--------+--------+---------------------+------+---------+
```

รูปภาพที่ 3-14 หน้าการดูข้อมูลการเช่ารถ

บทที่ 4
ผลการดำเนินงาน

4.1 ฟังก์ชันในงานพื้นฐานในระบบเช่ารถ

4.1.1 struct กำหนดรูปแบบ record และ pack/unpack

ไฟล์ cars.dat, customers.dat, rentals.dat เก็บข้อมูลแบบ fixed-length records โดยใช้โมดูล struct เพื่อแปลงข้อมูล Python ⟶ bytes (pack) และ bytes ⟶ Python (unpack)

```
ENDIAN = '<'  # little-endian

CARS_STRUCT_FMT = ENDIAN + 'i i i i f i 12s 12s 16s i i'
CARS_RECORD_SIZE = struct.calcsize(CARS_STRUCT_FMT)
```

รูปภาพที่ 4-1 struct

4.1.1.1 อธิบาย format 1. < : little-endian (กำหนด byte order) 2. i : 4-byte signed integer 3. f : 4-byte float 4. 12s, 16s : fixed-length byte string (12 bytes / 16 bytes)

4.1.1.2 การแมปฟิลด์ (cars)

1. car_id (i)
2. status (i) — 1=active, 0=deleted
3.is_rented (i) — 1=rented, 0=free
4. year (i)
5. daily_rate_thb (f)
6. odometer_km (i)
7. license_plate (12s)
8. brand (12s)
9. model (16s)
10. created_at (i) — unix timestamp (int)
11. updated_at (i)

4.1.1.3 ตัวอย่างฟังก์ชัน pack / unpack

```python
def pack_car(car: dict) -> bytes:
    return struct.pack(
        CARS_STRUCT_FMT,
        car['car_id'],
        car['status'],
        car['is_rented'],
        car['year'],
        float(car['daily_rate_thb']),
        car['odometer_km'],
        str_to_fixed_bytes(car.get('license_plate',''), 12),
        str_to_fixed_bytes(car.get('brand',''), 12),
        str_to_fixed_bytes(car.get('model',''), 16),
        car['created_at'],
        car['updated_at']
    )

def unpack_car(raw: bytes) -> dict:
    vals = struct.unpack(CARS_STRUCT_FMT, raw)
    return {
        'car_id': vals[0],
        'status': vals[1],
        'is_rented': vals[2],
        'year': vals[3],
        'daily_rate_thb': float(vals[4]),
        'odometer_km': vals[5],
        'license_plate': fixed_bytes_to_str(vals[6]),
        'brand': fixed_bytes_to_str(vals[7]),
        'model': fixed_bytes_to_str(vals[8]),
        'created_at': vals[9],
        'updated_at': vals[10]
    }
```

รูปภาพที่ 4-2 pack และ unpack car

4.1.2 Helper: fixed-length string (padding / truncate)

ในโค้ดระบบจัดการการเช่ารถนี้มี 2 ฟังก์ชันสำคัญ ดังนี้

```python
def str_to_fixed_bytes(s: str, length: int) -> bytes:
    b = s.encode('utf-8')
    if len(b) > length:
        return b[:length]
    return b.ljust(length, b'\x00')

def fixed_bytes_to_str(b: bytes) -> str:
    return b.split(b'\x00', 1)[0].decode('utf-8', errors='ignore')
```

รูปภาพที่ 4-3 pack และ unpack car

4.1.2.1 อธิบายฟังก์ชัน

4.1.2.1.1 str_to_fixed_bytes 1. แปลง s ⟶ UTF-8 bytes 2. ถ้ายาวกว่า length ให้ตัดทิ้ง (truncate) ระวัง ตัดใน byte-level อาจตัดเพียงส่วนของ multi-byte char 3. ถ้าสั้นกว่าก็ ljust เติม \x00 (null bytes) จนได้ความยาวที่กำหนด

4.1.2.2 fixed_bytes_to_str 1. ตัดที่ null byte ตัวแรก (padding) แล้ว decode เป็น string 2. ใช้ errors='ignore' เพื่อไม่ให้เกิด exception หาก byte sequence ไม่สมบูรณ์

4.1.2.2 ตัวอย่างผลลัพธ์

"TOYOTA" ⟶ b'TOYOTA\x00\x00\x00...' (เต็มความยาว 12/16)

b'TOYOTA\x00...' ⟶ "TOYOTA"

4.1.2.3 ฟังก์ชันเสริมแนะนำ (เพื่อดูตัวอย่าง byte/padding เมื่อ debug):

```
def debug_fixed_bytes(s, length):
    b = str_to_fixed_bytes(s, length)
    print(repr(b))
    print(len(b))
```

รูปภาพที่ 4-4 debug

4.1.3 Header management — เก็บ metadata ในส่วนหัวของไฟล์

4.1.3.1 ระบบสำรองที่ HEADER_SIZE = 256 ไบต์สำหรับเก็บ JSON header (เช่น version, count, next_id)

```
def ensure_file(filename: str, record_size: int):
    if not os.path.exists(filename):
        with open(filename, 'wb') as f:
            meta = {
                'version': '1.0',
                'record_size': record_size,
                'created_at': int(time.time()),
                'count': 0,
                'next_id': 1001,
                'free_list': []
            }
            raw = json.dumps(meta, ensure_ascii=False).encode('utf-8')
            f.write(raw)
            f.write(b'\x00' * (HEADER_SIZE - len(raw)))
            f.flush(); os.fsync(f.fileno())
```

รูปภาพที่ 4-5 ensure

4.1.3.2 และการเขียน/อ่าน header ปัจจุบัน

```python
def write_header(filename: str, meta: dict):
    with open(filename, 'r+b') as f:
        raw = json.dumps(meta, ensure_ascii=False).encode('utf-8')
        if len(raw) > HEADER_SIZE:
            raise ValueError('Header too large')
        f.seek(0)
        f.write(raw)
        f.write(b'\x00' * (HEADER_SIZE - len(raw)))
        f.flush(); os.fsync(f.fileno())

def read_header(filename: str) -> dict:
    if not os.path.exists(filename):
        return {}
    with open(filename, 'rb') as f:
        data = f.read(HEADER_SIZE)
        try:
            s = data.split(b'\x00', 1)[0].decode('utf-8')
            if not s:
                return {}
            return json.loads(s)
        except Exception:
            return {}
```

รูปภาพที่ 4-6 write และ read

4.1.3.1 อธิบายฟังก์ชัน

Header เป็น JSON แบบง่ายๆ เก็บ count (จำนวน record ที่มี), next_id (id ถัดไป), last_updated (อาจถูกเพิ่มโดย write_header) 2. เมื่อสร้างไฟล์ใหม่ ensure_file จะเขียน header เริ่มต้น + padding ให้พอดี 256 ไบต์ 3. write_header จะอัปเดตข้อมูล meta ในตำแหน่งเริ่มต้นของไฟล์เสมอ (r+b) 4. การจำกัด HEADER_SIZE ป้องกัน header ใหญ่เกิน จึงเป็น good practice

4.1.3.2 ข้อเสนอแนะเพิ่มเติม

ถ้าต้องการ header ขนาดใหญ่ขึ้น ให้เพิ่ม HEADER_SIZE และระวังให้ read_header / write_header สอดคล้องกัน 2. เก็บ free_list เพื่อ reuse บันทึกที่ลบแบบ logical

4.1.4 Low-level record access (offsets, append, write-at, read-at)

1. การคำนวณตำแหน่ง record ในไฟล์:

```python
def get_record_offset(index: int, record_size: int) -> int:
    return HEADER_SIZE + index * record_size
```

รูปภาพที่ 4-7 get_record

2. append_record (เพิ่ม record ท้ายไฟล์ + อัปเดต header)

```python
def append_record(filename: str, record_bytes: bytes, record_size: int):
    with open(filename, 'r+b') as f:
        meta = read_header(filename)
        count = meta.get('count', 0)
        f.seek(HEADER_SIZE + count * record_size)
        f.write(record_bytes)
        meta['count'] = count + 1
        meta['next_id'] = meta.get('next_id', 1001) + 1
        meta['last_updated'] = int(time.time())
        write_header(filename, meta)
```

รูปภาพที่ 4-8 append

3. write_record_at (เขียนซ้ำที่ index ที่กำหนด)

```python
def write_record_at(filename: str, index: int, record_bytes: bytes, record_size: int):
    with open(filename, 'r+b') as f:
        f.seek(get_record_offset(index, record_size))
        f.write(record_bytes)
        meta = read_header(filename)
        meta['last_updated'] = int(time.time())
        write_header(filename, meta)
```

รูปภาพที่ 4-9 write_record

4.1.4.1 อธิบายฟังก์ชัน

append_record อ่าน header เพื่อดู count แล้วเขียนบันทึกที่ offset = HEADER + count*record_size 2. อัปเดต meta['count'], meta['next_id'] และ last_updated ⟶ แล้วเรียก write_header 3. write_record_at ใช้เขียนซ้ำ record เดิม (ใช้เมื่อ update หรือลบแบบ logical) 4. read_record_at เช็กขนาดไฟล์ก่อนอ่าน เพื่อไม่ให้อ่านนอกขอบเขต

4.1.5 Find helpers — ค้นหา index ของ record ตาม id

```python
def find_car_index_by_id(car_id: int) -> int | None:
    meta = read_header(CARS_FILE)
    count = meta.get('count', 0)
    for idx in range(count):
        raw = read_record_at(CARS_FILE, idx, CARS_RECORD_SIZE)
        if not raw: break
        car = unpack_car(raw)
        if car['car_id'] == car_id:
            return idx
    return None
```

รูปภาพที่ 4-10 find

4.1.5.1 อธิบายฟังก์ชัน

1. อ่าน header เพื่อทราบจำนวน count 2. วน loop ตั้งแต่ 0..count-1: อ่าน raw record แต่ละตำแหน่ง ⟶ unpack ⟶ เช็ก id 3. คืนค่า index เมื่อพบ (ใช้ต่อกับ read_record_at / write_record_at) 4. ถ้าไม่พบคืน None

4.1.6 เวลา (timestamp) — เก็บ, แปลง และใช้งาน

ระบบใช้ int unix timestamp (วินาที) เก็บเวลา created_at/updated_at/pickup_ts

```
ts = int(time.time())
pickup_ts = int(time.mktime(pickup_dt.timetuple()))
```

รูปภาพที่ 4-11 Time

การแปลงกลับเป็นวันที่สำหรับแสดง

```
datetime.datetime.fromtimestamp(r['pickup_ts']).strftime('%Y-%m-%d')
```

รูปภาพที่ 4-12 datetime

4.1.6.1 อธิบายฟังก์ชัน

time.time() ⟶ timestamp ปัจจุบัน (float) ⟶ แปลงเป็น int เพื่อบันทึก 2. time.mktime(date.timetuple()) ⟶ แปลง datetime.date เป็น timestamp (local time) 3. fromtimestamp ⟶ แปลง timestamp ⟶ datetime เพื่อแสดง human-readable 4. ข้อควรระวังเรื่อง timezone: โค้ดนี้ทำงานบน timezone ของเครื่อง (local); ถ้าต้องการ cross 5. timezone ควรเก็บ UTC หรือใช้ datetime ที่มี timezone-aware

4.2 ฟังก์ชัน Pack และ Unpack

ระบบจัดการเช่ารถนี้ ใช้การจัดเก็บข้อมูลลงไฟล์ .dat ในรูปแบบ fixed-length binary record

4.2.1 Pack: เอา dict ⟶ struct.pack ⟶ bytes

4.2.2 Unpack: เอา raw bytes ⟶ struct.unpack ⟶ dict

การทำแบบนี้ทำให้ระบบประหยัดพื้นที่และเข้าถึง record ได้รวดเร็ว (random access) เพราะทุก record มีขนาดเท่ากัน

4.2.1 Cars (รถ)

โครงสร้าง record (จาก CARS_STRUCT_FMT) มี car_id | status | is_rented | year | daily_rate | odometer | license_plate | brand | model | created_at | updated_at

โค้ด pack/unpack:

```python
def pack_car(car: dict) -> bytes:
    return struct.pack(
        CARS_STRUCT_FMT,
        car['car_id'],
        car['status'],
        car['is_rented'],
        car['year'],
        float(car['daily_rate_thb']),
        car['odometer_km'],
        str_to_fixed_bytes(car.get('license_plate',''), 12),
        str_to_fixed_bytes(car.get('brand',''), 12),
        str_to_fixed_bytes(car.get('model',''), 16),
        car['created_at'],
        car['updated_at']
    )

def unpack_car(raw: bytes) -> dict:
    vals = struct.unpack(CARS_STRUCT_FMT, raw)
    return {
        'car_id': vals[0],
        'status': vals[1],
        'is_rented': vals[2],
        'year': vals[3],
        'daily_rate_thb': float(vals[4]),
        'odometer_km': vals[5],
        'license_plate': fixed_bytes_to_str(vals[6]),
        'brand': fixed_bytes_to_str(vals[7]),
        'model': fixed_bytes_to_str(vals[8]),
        'created_at': vals[9],
        'updated_at': vals[10]
```

รูปภาพที่ 4-13 Pack and unpack (car)

4.2.1.1 การทำงานทีละขั้นตอน (pack_car):

1. รับ dictionary car เช่น {'car_id':1001, 'brand':'Toyota', ...} 2. แปลงค่า string เป็น fixed-length bytes เช่น brand ⟶ b'Toyota\x00\x00... (12 bytes) 3. ใช้ struct .pack รวมทั้งหมดเป็น binary record 4. คืนค่าเป็น bytes ⟶ พร้อมเขียนลงไฟล์

4.2.1.2 การทำงานทีละขั้นตอน (unpack_car):

1. อ่าน raw bytes จากไฟล์ (ยาว 72 bytes) 2. ใช้ struct.unpack แตกเป็น tuple (ค่าตามลำดับ) 3. นำมา map กลับเป็น dict พร้อมแปลง string (fixed_bytes ⟶ str)

4.2.2 Customers (ลูกค้า)

โครงสร้าง record (CUST_STRUCT_FMT) มี cust_id | status | name(32) | phone (16) | email(32) | created_at | updated_at

โค้ด pack/unpack

```python
def pack_customer(cust: dict) -> bytes:
    return struct.pack(
        CUST_STRUCT_FMT,
        cust['cust_id'],
        cust['status'],
        str_to_fixed_bytes(cust.get('name',''), 32),
        str_to_fixed_bytes(cust.get('phone',''), 16),
        str_to_fixed_bytes(cust.get('email',''), 32),
        cust['created_at'],
        cust['updated_at']
    )

def unpack_customer(raw: bytes) -> dict:
    vals = struct.unpack(CUST_STRUCT_FMT, raw)
    return {
        'cust_id': vals[0],
        'status': vals[1],
        'name': fixed_bytes_to_str(vals[2]),
        'phone': fixed_bytes_to_str(vals[3]),
        'email': fixed_bytes_to_str(vals[4]),
        'created_at': vals[5],
        'updated_at': vals[6]
    }
```

รูปภาพที่ 4-14 Pack and unpack (customer)

4.2.2.1 อธิบายฟังก์ชัน

1. name ความยาวสูงสุด 32 ตัวอักษร, phone 16, email 32 2. pack: แปลงเป็น bytes + padding 3. unpack: ตัด null byte ออกแล้ว decode

4.2.3 Rentals (การเช่า)

โครงสร้าง record (RENT_STRUCT_FMT) มี rent_id | status | car_id | cust_id | pickup_ts | daily_rate | days | is_returned

โค้ด pack/unpack:

```python
def pack_rental(r: dict) -> bytes:
    return struct.pack(
        RENT_STRUCT_FMT,
        r['rent_id'],
        r['status'],
        r['car_id'],
        r['cust_id'],
        int(r['pickup_ts']),
        float(r['daily_rate']),
        int(r['days']),
        int(r['is_returned'])
    )

def unpack_rental(raw: bytes) -> dict:
    vals = struct.unpack(RENT_STRUCT_FMT, raw)
    return {
        'rent_id': vals[0],
        'status': vals[1],
        'car_id': vals[2],
        'cust_id': vals[3],
        'pickup_ts': vals[4],
        'daily_rate': float(vals[5]),
        'days': vals[6],
        'is_returned': vals[7]
    }
```

รูปภาพที่ 4-15 Pack and unpack (rentel)

4.2.3.1 อธิบายฟังก์ชัน

1. pickup_ts เก็บเป็น timestamp (int) เช่น 1696005000 2. is_returned เป็น flag 1=คืนแล้ว, 0=ยังไม่คืน 3. ใช้ daily_rate กับ days เพื่อคำนวณค่าเช่ารวม

4.2.4 ภาพรวมการใช้งาน Pack/Unpack

4.2.4.1 ตอน เพิ่มข้อมูลใหม่ ⟶ สร้าง dict ⟶ pack ⟶ append_record() ⟶ เขียนลงไฟล์

4.2.4.2 ตอน อ่านข้อมูล ⟶ read_record_at() ⟶ unpack ⟶ dict ⟶ ใช้งานในโปรแกรม

4.2.4.3 ตอน แก้ไข/ลบ ⟶ read_record_at ⟶ unpack ⟶ แก้ค่า ⟶ pack ⟶ write_record_at

4.3 ฟังก์ชันจัดการข้อมูล (CRUD)

ในระบบเช่ารถนี้ มีข้อมูลหลัก 3 ประเภท:

1. Cars (ข้อมูลรถ)
2. Customers (ข้อมูลลูกค้า)
3. Rentals (ข้อมูลการเช่า)

แต่ละกลุ่มจะมี CRUD (Create, Read, Update, Delete) หรือบางกรณีเป็น CRU (เพราะลบแบบ logical เท่านั้น)

4.3.1 การจัดการ Cars (รถ)

4.3.1.1 เพิ่มข้อมูลรถ add_car_interactive()

```
def add_car_interactive():
    meta = read_header(CARS_FILE)
    next_id = meta.get('next_id', 1001)
    car = {}
    car['car_id'] = next_id
    car['status'] = 1
    car['is_rented'] = 0
    try:
        car['year'] = int(input('Year (e.g., 2021): ').strip())
    except:
        car['year'] = 2020
    try:
        car['daily_rate_thb'] = float(input('Daily rate (THB): ').strip())
    except:
        car['daily_rate_thb'] = 1000.0
    try:
        car['odometer_km'] = int(input('Odometer (km): ').strip())
    except:
        car['odometer_km'] = 0
    car['license_plate'] = input('License plate: ').strip()
    car['brand'] = input('Brand: ').strip()
    car['model'] = input('Model: ').strip()
    ts = int(time.time())
    car['created_at'] = ts
    car['updated_at'] = ts
    raw = pack_car(car)
    append_record(CARS_FILE, raw, CARS_RECORD_SIZE)
    print(f"Added car_id={car['car_id']}")
```

รูปภาพที่ 4-16 add_car

4.3.1.2 ขั้นตอนการทำงาน

4.3.1.2.1 อ่าน next_id จาก header ⟶ ใช้เป็น car_id

4.3.1.2.2 รับค่าจากผู้ใช้ (ปี, ราคาเช่าต่อวัน, เลขไมล์, ป้ายทะเบียน, ยี่ห้อ, รุ่น)

4.3.1.2.3 ถ้าผู้ใช้กรอกผิด ⟶ ใส่ค่า default (year=2020, daily_rate =1000, odometer=0)

4.3.1.2.1 เพิ่ม timestamp created_at, updated_at

4.3.1.2.2 pack_car ⟶ แปลงเป็น binary record

4.3.1.2.3 เขียนไฟล์ด้วย append_record

4.3.1.2.4 แสดงผล “Added car_id=...”

4.3.1.3 แก้ไขข้อมูลรถ — update_car_interactive()

```
def update_car_interactive():
    try:
        cid = int(input('Enter car_id to update: ').strip())
    except:
        print('Invalid id'); return
    idx = find_car_index_by_id(cid)
    if idx is None:
        print('Car not found'); return
    raw = read_record_at(CARS_FILE, idx, CARS_RECORD_SIZE)
    car = unpack_car(raw)
    print('Current:', car)
    s = input(f"Year [{car['year']}]: ").strip()
    if s: car['year'] = int(s)
    s = input(f"Daily rate [{car['daily_rate_thb']}]: ").strip()
    if s: car['daily_rate_thb'] = float(s)
    s = input(f"Odometer [{car['odometer_km']}]: ").strip()
    if s: car['odometer_km'] = int(s)
    s = input(f"License plate [{car['license_plate']}]: ").strip()
    if s: car['license_plate'] = s
    s = input(f"Brand [{car['brand']}]: ").strip()
    if s: car['brand'] = s
    s = input(f"Model [{car['model']}]: ").strip()
    if s: car['model'] = s
    car['updated_at'] = int(time.time())
    raw2 = pack_car(car)
    write_record_at(CARS_FILE, idx, raw2, CARS_RECORD_SIZE)
    print('Updated')
```

รูปภาพที่ 4-17 update_car_interactive

4.3.1.3.1 อธิบายฟังก์ชัน

4.3.1.3.1 ค้นหา car_id ⟶ ถ้าไม่เจอ return

4.3.1.3.1 อ่าน record เดิม ⟶ แสดงค่า

4.3.1.3.1 รับค่าจากผู้ใช้ ⟶ ถ้าไม่ใส่จะคงค่าเดิม

4.3.1.3.1 update timestamp + pack_car ⟶ write_record_at

4.3.1.4 ลบข้อมูลรถ (logical delete) — delete_car_interactive()

```python
def delete_car_interactive():
    try:
        cid = int(input('Enter car_id to delete (logical): ').strip())
    except:
        print('Invalid id'); return
    idx = find_car_index_by_id(cid)
    if idx is None:
        print('Car not found'); return
    raw = read_record_at(CARS_FILE, idx, CARS_RECORD_SIZE)
    car = unpack_car(raw)
    if car['status'] == 0:
        print('Already deleted'); return
    car['status'] = 0
    car['updated_at'] = int(time.time())
    write_record_at(CARS_FILE, idx, pack_car(car), CARS_RECORD_SIZE)
    print('Deleted (logical)')
```

รูปภาพที่ 4-18 delete_car

4.3.1.5 ดูข้อมูลรถ

4.3.1.5.1 view_all_cars(filter_active=None) ดูทั้งหมด/เฉพาะ Active/เฉพาะ Deleted

4.3.1.5.2 view_one_car() — ดูรถคันเดียว (json format)

```python
def view_all_cars(filter_active: bool | None = None):
    meta = read_header(CARS_FILE)
    count = meta.get('count', 0)
    rows = []
    for idx in range(count):
        raw = read_record_at(CARS_FILE, idx, CARS_RECORD_SIZE)
        if not raw: break
        car = unpack_car(raw)
        if filter_active is True and car['status'] != 1:
            continue
        if filter_active is False and car['status'] != 0:
            continue
        rows.append(car)
    print('+------+------------+----------+----------+------+----------+--------+')
    print('| ID   | Plate      | Brand    | Model    | Year | Rate     | Status |')
    print('+------+------------+----------+----------+------+----------+--------+')
    for c in rows:
        st = 'Active' if c['status']==1 else 'Deleted'
        print(f"| {c['car_id']:<4} | {c['license_plate'][:10]:<10} | {c['brand'][:9]:<9} | {c['model'][:9]:<9} | {c['year']:<4} | {c['daily_rate_thb']:<8.2f} | {st:<6}|")
    print('+------+------------+----------+----------+------+----------+--------+')
```

รูปภาพที่ 4-19 view_all_cars

4.3.2 การจัดการ Customers (ลูกค้า)

4.3.2.1 เพิ่มลูกค้า add_customer()

```python
from __future__ import annotations
import struct
import os
import sys
import time
import datetime
import random
import json
from typing import Tuple, List

def add_customer(name: str, phone: str = '', email: str = '') -> int
    ensure_file(CUST_FILE, CUST_RECORD_SIZE)
    meta = read_header(CUST_FILE)
    cust_id = meta.get('next_id', 1001)
    ts = int(time.time())
    cust = {
        'cust_id': cust_id,
        'status': 1,
        'name': name,
        'phone': phone,
        'email': email,
        'created_at': ts,
        'updated_at': ts
    }
    append_record(CUST_FILE, pack_customer(cust), CUST_RECORD_SIZE)
    return cust_id
```

รูปภาพที่ 4-20 add_customer

4.3.2.1.1 อธิบายฟอร์ม

4.3.2.1.1 ใช้ header next_id ⟶ กำหนด customer id

4.3.2.1.2 บันทึกข้อมูลลูกค้า (status=1)

4.3.2.1.3 ใช้ pack_customer ⟶ append ลงไฟล์

4.3.2.2 ดูลูกค้าทั้งหมด view_all_customers()

```python
def view_all_customers():
    meta = read_header(CUST_FILE)
    count = meta.get('count', 0)
    rows = []
    for idx in range(count):
        raw = read_record_at(CUST_FILE, idx, CUST_RECORD_SIZE)
        if not raw: break
        rows.append(unpack_customer(raw))
    print('+------+------------------------------+---------------+--------------------+')
    print('| ID   | Name                         | Phone         | Email              |')
    print('+------+------------------------------+---------------+--------------------+')
    for c in rows:
        print(f"| {c['cust_id']:<4} | {c['name'][:30]:<30} | {c['phone'][:14]:<14} | {c['email'][:18]:<18} |")
    print('+------+------------------------------+---------------+--------------------+')
```

รูปภาพที่ 4-21 view_all_customers

4.3.2.2.1 อธิบาย

อ่านทุก record ⟶ แปลงเป็น dict ⟶ แสดงเป็นตาราง

4.3.3 การจัดการ Rentals (การเช่า)

4.3.3.1 เพิ่มการเช่า add_rental()

```python
def add_rental(car_id: int, cust_id: int, pickup_dt: datetime.date, days: int, daily_rate: float):
    ensure_file(RENT_FILE, RENT_RECORD_SIZE)
    meta = read_header(RENT_FILE)
    rent_id = meta.get('next_id', 1001)
    pickup_ts = int(time.mktime(pickup_dt.timetuple()))
    rent = {
        'rent_id': rent_id,
        'status': 1,
        'car_id': car_id,
        'cust_id': cust_id,
        'pickup_ts': pickup_ts,
        'daily_rate': float(daily_rate),
        'days': int(days),
        'is_returned': 0
    }
    append_record(RENT_FILE, pack_rental(rent), RENT_RECORD_SIZE)
    # mark car as rented
    idx = find_car_index_by_id(car_id)
    if idx is not None:
        car = unpack_car(read_record_at(CARS_FILE, idx, CARS_RECORD_SIZE))
        car['is_rented'] = 1
        car['updated_at'] = int(time.time())
        write_record_at(CARS_FILE, idx, pack_car(car), CARS_RECORD_SIZE)
    return rent_id
```

รูปภาพที่ 4-22 add_rental

4.3.3.1.1 ขั้นตอน

4.3.3.1.1.1 กำหนด rent_id ใหม่จาก header

4.3.3.1.1.2 เก็บข้อมูลการเช่า: car_id, cust_id, pickup_ts, daily _rate, days

4.3.3.1.1.3 เขียนลง rentals.dat

4.3.3.1.1.4 อัปเดตสถานะรถ ⟶ is_rented=1

4.3.3.1.1.5 คืนค่า rent_id

4.3.3.2 ดูการเช่าทั้งหมด — view_all_rentals()

```python
def view_all_rentals():
    meta = read_header(RENT_FILE)
    count = meta.get('count', 0)
    rows = []
    for idx in range(count):
        raw = read_record_at(RENT_FILE, idx, RENT_RECORD_SIZE)
        if not raw: break
        rows.append(unpack_rental(raw))
    print('+------+--------+--------+---------------------+------+---------+')
    print('| Rent | Car ID | CustID | Pick-up (YYYY-MM-DD) | Days | Returned |')
    print('+------+--------+--------+---------------------+------+---------+')
    for r in rows:
        dt = datetime.datetime.fromtimestamp(r['pickup_ts']).strftime('%Y-%m-%d')
        print(f"| {r['rent_id']:<4} | {r['car_id']:<6} | {r['cust_id']:<6} | {dt:<19} | {r['days']:<4} | {r['is_returned']:<7} |")
    print('+------+--------+--------+---------------------+------+---------+')
```

รูปภาพที่ 4-23 view_all_rentals

4.4 ฟังก์ชันการสร้างรายงาน

ฟังก์ชันหลักคือ

```
def generate_report_all():
    ...
```

รูปภาพที่ 4-24 generate_report

ใช้สร้างไฟล์ report.txt โดยมี 3 ส่วนสำคัญ:

1. Report A: Rentals by Customer ⟶ สรุปการเช่ารถของลูกค้า
2. Report B: Rental Details ⟶ รายการเช่ารถทั้งหมด
3. Report C: Car Summary ⟶ รายงานรถที่มีอยู่ + สรุปสถิติ

4.4.1 การโหลดข้อมูลก่อนทำรายงาน

ก่อนทำรายงาน ระบบต้องโหลดข้อมูลจากทั้ง 3 ไฟล์

```
# โหลด cars
cars_meta = read_header(CARS_FILE); cars_count = cars_meta.get('count', 0)
cars = {}
for i in range(cars_count):
    raw = read_record_at(CARS_FILE, i, CARS_RECORD_SIZE)
    if not raw: break
    c = unpack_car(raw); cars[c['car_id']] = c

# โหลด customers
cust_meta = read_header(CUST_FILE); cust_count = cust_meta.get('count', 0)
customers = {}
for i in range(cust_count):
    raw = read_record_at(CUST_FILE, i, CUST_RECORD_SIZE)
    if not raw: break
    cu = unpack_customer(raw); customers[cu['cust_id']] = cu

# โหลด rentals
rent_meta = read_header(RENT_FILE); rent_count = rent_meta.get('count', 0)
rentals = []
for i in range(rent_count):
    raw = read_record_at(RENT_FILE, i, RENT_RECORD_SIZE)
    if not raw: break
    r = unpack_rental(raw); rentals.append(r)
```

รูปภาพที่ 4-25 โหลดข้อมูล

4.4.1.1 อธิบาย

4.4.1.1.1 ใช้ read_header ⟶ ดูจำนวน record

4.4.1.1.2 วน loop อ่าน record ทั้งหมด

4.4.1.1.3 ใช้ unpack_* แปลงเป็น dict

4.4.1.1.4 cars, customers เก็บใน dict (key=id) เพื่อค้นหาง่าย

4.4.1.1.5 rentals เก็บเป็น list

4.4.2 Report A: Rentals by Customer

เป็นรายงานสรุปว่า ลูกค้าคนไหน เช่ารถอะไร วันที่เท่าไร และค่าใช้จ่ายรวม

```
lines.append('=== Report A: Rentals by Customer (sample) ===')
lines.append('+-------------+-------------------------+----------------+------------+------------+------+---------------+')
lines.append('| Customer ID | Customer Name           | Car Rented     | Pick-up    | Return     | Days | Total Charge  |')
lines.append('+-------------+-------------------------+----------------+------------+------------+------+---------------+')

rentals_sorted = sorted(rentals, key=lambda x: x['rent_id'])
example_rows = []
for r in rentals_sorted[:3]:
    cust = customers.get(r['cust_id'], {'name':'Unknown'})
    car = cars.get(r['car_id'], {'brand':'Unknown','model':'Unknown'})
    pickup_dt = datetime.datetime.fromtimestamp(r['pickup_ts']).date()
    return_dt = pickup_dt + datetime.timedelta(days=r['days'])
    total = r['daily_rate'] * r['days']
    example_rows.append((
        f"C{r['cust_id']}",
        cust.get('name','Unknown'),
        f"{car.get('brand','')} {car.get('model','')}".strip(),
        pickup_dt.strftime('%Y-%m-%d'),
        return_dt.strftime('%Y-%m-%d'),
        str(r['days']),
        f"{total:.2f}"
    ))
```

รูปภาพที่ 4-26 Report A

4.4.2.1 อธิบายทีละขั้นตอน

4.4.2.1.1 สร้าง header ตาราง

4.4.2.1.2 ดึง rentals (ล่าสุด 3 รายการ)

4.4.2.1.3 หาลูกค้า + รถ จาก dict

4.4.2.1.4 คำนวณ วันที่คืนรถ = pickup + days

4.4.2.1.5 คำนวณ ค่าใช้จ่ายรวม = daily_rate × days

4.4.2.1.5เพิ่มเป็น row ของรายงาน

4.4.2.2 ตัวอย่าง Output

```
=== Report A: Rentals by Customer (sample) ===
+-------------+-------------------------+----------------+------------+------------+------+---------------+
| Customer ID | Customer Name           | Car Rented     | Pick-up    | Return     | Days | Total Charge  |
+-------------+-------------------------+----------------+------------+------------+------+---------------+
| C1001       | Somchai Jaidee          | Toyota Vios    | 2025-09-22 | 2025-09-25 | 3    |       2700.00 |
| C1004       | Ms.Lileana Park         | Isuzu D-Max    | 2025-09-23 | 2025-09-25 | 2    |       3000.00 |
| C1003       | Mr.Anan Srisuk          | Honda Civic    | 2025-09-23 | 2025-09-29 | 6    |       7200.00 |
+-------------+-------------------------+----------------+------------+------------+------+---------------+
```

รูปภาพที่ 4-27 Output Report A

4.4.3 Report B: Rental Details

รายงานนี้เป็น รายการเช่ารถทั้งหมด (หรืออย่างน้อย 6 รายการล่าสุด)

```
lines.append('=== Report B: Rental Details (sample) ===')
lines.append('+--------+---------+-------------+------------+------------+------+----------+')
lines.append('| RentID | Car ID  | Customer ID | Pick-up    | Return     | Days | Returned |')
lines.append('+--------+---------+-------------+------------+------------+------+----------+')

detail_rows = rentals_sorted[:6]
for r in detail_rows:
    pickup_dt = datetime.datetime.fromtimestamp(r['pickup_ts']).date()
    return_dt = pickup_dt + datetime.timedelta(days=r['days'])
    lines.append(f"| {r['rent_id']:<6} | {r['car_id']:<7} | C{r['cust_id']:<10} | {pickup_dt.strftime('%Y-%m-%d')} | {return_dt
```

รูปภาพที่ 4-28 Report B

4.4.3.1 ตัวอย่าง Output

```
=== Report B: Rental Details (sample) ===
+--------+---------+-------------+------------+------------+------+----------+
| RentID | Car ID  | Customer ID | Pick-up    | Return     | Days | Returned |
+--------+---------+-------------+------------+------------+------+----------+
| 1001   | 1001    | C1001       | 2025-09-22 | 2025-09-25 | 3    | No       |
| 1002   | 1002    | C1004       | 2025-09-23 | 2025-09-25 | 2    | No       |
| 1003   | 1003    | C1003       | 2025-09-23 | 2025-09-29 | 6    | No       |
| 1004   | 1004    | C1002       | 2025-09-28 | 2025-09-29 | 1    | No       |
| 1005   | 1005    | C1005       | 2025-10-03 | 2025-10-07 | 4    | No       |
+--------+---------+-------------+------------+------------+------+----------+
```

รูปภาพที่ 4-29 Output Report B

4.4.4 Report C: Car Summary

รายงานนี้สรุป รถทั้งหมดที่ Active + จำนวนรถที่เช่าแล้ว/ว่างอยู่ + ค่าเช่าต่ำสุด/สูงสุด/เฉลี่ย + จำนวนรถตามแบรนด์

```
lines.append('=== Report C: Car Summary (Active only) ===')
lines.append('+-------+-----------+----------+----------+------+-----------+---------+')
lines.append('| CarID | Plate     | Brand    | Model    | Year | Rate (THB) | Status  |')
lines.append('+-------+-----------+----------+----------+------+-----------+---------+')

car_list = [v for v in cars.values() if v['status']==1]
car_list = sorted(car_list, key=lambda x: x['car_id'])
for c in car_list:
    st = 'Active' if c['status']==1 else 'Deleted'
    lines.append(f"| {c['car_id']:<5} | {c['license_plate'][:10]:<10} | {c['brand'][:9]:<9} | {c['model'][:9]:<9} | {c['year']:<4} | {c['daily_rate_thb']:<10.2f} | {st:<7} |")

# Summary
total = len(car_list)
rented = sum(1 for c in car_list if c['is_rented']==1)
available = total - rented
lines.append('Summary:')
lines.append(f'- Active Cars     : {total}')
lines.append(f'- Currently Rented: {rented}')
lines.append(f'- Available Now   : {available}')
if car_list:
    rates = [c['daily_rate_thb'] for c in car_list]
    lines.append(f'- Rate Min/Max/Avg: {min(rates):.2f} / {max(rates):.2f} / {sum(rates)/len(rates):.2f}')

# Cars by brand
brands = {}
for c in car_list:
    b = c['brand'] or 'Unknown'
    brands[b] = brands.get(b, 0) + 1
if brands:
    lines.append('Cars by Brand:')
    for k,v in sorted(brands.items(), key=lambda x: -x[1]):
        lines.append(f'- {k} : {v}')
```

รูปภาพที่ 4-30 Report C

4.4.4.1 อธิบาย

4.4.4.1.1 แสดงตารางรถที่ active เท่านั้น

4.4.4.1.2 นับจำนวนรถ ⟶ Active, Rented, Available

4.4.4.1.3 คำนวณ min/max/avg ของ daily_rate

4.4.4.1.4 นับจำนวนรถตามยี่ห้อ (brand)

4.4.4.2 ตัวอย่าง Output

```
=== Report C: Car Summary (Active only) ===
+-------+------------+-----------+-----------+------+------------+---------+
| CarID | Plate      | Brand     | Model     | Year | Rate (THB) | Status  |
+-------+------------+-----------+-----------+------+------------+---------+
| 1001  | ABX-1234   | Toyota    | Vios      | 2019 | 900.00     | Active  |
| 1003  | BCD-5678   | Honda     | Civic     | 2020 | 1200.00    | Active  |
| 1004  | ทพพ-76     | Mazda     | 2         | 2021 | 1000.00    | Active  |
| 1005  | EHI-7890   | Ford      | Ranger    | 2017 | 1300.00    | Active  |
| 1006  | BCD-5682   | Honda     | Civic     | 2018 | 500.00     | Active  |
| 1007  | REGH-5826  | FORT      | Mirasuki  | 2000 | 200.00     | Active  |
+-------+------------+-----------+-----------+------+------------+---------+

Summary:
- Active Cars     : 6
- Currently Rented: 4
- Available Now   : 2
- Rate Min/Max/Avg: 200.00 / 1300.00 / 850.00

Cars by Brand:
- Honda : 2
- Toyota : 1
- Mazda : 1
- Ford : 1
- FORT : 1
```

รูปภาพที่ 4-31 Output Report C and Car Summary

4.4.5 การเขียนไฟล์ Report

สุดท้าย รายงานทั้งหมดถูกบันทึกลง report.txt

```python
with open(REPORT_FILE, 'w', encoding='utf-8') as f:
    f.write('\n'.join(lines))
print(f"Report written to {REPORT_FILE}.")
```

รูปภาพที่ 4-32 การเขียนไฟล์ Report

4.5 ฟังก์ชันสร้างข้อมูลตัวอย่าง (Sample Data Generators)

ระบบเช่ารถนี้มีฟังก์ชันสำหรับ สร้างข้อมูลตัวอย่างอัตโนมัติ เพื่อช่วยในการทดสอบการทำงาน โดยข้อมูลตัวอย่างจะถูกบันทึกลงไฟล์ .dat เหมือนกับข้อมูลจริง ฟังก์ชันที่มีได้แก่:

1. create_sample_cars() สร้างข้อมูลรถ
2. create_sample_customers() สร้างข้อมูลลูกค้า
3. create_sample_rentals() สร้างข้อมูลการเช่า

4.5.1 สร้างข้อมูลรถ create_sample_cars()

```python
BRANDS = ['Toyota','Honda','Nissan','Mazda','BMW','Mercedes','MG','Isuzu','Ford']
MODELS = ['Camry','Civic','Almera','2','Fortuner','530e','Yaris','Accord','C300','March']

def create_sample_cars(n: int = 50):
    ensure_file(CARS_FILE, CARS_RECORD_SIZE)
    meta = read_header(CARS_FILE)
    start_id = meta.get('next_id', 1001)
    for i in range(n):
        cid = start_id + i
        car = {
            'car_id': cid,
            'status': 1,
            'is_rented': 1 if random.random() < 0.3 else 0,
            'year': random.randint(2015, 2023),
            'daily_rate_thb': float(random.choice([800,900,1000,1200,1500,1800,2200,2500,3000])),
            'odometer_km': random.randint(5000,150000),
            'license_plate': ''.join(random.choices('ABCDEFGHIJKLMNOPQRSTUVWXYZ', k=3)) + '-' + str(random.randint(1000,9999)),
            'brand': random.choice(BRANDS),
            'model': random.choice(MODELS),
            'created_at': int(time.time()),
            'updated_at': int(time.time())
        }
        append_record(CARS_FILE, pack_car(car), CARS_RECORD_SIZE)
    print(f'Created {n} sample car records (starting id {start_id})')
```

รูปภาพที่ 4-33 sample_cars

4.5.1.1 การทำงาน

4.5.1.1.1 กำหนดจำนวน record ที่ต้องการ (n=50)

4.5.1.1.2 สุ่มค่า brand, model, year, daily_rate, odometer, license_plate

4.5.1.1.3 กำหนด status=1 (active), is_rented = 30% โอกาสที่รถจะถูกเช่าอยู่

4.5.1.1.4 ใช้ append_record บันทึกลงไฟล์

4.5.1.2 ผลลัพธ์

4.5.1.2.1 ได้รถตัวอย่าง 50 คันขึ้นไป

4.5.1.2.3 มีความหลากหลายของยี่ห้อ, รุ่น, ราคา

4.5.2 สร้างข้อมูลลูกค้า create_sample_customers()

```
datetime.datetime.fromtimestamp(r['pickup_ts']).strftime('%Y-%m-%d')

def create_sample_customers(n: int = 10):
    ensure_file(CUST_FILE, CUST_RECORD_SIZE)
    sample_names = [
        "Mr. Somchai Jaidee","Ms. Kamoltip Nimnuan","Mr. Anan Srisuk",
        "Ms. Lina Park","Mr. John Smith","Ms. Anna Lee","Mr. David Brown",
        "Ms. Sara Kim","Mr. Tom Jones","Ms. Maria Gomez"
    ]
    meta = read_header(CUST_FILE)
    start_id = meta.get('next_id', 1001)
    for i in range(n):
        cid = start_id + i
        name = sample_names[i % len(sample_names)]
        cust = {
            'cust_id': cid,
            'status': 1,
            'name': name,
            'phone': f'080-000-{1000 + i}',
            'email': f'user{cid}@example.com',
            'created_at': int(time.time()),
            'updated_at': int(time.time())
        }
        append_record(CUST_FILE, pack_customer(cust), CUST_RECORD_SIZE)
    print(f'Created {n} sample customers (starting id {start_id})')
```

รูปภาพที่ 4-34 sample_customers

4.5.2.1 การทำงาน

4.5.2.1.1 ใช้ชื่อสมมติจาก list เช่น Somchai, Kamoltip, Anan, Lina

4.5.2.1.2 สร้างหมายเลขโทรศัพท์ในรูปแบบ 080-000-xxxx

4.5.2.1.3 สร้างอีเมลสมมติ userxxxx@example.com

4.5.2.1.4 บันทึกลงไฟล์ด้วย append_record

4.5.2.2 ผลลัพธ์

4.5.2.2.1 ได้ลูกค้าตัวอย่าง 10 คน

4.5.2.2.2 แต่ละคนมีข้อมูลพื้นฐานครบ: ชื่อ, เบอร์, อีเมล

4.6 ฟังก์ชันเมนูหลัก (Main Menu และ Sub-menu)

โค้ดในไฟล์ Car Rental.py ได้จัดทำเมนูแบบ Text-based (Console menu) ที่ใช้การวนลูป while True เพื่อรอให้ผู้ใช้เลือกคำสั่ง

4.6.1 Main Menu

```
MENU = '''
Main Menu:
1) Add Car
2) Update Car
3) Delete Car (logical)
4) View Cars
   a) View one
   b) View all
   c) View only Active
   d) View only Deleted
5) Generate Report (3 examples -> report.txt)
6) Create Sample Data
7) Customers (view/add)
8) Rentals (view/add)
0) Exit
Choose: '''
```

รูปภาพที่ 4-35 Main Menu

4.6.1.1 อธิบายเมนู

1–3 ⟶ จัดการรถ (เพิ่ม, แก้ไข, ลบ)

4 ⟶ ดูรถ (แบบย่อย a–d)

5 ⟶ สร้างรายงาน

6 ⟶ สร้างข้อมูลตัวอย่าง

7 ⟶ จัดการลูกค้า

8 ⟶ จัดการการเช่า

0 ⟶ ออกจากโปรแกรม

4.6.2 ฟังก์ชัน main_loop()

```python
def main_loop():
    initialize_all_files()
    while True:
        choice = input(MENU).strip()
        if choice == '1':
            add_car_interactive()
        elif choice == '2':
            update_car_interactive()
        elif choice == '3':
            delete_car_interactive()
        elif choice == '4':
            sub = input('a/b/c/d: ').strip().lower()
            if sub == 'a': view_one_car()
            elif sub == 'b': view_all_cars(None)
            elif sub == 'c': view_all_cars(True)
            elif sub == 'd': view_all_cars(False)
            else: print('Unknown')
        elif choice == '5':
            print('Generate Report:')
            print('a) Generate combined report (A,B,C) -> report.txt')
            print('b) Generate only Car Summary (old style)')
            c = input('Choice (a/b): ').strip().lower()
            if c == 'a':
                generate_report_all()
            else:
                generate_report_all()  # simplified
        elif choice == '6':
            sample_data_menu()
        elif choice == '7':
            customers_menu()
        elif choice == '8':
            rentals_menu()
        elif choice == '0':
            print('Exiting. Bye.')
            break
        else:
            print('Unknown choice')
```

รูปภาพที่ 4-36 main_loop

4.6.2.1 อธิบายการทำงานทีละขั้นตอน

1. .เริ่มจาก initialize_all_files() ตรวจสอบว่ามีไฟล์ .dat หรือยัง ถ้าไม่มีจะสร้างใหม่
2. แสดงเมนู รับค่า choice จากผู้ใช้
3. ใช้ if-elif แยกทำงานตามที่เลือก
4. เมื่อเลือก 4 (View Cars) มี sub-menu (a–d) ให้เลือกดูข้อมูล
5. เมื่อเลือก 5 (Generate Report) เลือกว่าจะทำรายงานเต็ม (A,B,C) หรือเฉพาะ Car Summary
6. เมื่อเลือก 6–8 เข้าสู่ sub-menu ของ Sample Data, Customers, Rentals
7. ถ้าเลือก 0 ⟶ ออกจากลูป

4.6.3 Sub-menu สำหรับ Sample Data

```
def sample_data_menu():
    print('Sample Data Menu:')
    print('1) Create 50 sample cars')
    print('2) Create 10 sample customers')
    print('3) Create 20 sample rentals (requires sample cars & customers)')
    print('0) Back')
    cmd = input('Choice: ').strip()
    if cmd == '1':
        create_sample_cars(50)
    elif cmd == '2':
        create_sample_customers(10)
    elif cmd == '3':
        create_sample_rentals(20)
    else:
        return
```

รูปภาพที่ 4-37 Sample Data Menu

4.6.3.1 อธิบายฟังก์ชัน

4.6.3.1.1 เลือก 1–3 เพื่อสร้างข้อมูลตัวอย่าง (cars/customers/rentals)

4.6.3.1.2 เลือก 0 ⟶ กลับไปเมนูหลัก

4.6.4 Sub-menu สำหรับ Customers

```
def customers_menu():
    print('Customers Menu:')
    print('1) View all customers')
    print('2) Add a customer (interactive)')
    print('0) Back')
    cmd = input('Choice: ').strip()
    if cmd == '1':
        view_all_customers()
    elif cmd == '2':
        name = input('Name: ').strip()
        phone = input('Phone: ').strip()
        email = input('Email: ').strip()
        cid = add_customer(name, phone, email)
        print(f'Added customer id {cid}')
    else:
        return
```

รูปภาพที่ 4-38 Customers Menu

4.6.4.1 อธิบาย

4.6.4.1.1 ดูข้อมูลลูกค้าทั้งหมด

4.6.4.1.2 เพิ่มลูกค้าใหม่ (กรอกชื่อ, เบอร์, อีเมล)

4.6.5 Sub-menu สำหรับ Rentals

```python
def rentals_menu():
    print('Rentals Menu:')
    print('1) View all rentals')
    print('2) Add a rental (interactive)')
    print('0) Back')
    cmd = input('Choice: ').strip()
    if cmd == '1':
        view_all_rentals()
    elif cmd == '2':
        try:
            car_id = int(input('Car ID: ').strip())
            cust_id = int(input('Customer ID: ').strip())
            pickup_str = input('Pick-up date (YYYY-MM-DD): ').strip()
            days = int(input('Days: ').strip())
            pickup_dt = datetime.datetime.strptime(pickup_str, '%Y-%m-%d').date()
            idx = find_car_index_by_id(car_id)
            rate = 1000.0
            if idx is not None:
                rate = unpack_car(read_record_at(CARS_FILE, idx, CARS_RECORD_SIZE))['daily_rate_thb']
            rid = add_rental(car_id, cust_id, pickup_dt, days, rate)
            print(f'Added rental id {rid}')
        except Exception as e:
            print('Invalid input or error:', e)
    else:
        return
```

รูปภาพที่ 4-39 Rentals Menu

4.6.5.1 อธิบาย

4.6.5.1.1 ดูรายการเช่าทั้งหมด

4.6.5.1.2 เพิ่มการเช่าใหม่:

4.6.5.1.2.1 ระบุ car_id และ cust_id

4.6.5.1.2.2 กำหนดวันเริ่มต้น (pickup date) และจำนวนวัน

4.6.5.1.2.3 ใช้ daily_rate ของรถจริง (ถ้ามี)

4.6.5.1.2.4 เรียก add_rental() เพื่อบันทึก

4.7 สรุปการทำงานของระบบทั้งหมด

ระบบ Car Rental Management System ที่พัฒนาด้วยภาษา Python และใช้ Binary File ในการเก็บข้อมูล ได้ถูกออกแบบให้ทำงานตามลำดับขั้นตอนและฟังก์ชันหลักดังนี้

4.7.1 โครงสร้างข้อมูล (Data Structures)

ระบบเก็บข้อมูล 3 ประเภทหลักในไฟล์แยกกัน:

4.7.1.1 Cars เก็บข้อมูลรถแต่ละคัน เช่น id, brand, model, year, daily_rate, status

4.7.1.2 Customers เก็บข้อมูลลูกค้า เช่น id, name, phone, email

4.7.1.3 Rentals เก็บข้อมูลการเช่า เช่น id, car_id, cust_id, pickup_date, days, is_returned

4.7.1.4 รูปแบบไฟล์: Binary File (fixed-length records)

4.7.1.4.1. ใช้ pack_* และ unpack_* ในการแปลงระหว่าง dict กับ binary

4.7.1.4.2. แต่ละไฟล์มี header เพื่อเก็บ metadata (เช่น count, next_id)

4.7.2 การทำงานของฟังก์ชันหลัก

4.7.2.1 ฟังก์ชันพื้นฐาน (4.1)

4.7.2.1.1 pack_car(), pack_customer(), pack_rental() ⟶ แปลง dict ⟶ binary

4.7.2.1.2 unpack_car() ฯลฯ ⟶ binary ⟶ dict

4.7.2.1.3. append_record(), read_record_at(), write_record_at() ⟶ จัดการบันทึก/แก้ไขไฟล์

4.7.2.2 ฟังก์ชันอ่าน–เขียนไฟล์ (4.2)

4.7.2.2.1 จัดการ header ของไฟล์ (count, next_id)

4.7.2.2.2 จัดเก็บ record แบบ fixed size

4.7.2.2.3 รองรับการแก้ไขและลบแบบ logical delete

4.7.2.3 ฟังก์ชัน CRUD (4.3)

4.7.2.3.1 Cars: เพิ่ม, แก้ไข, ลบ (logical), ดูข้อมูล

4.7.2.3.2 Customers: เพิ่ม, ดูข้อมูล

4.7.2.3.3 Rentals: เพิ่ม (พร้อมอัปเดตสถานะรถ), ดูข้อมูล

4.7.2.4 ฟังก์ชันรายงาน (4.4)

4.7.2.4.1 Report A: Rentals by Customer ⟶ สรุปว่าลูกค้าเช่าอะไร, วันไหน, รวมค่าใช้จ่าย

4.7.2.4.2 Report B: Rental Details ⟶ รายละเอียดการเช่าทั้งหมด

4.7.2.4.3 Report C: Car Summary ⟶ รายการรถทั้งหมด Active/Deleted, ค่าธรรมเนียม min/max/avg, จำนวนรถแต่ละแบรนด์

4.7.2.5 ฟังก์ชันสร้างข้อมูลตัวอย่าง (4.5)

4.7.2.5.1. create_sample_cars() ⟶ สร้างรถตัวอย่าง 50 คัน

4.7.2.5.2. create_sample_customers() ⟶ ลูกค้าตัวอย่าง 10 คน

4.7.2.5.3. create_sample_rentals() ⟶ ประวัติการเช่า 10–20 รายการ

4.7.2.6 ฟังก์ชันเมนูหลัก (4.6)

4.7.2.6.1 Main Menu ผู้ใช้เลือกการทำงาน: Cars, Customers, Rentals, Report, Sample Data

4.7.2.6.2 Sub-menu แบ่งย่อยแต่ละส่วน (view, add, update, delete)

4.7.2.6.3 ใช้ while True ให้ผู้ใช้สั่งงานต่อเนื่องจนกว่าจะ exit

4.7.3 การทำงานเชื่อมโยง (Workflow)

ภาพรวมการทำงานของระบบคือ

4.7.3.1 ผู้ใช้เปิดโปรแกรม

4.7.3.1.1 ระบบตรวจสอบไฟล์ .dat ทั้งหมด

4.7.3.1.2 ถ้าไม่มี สร้างไฟล์ใหม่พร้อม header

4.7.3.2 ผู้ใช้เลือกเมนู

4.7.3.2.1 จัดการรถ ⟶ เพิ่ม/แก้ไข/ลบ/ดู

4.7.3.2.2 จัดการลูกค้า ⟶ เพิ่ม/ดู

4.7.3.2.3 จัดการการเช่า ⟶ เพิ่ม/ดู

4.7.3.2.4 สร้างรายงาน ⟶ สรุปผลรวมใน report.txt

4.7.3.2.5 สร้างข้อมูลตัวอย่าง ⟶ เติมข้อมูลจำลองเพื่อทดสอบ

4.7.3.3 ระบบทำงานกับไฟล์

4.7.3.3.1 ทุกการเปลี่ยนแปลง (เพิ่ม, แก้ไข, ลบ) เขียนลง binary file โดยตรง

4.7.3.3.2 ทุกการดูข้อมูล ⟶ อ่านจากไฟล์ + แปลงเป็นตาราง

4.7.3.4 ผู้ใช้สิ้นสุดการทำงาน

4.7.3.4.1 กด 0 (Exit) ⟶ โปรแกรมจบการทำงาน

4.7.4 จุดเด่นของระบบ

4.7.4.1 ใช้ Binary File + Fixed-Length Record ⟶ ข้อมูลเข้าถึงได้เร็วและไม่ต้องพึ่งฐานข้อมูล

4.7.4.2 รองรับ CRUD ครบ ⟶ Cars, Customers, Rentals

4.7.4.3 สามารถสร้าง Report สรุปแบบอัตโนมัติ ⟶ ช่วยผู้ดูแลตรวจสอบสถานะรถและลูกค้าได้ง่าย

4.7.4.4 มี ฟังก์ชันสร้างข้อมูลจำลอง ⟶ สะดวกสำหรับการทดสอบ

4.7.4.5 ใช้ เมนูแบบ Text-based ⟶ เข้าใจง่ายและใช้งานได้ทันที

4.7.5 ข้อสังเกตและข้อจำกัด

4.7.5.1 ระบบใช้ไฟล์แบบ binary ⟶ ยากต่อการแก้ไขข้อมูลโดยตรง (ต้องผ่านโปรแกรม)

4.7.5.2 การลบเป็น Logical Delete (status=0) ⟶ ข้อมูลยังอยู่ในไฟล์ ไม่ได้ถูกลบจริง

4.7.5.3 ไม่มีระบบ ตรวจสอบสิทธิ์ (Authentication) ⟶ ใครก็เข้ามาแก้ไขได้

4.7.5.4 ยังไม่รองรับการทำงานแบบ หลายผู้ใช้พร้อมกัน (Concurrency)

บทที่ 5
สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

ระบบจัดการการเช่ารถที่พัฒนาขึ้นนี้ สามารถดำเนินการจัดเก็บและจัดการข้อมูล รถ ลูกค้า และรายการเช่าได้อย่างมีประสิทธิภาพ โดยมีการจัดเก็บข้อมูลในรูปแบบไฟล์ไบนารี (binary files) ซึ่งช่วยให้การเข้าถึงข้อมูลรวดเร็ว และทำให้สามารถสร้างรายการสรุปผลการดำเนินงาน เช่น จำนวนรถที่มีอยู่ ข้อมูลลูกค้า และสถิติการเช่าโดยรวมได้อย่างสะดวก ข้อมูลเหล่านี้มีประโยชน์อย่างยิ่งต่อการบริหารจัดการระบบการเช่ารถ

5.2 ปัญหาและอุปสรรคในการทำงาน

ในการพัฒนาระบบนี้ ปัญหาหลัก คือความซับซ้อนของการจัดการไฟล์ไบนารี (binary files) เนื่องจากต้องกำหนดโครงสร้างข้อมูล (struct format) สำหรับไฟล์ข้อมูลรถ (cars.dat) ลูกค้า (customers.dat) และการเช่า (rentals.dat) อย่างแม่นยำ ซึ่งอาจเกิดข้อผิดพลาดในการเขียนหรือถอดรหัสข้อมูลที่ไม่ถูกต้องได้ นอกจากนี้ การจัดการข้อมูลแบบไฟล์ไบนารียังไม่รองรับการทำงานพร้อมกันของผู้ใช้งานหลายคน (multi-user concurrency) ทำให้เกิดข้อจำกัดในการจัดการข้อมูลจำนวนมากหรือการเข้าถึงพร้อมกัน

5.3 ข้อเสนอแนะ

เพื่อปรับปรุงและขยายขีดความสามารถของระบบให้พร้อมใช้งานในอนาคต มีข้อเสนอแนะ ดังนี้:

5.3.1 ปรับเปลี่ยนมาใช้ระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database) เช่น MySQL หรือ SQLite เพื่อรองรับการจัดเก็บข้อมูลจำนวนมากและให้การเข้าถึงข้อมูลหลายผู้ใช้พร้อมกัน

5.3.2 เพิ่มเติมฟังก์ชันการค้นหาและกรองข้อมูล เช่น ค้นหารถตามยี่ห้อ รุ่น หรือราคา, ค้นหาลูกค้าตามชื่อ หรือ ID

5.3.3 ปรับปรุงระบบยืนยันตัวตน และกำหนดสิทธิ์การเข้าถึงของผู้ใช้แต่ละประเภท** (เช่น พนักงาน, ผู้จัดการ) เพื่อเพิ่มความปลอดภัย

5.3.4 พัฒนาโปรแกรมให้มีส่วนติดต่อผู้ใช้แบบกราฟิก (GUI) หรือเว็บแอปพลิเคชัน เพื่อเพิ่มความสะดวกในการใช้งานให้แก่ผู้ใช้

5.4 สิ่งที่ผู้จัดทำได้รับในการพัฒนาโครงการ

จากการพัฒนาโครงงานระบบจัดการการเช่ารถในครั้งนี้ ผู้จัดทำได้รับความรู้และประสบการณ์ด้านการออกแบบระบบ การเขียนโปรแกรมด้วยภาษา Python การใช้โครงสร้างข้อมูลแบบไบนารี (binary data structures) รวมถึงการคิดวิเคราะห์และแก้ไขปัญหาเชิงตรรกะ นอกจากนี้ยังได้ฝึกทักษะการทำงานเป็นทีม การแบ่งหน้าที่รับผิดชอบ และการจัดเวลาให้สอดคล้องกับแผนงาน ทำให้ผู้จัดทำมีความเข้าใจในกระบวนการพัฒนาระบบซอฟต์แวร์มากยิ่งขึ้น และสามารถนำไปประยุกต์ใช้ในโครงการหรือการทำงานจริงในอนาคตได้เป็นอย่างดี