ระบบยืม - คืนคีย์การ์ดในโรงแรม

Borrowing and return key card in hotel

นางสาวพัทธนันต์ ขันธลักษณา รหัส6806022510181 Sce2

นายศิริโรจน์ อุดมเดช รหัส6806022510271 Sce2

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ปีการศึกษา 2568

ลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

คำนำ

โครงงาน “ระบบยืม–คืนคีย์การ์ดในโรงแรม” จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของรายวิชา Computer Programming หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ โดยมีวัตถุประสงค์เพื่อให้นักศึกษาได้บูรณาการความรู้ที่ได้ศึกษา มาประยุกต์ใช้ในการพัฒนาโปรแกรมที่สามารถนำไปใช้งานได้จริง โครงงานฉบับนี้มุ่งเน้นการออกแบบและพัฒนาโปรแกรมด้วยภาษา Python ซึ่งเป็นภาษาที่ใช้ในการเรียนการสอนของรายวิชา เพื่อเสริมสร้างทักษะการคิดวิเคราะห์ การแก้ปัญหาทางเทคนิค ตลอดจนเพิ่มพูนศักยภาพด้านวิศวกรรมสารสนเทศและเครือข่ายอันจะเป็นประโยชน์ต่อการประกอบวิชาชีพในอนาคตคณะผู้จัดทำใคร่ขอกราบขอบพระคุณอาจารย์ผู้สอน ที่ได้กรุณาให้คำแนะนำและข้อเสนอแนะตลอดระยะเวลาการดำเนินโครงงาน จนทำให้รายงานฉบับนี้สำเร็จลุล่วงไปด้วยดี

ทั้งนี้คณะผู้จัดทำหวังเป็นอย่างยิ่งว่า รายงานฉบับนี้จะเป็นประโยชน์แก่ผู้อ่าน นักเรียน และนักศึกษาที่สนใจศึกษาค้นคว้าเพิ่มเติม หากมีข้อผิดพลาดหรือบกพร่องประการใด คณะผู้จัดทำขอน้อมรับไว้ด้วยความเคารพ และขออภัยมา ณ โอกาสนี้

สารบัญ

สารบัญภาพ

สารบัญตาราง

บทที่ 1 บทนำ

1.1 ความเป็นมาและความสำคัญ

ในปัจจุบัน ธุรกิจโรงแรมมีการบริหารจัดการที่ซับซ้อนขึ้น การควบคุมการเข้าถึงห้องพักและสิ่งอำนวยความสะดวกด้วย **คีย์การ์ด (Key Card)** เป็นมาตรฐานที่สำคัญ แต่การจัดการคีย์การ์ดจำนวนมาก เช่น การออก การคืน และการติดตามสถานะ ยังคงต้องพึ่งพาระบบเอกสารหรือการบันทึกข้อมูลแบบแมนนวล ซึ่งอาจนำไปสู่ข้อผิดพลาด ความล่าช้าในการบริการ หรือปัญหาด้านความปลอดภัย โครงงาน **“ระบบยืม – คืนคีย์การ์ดในโรงแรม”** นี้ จึงถูกพัฒนาขึ้นเพื่อนำเทคโนโลยีสารสนเทศมาช่วยในการจัดการกระบวนการเหล่านี้ให้เป็นระบบอัตโนมัติ (Automated) มากขึ้น โดยใช้ความรู้ด้าน **Computer Programming ด้วยภาษา Python** มาสร้างโปรแกรมที่สามารถบันทึกข้อมูลการยืม-คืนคีย์การ์ดได้อย่างรวดเร็ว ถูกต้อง และตรวจสอบย้อนหลังได้ เพื่อเพิ่มประสิทธิภาพในการดำเนินงานของพนักงานต้อนรับ และยกระดับมาตรฐานการบริการของโรงแรม

1.2 วัตถุประสงค์ของโครงงาน

โครงงานนี้มีวัตถุประสงค์หลักดังต่อไปนี้:

1 เพื่อพัฒนาโปรแกรมระบบยืม – คืนคีย์การ์ด ในโรงแรมที่สามารถทำงานได้จริง ด้วยการออกแบบและเขียนโค้ดโดยใช้ ภาษา Python และหลักการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming)

2 เพื่อสร้างฐานข้อมูล สำหรับจัดเก็บข้อมูลคีย์การ์ด ข้อมูลห้องพัก ข้อมูลพนักงาน และบันทึกประวัติการยืม – คืนคีย์การ์ดได้อย่างมีประสิทธิภาพ

3 เพื่อให้นักศึกษาได้ประยุกต์ใช้ ความรู้และทักษะด้านวิศวกรรมสารสนเทศและเครือข่าย ทั้งการวิเคราะห์ระบบ การออกแบบส่วนต่อประสานกับผู้ใช้ (User Interface) และการแก้ไขปัญหาทางเทคนิค

1.3 ขอบเขตของโครงงาน

เพื่อให้การพัฒนาโปรแกรมเป็นไปอย่างมีประสิทธิภาพ โครงงานนี้จึงกำหนดขอบเขตการดำเนินงานไว้ดังนี้:

1.3.1 ขอบเขตด้านการทำงานของระบบ (Functional Scope)

- การจัดการข้อมูลหลัก: ระบบสามารถเพิ่ม ลบ แก้ไข และค้นหาข้อมูลคีย์การ์ด ข้อมูลห้องพัก และข้อมูลพนักงานได้
- การบันทึกการยืม: พนักงานสามารถบันทึกการจ่ายคีย์การ์ดให้กับลูกค้าหรือพนักงานที่เกี่ยวข้อง โดยระบุหมายเลขคีย์การ์ด หมายเลขห้องพัก และช่วงเวลายืม

- การบันทึกการคืน: พนักงานสามารถบันทึกการรับคืนคีย์การ์ด และปรับสถานะคีย์การ์ดให้พร้อมใช้งาน (Available) ทันที
- การตรวจสอบสถานะ: ระบบสามารถแสดงสถานะปัจจุบันของคีย์การ์ดแต่ละใบ (เช่น พร้อมใช้งาน, ถูกยืม, สูญหาย) ได้อย่างรวดเร็ว
- การออกรายงาน: ระบบสามารถสร้างรายงานสรุปประวัติการยืม-คืนในช่วงเวลาที่กำหนดได้

1.3.2 ขอบเขตด้านเครื่องมือและภาษา (Tools and Language Scope)

- ภาษาโปรแกรม: ใช้ ภาษา Python ในการพัฒนาโปรแกรมหลัก
- ฐานข้อมูล: ใช้ SQLite หรือ MySQL ในการจัดเก็บข้อมูลหลักและบันทึกประวัติการทำรายการ
- ส่วนต่อประสานกับผู้ใช้ (UI): พัฒนาด้วยไลบรารีของ Python เช่น Tkinter หรือ PyQt สำหรับการทำงานบนระบบปฏิบัติการ Windows/Linux

1.4 ประโยชน์ที่คาดว่าจะได้รับ

การพัฒนาโครงงานนี้จะก่อให้เกิดประโยชน์ดังต่อไปนี้:

1. ลดความผิดพลาดและประหยัดเวลา: ช่วยลดขั้นตอนการบันทึกข้อมูลด้วยมือ ทำให้การยืมและคืนคีย์การ์ดรวดเร็วและแม่นยำยิ่งขึ้น
2. เพิ่มประสิทธิภาพการบริหารจัดการ: ทำให้ผู้ดูแลสามารถตรวจสอบสถานะคีย์การ์ดและประวัติการใช้งานได้อย่างง่ายดาย นำไปสู่การบริหารจัดการทรัพย์สินของโรงแรมที่ดีขึ้น
3. ความพร้อมทางอาชีพ: นักศึกษาได้ฝึกฝนทักษะการออกแบบระบบ การเขียนโปรแกรมด้วย Python และการแก้ไขปัญหาทางเทคนิค ซึ่งเป็นทักษะที่จำเป็นต่อการทำงานในสายงานวิศวกรรมสารสนเทศและเครือข่าย
4. เป็นต้นแบบในการพัฒนา: สามารถนำระบบนี้ไปพัฒนาต่อยอดให้ครอบคลุมการจัดการทรัพย์สินอื่น ๆ ในโรงแรมได้

1.5 เครื่องมือและเทคโนโลยีที่คาดว่าจะต้องใช้

ในการพัฒนาโครงงาน **“ระบบยืม – คืนคีย์การ์ดในโรงแรม”** คณะผู้จัดทำได้กำหนดเครื่องมือและเทคโนโลยีหลักที่คาดว่าจะนำมาใช้ดังต่อไปนี้:

1. ภาษาโปรแกรม: โครงงานนี้จะใช้ ภาษา Python เป็นภาษาหลักในการเขียนโค้ดและพัฒนาฟังก์ชันการทำงานทั้งหมดของระบบ เนื่องจากเป็นภาษาที่เน้นการอ่านง่ายและตรงตามข้อกำหนดของรายวิชา Computer Programming

2. ระบบจัดการฐานข้อมูล: เพื่อให้ระบบสามารถจัดเก็บข้อมูลหลัก เช่น ข้อมูลคีย์การ์ด ข้อมูลห้องพัก และบันทึกประวัติการทำรายการยืม-คืนได้อย่างมีประสิทธิภาพ คาดว่าจะเลือกใช้ระบบจัดการฐานข้อมูลแบบเชิงสัมพันธ์ เช่น SQLite (สำหรับความง่ายในการติดตั้งและใช้งานในเบื้องต้น) หรือ MySQL (หากต้องการรองรับการใช้งานพร้อมกันหลายเครื่องในอนาคต)
3. ส่วนต่อประสานกับผู้ใช้ (GUI): ในการสร้างหน้าจอโปรแกรมที่ใช้งานง่ายและเป็นมิตรกับผู้ใช้ จะใช้ไลบรารีมาตรฐานของ Python เช่น Tkinter หรืออาจพิจารณาใช้ PyQt/Kivy เพื่อพัฒนาส่วนติดต่อผู้ใช้งานแบบกราฟิก (GUI) ให้สามารถทำงานบนระบบปฏิบัติการ Windows/Linux ได้อย่างมีประสิทธิภาพ
4. สภาพแวดล้อมการพัฒนา (IDE): จะใช้โปรแกรมอย่าง Visual Studio Code (VS Code) หรือ PyCharm ในการเขียน ตรวจสอบ และทดสอบโค้ด เพื่อเพิ่มความสะดวกและรวดเร็วในกระบวนการพัฒนา
5. ไลบรารีที่จำเป็น: จะมีการประยุกต์ใช้ไลบรารีมาตรฐานและไลบรารีภายนอกของ Python เช่น โมดูล sqlite3 หรือ Connector ของฐานข้อมูลที่เลือกใช้สำหรับการเชื่อมต่อและจัดการข้อมูล รวมถึงไลบรารีอื่น ๆ เช่น datetime สำหรับการจัดการวันที่และเวลา และอาจใช้ pandas สำหรับการประมวลผลข้อมูลเบื้องต้นเพื่อออกรายงานสรุป

การเลือกใช้เครื่องมือเหล่านี้จะช่วยให้การพัฒนาระบบเป็นไปตามขอบเขตและวัตถุประสงค์ที่กำหนดไว้ และทำให้นักศึกษาได้ฝึกฝนทักษะการทำงานร่วมกับเทคโนโลยีที่เป็นที่นิยมในอุตสาหกรรม

บทที่ 2 เอกสารและงานวิจัยที่เกี่ยวข้อง

บทที่ 2 นี้ เป็นการนำเสนอการออกแบบและวิเคราะห์โครงสร้างข้อมูลและแฟ้มข้อมูลหลัก (Data Files) ที่ใช้ในการจัดเก็บข้อมูลของระบบยืม – คืนคีย์การ์ดในโรงแรม ระบบนี้พัฒนาขึ้นโดยใช้การจัดการข้อมูลแบบไบนารี (Binary File) ผ่าน Module struct ของภาษา Python

2.1 โครงสร้างแฟ้มข้อมูลของระบบยืม – คืนคีย์การ์ด

ระบบถูกออกแบบให้มีการจัดเก็บข้อมูลหลักออกเป็น 4 แฟ้มข้อมูล (Data Files) ในรูปแบบไฟล์ไบนารี โดยมีการกำหนดขนาดและชนิดของข้อมูลที่แน่นอนตามที่ใช้ใน Module struct เพื่อให้เกิดประสิทธิภาพสูงสุดในการจัดเก็บและเรียกใช้ข้อมูล

2.1.1 แฟ้มข้อมูลห้อง (rooms.dat)

แฟ้มข้อมูลนี้ทำหน้าที่จัดเก็บรายละเอียดคงที่และสถานะของห้องพักแต่ละห้องในโรงแรม

ฟิลด์ (Field)	ชนิด (Type)	ขนาด (Bytes)	คำอธิบาย
room id	I	4	รหัสหลัก ใช้ระบุห้องพักแต่ละห้อง
status	I	4	สถานะของห้อง (1=Active, 0=Deleted)
room_type	20s	20	ประเภทห้อง (เช่น STD, DELUXE, SUITE)
floor	I	4	ชั้นที่ตั้งของห้อง
capacity	I	4	ความจุคนพักสูงสุด
max_cards	I	4	จำนวนคีย์การ์ดสูงสุดที่ออกให้ได้
created at	I	4	timestamp ที่สร้างข้อมูลห้อง
updated_at	I	4	timestamp ล่าสุดที่มีการแก้ไขข้อมูล

ตารางที่ 2.1 แฟ้มข้อมูลห้อง

คำอธิบายฟิลด์:

- room id: เป็น รหัสหลัก (Primary Key) ที่ไม่ซ้ำกัน ใช้ระบุห้องพักแต่ละห้อง และเป็นกุญแจสำคัญในการเชื่อมโยงข้อมูลไปยังแฟ้ม stays.dat
- status: ใช้เพื่อระบุสถานะการใช้งานของห้องพักในระบบ หากเป็น 0 (Deleted) จะถูกซ่อนจากการแสดงผลในเมนูการจอง แต่ข้อมูลยังคงอยู่ในแฟ้ม
- room_type: ระบุประเภทของห้องพัก ซึ่งอาจมีผลต่ออัตราค่าห้องพัก
- floor: ระบุชั้นที่ตั้งของห้อง ใช้ในการจัดกลุ่มและค้นหาห้อง

- capacity: ระบุความจุคนพักสูงสุดของห้อง
- max_cards: เป็นขีดจำกัดสูงสุดของจำนวนคีย์การ์ดที่ระบบอนุญาตให้ออกสำหรับห้องนี้ ใช้ในการควบคุมกับฟิลด์ cards_issued ในแฟ้ม stays.dat
- created at: บันทึกเวลาที่ห้องนี้ถูกเพิ่มเข้าสู่ระบบครั้งแรก
- updated_at: บันทึกเวลาล่าสุดที่มีการแก้ไขข้อมูลห้องพัก

2.1.2 แฟ้มข้อมูลผู้เข้าพัก (guests.dat)

แฟ้มข้อมูลนี้จัดเก็บข้อมูลส่วนบุคคลของลูกค้าที่เคยเข้าพักหรือลงทะเบียนไว้กับระบบ

ฟิลด์ (Field)	ชนิด (Type)	ขนาด (Bytes)	คำอธิบาย
guest_id	I	4	รหัสหลัก ใช้ระบุตัวตนของแขก
status	I	4	สถานะของแขก (Active/Deleted)
full name	50s	50	ชื่อ-สกุลของผู้เข้าพัก
phone	15s	15	เบอร์โทรศัพท์สำหรับติดต่อ
id_no	20s	20	เลขบัตรประชาชน หรือ Passport
max_cards	I	4	จำนวนคีย์การ์ดสูงสุดที่ออกให้ได้
created at	I	4	timestamp ที่สร้างข้อมูลแขก
updated_at	I	4	timestamp ล่าสุดที่มีการแก้ไขข้อมูล

ตารางที่ 2.2 แฟ้มข้อมูลผู้เข้าพัก

คำอธิบายฟิลด์:

- guest_id: เป็นรหัสหลัก (Primary Key) ใช้ในการอ้างอิงไปยังแฟ้ม stays.dat เพื่อเชื่อมโยงว่าแขกคนใดเป็นผู้เข้าพักในรายการใด
- status: ใช้ในการจัดการข้อมูลลูกค้าที่ไม่ได้ใช้งานแล้ว
- full name: ใช้ในการแสดงผลและยืนยันตัวตนของแขก โดยต้องมีการจัดการการเข้ารหัส (Encoding) ที่ถูกต้องเมื่อบันทึกลงในไฟล์ไบนารี
- phone: บันทึกเบอร์โทรศัพท์สำหรับติดต่อ
- id_no: บันทึกเลขบัตรประชาชน หรือ Passport ใช้สำหรับลงทะเบียน
- created at: บันทึกเวลาที่แขกถูกเพิ่มเข้าสู่ระบบ
- updated_at: บันทึกเวลาล่าสุดที่มีการแก้ไขข้อมูลแขก

2.1.3 แฟ้มข้อมูลการเข้าพัก (stays.dat)

แฟ้มข้อมูลนี้เป็นแฟ้มธุรกรรมที่บันทึกรายการเข้าพักที่กำลังดำเนินการ (Open) และที่สิ้นสุดแล้ว (Closed) โดยเป็นแกนหลักในการติดตามสถานะคีย์การ์ด

ฟิลด์ (Field)	ชนิด (Type)	ขนาด (Bytes)	คำอธิบาย
stay_id	I	4	รหัสหลัก ใช้ระบุธุรกรรมนี้โดยเฉพาะ
status	I	4	สถานะของแขก (1=Open, 0=Closed)
guest_id	I	4	รหัสอ้างอิงแขก
room id	I	4	รหัสอ้างอิงห้อง
checkin_date	10s	10	วันที่ Check-in (YYYY-MM-DD)
checkout date	10s	10	วันที่ Check-out (ถ้า Check-out แล้ว)
cards_issued	I	4	จำนวนคีย์การ์ดที่ออกให้ ณ ปัจจุบัน
cards returned	I	4	จำนวนคีย์การ์ดที่คืนแล้ว
updated_at	I	4	timestamp ล่าสุดของการทำรายการ

ตารางที่ 2.3 แฟ้มข้อมูลการเข้าพัก

คำอธิบายฟิลด์:

- stay_id: เป็น รหัสหลัก (Primary Key) ที่ใช้ระบุรายการธุรกรรมการเข้าพัก และเป็นรหัสที่ใช้เชื่อมโยงกับบันทึกเหตุการณ์ในแฟ้ม events.dat
- status: ใช้บ่งชี้ว่ารายการเข้าพักยังคงดำเนินการอยู่ (1=Open) หรือได้ทำการ Check-out แล้ว (0=Closed)
- checkin_date และ checkout date: ใช้ในการกำหนดช่วงเวลาการเข้าพักและคำนวณระยะเวลา
- cards_issued: มีการเพิ่มค่าเมื่อมีการออกคีย์การ์ดใหม่ และใช้ในการตรวจสอบว่าไม่เกิน max_cards ของห้องนั้น
- cards returned: มีการเพิ่มค่าเมื่อมีการคืนคีย์การ์ด ใช้ในการตรวจสอบความรับผิดชอบของแขก

2.1.4 แฟ้มข้อมูลบันทึกการใช้งาน (events.dat)

แฟ้มข้อมูลนี้ทำหน้าที่เป็น Log File หรือบันทึกเหตุการณ์ (Event Logging) ทุกครั้งที่มีการดำเนินการสำคัญในระบบ ใช้สำหรับตรวจสอบย้อนหลัง (Audit Trail)

ฟิลด์ (Field)	ชนิด (Type)	ขนาด (Bytes)	คำอธิบาย

ts	I	4	เวลาที่เกิดเหตุการณ์ (timestamp)
op_code	I	4	รหัสการดำเนินการ (เช่น CHECKIN, RETURN CARD
op_code	I	4	รหัสอ้างอิงรายการเข้าพักที่เกี่ยวข้อง
room_id	I	4	ห้องที่เกี่ยวข้องกับเหตุการณ์
guest_id	I	4	แขกที่เกี่ยวข้องกับเหตุการณ์
from date	10s	10	วันที่เริ่มต้น (สำหรับ CHECKIN)
to date	10s	10	วันที่สิ้นสุด (สำหรับ CHECKOUT)
status_after	I	4	สถานะของรายการหลังเกิดเหตุการณ์

ตารางที่ 2.4 แฟ้มข้อมูลบันทึกการใช้งาน

คำอธิบายฟิลด์:

- ts (timestamp): บันทึกเวลาที่แน่นอนที่เกิดกิจกรรม ใช้เพื่อเรียงลำดับเหตุการณ์อย่างแม่นยำ
- op_code: รหัสการดำเนินการ (Operation Code) ใช้ในการจำแนกประเภทของกิจกรรมที่เกิดขึ้นอย่างชัดเจน (เช่น 1=CHECKIN, 4=RETURN CARD, 5=CRUD)
- from date และ to date: ฟิลด์ที่ใช้ในการบันทึกช่วงวันที่ที่เกี่ยวข้องกับกิจกรรมนั้น ๆ (ส่วนใหญ่ใช้ในการทำรายการ Check-in และ Check-out)
- status_after: บันทึกสถานะล่าสุดของรายการเข้าพักหรือห้องหลังจากการดำเนินการนั้น เพื่อช่วยในการตรวจสอบความถูกต้องของข้อมูลในแฟ้ม stays.dat

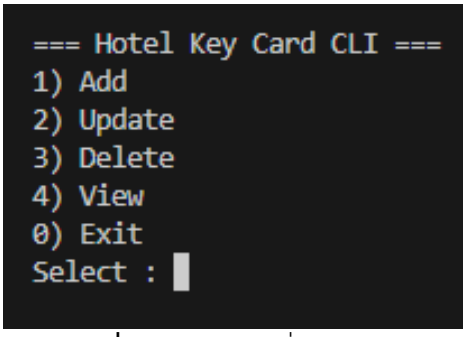
บทที่ 3 การใช้งานระบบยืม – คืนคีย์การ์ดในโรงแรม

โปรแกรมระบบยืม – คืนคีย์การ์ดในโรงแรมนี้ถูกพัฒนาขึ้นเพื่อช่วยให้ กระบวนการ Check-in และ Check-out รวมถึงการ ออกและรับคืนคีย์การ์ด สำหรับผู้เข้าพักเป็นไปอย่างสะดวก รวดเร็ว และเป็นระบบมากยิ่งขึ้น และยังช่วยให้โรงแรมสามารถ ติดตามสถานะของห้องพักและการเข้าพัก ได้อย่างชัดเจน โดยมีการจัดทำ รายงานสรุปเหตุการณ์ (Log File) ไปเก็บไว้ยังแฟ้มข้อมูลเพื่อใช้ในการตรวจสอบย้อนหลัง

โปรแกรมนี้สามารถทำงานได้อย่างครบวงจร โดยประกอบไปด้วยฟังก์ชันหลักที่สำคัญ ได้แก่ การเพิ่มข้อมูล สำหรับรายการหลักทั้งหมด ทั้งข้อมูลห้องพัก ข้อมูลผู้เข้าพัก และรายการการเข้าพัก, การแสดงข้อมูล ห้องพัก รายการเข้าพัก หรือข้อมูลผู้เข้าพักทั้งหมดในโปรแกรม, การค้นหาข้อมูลรายการเข้าพักหรือห้องพักที่ต้องการได้อย่างรวดเร็ว โดยใช้ room id หรือ guest id ในการค้นหา, การอัปเดตข้อมูล ที่ผู้ใช้สามารถเปลี่ยนแปลงข้อมูลได้ และถ้าไม่ต้องการแก้ไขข้อมูลบางส่วนสามารถกด Enter เพื่อข้ามการเปลี่ยนแปลงในฟิลด์นั้น ๆ ได้, การลบข้อมูล โดยการใช้ room id หรือ guest id ในการลบข้อมูลทั้งหมดของรหัสอ้างอิงนั้น, การทำรายการยืม – คืน โดยเฉพาะการบันทึกรายการ Check-in, Check-out, ออกคีย์การ์ด และ รับคืนคีย์การ์ด, การสร้างรายงาน เพื่อทำสรุป บันทึกเหตุการณ์ (Log File) การยืม – คืนคีย์การ์ดและการเข้าพักทั้งหมดในโปรแกรม และสุดท้ายคือ จบการทำงาน ของโปรแกรม

3.1 ภาพรวมหน้าเมนูหลัก

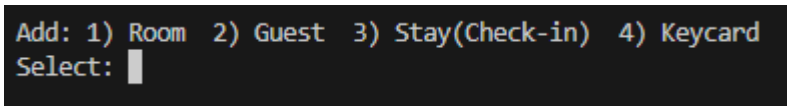
3.1.1 เมื่อรันคำสั่ง python hotel.py จะแสดงเมนูหลักดังนี้



รูปภาพที่ 3.1 หน้าต่างเริ่มต้นโปรแกรม

3.2 เพิ่มข้อมูล (เมนู Add)

3.2.1 จากเมนูหลัก กด 1 เพื่อเข้าสู่เมนู Add แล้วเลือกรายการต่อไปนี้



รูปภาพที่3.2 เลือกการใช้งานของเมนู Add

3.2.2 เพิ่มห้องพัก (Add Room)

```
Add: 1) Room  2) Guest  3) Stay(Check-in)  4) Keycard
Select: 1
```

รูปภาพที่3.3 หน้าต่างการเพิ่มห้องพัก

3.2.2.1 ระบบแสดงตารางห้องที่มีอยู่ เพื่อดูภาพรวมก่อนเพิ่ม

```
=== Existing Rooms ===
ID   | ประเภท  | ชั้น  | ความจุ  | จำนวนคีย์การ์ด  | สถานะ
-----------------------------------------------------------------
2    | DELUXE  | 5     | 3       | 3                | Occupied
3    | SUITE   | 10    | 4       | 4                | Vacant
4    | STD     | 1     | 2       | 2                | Vacant
5    | DELUXE  | 2     | 2       | 2                | Vacant
6    | STD     | 1     | 10      | 2                | Vacant
7    | STD     | 2     | 2       | 2                | Vacant
8    | DELUXE  | 5     | 3       | 3                | Occupied
9    | STD     | 2     | 2       | 2                | Occupied
10   | DELUXE  | 5     | 3       | 3                | Occupied
11   | STD     | 2     | 2       | 2                | Occupied
12   | DELUXE  | 5     | 3       | 3                | Occupied
13   | STD     | 2     | 2       | 2                | Occupied
14   | DELUXE  | 5     | 3       | 3                | Occupied

=== Add New Room ===
Room Type (STD/DELUXE/SUITE/..):
```

รูปภาพที่ 3.4 หน้าต่างตารางห้องที่มีอยู่

3.2.2.2 กรอกข้อมูล: Room Type (ตัวอย่าง: STD/DELUXE/SUITE), Floor, Capacity, Max keycards กด Enter ระบบจะบันทึกเรคคอร์ดใหม่ สถานะเริ่มต้น Vacant และอัปเดตเวลา created_at / updated_at

```
=== Add New Room ===
Room Type (STD/DELUXE/SUITE/..): STD
Floor: 2
Capacity: 1
Max keycards: 2
Room added: Room(room_id=15, status=1, room_type='STD', floor=2, capacity=1, max_cards=2, created_at=1759430915, updated_at=1759430915)
```

รูปภาพที่ 3.5 ตัวอย่างการเพิ่มห้องใหม่

3.2.3 เพิ่มผู้เข้าพัก (Add Guest)

```
Add: 1) Room  2) Guest  3) Stay(Check-in)  4) Keycard
Select: 2
```

รูปภาพที่ 3.6 เพิ่มผู้เข้าพัก

3.2.3.1 ระบบแสดงรายชื่อผู้เข้าพักปัจจุบัน กรอก Full name, Phone, ID/Passport บันทึกเรคคอร์ดใหม่ สถานะเริ่มต้น “Active”

```
=== Existing Guests ===
ID   | Full Name          | Phone       | ID Number      | Status
-------------------------------------------------------------------
1    | Thunyaluck Suksom  | 0818875555  | 15799          | Active
2    | Jane Doe           | 0899999999  | B9876543210    | Active
3    | Sirirod            | 0818837518  | 15799          | Active
4    | Sirirod Udomdach   | 0818870000  | 15899010       | Active
5    | John Smith         | 0812345678  | A1234567890    | Active
6    | Jane Doe           | 0899999999  | B9876543210    | Active
8    | Jane Doe           | 0899999999  | B9876543210    | Active
9    | John Smith         | 0812345678  | A1234567890    | Active
10   | Jane Doe           | 0899999999  | B9876543210    | Active
11   | John Smith         | 0812345678  | A1234567890    | Active
12   | Jane Doe           | 0899999999  | B9876543210    | Active
13   | test subject       | 081883      | 15799010       | Active

=== Add New Guest ===
Full name:
```

รูปภาพที่ 3.7 เพิ่มข้อมูลของผู้เข้าพัก

```
=== Add New Guest ===
Full name: test subject
Phone: 081883
ID/Passport: 15799010
Guest added: Guest(guest_id=13, status=1, full_name='test subject', phone='081883', id_no='15799010', created_at=1759431033, updated_at=1759431033)
```

รูปภาพที่ 3.8 ตัวอย่างการเพิ่มข้อมูลผู้เข้าพัก

3.2.4 เช็คอิน (Add Stay / Check-in)

3.2.4.1 ห้องต้องอยู่สถานะ Vacant, ผู้เข้าพัก Active, และจำนวนบัตรที่ขอไม่เกิน Max keycards ของห้องนั้น

```
Add: 1) Room  2) Guest  3) Stay(Check-in)  4) Keycard
Select: 1
```

รูปภาพที่ 3.9 เมนูเพิ่มรหัสห้องและรหัสผู้เข้าห้อง

3.2.4.2 เลือก Guest ID และ Room ID ระบุวันที่เช็คอิน YYYY-MM-DD (เว้นว่าง = วันนี้) ระบุ Cards to issue (1..Max keycards) ระบบจะสร้าง Stay (สถานะ Open), ออก Keycard อัตโนมัติ และเปลี่ยนห้องเป็น Occupied

```
=== Active Guests ===
ID   | Full Name          | Phone       | ID Number
-----------------------------------------------------
1    | Thunyaluck Suksom  | 0818875555  | 15799
2    | Jane Doe           | 0899999999  | B9876543210
3    | Sirirod            | 0818837518  | 15799
4    | Sirirod Udomdach   | 0818870000  | 15899010
5    | John Smith         | 0812345678  | A1234567890
6    | Jane Doe           | 0899999999  | B9876543210
7    | John Smith         | 0812345678  | A1234567890
8    | Jane Doe           | 0899999999  | B9876543210
9    | John Smith         | 0812345678  | A1234567890
10   | Jane Doe           | 0899999999  | B9876543210
11   | John Smith         | 0812345678  | A1234567890
12   | Jane Doe           | 0899999999  | B9876543210
13   | test subject       | 081883      | 15799010

=== Perform Check-in ===
Guest ID:
```

รูปภาพที่ 3.10 เมนูการ check-in

```
=== Perform Check-in ===
Guest ID: 12
Room ID: 4

Check-in Information:
Room: STD (Room 4) - Floor 1
Guest: Jane Doe
Maximum key cards: 2
Check-in date (YYYY-MM-DD) [today]:
Cards to issue (1-2) [1]: 1
Check-in successful: StayID=13
Room 4 status changed to 'Occupied'
Issued keycard serials: KC00401210
```

รูปภาพที่ 3.11 ตัวอย่างหลังจากเพิ่มรายการ check-in

3.2.5 เพิ่มคีย์การ์ด (Add Keycard)

```
Add: 1) Room  2) Guest  3) Stay(Check-in)  4) Keycard
Select: 4
```

รูปภาพที่ 3.12 เมนูเพิ่มคีย์การ์ด

3.2.5.1 ใช้เติมคีย์การ์ดเข้าสต๊อกของห้อง กรอก Room ID และ Serial ($\leq$10 ตัว)

```
=== Add New Keycard ===
ID   | Type     | Floor   | Capacity   | Max Cards   | Status
-------------------------------------------------------------
2    | DELUXE   | 5       | 3          | 3           | Occupied
3    | SUITE    | 10      | 4          | 4           | Vacant
4    | STD      | 1       | 2          | 2           | Occupied
5    | DELUXE   | 2       | 2          | 2           | Vacant
6    | STD      | 1       | 10         | 2           | Vacant
8    | DELUXE   | 5       | 3          | 3           | Occupied
9    | STD      | 2       | 2          | 2           | Occupied
10   | DELUXE   | 5       | 3          | 3           | Vacant
11   | STD      | 2       | 2          | 2           | Occupied
12   | DELUXE   | 5       | 3          | 3           | Occupied
13   | STD      | 2       | 2          | 2           | Occupied
14   | DELUXE   | 5       | 3          | 3           | Occupied
15   | STD      | 2       | 1          | 2           | Vacant
Room ID:
```

รูปภาพที่ 3.13 เพิ่มคีย์การ์ดของแต่ละห้อง

```
Room ID: 2
Serial Number: KM-99
Keycard added: Keycard(keycard_id=18, status=1, room_id=2, serial='KM-99', created_at=1759431087, updated_at=1759431087)
```

รูปภาพที่ 3.14 ตัวอย่างหลักจากเพิ่มข้อมูลของห้อง

3.3 แก้ไข/ปรับปรุง (เมนู Update)

```
=== Hotel Key Card CLI ===
1) Add
2) Update
3) Delete
4) View
0) Exit
Select : 2
```

รูปภาพที่ 3.15 หน้าต่างการแก้ไข ปรับปรุง

3.3.1 แก้ไขห้องพัก (Edit Room)

จากเมนูหลัก กด 2 เพื่อเข้าสู่เมนู Update

```
Update: 1) Room  2) Guest  3) Stay(Check-out)  4) Keycard
Select: 1
```

รูปภาพที่ 3.16 เมนูแก้ไขห้องพัก

3.3.2 แสดงตารางห้องทั้งหมด เลือก Room ID ที่ต้องการแก้ไข ป้อนค่าที่ต้องการ (เว้นว่างเพื่อคงค่าเดิม) ระบบอัปเดตข้อมูลและเวลา updated_at

```
All Rooms:
ID | Type | Floor | Capacity | Max Cards | Status
------------------------------------------------------------------------
2 | DELUXE | 5 | 3 | 3 | Occupied
3 | SUITE | 10 | 4 | 4 | Vacant
4 | STD | 1 | 2 | 2 | Occupied
5 | DELUXE | 2 | 2 | 2 | Vacant
6 | STD | 1 | 10 | 2 | Vacant
7 | STD | 2 | 2 | 2 | Vacant
8 | DELUXE | 5 | 3 | 3 | Occupied
9 | STD | 2 | 2 | 2 | Occupied
10 | DELUXE | 5 | 3 | 3 | Occupied
11 | STD | 2 | 2 | 2 | Occupied
12 | DELUXE | 5 | 3 | 3 | Occupied
13 | STD | 2 | 2 | 2 | Occupied
14 | DELUXE | 5 | 3 | 3 | Occupied
15 | STD | 2 | 1 | 2 | Vacant
------------------------------------------------------------------------

Select Room ID to edit:
```

รูปภาพที่ 3.17 แสดงตารางห้องทั้งหมด

```
Select Room ID to edit: 7

Current Information:
Room Type: STD
Floor: 2
Capacity: 2
Max Key Cards: 2

Enter new information (leave blank to keep current):
Room Type: DELUXE
Floor: 7
Capacity: 2
Max Key Cards: 2

Data updated successfully
New information: Room(room_id=7, status=1, room_type='DELUXE', floor=7, capacity=2, max_cards=2, created_at=1759375532, updated_at=1759431287)
```

รูปภาพที่ 3.18 ตัวอย่างหลังจากแก้ไขข้อมูลห้อง

3.3.2 แก้ไขผู้เข้าพัก (Edit Guest)

```
Update: 1) Room  2) Guest  3) Stay(Check-out)  4) Keycard
Select: 2
```

รูปภาพที่ 3.19 เมนูแก้ไขผู้เข้าพัก

3.3.3 แสดงตารางแขกทั้งหมด เลือก Guest ID ที่ต้องการแก้ไข ป้อนค่าที่ต้องการ (เว้นว่างเพื่อคงค่าเดิม) ระบบอัปเดตข้อมูลและเวลา updated_at

```
All Guests:
ID | Full Name | Phone | ID/Passport
--------------------------------------------------------------------
1 | Thunyaluck Suksom | 0818875555 | 15799
2 | Jane Doe | 0899999999 | B9876543210
3 | Sirirod | 0818837518 | 15799
4 | Sirirod Udomdach | 0818870000 | 15899010
5 | John Smith | 0812345678 | A1234567890
6 | Jane Doe | 0899999999 | B9876543210
7 | John Smith | 0812345678 | A1234567890
8 | Jane Doe | 0899999999 | B9876543210
9 | John Smith | 0812345678 | A1234567890
10 | Jane Doe | 0899999999 | B9876543210
11 | John Smith | 0812345678 | A1234567890
12 | Jane Doe | 0899999999 | B9876543210
13 | test subject | 081883 | 15799010
--------------------------------------------------------------------

Select Guest ID to edit:
```

รูปภาพที่ 3.20 ตารางแสดงข้อมูลของผู้เข้าพักทั้งหมด

```
Select Guest ID to edit: 7

Current Information:
Full Name: John Smith
Phone: 0812345678
ID/Passport: A1234567890

Enter new information (leave blank to keep current):
Full Name: RodRod
Phone: 087777
ID/Passport: 124556

Data updated successfully
New information: Guest(guest_id=7, status=1, full_name='RodRod', phone='087777', id_no='124556', created_at=1759375548, updated_at=1759431370)
```

รูปภาพที่ 3.21 ตัวอย่างหลังจากแก้ไขข้อมูลของผู้เข้าพัก

3.3.3 เช็คเอาต์ (Stay Check-out)

```
Update: 1) Room  2) Guest  3) Stay(Check-out)  4) Keycard
Select: 3
```

รูปภาพที่ 3.22 เมนูแก้ไขของการเช็คเอาต์

3.3.4 เลือก Stay ID เพื่อทำการ Check-out (Stay Check-out)

```
Stays not yet checked out:
Stay ID | Room | Guest | Phone | ID Number | Keycard Serials | Check-in Date
--------------------------------------------------------------------------------------------------------------
6 | STD (Room 9) | Unknown | N/A | N/A | KC00900755, KC00900755 | 2024-01-15
8 | STD (Room 11) | John Smith | 0812345678 | A1234567890 | KC01100962, KC01100962 | 2024-01-15
9 | DELUXE (Room 12) | Jane Doe | 0899999999 | B9876543210 | KC01201062 | 2024-01-16
10 | STD (Room 13) | John Smith | 0812345678 | A1234567890 | KC01301163, KC01301163 | 2024-01-15
11 | DELUXE (Room 14) | Jane Doe | 0899999999 | B9876543210 | KC01401263 | 2024-01-16
13 | STD (Room 4) | Jane Doe | 0899999999 | B9876543210 | KC00401210 | 2025-10-03
--------------------------------------------------------------------------------------------------------------

Select Stay ID for check-out:
```

รูปภาพที่ 3.23 ตารางแสดงข้อมูลของผู้เช็คเอาต์ทั้งหมด

```
Select Guest ID to edit: 7

Current Information:
Full Name: John Smith
Phone: 0812345678
ID/Passport: A1234567890

Enter new information (leave blank to keep current):
Full Name: RodRod
Phone: 087777
ID/Passport: 124556

Data updated successfully
New information: Guest(guest_id=7, status=1, full_name='RodRod', phone='087777', id_no='124556', created_at=1759375548, updated_at=1759431370)
```

รูปภาพที่ 3.24 ตัวอย่างการเช็คเอาต์สำเร็จ

3.3.4 แก้ไขคีย์การ์ด (Edit Keycard)

```
Update: 1) Room  2) Guest  3) Stay(Check-out)  4) Keycard
Select: 4
```

รูปภาพที่ 3.25 เมนูแก้ไขคีย์การ์ด

3.3.5 แสดงรายชื่อคีย์การ์ดทั้งหมด ⟶ เลือก Keycard ID ⟶ เปลี่ยน Room ID/Serial ตามต้องการ

```
=== Update Keycard ===
Keycard ID   | Room ID   | Serial       | Status
-----------------------------------------------
1            | 2         | K-2-5        | Active
4            | 8         | KC00800655   | Active
5            | 9         | KC00900755   | Active
6            | 9         | KC00900755   | Active
8            | 11        | KC01100962   | Active
9            | 11        | KC01100962   | Active
10           | 12        | KC01201062   | Active
11           | 13        | KC01301163   | Active
12           | 13        | KC01301163   | Active
13           | 14        | KC01401263   | Active
17           | 4         | KC00401210   | Active
18           | 2         | KM-99        | Active
Keycard ID to update: 18
```

รูปภาพที่ 3.26 หน้าต่างแก้ไข คีย์การ์ด

```
Keycard ID to update: 18

Current Information:
Room ID: 2
Serial: KM-99

Enter new information (leave blank to keep current):
New Room ID: 9
New Serial: KM-100
Keycard updated successfully
```

รูปภาพที่ 3.27 ตัวอย่างการแก้ใข คีย์การ์ด

3.4 ลบข้อมูล (เมนู Delete) — Soft delete

3.4.1 จากเมนูหลัก กด 3 เพื่อเข้าสู่การลบแบบเชิงตรรกะ (ไม่ลบจริงในไฟล์ สามารถเลือกหัวข้อที่ต้องการลบผ่านการเลือก 1) Room, 2) Guest, 3) Stay, 4) Keycard หลังจากนั้นให้ใส่เลขอ้างอิง

```
Delete (soft): 1) Room  2) Guest  3) Stay  4) Keycard
Select:
```

รูปภาพที่ 3.28 หน้าต่างหลักส่วน ลบ

```
Delete (soft): 1) Room  2) Guest  3) Stay  4) Keycard
Select: 1
```

รูปภาพที่ 3.29 หน้าต่างลบข้อมูลห้อง

```
Delete (soft): 1) Room  2) Guest  3) Stay  4) Keycard
Select: 1
Room ID: 7
Deleted
```

รูปภาพที่ 3.30 ตัวอย่างการลบ ห้อง

```
Delete (soft): 1) Room  2) Guest  3) Stay  4) Keycard
Select: 2
Guest ID:
```

รูปภาพที่ 3.31 หน้าต่างลบผู้ใช้

```
Delete (soft): 1) Room  2) Guest  3) Stay  4) Keycard
Select: 2
Guest ID: 7
Deleted
```

รูปภาพที่ 3.32 ตัวอย่างการลบ ผู้ใช้

```
Delete (soft): 1) Room  2) Guest  3) Stay  4) Keycard
Select: 3
Stay ID: 5
Deleted
```

รูปภาพที่ 3.33 ตัวอย่างการลบเช็คอิน

```
All Keycards:
Keycard ID   | Room ID   | Serial       | Status
-----------------------------------------------
1            | 2         | K-2-5        | Active
4            | 8         | KC00800655   | Active
5            | 9         | KC00900755   | Active
6            | 9         | KC00900755   | Active
8            | 11        | KC01100962   | Active
9            | 11        | KC01100962   | Active
10           | 12        | KC01201062   | Active
11           | 13        | KC01301163   | Active
12           | 13        | KC01301163   | Active
13           | 14        | KC01401263   | Active
17           | 4         | KC00401210   | Active
18           | 9         | KM-100       | Active
Keycard ID to delete: 18
Deleted
```

รูปภาพที่ 3.34 ตัวอย่างการลบ คีย์การ์ด

3.5 ดูข้อมูล/สรุป/รายงาน (เมนู View)

จากเมนูหลัก กด 4 เพื่อดูข้อมูลได้หลายรูปแบบ:

```
View:
  1) View Single Record
  2) View All Records
  3) View Filtered Records
  4) Summary Statistics + Export Report
  5) View Keycards

Select:
```

รูปภาพที่ 3.35 หน้าต่างแสดงข้อมูล

3.5.1 View Single Record : แสดงทีละรายการ (Room/Guest/Stay)

```
View:
  1) View Single Record
  2) View All Records
  3) View Filtered Records
  4) Summary Statistics + Export Report
  5) View Keycards

Select: 1
Select: 1) Room  2) Guest  3) Stay :
```

รูปภาพที่ 3.36 หน้าต่างเลือกแสดงข้อมูลแบบแถวเดียว

```
All Rooms:
ID   | Type     | Floor   | Capacity   | Max Cards   | Status
-------------------------------------------------------------
2    | DELUXE   | 5       | 3          | 3           | Occupied
3    | SUITE    | 10      | 4          | 4           | Vacant
4    | STD      | 1       | 2          | 2           | Occupied
5    | DELUXE   | 2       | 2          | 2           | Vacant
6    | STD      | 1       | 10         | 2           | Vacant
8    | DELUXE   | 5       | 3          | 3           | Occupied
9    | STD      | 2       | 2          | 2           | Occupied
10   | DELUXE   | 5       | 3          | 3           | Vacant
11   | STD      | 2       | 2          | 2           | Occupied
12   | DELUXE   | 5       | 3          | 3           | Occupied
13   | STD      | 2       | 2          | 2           | Occupied
14   | DELUXE   | 5       | 3          | 3           | Occupied
15   | STD      | 2       | 1          | 2           | Vacant

Select Room ID to view: 1

Selected Room Information:
ID   | Type   | Floor   | Capacity   | Max Cards   | Status
-------------------------------------------------------------
1    | STD    | 2       | 2          | 2           | Deleted
```

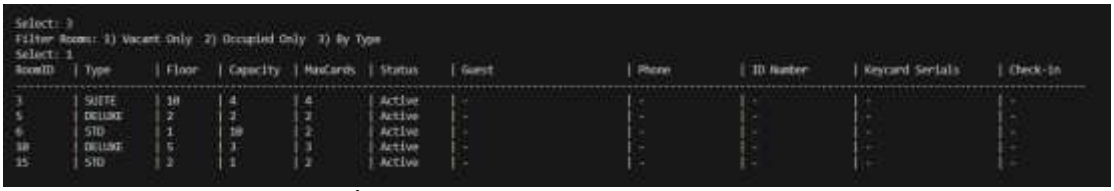
รูปภาพที่ 3.37 ตัวอย่างหน้าแสดงแบบเจาะจง

3.5.2 View All Records : แสดงทั้งหมดของชนิดที่เลือก

```
Select: 2
Select: 1) Rooms  2) Guests  3) Stays : 2

All Guests:
ID   | Full Name          | Phone       | ID Number     | Status
-------------------------------------------------------------------
1    | Thunyaluck Suksom  | 0818875555  | 15799         | Active
2    | Jane Doe           | 0899999999  | B9876543210   | Active
3    | Sirirod            | 0818837518  | 15799         | Active
4    | Sirirod Udomdach   | 0818870000  | 15899010      | Active
5    | John Smith         | 0812345678  | A1234567890   | Active
6    | Jane Doe           | 0899999999  | B9876543210   | Active
8    | Jane Doe           | 0899999999  | B9876543210   | Active
9    | John Smith         | 0812345678  | A1234567890   | Active
10   | Jane Doe           | 0899999999  | B9876543210   | Active
11   | John Smith         | 0812345678  | A1234567890   | Active
12   | Jane Doe           | 0899999999  | B9876543210   | Active
13   | test subject       | 081883      | 15799010      | Active
```

รูปภาพที่ 3.38 ตัวอย่างหน้าแสดงผล ทั้งหมด

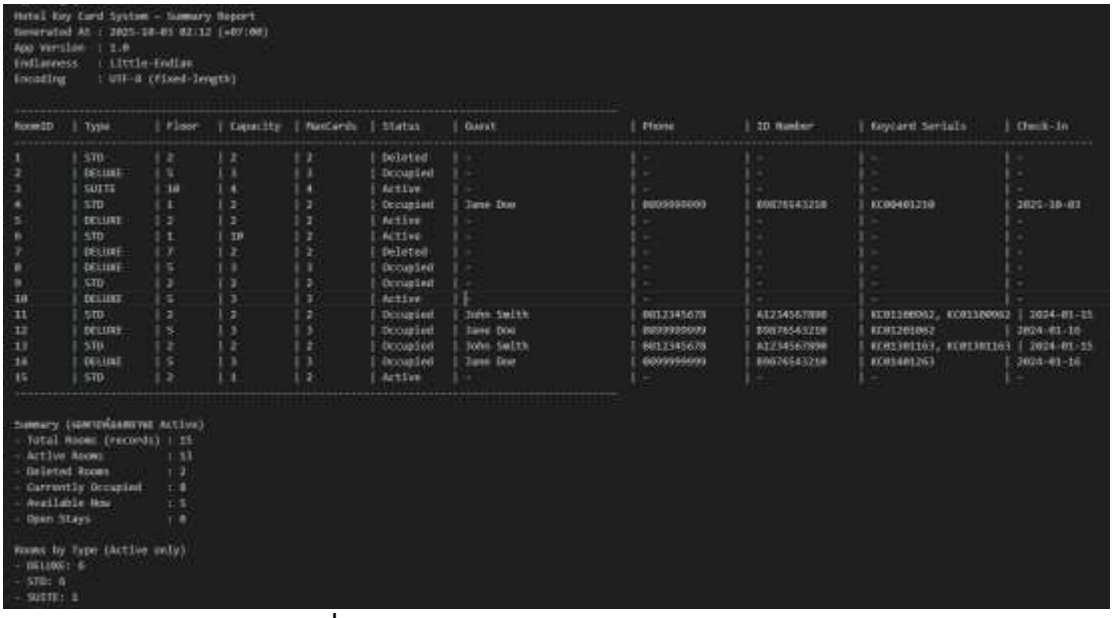
3.5.3 View Filtered Records (Rooms) : กรองแบบ Vacant Only / Occupied Only / By Type



รูปภาพที่ 3.39 ตัวอย่างหน้าต่างแสดงผลแบบคัดกรอง

3.5.4 Summary Statistics + Export Report : รวมตารางห้อง + สถิติ + Rooms by Type และบันทึกไฟล์ hotel_report.txt

รูปภาพที่ 3.40 หน้าต่างแสดงผลรายงาน (report)



รูปภาพที่ 3.41 ตัวอย่างหน้าต่างแสดงผลรายงาน (report)

3.5.5 View Keycards : ทั้งหมด / ตามห้อง / ตามสถานะ

```
Keycard View: 1) All Keycards  2) By Room  3) By Status
Select: 1

All Keycards:
Keycard ID   | Room ID   | Serial       | Status   | Created
-------------------------------------------------------------------
1            | 2         | K-2-5        | Active   | 2025-10-02 10:18
4            | 8         | KC00800655   | Active   | 2025-10-02 10:25
5            | 9         | KC00900755   | Active   | 2025-10-02 10:25
6            | 9         | KC00900755   | Active   | 2025-10-02 10:25
8            | 11        | KC01100962   | Active   | 2025-10-02 10:37
9            | 11        | KC01100962   | Active   | 2025-10-02 10:37
10           | 12        | KC01201062   | Active   | 2025-10-02 10:37
11           | 13        | KC01301163   | Active   | 2025-10-02 10:38
12           | 13        | KC01301163   | Active   | 2025-10-02 10:38
13           | 14        | KC01401263   | Active   | 2025-10-02 10:38
17           | 4         | KC00401210   | Active   | 2025-10-03 01:51
```

รูปภาพที่ 3.42 ตัวอย่างการแสดงผล คีย์การ์ดแบบทั้งหมด

บทที่ 4 อธิบายการทำงานของโค้ด

4.1 ฟังก์ชัน/โมดูลพื้นฐานในระบบยืม–คืนคีย์การ์ด

4.1.1 โมดูล struct (แปลงข้อมูลเป็นไบนารีคงที่)

struct ใช้กำหนด รูปแบบเรคคอร์ดแบบความยาวคงที่ (fixed-length binary record) และแปลงข้อมูลระหว่างชนิด Python ↔ ไบนารีบนไฟล์ โดยใช้ endianness แบบ little-endian (สัญลักษณ์ < ในฟอร์แมตสตริง) เพื่อให้ ทุกเรคคอร์ดมีขนาดเท่ากันบนดิสก์ ⟶ คำนวณตำแหน่งอ่าน/เขียนได้แม่น สตริงเข้ารหัส UTF-8 ตัด/เติมด้วย \x00 ให้ได้ความยาวตามที่กำหนด

```
import struct
```

รูปภาพที่ 4.1 Module struct

4.1.2 โมดูล time

ใช้สำหรับอ่านเวลาปัจจุบันเป็น Unix timestamp (int) เพื่อบันทึกฟิลด์ created_at และ updated_at ในทุกเรคคอร์ด

```
import time
```

รูปภาพที่ 4.2 Module time

4.1.3 คลาส datetime (เสริม)

ใช้ จัดรูปแบบเวลา จาก timestamp เป็นสตริง YYYY-MM-DD HH:MM สำหรับแสดงผลในรายงาน/ตาราง CLI รวมถึงการเก็บวันเช็คอิน/เอาต์ในฟิลด์สตริง YYYY-MM-DD ของเรคคอร์ดการเข้าพัก (Stay)

```
import datetime
```

รูปภาพที่ 4.3 Module datetime

4.1.4 โมดูล OS (การตรวจสอบและจัดการไฟล์) ตรวจสอบว่ามีไฟล์ .dat อยู่แล้วหรือไม่ เช่น os.path.exists("members.dat") (เสริม)

4.1.5 โมดูล typing (Type Hints) กำหนดชนิดข้อมูลของฟังก์ชัน เช่น def read_all_members() -> List[Dict]: (เสริม)

4.1.6 ฟีเจอร์ __future__.annotations ทำให้ type hint ในไฟล์ถูกประเมินเป็น string ⟶ สามารถอ้างอิง class หรือฟังก์ชันที่ประกาศภายหลังได้ (เสริม)

4.2 ฟังก์ชันอรรถประโยชน์ (Utilities)

4.2.1 now_ts()

หน้าที่: คืนค่าเวลาปัจจุบันในรูปแบบ Unix timestamp (ชนิด int) สำหรับประทับเวลา created_at/updated_at ของทุกรายการในระบบ

```
def now_ts() -> int:
    return int(time.time())
```

รูปภาพที่ 4.4 ฟังก์ชัน now_ts

4.2.2 fmt_date(ts: int) -> str

หน้าที่: แปลง timestamp เป็นสตริงวันที่-เวลา "YYYY-MM-DD HH:MM" เพื่อแสดงผลรายงาน/ตารางใน CLIตัวอย่างการใช้: แสดงคอลัมน์ “Created” ในหน้าดูคีย์การ์

```
def fmt_date(ts: int) -> str:
    return datetime.fromtimestamp(ts).strftime("%Y-%m-%d %H:%M")
```

รูปภาพที่ 4.5 ฟังก์ชัน fmt_date

4.2.3 fix_bytes(s: str, size: int)

fix_bytes แปลงสตริงเป็นไบต์แบบ UTF-8, ตัดความยาวเกินที่กำหนด และ เติม \x00 จนครบขนาด เพื่อให้ทุกฟิลด์บนดิสก์เป็น ความยาวคงที่

```
def fix_bytes(s: str, size: int) -> bytes:
    b = s.encode("utf-8", errors="ignore")[:size]
    return b.ljust(size, b"\x00")
```

รูปภาพที่ 4.6 ฟังก์ชัน fix_bytes

4.2.4 read_str(b: bytes) -> str ตัด \x00 ด้านขวาแล้วถอดรหัสกลับเป็นสตริง

ทำให้เรคคอร์ดบนไฟล์เป็น fixed-length ใช้ร่วมกับ struct.Struct เพื่อคำนวณ offset อ่าน/เขียนแม่นยำ (แนวคิดเดียวกับเอกสารตัวอย่างที่บังคับความยาวแล้ว pack/unpack)

```
def read_str(b: bytes) -> str:
```

รูปภาพที่ 4.7 ฟังก์ชัน read_str

4.3 โครงสร้างเรคคอร์ด (Struct Layouts) และค่าคงที่สถานะ

กำหนดรูปแบบเรคคอร์ดไบนารีสำหรับ Room/Guest/Stay/Keycard โดยใช้ endianness แบบ little-endian ("<")กำหนด ขนาดเรคคอร์ด (เช่น ROOM_SIZE=64) เพื่อรองรับการ seek ตำแหน่งที่แน่นอน

```
# ---------------------------- Record Layouts (struct) -----------------------

# Room record (64 bytes)
# <II20sIIIII -> 4+4+20+4+4+4+4+4 = 48  (pad to 64 on disk)
ROOM_STRUCT = struct.Struct("<II20sIIIII")
ROOM_SIZE = 64

# Guest record (112 bytes)
# <II50s15s20sII -> 4+4+50+15+20+4+4 = 101 (pad 112)
GUEST_STRUCT = struct.Struct("<II50s15s20sII")
GUEST_SIZE = 112

# Stay record (64 bytes)
# <IIII10s10sIII -> 4+4+4+4+10+10+4+4+4 = 44 (pad 64)
STAY_STRUCT = struct.Struct("<IIII10s10sIII")
STAY_SIZE = 64

# Keycard record (32 bytes)
# <III10sII -> 4+4+4+10+4+4 = 30 (pad 32)
KEYCARD_STRUCT = struct.Struct("<III10sII")
KEYCARD_SIZE = 32

# Status constants
ROOM_DELETED = 0
ROOM_ACTIVE_VACANT = 1
ROOM_ACTIVE_OCCUPIED = 2

GUEST_DELETED = 0
GUEST_ACTIVE = 1

STAY_DELETED = 9
STAY_OPEN = 1
STAY_CLOSED = 0

KEYCARD_DELETED = 0
KEYCARD_ACTIVE = 1
```

รูปภาพที่ 4.8 โครงสร้างเรคคอร์ดต่างๆ (struct)

4.4 คลาสข้อมูลและเมธอดแพ็ก/อันแพ็ก

4.4.1 @dataclass Room + pack()/unpack()

แทนข้อมูลห้องและวิธีแปลงไป-กลับระหว่างอ็อบเจ็กต์ ↔ ไบนารี ด้วย ROOM_STRUCTฟิลด์สตริง (room_type) แปลงด้วย fix_bytes(..., 20)pack() เติมแพดดิ้งให้ครบ ROOM_SIZEunpack() แปลงไบต์กลับเป็นชนิดจริง (ตัด \x00 ที่ปลาย)

```python
@dataclass
class Room:
    room_id: int
    status: int
    room_type: str
    floor: int
    capacity: int
    max_cards: int
    created_at: int
    updated_at: int

    def pack(self) -> bytes:
        raw = ROOM_STRUCT.pack(
            self.room_id,
            self.status,
            fix_bytes(self.room_type, 20),
            self.floor,
            self.capacity,
            self.max_cards,
            self.created_at,
            self.updated_at,
        )
        return raw.ljust(ROOM_SIZE, b"\x00")

    @staticmethod
    def unpack(buf: bytes) -> "Room":
        t = ROOM_STRUCT.unpack(buf[:ROOM_STRUCT.size])
        return Room(
            room_id=t[0],
            status=t[1],
            room_type=read_str(t[2]),
            floor=t[3],
            capacity=t[4],
            max_cards=t[5],
            created_at=t[6],
            updated_at=t[7],
        )
```

รูปภาพที่ 4.9 @dataclass ของคลาส Room

4.3.2 @dataclass Guest + pack()/unpack()

เหมือน Room แต่มีฟิลด์สตริงหลายช่อง (full_name/phone/id_no) ที่ต้อง บังคับความยาว (50/15/20) ก่อน pack; ใช้ GUEST_STRUCT และขนาด GUEST_SIZE

```python
@dataclass
class Guest:
    guest_id: int
    status: int
    full_name: str
    phone: str
    id_no: str
    created_at: int
    updated_at: int

    def pack(self) -> bytes:
        raw = GUEST_STRUCT.pack(
            self.guest_id,
            self.status,
            fix_bytes(self.full_name, 50),
            fix_bytes(self.phone, 15),
            fix_bytes(self.id_no, 20),
            self.created_at,
            self.updated_at,
        )
        return raw.ljust(GUEST_SIZE, b"\x00")

    @staticmethod
    def unpack(buf: bytes) -> "Guest":
        t = GUEST_STRUCT.unpack(buf[:GUEST_STRUCT.size])
        return Guest(
            guest_id=t[0],
            status=t[1],
            full_name=read_str(t[2]),
            phone=read_str(t[3]),
            id_no=read_str(t[4]),
            created_at=t[5],
            updated_at=t[6],
        )
```

รูปภาพที่ 4.10 @dataclass ของคลาส Guest

4.3.3 @dataclass Stay + pack()/unpack()

บันทึกเหตุการณ์เข้าพัก (Guest x Room) พร้อมจำนวนบัตรที่ออก/คืน และวันที่ check-in/out (เก็บเป็นสตริงความยาว 10 ไบต์) ใช้ STAY_STRUCT และ STAY_SIZE, วันที่ใช้ fix_bytes(...,10)

```python
@dataclass
class Stay:
    stay_id: int
    status: int
    guest_id: int
    room_id: int
    checkin_date: str  # 'YYYY-MM-DD'
    checkout_date: str # 'YYYY-MM-DD' or empty
    cards_issued: int
    cards_returned: int
    updated_at: int

    def pack(self) -> bytes:
        raw = STAY_STRUCT.pack(
            self.stay_id,
            self.status,
            self.guest_id,
            self.room_id,
            fix_bytes(self.checkin_date, 10),
            fix_bytes(self.checkout_date, 10),
            self.cards_issued,
            self.cards_returned,
            self.updated_at,
        )
        return raw.ljust(STAY_SIZE, b"\x00")

    @staticmethod
    def unpack(buf: bytes) -> "Stay":
        t = STAY_STRUCT.unpack(buf[:STAY_STRUCT.size])
        return Stay(
            stay_id=t[0],
            status=t[1],
            guest_id=t[2],
            room_id=t[3],
            checkin_date=read_str(t[4]),
            checkout_date=read_str(t[5]),
            cards_issued=t[6],
            cards_returned=t[7],
            updated_at=t[8],
        )
```

รูปภาพที่ 4.11 @dataclass ของคลาส Stay

4.3.4 @dataclass Keycard + pack()/unpack()

บัญชีคีย์การ์ดที่ออกให้ห้องหนึ่ง ๆ พร้อมหมายเลข serial, สถานะ และเวลา created_at/updated_at ใช้ KEYCARD_STRUCT (<III10sII>) และ KEYCARD_SIZE=32; serial บังคับความยาว 10 ไบต์

```python
@dataclass
class Keycard:
    keycard_id: int
    status: int
    room_id: int
    serial: str
    created_at: int
    updated_at: int

    def pack(self) -> bytes:
        raw = KEYCARD_STRUCT.pack(
            self.keycard_id,
            self.status,
            self.room_id,
            fix_bytes(self.serial, 10),
            self.created_at,
            self.updated_at,
        )
        return raw.ljust(KEYCARD_SIZE, b"\x00")

    @staticmethod
    def unpack(buf: bytes) -> "Keycard":
        t = KEYCARD_STRUCT.unpack(buf[:KEYCARD_STRUCT.size])
        return Keycard(
            keycard_id=t[0],
            status=t[1],
            room_id=t[2],
            serial=read_str(t[3]),
            created_at=t[4],
            updated_at=t[5],
        )
```

รูปภาพที่ 4.12 @dataclass ของคลาส Keycard

4.4 ชั้นจัดการไฟล์ไบนารีคงที่ (Binary Stores)

4.4.1 __len__()

นับเรคคอร์ด = ขนาดไฟล์ / ขนาดเรคคอร์ด

```python
def __len__(self) -> int:
    return os.path.getsize(self.path) // self.size
```

รูปภาพที่ 4.13 ฟังก์ชัน __len__

4.4.2 _read_at(index)

ฟังก์ชัน **_read_at** เป็นฟังก์ชันที่ใช้สำหรับ **อ่านข้อมูลไบนารีหนึ่งระเบียน** จากไฟล์ที่ถูกจัดเก็บในรูปแบบขนาดคงที่: มันจะเปิดไฟล์ในโหมดอ่านไบนารี ("rb") จากนั้นใช้ f.seek เพื่อย้ายตัวชี้ไฟล์ไปยังตำแหน่งที่คำนวณจาก **index** ที่ระบุคูณด้วยขนาดของระเบียน (self.size) และใช้ f.read เพื่ออ่านข้อมูลขนาดเท่ากับ self.size ออกมาสุดท้ายจะคืนค่าข้อมูลไบนารีที่อ่านได้ถ้าขนาดถูกต้อง มิฉะนั้นจะคืนค่า None

```python
def _read_at(self, index: int) -> Optional[bytes]:
    with open(self.path, "rb") as f:
        f.seek(index * self.size)
        b = f.read(self.size)
        return b if len(b) == self.size else None
```

รูปภาพที่ 4.14 ฟังก์ชัน _read_at

4.4.3 _write_at(index, data)

ฟังก์ชัน `_write_at` ใช้สำหรับเขียนทับข้อมูลไบนารีหนึ่งระเบียนในไฟล์ตามตำแหน่งที่ระบุ: มันจะตรวจสอบก่อนว่าขนาดของ `data` ที่รับเข้ามาเท่ากับขนาดระเบียน (`self.size`) จากนั้นเปิดไฟล์ในโหมดอ่าน/เขียนไบนารี (`"r+b"`) ใช้ `f.seek` เพื่อย้ายตัวชี้ไฟล์ไปยังตำแหน่งของ `index` ที่ต้องการ ก่อนจะใช้ `f.write` เพื่อเขียนข้อมูลทับลงไป และสุดท้ายเรียกใช้ `f.flush()` และ `os.fsync()` เพื่อบังคับให้ข้อมูลถูกบันทึกสู่ดิสก์ทันที เพื่อรับประกันความสมบูรณ์ของข้อมูล

```python
def _write_at(self, index: int, data: bytes) -> None:
    assert len(data) == self.size
    with open(self.path, "r+b") as f:
        f.seek(index * self.size)
        f.write(data)
        f.flush()
        os.fsync(f.fileno())
```

รูปภาพที่ 4.15 ฟังก์ชัน _write_at

4.4.4 append(obj)

ฟังก์ชัน `append` ใช้สำหรับเพิ่มวัตถุใหม่ต่อท้ายแหล่งเก็บข้อมูลไบนารี: มันจะคำนวณดัชนี (`idx`) ใหม่ที่จะบันทึกโดยใช้ความยาวปัจจุบันของแหล่งเก็บข้อมูล (`len(self)`), จากนั้นแปลงวัตถุ (`obj`) ให้เป็นข้อมูลไบนารีโดยใช้เมธอด `obj.pack()` และเรียกใช้ `self._write_at` เพื่อเขียนข้อมูลนั้นลงในไฟล์ที่ตำแหน่งดัชนีใหม่ สุดท้ายฟังก์ชันจะคืนค่าดัชนี (`idx`) ของข้อมูลที่เพิ่งถูกเพิ่มเข้าไป

```
def append(self, obj) -> int:
    idx = len(self)
    self._write_at(idx, obj.pack())
    return idx
```

รูปภาพที่ 4.16 **ฟังก์ชัน** append

4.4.5 update(index, obj)

เขียนทับเรคคอร์ดเดิม

```
def update(self, index: int, obj) -> None:
    self._write_at(index, obj.pack())
```

รูปภาพที่ 4.17 ฟังก์ชัน update

4.4.6 ter()

ไอเทอเรตทุกเรคคอร์ด ⟶ คืน (index, object) ที่แปลงแล้ว

```
def iter(self) -> Iterable[Tuple[int, object]]:
    total = len(self)
    for i in range(total):
        b = self._read_at(i)
        if not b:
            continue
        yield i, self.cls.unpack(b)
```

รูปภาพที่ 4.18 ฟังก์ชัน iter

4.4.7 find_first(keyfn)

ฟังก์ชัน find_first เป็นฟังก์ชันอำนวยความสะดวกที่ใช้สำหรับค้นหาและคืนค่าระเบียนแรกที่ตรงตามเงื่อนไข: มันจะวนซ้ำผ่านระเบียนทั้งหมดในแหล่งเก็บข้อมูล (self.iter()) และใช้ฟังก์ชันเงื่อนไข (keyfn) ที่รับเข้ามาเพื่อตรวจสอบแต่ละระเบียน (rec) หากเงื่อนไขเป็นจริง (คืนค่า True) จะคืนค่า ดัชนี (i) และตัววัตถุระเบียน (rec) นั้นทันที ถ้าวนซ้ำจนครบแล้วไม่พบระเบียนใดตรงตามเงื่อนไข จะคืนค่า None

```python
# convenience
def find_first(self, keyfn) -> Optional[Tuple[int, object]]:
    for i, rec in self.iter():
        if keyfn(rec):
            return i, rec
    return None
```

รูปภาพที่ 4.19 ฟังก์ชัน find_first

4.4.8 RoomStore/GuestStore/StayStore/KeycardStore

ผูก cls (ชนิดเรคคอร์ด) และขนาด (ROOM_SIZE ฯลฯ) เพื่อใช้เมธอดจาก FixedStore ได้ทันที

```python
class RoomStore(FixedStore):
    cls = Room
    def __init__(self, path: str):
        super().__init__(path, ROOM_SIZE)

class GuestStore(FixedStore):
    cls = Guest
    def __init__(self, path: str):
        super().__init__(path, GUEST_SIZE)

class StayStore(FixedStore):
    cls = Stay
    def __init__(self, path: str):
        super().__init__(path, STAY_SIZE)

class KeycardStore(FixedStore):
    cls = Keycard
    def __init__(self, path: str):
        super().__init__(path, KEYCARD_SIZE)
```

รูปภาพที่ 4.20 คลาสที่ใช้เมธอด FixedStore

4.5 บริการโดเมน (Domain Services) — class HotelService

รูปแบบ “เมนู/การทำงานหลัก” อ้างอิงแนวทางอธิบายในบทเมนูของเอกสารตัวอย่าง (เช่น เมนู Books/Members/Loans และเมธอดจัดการแต่ละส่วน)

4.5.1 ตัวสร้าง __init__

เปิดสโตร์สำหรับ rooms/guests/stays/keycards ชี้ไปไฟล์ data/.dat (สร้างโฟลเดอร์อัตโนมัติในส่วน Utilities)

```
def __init__(self):
    self.rooms = RoomStore(os.path.join(DATA_DIR, "rooms.dat"))
    self.guests = GuestStore(os.path.join(DATA_DIR, "guests.dat"))
    self.stays = StayStore(os.path.join(DATA_DIR, "stays.dat"))
    self.keycards = KeycardStore(os.path.join(DATA_DIR, "keycards.dat"))
```

รูปภาพที่ 4.21 ฟังก์ชัน __init__ ในคลาส HotelService

4.5.2 _next_id(store, attr)

สแกนเรคคอร์ดในสโตร์ กำหนดรหัสใหม่ = max(attr) + 1 ใช้กับทุกเอนทิตี

```
# ---- Helpers ----
def _next_id(self, store: FixedStore, attr: str) -> int:
    max_id = 0
    for _, rec in store.iter():
        rid = getattr(rec, attr)
        max_id = max(max_id, rid)
    return max_id + 1
```

รูปภาพที่ 4.22 ฟังก์ชัน _next_id

4.5.3 กลุ่ม “ห้องพัก” (Rooms)

add_room(room_type, floor, capacity, max_cards) สร้าง Room ใหม่ สถานะเริ่มต้นเป็น ว่าง (ROOM_ACTIVE_VACANT) และประทับเวลา แล้ว append ลงไฟล์

```python
# ---- CRUD Rooms ----
def add_room(self, room_type: str, floor: int, capacity: int, max_cards: int) -> Room:
    room = Room(
        room_id=self._next_id(self.rooms, "room_id"),
        status=ROOM_ACTIVE_VACANT,
        room_type=room_type,
        floor=floor,
        capacity=capacity,
        max_cards=max_cards,
        created_at=now_ts(),
        updated_at=now_ts(),
    )
    self.rooms.append(room)
    return room
```

รูปภาพที่ 4.23 ฟังก์ชัน add_room

4.5.3.1 update_room(room_id, fields)

ค้นหาด้วย find_first ⟶ อัปเดตเฉพาะฟิลด์ที่อนุญาต และปรับ updated_at

```python
def update_room(self, room_id: int, **fields) -> Optional[Room]:
    pos = self.rooms.find_first(lambda r: r.room_id == room_id)
    if not pos: return None
    idx, room = pos
    for k, v in fields.items():
        if hasattr(room, k) and k not in ("room_id", "created_at"):
            setattr(room, k, v)
    room.updated_at = now_ts()
    self.rooms.update(idx, room)
    return room
```

รูปภาพที่ 4.24 ฟังก์ชัน update_room

4.5.3.2 delete_room(room_id)

ฟังก์ชันนี้ใช้สำหรับทำ **Soft Delete** (ทำเครื่องหมายว่าถูกลบ) ข้อมูลห้องพัก: มันจะค้นหาห้องด้วย **room_id** แล้วเปลี่ยนสถานะ (room.status) เป็น **ROOM_DELETED** และบันทึกเวลาที่อัปเดต (room.updated_at) จากนั้นจึงอัปเดตข้อมูลในแหล่งเก็บข้อมูล หากไม่พบห้อง จะคืนค่า **False**, แต่ถ้าสำเร็จจะคืนค่า **True**

```python
def delete_room(self, room_id: int) -> bool:
    pos = self.rooms.find_first(lambda r: r.room_id == room_id)
    if not pos: return False
    idx, room = pos
    room.status = ROOM_DELETED
    room.updated_at = now_ts()
    self.rooms.update(idx, room)
    return True
```

รูปภาพที่4.25 ฟังก์ชัน delete_room

4.5.3.3 add_guest(full_name, phone, id_no)

ฟังก์ชัน add_guest ใช้สำหรับสร้างและเพิ่มข้อมูลแขกใหม่เข้าสู่ระบบ: มันจะสร้าง ID แขกใหม่โดยอัตโนมัติ (guest_id), กำหนดสถานะเป็นพร้อมใช้งาน (GUEST_ACTIVE), บันทึกรายละเอียดที่รับเข้ามา (ชื่อ, โทรศัพท์, เลขบัตร), บันทึกเวลาที่สร้างและอัปเดต จากนั้นจึงบันทึกวัตถุแขกใหม่เข้าสู่แหล่งเก็บข้อมูลและคืนค่าวัตถุแขกนั้น

```python
# ---- CRUD Guests ----
def add_guest(self, full_name: str, phone: str, id_no: str) -> Guest:
    guest = Guest(
        guest_id=self._next_id(self.guests, "guest_id"),
        status=GUEST_ACTIVE,
        full_name=full_name,
        phone=phone,
        id_no=id_no,
        created_at=now_ts(),
        updated_at=now_ts(),
    )
    self.guests.append(guest)
    return guest
```

รูปภาพที่ 4.26 ฟังก์ชัน add_guest

4.5.3.4 update_guest(guest_id, fields)

ฟังก์ชันนี้ใช้สำหรับ ค้นหาและอัปเดตข้อมูลแขก ในระบบ: มันจะค้นหาแขกด้วย guest_id แล้ววนลูปเพื่อนำค่าฟิลด์ที่ต้องการแก้ไข (fields) ไปตั้งค่าใหม่ให้กับวัตถุแขก (g) โดยจะ ข้ามการแก้ไขฟิลด์สำคัญ เช่น guest_id และ created_at จากนั้นจะบันทึกเวลาที่อัปเดตล่าสุดและอัปเดตข้อมูลแขกในแหล่งเก็บข้อมูล. หากไม่พบแขก จะคืนค่า None

```python
def update_guest(self, guest_id: int, **fields) -> Optional[Guest]:
    pos = self.guests.find_first(lambda g: g.guest_id == guest_id)
    if not pos: return None
    idx, g = pos
    for k, v in fields.items():
        if hasattr(g, k) and k not in ("guest_id", "created_at"):
            setattr(g, k, v)
    g.updated_at = now_ts()
    self.guests.update(idx, g)
    return g
```

รูปภาพที่ 4.27 ฟังก์ชัน update_guest

4.5.3.5 delete_guest(guest_id)

สองฟังก์ชันนี้ใช้สำหรับดึงข้อมูลคีย์การ์ด: **get_keycards** ดึงรายการทั้งหมด โดยค่าเริ่มต้นจะกรองคีย์การ์ดที่ถูกลบออก (KEYCARD_DELETED) ส่วน **get_keycards_by_room** จะกรองผลลัพธ์เพื่อคืนเฉพาะคีย์การ์ดของหมายเลขห้อง (room_id) ที่ระบุเท่านั้น

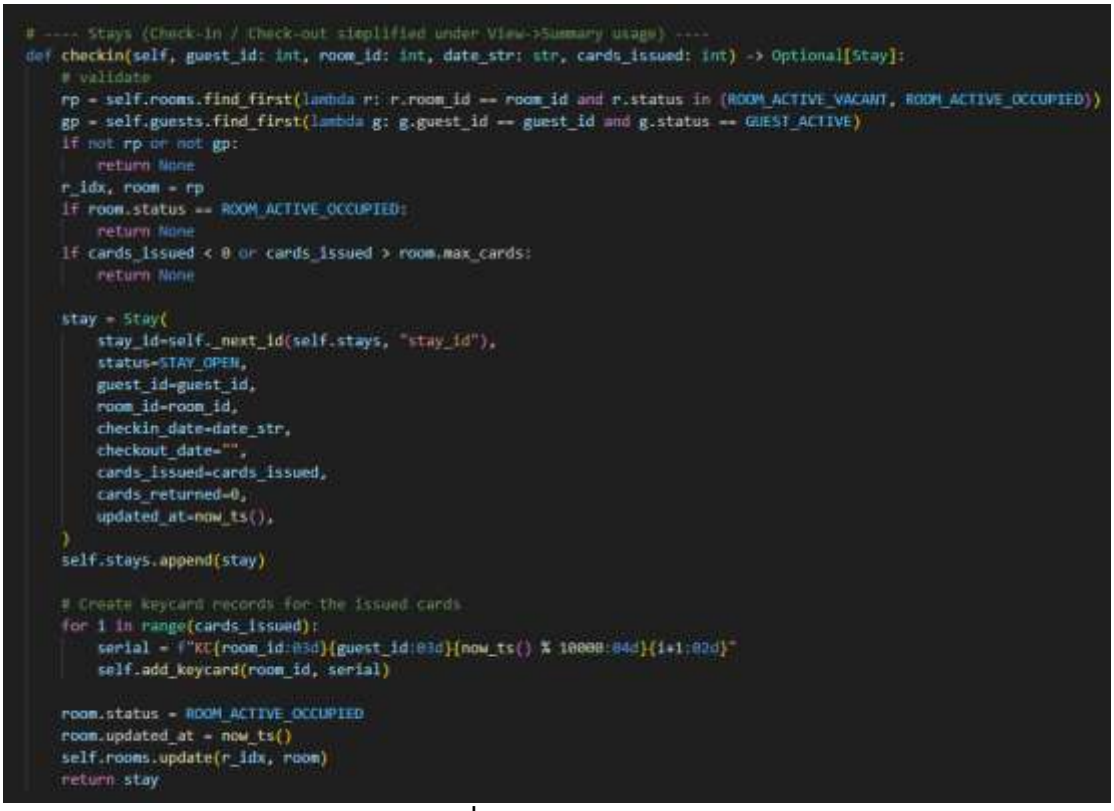
```python
def delete_guest(self, guest_id: int) -> bool:
    pos = self.guests.find_first(lambda g: g.guest_id == guest_id)
    if not pos: return False
    idx, g = pos
    g.status = GUEST_DELETED
    g.updated_at = now_ts()
    self.guests.update(idx, g)
    return True
```

รูปภาพที่ 4.28 ฟังก์ชัน delete_guest

4.5.4 กลุ่ม “การเข้าพัก” (Stays) — Check-in/Check-out

4.5.4.1 checkin(guest_id, room_id, date_str, cards_issued)

ตรวจสอบห้อง/ผู้เข้าพักและสถานะห้อง สร้าง Stay สถานะ STAY_OPEN ออกคีย์การ์ดจำนวน cards_issued โดยสร้าง serial อัตโนมัติ เปลี่ยนสถานะห้องเป็น Occupied และบันทึก



```
# ---- Stays (Check-in / Check-out simplified under View->Summary usage) ----
def checkin(self, guest_id: int, room_id: int, date_str: str, cards_issued: int) -> Optional[Stay]:
    # validate
    rp = self.rooms.find_first(lambda r: r.room_id == room_id and r.status in (ROOM_ACTIVE_VACANT, ROOM_ACTIVE_OCCUPIED))
    gp = self.guests.find_first(lambda g: g.guest_id == guest_id and g.status == GUEST_ACTIVE)
    if not rp or not gp:
        return None
    r_idx, room = rp
    if room.status == ROOM_ACTIVE_OCCUPIED:
        return None
    if cards_issued < 0 or cards_issued > room.max_cards:
        return None

    stay = Stay(
        stay_id=self._next_id(self.stays, "stay_id"),
        status=STAY_OPEN,
        guest_id=guest_id,
        room_id=room_id,
        checkin_date=date_str,
        checkout_date="",
        cards_issued=cards_issued,
        cards_returned=0,
        updated_at=now_ts(),
    )
    self.stays.append(stay)

    # Create keycard records for the issued cards
    for i in range(cards_issued):
        serial = f"KC{room_id:03d}{guest_id:03d}{now_ts() % 10000:04d}{i+1:02d}"
        self.add_keycard(room_id, serial)

    room.status = ROOM_ACTIVE_OCCUPIED
    room.updated_at = now_ts()
    self.rooms.update(r_idx, room)
    return stay
```

รูปภาพที่ 4.29 ฟังก์ชัน checkin

4.5.4.2 checkout(stay_id, date_str)

หา Stay สถานะเปิด ⟶ เปลี่ยนเป็น STAY_CLOSED ใส่วันที่ออก นับบัตรคืนเต็มจำนวน soft-delete คีย์การ์ดที่ออกสำหรับห้องนั้น เปลี่ยนสถานะห้องกลับเป็น Vacant

```python
def checkout(self, stay_id: int, date_str: str) -> bool:
    pos = self.stays.find_first(lambda s: s.stay_id == stay_id and s.status == STAY_OPEN)
    if not pos: return False
    idx, st = pos
    st.status = STAY_CLOSED
    st.checkout_date = date_str
    st.cards_returned = max(st.cards_returned, st.cards_issued)
    st.updated_at = now_ts()
    self.stays.update(idx, st)

    # Mark keycards as returned (soft delete)
    room_keycards = self.get_keycards_by_room(st.room_id)
    for keycard in room_keycards:
        if keycard.status == KEYCARD_ACTIVE:
            self.delete_keycard(keycard.keycard_id)

    # free room
    rp = self.rooms.find_first(lambda r: r.room_id == st.room_id)
    if rp:
        r_idx, room = rp
        if room.status != ROOM_DELETED:
            room.status = ROOM_ACTIVE_VACANT
            room.updated_at = now_ts()
            self.rooms.update(r_idx, room)
    return True
```

รูปภาพที่ 4.30 ฟังก์ชัน checkout

4.5.4.3 delete_stay(stay_id)

ฟังก์ชันนี้ใช้สำหรับทำ Soft Delete (ทำเครื่องหมายว่าถูกลบ) รายการเข้าพัก: มันจะค้นหารายการเข้าพักด้วย stay_id แล้วเปลี่ยนสถานะ (st.status) เป็น STAY_DELETED และบันทึกการเปลี่ยนแปลง ก่อนจะคืนค่า True เมื่อสำเร็จ

```python
def delete_stay(self, stay_id: int) -> bool:
    pos = self.stays.find_first(lambda s: s.stay_id == stay_id)
    if not pos: return False
    idx, st = pos
    st.status = STAY_DELETED
    st.updated_at = now_ts()
    self.stays.update(idx, st)
    return True
```

รูปภาพที่ 4.31 ฟังก์ชัน delete_stay

4.5.5 กลุ่ม “คีย์การ์ด” (Keycards)

4.5.5.1 add_keycard(room_id, serial)

ฟังก์ชันนี้ใช้สำหรับสร้างคีย์การ์ดใหม่ กำหนด ID อัตโนมัติ สถานะเป็น พร้อมใช้งาน และบันทึกเข้าสู่ระบบฐานข้อมูล ก่อนจะคืนค่าคีย์การ์ดนั้น

```python
# ---- CRUD Keycards ----
def add_keycard(self, room_id: int, serial: str) -> Keycard:
    keycard = Keycard(
        keycard_id=self._next_id(self.keycards, "keycard_id"),
        status=KEYCARD_ACTIVE,
        room_id=room_id,
        serial=serial,
        created_at=now_ts(),
        updated_at=now_ts(),
    )
    self.keycards.append(keycard)
    return keycard
```

รูปภาพที่ 4.32 ฟังก์ชัน add_keycard

4.5.5.2 update_keycard(keycard_id, fields)

ฟังก์ชันนี้ใช้สำหรับค้นหาและแก้ไขข้อมูลบางฟิลด์ของคีย์การ์ดตามรหัสที่กำหนด โดยมีการตรวจสอบไม่ให้แก้ไขฟิลด์สำคัญ และบันทึกเวลาที่ทำการอัปเดตไว้

```python
def update_keycard(self, keycard_id: int, **fields) -> Optional[Keycard]:
    pos = self.keycards.find_first(lambda k: k.keycard_id == keycard_id)
    if not pos: return None
    idx, k = pos
    for key, val in fields.items():
        if hasattr(k, key) and key not in ("keycard_id", "created_at"):
            setattr(k, key, val)
    k.updated_at = now_ts()
    self.keycards.update(idx, k)
    return k
```

รูปภาพที่ 4.33 ฟังก์ชัน update_keycard

4.5.5.3 delete_keycard(keycard_id)

ฟังก์ชันนี้ใช้รหัสคีย์การ์ดเพื่อค้นหาข้อมูล หากพบจะไม่ลบข้อมูลออกจริง แต่จะเปลี่ยนสถานะเป็น KEYCARD_DELETED (Soft Delete) และบันทึกเวลาอัปเดต จากนั้นจึงคืนค่า True เพื่อระบุว่าการทำเครื่องหมายว่าถูกลบสำเร็จแล้ว

```python
def delete_keycard(self, keycard_id: int) -> bool:
    pos = self.keycards.find_first(lambda k: k.keycard_id == keycard_id)
    if not pos: return False
    idx, k = pos
    k.status = KEYCARD_DELETED
    k.updated_at = now_ts()
    self.keycards.update(idx, k)
    return True
```

รูปภาพที่ 4.34 ฟังก์ชัน delete_keycard

4.5.5.4 get_keycards() และ get_keycards_by_room(room_id)

สองฟังก์ชันนี้ใช้สำหรับดึงข้อมูลคีย์การ์ด: **get_keycards** ดึงรายการทั้งหมด โดยค่าเริ่มต้นจะกรองคีย์การ์ดที่ถูกลบออก (KEYCARD_DELETED) ส่วน **get_keycards_by_room** จะกรองผลลัพธ์เพื่อคืนเฉพาะคีย์การ์ดของหมายเลขห้อง (room_id) ที่ระบุเท่านั้น

```python
def get_keycards(self, include_deleted=False) -> List[Keycard]:
    res = []
    for _, k in self.keycards.iter():
        if include_deleted or k.status != KEYCARD_DELETED:
            res.append(k)
    return res

def get_keycards_by_room(self, room_id: int) -> List[Keycard]:
    return [k for k in self.get_keycards() if k.room_id == room_id]
```

รูปภาพที่ 4.35 ฟังก์ชัน get_keycards

4.6 ฟังก์ชันดึงข้อมูลสรุปสำหรับ View/Report

4.6.1 get_rooms(...), get_guests(...), get_stays(...)

วนอ่านทุกเรคคอร์ดจากสโตร์ แล้วกรองสถานะที่ไม่ถูกลบ (หรือรวมที่ถูกลบ เมื่อส่ง include_deleted=True)

```python
# ---- Queries for View/Report ----
def get_rooms(self, include_deleted=False) -> List[Room]:
    res = []
    for _, r in self.rooms.iter():
        if include_deleted or r.status != ROOM_DELETED:
            res.append(r)
    return res

def get_guests(self, include_deleted=False) -> List[Guest]:
    res = []
    for _, g in self.guests.iter():
        if include_deleted or g.status != GUEST_DELETED:
            res.append(g)
    return res

def get_stays(self, include_deleted=False) -> List[Stay]:
    res = []
    for _, s in self.stays.iter():
        if include_deleted or s.status != STAY_DELETED:
            res.append(s)
    return res
```

รูปภาพที่ 4.36 ฟังก์ชัน get_room, ฟังก์ชัน _next_id, ฟังก์ชัน _next_id

4.7 การทำรายงาน (Reporting) — class Report

รูปแบบ “เมนูรายงาน/สรุป” สอดคล้องกับส่วน “generate_report” ในเอกสารตัวอย่างที่อธิบายการดึงข้อมูล—จัดรูป—บันทึกไฟล์

4.7.1 _line(ch="-", width=100)

สร้างเส้นคั่นตามความกว้างที่ระบุ ใช้คั่นส่วนหัว ตาราง และสรุปในรายงานตัวอักษร

```python
def _line(self, ch: str = "-", width: int = 100) -> str:
    return ch * width
```

รูปภาพที่ 4.37 ฟังก์ชัน _line

4.7.2 _rooms_table(rooms: List[Room]) -> str

ประกอบ “ตารางห้อง” พร้อมข้อมูลชื่อผู้เข้าพัก เบอร์โทร เลขบัตร และวันที่เช็คอิน (ถ้าว่างจะแสดง “-”).ขั้นตอนการแสดงผล 1) เตรียมหัวคอลัมน์และคำนวณความกว้างรวมของตาราง 2) รวบรวม stays สถานะเปิด, แผนที่ guests, และคีย์การ์ดต่อห้อง 3) วนทุกรายการห้อง สร้างบรรทัดข้อมูลตามสถานะ/ความสัมพันธ์ 4)ผลลัพธ์: คืนสตริงตารางพร้อมใช้พิมพ์หรือเขียนไฟล์รายงาน

```
def _rooms_table(self, rooms: List[Room]) -> str:
    # columns with guest info, phone, ID, and keycard serials
    cols = ["RoomID","Type","Floor","Capacity","MaxCards","Status","Guest","Phone","ID Number","Keycard Serials","Check-in"]
    widths = [8,10,7,9,9,10,25,15,15,20,12]
    def row(vals):
        return " | ".join(str(v).ljust(w) for v,w in zip(vals, widths))
    header = " | ".join(c.ljust(w) for c,w in zip(cols, widths))
    # Calculate line length to match actual table width
    line = "-" * len(header)
    out = [header, line]

    # Get active stays for guest info
    stays = {s.room_id: s for s in self.svc.get_stays() if s.status == STAY_OPEN}
    # Get guests for name lookup
    guests = {g.guest_id: g for g in self.svc.get_guests()}
    # Get keycards for serial lookup
    keycards = {k.room_id: [kc for kc in self.svc.get_keycards() if kc.room_id == k.room_id] for k in self.svc.get_keycards()}

    for r in rooms:
        status = "Deleted" if r.status==ROOM_DELETED else ("Occupied" if r.status==ROOM_ACTIVE_OCCUPIED else "Active")
        # Get guest info if room is occupied
        guest_name = "-"
        guest_phone = "-"
        guest_id = "-"
        keycard_serials = "-"
        checkin_date = "-"
        if r.status == ROOM_ACTIVE_OCCUPIED and r.room_id in stays:
            stay = stays[r.room_id]
            if stay.guest_id in guests:
                guest = guests[stay.guest_id]
                guest_name = guest.full_name
                guest_phone = guest.phone
                guest_id = guest.id_no
                checkin_date = stay.checkin_date
                # Get keycard serials for this room
                if r.room_id in keycards:
                    keycard_serials = ", ".join([k.serial for k in keycards[r.room_id]])

        out.append(row([
            r.room_id, r.room_type, r.floor, r.capacity,
            r.max_cards, status, guest_name, guest_phone, guest_id, keycard_serials, checkin_date
        ]))
    return "\n".join(out)
```

รูปภาพที่ 4.38 ฟังก์ชัน _rooms_table

4.7.3 _summary(rooms, stays) -> str

สรุปจำนวนห้องทั้งหมด/Active/Deleted/Occupied/Vacant และจำนวน Open Stays สตริงบรรทัดสรุปแบบ bullet สั้น ๆ

```
def _summary(self, rooms: List[Room], stays: List[Stay]) -> str:
    total = len(rooms)
    active = sum(1 for r in rooms if r.status != ROOM_DELETED)
    deleted = sum(1 for r in rooms if r.status == ROOM_DELETED)
    occupied = sum(1 for r in rooms if r.status == ROOM_ACTIVE_OCCUPIED)
    vacant = sum(1 for r in rooms if r.status == ROOM_ACTIVE_VACANT)
    open_stays = sum(1 for s in stays if s.status == STAY_OPEN)
    return dedent(f"""
    Summary (เฉพาะห้องสถานะ Active)
    - Total Rooms (records) : {total}
    - Active Rooms          : {active}
    - Deleted Rooms         : {deleted}
    - Currently Occupied    : {occupied}
    - Available Now         : {vacant}
    - Open Stays            : {open_stays}
    """).strip()
```

รูปภาพที่ 4.39 ฟังก์ชัน _summary

4.7.4 _stats_by_type(rooms) -> str

นับจำนวนห้องตาม room_type (เฉพาะ Active) แล้วเรียงแสดงผล

```
def _stats_by_type(self, rooms: List[Room]) -> str:
    from collections import Counter
    c = Counter(r.room_type for r in rooms if r.status != ROOM_DELETED)
    lines = ["Rooms by Type (Active only)"]
    for k, v in sorted(c.items()):
        lines.append(f"- {k}: {v}")
    return "\n".join(lines)
```

รูปภาพที่ 4.40 ฟังก์ชัน _stats_by_type

4.7.5 build_text() -> str

รวม ส่วนหัวรายงาน (เวลาสร้าง, เวอร์ชัน, Endianness, Encoding) + ตารางห้อง + สรุป + สถิติประเภทห้อง พิมพ์วันที่ตามไทม์โซนระบบ (+07:00 ในตัวอย่าง) สตริงรายงานฉบับสมบูรณ์ ใช้แสดงบนหน้าจอ/บันทึกไฟล์

```
def build_text(self) -> str:
    rooms = self.svc.get_rooms(include_deleted=True)
    stays = self.svc.get_stays(include_deleted=True)

    header = dedent(f"""\
    Hotel Key Card System — Summary Report
    Generated At : {datetime.now().strftime("%Y-%m-%d %H:%M")} (+07:00)
    App Version  : {self.APP_VERSION}
    Endianness   : {self.ENDIAN}
    Encoding     : {self.ENCODING}
    """).rstrip()

    table = self._rooms_table(rooms)
    summary = self._summary(rooms, stays)
    bytype = self._stats_by_type(rooms)

    bigline = self._line("-", 95)
    return "\n".join([header, "", bigline, table, bigline, "", summary, "", bytype]).rstrip()

def save(self, path: str) -> str:
    txt = self.build_text()
    with open(path, "w", encoding="utf-8") as f:
        f.write(txt + "\n")
    return path
```

รูปภาพที่ 4.41 ฟังก์ชัน build_text

4.7.6 save(path: str) -> str

เขียนผล build_text() ลงไฟล์; คืนพาธไฟล์ที่บันทึกสำเร็จ

```
def save(self, path: str) -> str:
    txt = self.build_text()
    with open(path, "w", encoding="utf-8") as f:
        f.write(txt + "\n")
    return path
```

รูปภาพที่ 4.42 ฟังก์ชัน save

4.8 ส่วนติดต่อผู้ใช้แบบบรรทัดคำสั่ง (CLI)

แนวการแบ่งเมนูและผูกเมนูย่อยให้ผู้ใช้เลือกซ้ำ ๆ สไตล์เดียวกับ “เมนู Book/Members/Loans” และ “main_menu” ในเอกสารตัวอย่าง

4.8.1 input_int(prompt, default=None) -> int

รับตัวเลขจากผู้ใช้ (รองรับค่าเริ่มต้น) และตรวจสอบความถูกต้องก่อนคืนค่า

```python
def input_int(self, prompt: str, default: Optional[int]=None) -> int:
    while True:
        s = input(prompt + (f" [{default}]" if default is not None else "") + ": ").strip()
        if not s and default is not None:
            return default
        if s.isdigit():
            return int(s)
        print("Please enter a valid number")
```

รูปภาพที่ 4.43 ฟังก์ชัน input_int

4.8.2 main_menu()

เมนูหลัก วนลูปรับตัวเลือก Add/Update/Delete/View/Exit แล้วเรียกเมนูย่อยตามที่เลือก

```python
def main_menu(self):
    while True:
        print("\n=== Hotel Key Card CLI ===")
        print("1) Add\n2) Update\n3) Delete\n4) View\n0) Exit")
        choice = input("Select : ").strip()
        if choice == "1":
            self.menu_add()
        elif choice == "2":
            self.menu_update()
        elif choice == "3":
            self.menu_delete()
        elif choice == "4":
            self.menu_view()
        elif choice == "0":
            print("Goodbye!")
            return
        else:
            print("Invalid menu option")
```

รูปภาพที่ 4.44 ฟังก์ชัน main_menu

4.8.3 menu_add()

เมนูเพิ่มข้อมูล: เพิ่มห้อง: แสดงรายการห้องปัจจุบันในตาราง ⟶ รับค่าห้องใหม่ ⟶ เรียก svc.add_room() เพิ่มผู้เข้าพัก: แสดงรายการเดิม ⟶ รับชื่อ/เบอร์/เลขบัตร ⟶ svc.add_guest()เช็คอิน: แสดง “ห้องว่าง” + “ผู้เข้าพัก Active” ⟶ รับ GuestID/RoomID ⟶ จำนวนบัตร ⟶ svc.checkin() (และแจ้งหมายเลข serial ที่ออก) เพิ่มคีย์การ์ด: เลือกห้อง + กรอก serial ⟶ svc.add_keycard() การแสดงผล: ใช้เมธอด _format_table เพื่อพิมพ์ตาราง (ฟังก์ชันจัดรูปตารางภายในคลาส)

```
# ----------------- Add -----------------
def menu_add(self):
    print("\nAdd: 1) Room  2) Guest  3) Stay(Check-in)  4) Keycard")
    c = input("Select: ").strip()
    if c == "1":
        # Show existing rooms
        print("\n=== Existing Rooms ===")
        rooms = self.svc.get_rooms(include_deleted=False)
        if rooms:
            headers = ["ID", "ประเภท", "ชั้น", "ความจุ", "จำนวนคีย์การ์ด", "สถานะ"]
            rows = [self._format_room_row(r) for r in rooms]
            print(self._format_table(headers, rows))
        else:
            print("No rooms in the system yet")

        print("\n=== Add New Room ===")
        rt = input("Room Type (STD/DELUXE/SUITE/..): ").strip()[:20]
        if not rt:
            print("Return to main menu")
            return
        floor = self.input_int("Floor")
        cap = self.input_int("Capacity")
        mx = self.input_int("Max keycards")
        room = self.svc.add_room(rt, floor, cap, mx)
        print(f"Room added: {room}")
```

รูปภาพที่ 4.45 ฟังก์ชัน menu_add

```
elif c == "2":
    # Show existing guests
    print("\n=== Existing Guests ===")
    guests = self.svc.get_guests(include_deleted=False)
    if guests:
        headers = ["ID", "Full Name", "Phone", "ID Number", "Status"]
        rows = [self._format_guest_row(g) for g in guests]
        print(self._format_table(headers, rows))
    else:
        print("No guests in the system yet")

    print("\n=== Add New Guest ===")
    name = input("Full name: ").strip()[:50]
    if not name:
        print("Return to main menu")
        return
    phone = input("Phone: ").strip()[:15]
    idno = input("ID/Passport: ").strip()[:20]
    g = self.svc.add_guest(name, phone, idno)
    print(f"Guest added: {g}")
```

รูปภาพที่ 4.46 ฟังก์ชัน menu_add

```
elif c == "3":
    # Show available rooms and active guests
    print("\n=== Available Rooms ===")
    available_rooms = [r for r in self.svc.get_rooms(include_deleted=False)
                       if r.status == ROOM_ACTIVE_VACANT]
    if available_rooms:
        headers = ["ID", "Type", "Floor", "Capacity", "Max Cards"]
        rows = [[str(r.room_id), r.room_type, str(r.floor),
                 str(r.capacity), str(r.max_cards)] for r in available_rooms]
        print(self._format_table(headers, rows))
    else:
        print("No vacant rooms available")
        return

    print("\n=== Active Guests ===")
    active_guests = self.svc.get_guests(include_deleted=False)
    if active_guests:
        headers = ["ID", "Full Name", "Phone", "ID Number"]
        rows = [[str(g.guest_id), g.full_name, g.phone, g.id_no]
                for g in active_guests]
        print(self._format_table(headers, rows))
    else:
        print("No guests in the system")
        return

    print("\n=== Perform Check-in ===")
    gid = self.input_int("Guest ID")
    rid = self.input_int("Room ID")

    # Validate selected room and guest exist
    selected_room = next((r for r in available_rooms if r.room_id == rid), None)
    selected_guest = next((g for g in active_guests if g.guest_id == gid), None)

    if not selected_room:
        print("Selected room not found or not available")
        return
    if not selected_guest:
        print("Selected guest not found")
        return

    print(f"\nCheck-in Information:")
    print(f"Room: {selected_room.room_type} (Room {selected_room.room_id}) - Floor {selected_room.floor}")
    print(f"Guest: {selected_guest.full_name}")
    print(f"Maximum key cards: {selected_room.max_cards}")

    date = input("Check-in date (YYYY-MM-DD) [today]: ").strip() or datetime.now().strftime("%Y-%m-%d")
    cards = self.input_int(f"Cards to issue (1-{selected_room.max_cards})", 1)

    st = self.svc.checkin(gid, rid, date, cards)
    if st:
        print(f"Check-in successful: StayID={st.stay_id}")
        print(f"Room {rid} status changed to 'Occupied'")
        # Show issued keycard serials
        room_keycards = self.svc.get_keycards_by_room(rid)
        if room_keycards:
            print(f"Issued keycard serials: {', '.join([k.serial for k in room_keycards])}")
    else:
        print("Check-in failed (verify Guest/Room/Status/MaxCards)")
```

รูปภาพที่ 4.47 ฟังก์ชัน menu_add

```
elif c == "4":
    # Add new keycard
    print("\n=== Add New Keycard ===")
    # Show existing rooms
    rooms = self.svc.get_rooms(include_deleted=False)
    if rooms:
        headers = ["ID", "Type", "Floor", "Capacity", "Max Cards", "Status"]
        rows = [self._format_room_row(r) for r in rooms]
        print(self._format_table(headers, rows))
    else:
        print("No rooms in the system yet")
        return

    room_id = self.input_int("Room ID")
    serial = input("Serial Number: ").strip()[:10]
    if not serial:
        print("Serial number is required")
        return
    keycard = self.svc.add_keycard(room_id, serial)
    print(f"Keycard added: {keycard}")
else:
    print("Return to main menu")
```

รูปภาพที่ 4.48 ฟังก์ชัน menu_add

4.8.4 menu_update()

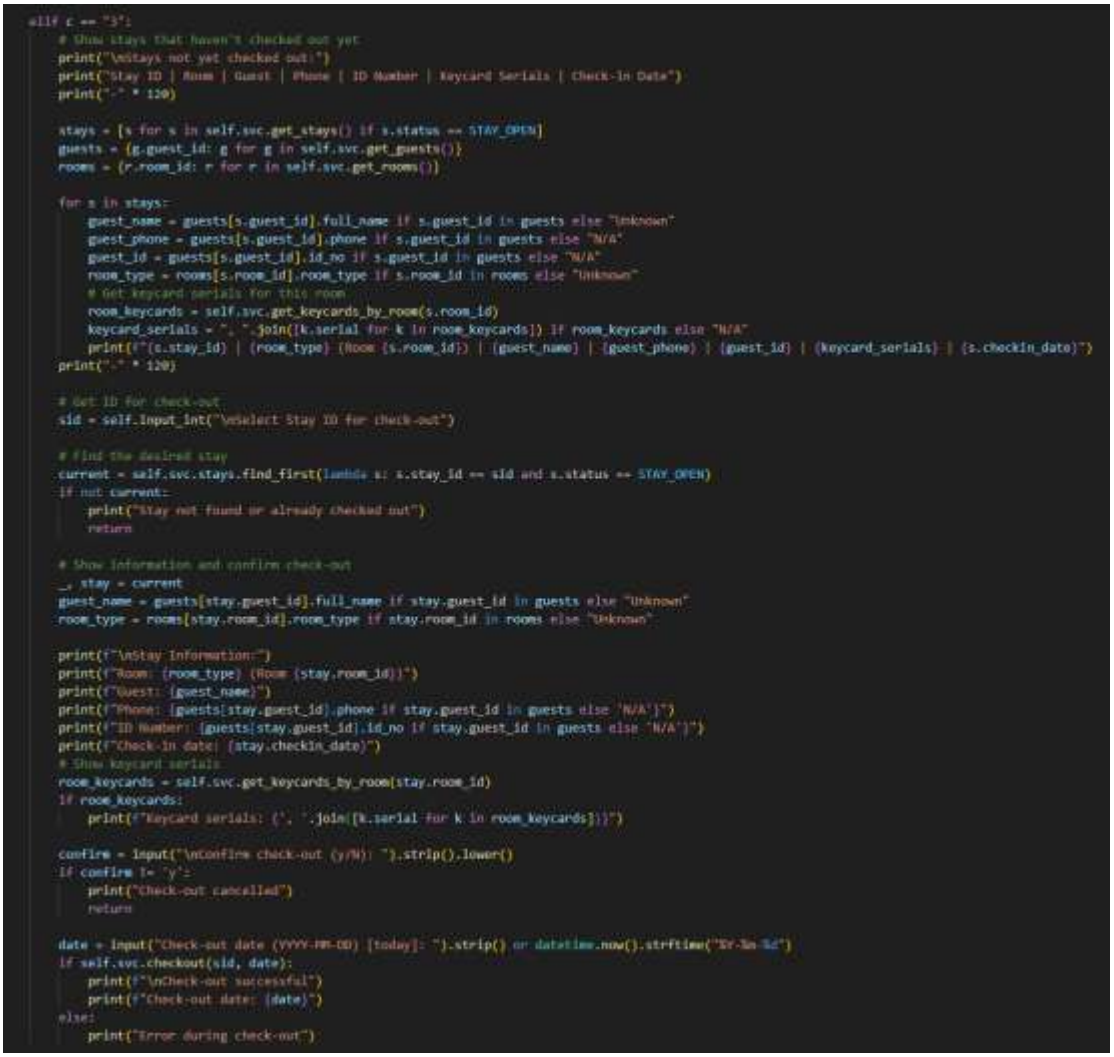
เมนูแก้ไข: ห้อง: แสดงรายการทั้งหมด ⟶ รับ RoomID ⟶ รับค่าสำหรับฟิลด์ที่จะเปลี่ยน ⟶ svc.update_room() ผู้เข้าพัก: ขั้นตอนคล้ายกัน ⟶ svc.update_guest() เช็คเอาท์: แสดงรายการเข้าพักที่ยังเปิด ⟶ เลือก StayID ⟶ ยืนยัน ⟶ svc.checkout() ข้อสังเกต: ก่อนแก้ไขจะแสดงข้อมูลเดิมและรองรับการเว้นว่างเพื่อคงค่าเดิม

```
# ---------- Update ----------
def menu_update(self):
    print("\nUpdate: 1) Room  2) Guest  3) Stay(Check-out)  4) Keycard")
    c = input("Select: ").strip()
    if c == "1":
        # Show all rooms
        print("\nAll Rooms:")
        print("ID | Type | Floor | Capacity | Max Cards | Status")
        print("-" * 70)
        rooms = self.svc.get_rooms(include_deleted=False)
        for r in rooms:
            status = "Vacant" if r.status == ROOM_ACTIVE_VACANT else "Occupied" if r.status == ROOM_ACTIVE_OCCUPIED else "Deleted"
            print(f"{r.room_id} | {r.room_type} | {r.floor} | {r.capacity} | {r.max_cards} | {status}")
        print("-" * 70)

        # Get room ID to edit
        rid = self.input_int("\nSelect Room ID to edit")

        # Find current room data
        current = self.svc.rooms.find_first(lambda r: r.room_id == rid)
        if not current:
            print("Room not found")
            return

        # Show current data
        _, room = current
        print(f"\nCurrent Information:")
        print(f"Room Type: {room.room_type}")
        print(f"Floor: {room.floor}")
        print(f"Capacity: {room.capacity}")
        print(f"Max Key Cards: {room.max_cards}")

        print("\nEnter new information (leave blank to keep current):")
        rt = input("Room Type: ").strip() or room.room_type
        floor = input("Floor: ").strip()
        cap = input("Capacity: ").strip()
        mx = input("Max Key Cards: ").strip()

        fields = {
            "room_type": rt[:20],
            "floor": int(floor) if floor.isdigit() else room.floor,
            "capacity": int(cap) if cap.isdigit() else room.capacity,
            "max_cards": int(mx) if mx.isdigit() else room.max_cards
        }

        upd = self.svc.update_room(rid, **fields)
        if upd:
            print("\nData updated successfully")
            print(f"New information: {upd}")
        else:
            print("Error updating data")
```

รูปภาพที่ 4.49 ฟังก์ชัน menu_update

```
elif c == "2":
    # Show all guests
    print("\nAll Guests:")
    print("ID | Full Name | Phone | ID/Passport")
    print("-" * 70)
    guests = self.svc.get_guests(include_deleted=False)
    for g in guests:
        print(f"{g.guest_id} | {g.full_name} | {g.phone} | {g.id_no}")
    print("-" * 70)

    # Get guest ID to edit
    gid = self.input_int("\nSelect Guest ID to edit")

    # Find current guest data
    current = self.svc.guests.find_first(lambda g: g.guest_id == gid)
    if not current:
        print("Guest not found")
        return

    # Show current data
    _, guest = current
    print(f"\nCurrent Information:")
    print(f"Full Name: {guest.full_name}")
    print(f"Phone: {guest.phone}")
    print(f"ID/Passport: {guest.id_no}")

    print("\nEnter new information (leave blank to keep current):")
    name = input("Full Name: ").strip() or guest.full_name
    phone = input("Phone: ").strip() or guest.phone
    idno = input("ID/Passport: ").strip() or guest.id_no

    # Update data
    fields = {
        "full_name": name[:50],
        "phone": phone[:15],
        "id_no": idno[:20]
    }

    upd = self.svc.update_guest(gid, **fields)
    if upd:
        print("\nData updated successfully")
        print(f"New information: {upd}")
    else:
        print("Error updating data")
```

รูปภาพที่ 4.50 ฟังก์ชัน menu_update



```
elif c == "3":
    # Show stays that haven't checked out yet
    print("\nStays not yet checked out:")
    print("Stay ID | Room | Guest | Phone | ID Number | Keycard Serials | Check-In Date")
    print("-" * 120)

    stays = [s for s in self.svc.get_stays() if s.status == STAY_OPEN]
    guests = {g.guest_id: g for g in self.svc.get_guests()}
    rooms = {r.room_id: r for r in self.svc.get_rooms()}

    for s in stays:
        guest_name = guests[s.guest_id].full_name if s.guest_id in guests else "Unknown"
        guest_phone = guests[s.guest_id].phone if s.guest_id in guests else "N/A"
        guest_id = guests[s.guest_id].id_no if s.guest_id in guests else "N/A"
        room_type = rooms[s.room_id].room_type if s.room_id in rooms else "Unknown"
        # Get keycard serials for this room
        room_keycards = self.svc.get_keycards_by_room(s.room_id)
        keycard_serials = ", ".join([k.serial for k in room_keycards]) if room_keycards else "N/A"
        print(f"{s.stay_id} | {room_type} {Room {s.room_id}} | {guest_name} | {guest_phone} | {guest_id} | {keycard_serials} | {s.checkIn_date}")
    print("-" * 120)

    # Get ID for check-out
    sid = self.input_int("\nSelect Stay ID for check-out")

    # Find the desired stay
    current = self.svc.stays.find_first(lambda s: s.stay_id == sid and s.status == STAY_OPEN)
    if not current:
        print("Stay not found or already checked out")
        return

    # Show information and confirm check-out
    _, stay = current
    guest_name = guests[stay.guest_id].full_name if stay.guest_id in guests else "Unknown"
    room_type = rooms[stay.room_id].room_type if stay.room_id in rooms else "Unknown"

    print(f"\nStay Information:")
    print(f"Room: {room_type} {Room {stay.room_id}}")
    print(f"Guest: {guest_name}")
    print(f"Phone: {guests[stay.guest_id].phone if stay.guest_id in guests else 'N/A'}")
    print(f"ID Number: {guests[stay.guest_id].id_no if stay.guest_id in guests else 'N/A'}")
    print(f"Check-in date: {stay.checkIn_date}")
    # Show keycard serials
    room_keycards = self.svc.get_keycards_by_room(stay.room_id)
    if room_keycards:
        print(f"Keycard serials: {', '.join([k.serial for k in room_keycards])}")

    confirm = input("\nConfirm check-out (y/N): ").strip().lower()
    if confirm != 'y':
        print("Check-out cancelled")
        return

    date = input("Check-out date (YYYY-MM-DD) [today]: ").strip() or datetime.now().strftime("%Y-%m-%d")
    if self.svc.checkout(sid, date):
        print(f"\nCheck-out successful")
        print(f"Check-out date: {date}")
    else:
        print("Error during check-out")
```

รูปภาพที่ 4.51 ฟังก์ชัน menu_update

```
elif c == "4":
    # Update keycard
    print("\n=== Update Keycard ===")
    # Show all keycards
    keycards = self.svc.get_keycards()
    if keycards:
        headers = ["Keycard ID", "Room ID", "Serial", "Status"]
        rows = []
        for k in keycards:
            status = "Active" if k.status == KEYCARD_ACTIVE else "Deleted"
            rows.append([str(k.keycard_id), str(k.room_id), k.serial, status])
        print(self._format_table(headers, rows))
    else:
        print("No keycards in the system")
        return

    keycard_id = self.input_int("Keycard ID to update")
    current = self.svc.keycards.find_first(lambda k: k.keycard_id == keycard_id)
    if not current:
        print("Keycard not found")
        return
    _, keycard = current
    print(f"\nCurrent Information:")
    print(f"Room ID: {keycard.room_id}")
    print(f"Serial: {keycard.serial}")

    print("\nEnter new information (leave blank to keep current):")
    new_room = input("New Room ID: ").strip()
    new_serial = input("New Serial: ").strip()
    fields = {}
    if new_room.isdigit():
        fields["room_id"] = int(new_room)
    if new_serial:
        fields["serial"] = new_serial[:10]
    if fields:
        updated = self.svc.update_keycard(keycard_id, **fields)
        if updated:
            print("Keycard updated successfully")
        else:
            print("Error updating keycard")
    else:
        print("No changes made")
else:
    print("Return to main menu")
```

รูปภาพที่ 4.52 ฟังก์ชัน menu_update

4.8.5 menu_view() (ดูข้อมูล/สรุป/รายงาน)

ดูรายละเอียด 1 รายการ: เลือก 1)Room 2) Guest 3) Stay, ดูทั้งหมด/กรอง: Rooms/Guests/Stays ทั้งหมด, รายงาน: ว่าง/ไม่ว่าง/ตามประเภท ⟶ พิมพ์ตารางด้วย Report._rooms_table(), ส่งออกไฟล์รายงาน: เรียก Report.save() แล้วพิมพ์พาธไฟล์, ดูคีย์การ์ด: ทั้งหมด/ตามห้อง/ตามสถานะ (แสดงเวลา “Created” ด้วย fmt_date)

```
def menu_view(self):
    print(dedent("""
    View:
      1) View Single Record
      2) View All Records
      3) View Filtered Records
      4) Summary Statistics + Export Report
      5) View Keycards
    """))
    c = input("Select: ").strip()
    if c == "1":
        sub = input("Select: 1) Room  2) Guest  3) Stay : ").strip()
        if sub == "1":
            # Show all rooms first
            print("\nAll Rooms:")
            rooms = self.svc.get_rooms()
            headers = ["ID", "Type", "Floor", "Capacity", "Max Cards", "Status"]
            rows = [self._format_room_row(r) for r in rooms]
            print(self._format_table(headers, rows))

            rid = self.input_int("\nSelect Room ID to view")
            pos = self.svc.rooms.find_first(lambda r: r.room_id == rid)
            if pos:
                _, room = pos
                print("\nSelected Room Information:")
                print(self._format_table(headers, [self._format_room_row(room)]))
            else:
                print("Room not found")
```

รูปภาพที่ 4.53 ฟังก์ชัน menu_view

```
    elif sub == "2":
        # Show all guests
        print("\nAll Guests:")
        guests = self.svc.get_guests()
        headers = ["ID", "Full Name", "Phone", "ID Number", "Status"]
        rows = [self._format_guest_row(g) for g in guests]
        print(self._format_table(headers, rows))

        gid = self.input_int("\nSelect Guest ID to view")
        pos = self.svc.guests.find_first(lambda g: g.guest_id == gid)
        if pos:
            _, guest = pos
            print("\nSelected Guest Information:")
            print(self._format_table(headers, [self._format_guest_row(guest)]))
        else:
            print("Guest not found")
    else:
        # Show all stays
        print("\nAll Stays:")
        stays = self.svc.get_stays()
        guests = {g.guest_id: g for g in self.svc.get_guests()}
        rooms = {r.room_id: r for r in self.svc.gut_rooms()}

        headers = ["StayID", "RoomID", "Room Type", "Guest Name", "Check-in", "Check-out", "Cards Issued", "Cards Returned", "Status"]
        rows = [self._format_stay_row(s, guests, rooms) for s in stays]
        print(self._format_table(headers, rows))

        sid = self.input_int("\nSelect Stay ID to view")
        pos = self.svc.stays.find_first(lambda s: s.stay_id == sid)
        if pos:
            _, stay = pos
            print("\nSelected Stay Information:")
            print(self._format_table(headers, [self._format_stay_row(stay, guests, rooms)]))
        else:
            print("Stay information not found")

elif c == "2":
    sub = input("Select: 1) Rooms  2) Guests  3) Stays : ").strip()
    if sub == "1":
        rooms = self.svc.get_rooms()
        headers = ["ID", "Type", "Floor", "Capacity", "Max Cards", "Status"]
        rows = [self._format_room_row(r) for r in rooms]
        print("\nAll Rooms:")
        print(self._format_table(headers, rows))
    elif sub == "2":
        guests = self.svc.get_guests()
        headers = ["ID", "Full Name", "Phone", "ID Number", "Status"]
        rows = [self._format_guest_row(g) for g in guests]
        print("\nAll Guests:")
        print(self._format_table(headers, rows))
    else:
        stays = self.svc.get_stays()
        guests = {g.guest_id: g for g in self.svc.get_guests()}
        rooms = {r.room_id: r for r in self.svc.get_rooms()}
        headers = ["StayID", "RoomID", "Room Type", "Guest Name", "Check-in", "Check-out", "Cards Issued", "Cards Returned", "Status"]
        rows = [self._format_stay_row(s, guests, rooms) for s in stays]
        print("\nAll Stays:")
        print(self._format_table(headers, rows))
```

รูปภาพที่ 4.54 ฟังก์ชัน menu_view

```
elif c == "3":
    print("Filter Rooms: 1) Vacant Only  2) Occupied Only  3) By Type")
    sub = input("Select: ").strip()
    rooms = self.svc.get_rooms(include_deleted=False)
    if sub == "1":
        rooms = [r for r in rooms if r.status == ROOM_ACTIVE_VACANT]
    elif sub == "2":
        rooms = [r for r in rooms if r.status == ROOM_ACTIVE_OCCUPIED]
    elif sub == "3":
        typ = input("Room Type: ").strip()
        rooms = [r for r in rooms if r.room_type == typ]
    rep = Report(self.svc)
    print(rep._rooms_table(rooms))
elif c == "4":
    path = os.path.join(REPORT_DIR, "hotel_report.txt")
    rep = Report(self.svc)
    rep.save(path)
    print(f"Export successful → {path}")
    print("\nReport header preview:\n")
    print(rep.build_text().split("\n", 8)[0:8])
```

รูปภาพที่ 4.55 ฟังก์ชัน menu_view

```
    elif c == "5":
        # View keycards
        print("\nKeycard View: 1) All Keycards  2) By Room  3) By Status")
        sub = input("Select: ").strip()
        if sub == "1":
            keycards = self.svc.get_keycards()
            if keycards:
                headers = ["Keycard ID", "Room ID", "Serial", "Status", "Created"]
                rows = []
                for k in keycards:
                    status = "Active" if k.status == KEYCARD_ACTIVE else "Deleted"
                    created = fmt_date(k.created_at)
                    rows.append([str(k.keycard_id), str(k.room_id), k.serial, status, created])
                print("\nAll Keycards:")
                print(self._format_table(headers, rows))
            else:
                print("No keycards in the system")
        elif sub == "2":
            room_id = self.input_int("Room ID")
            room_keycards = self.svc.get_keycards_by_room(room_id)
            if room_keycards:
                headers = ["Keycard ID", "Serial", "Status", "Created"]
                rows = []
                for k in room_keycards:
                    status = "Active" if k.status == KEYCARD_ACTIVE else "Deleted"
                    created = fmt_date(k.created_at)
                    rows.append([str(k.keycard_id), k.serial, status, created])
                print(f"\nKeycards for Room {room_id}:")
                print(self._format_table(headers, rows))
            else:
                print(f"No keycards found for Room {room_id}")
        elif sub == "3":
            status_filter = input("Status (1=Active, 0=Deleted): ").strip()
            if status_filter in ["0", "1"]:
                status_val = int(status_filter)
                keycards = [k for k in self.svc.get_keycards() if k.status == status_val]
                if keycards:
                    headers = ["Keycard ID", "Room ID", "Serial", "Status", "Created"]
                    rows = []
                    for k in keycards:
                        status = "Active" if k.status == KEYCARD_ACTIVE else "Deleted"
                        created = fmt_date(k.created_at)
                        rows.append([str(k.keycard_id), str(k.room_id), k.serial, status, created])
                    print(f"\nKeycards with status {status_val}:")
                    print(self._format_table(headers, rows))
                else:
                    print(f"No keycards found with status {status_val}")
            else:
                print("Invalid status filter")
        else:
            print("Return to main menu")
    else:
        print("Return to main menu")
```

รูปภาพที่ 4.56 ฟังก์ชัน menu_view

บทที่ 5 สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

โครงงาน "ระบบยืม – คืน คีย์การ์ดในโรงแรม" ประสบความสำเร็จในการบรรลุวัตถุประสงค์หลักที่ตั้งไว้ทั้งหมดได้อย่างสมบูรณ์ โดยสามารถพัฒนาโปรแกรมจัดการคีย์การ์ดโดยใช้ ภาษา Python และหลักการ Object-Oriented Programming (OOP) ทำให้โครงสร้างโปรแกรมมีความยืดหยุ่นและเป็นระบบ โปรแกรมสามารถทำงานได้บน Command-Line Interface (CLI) เพื่อรองรับการจัดการข้อมูลจริงได้อย่างครบถ้วน นอกจากนี้ยังบรรลุวัตถุประสงค์ด้านฐานข้อมูล โดยออกแบบและสร้างโครงสร้างแฟ้มข้อมูลหลัก 4 แฟ้ม ใช้ Binary File ร่วมกับ Module struct ในการจัดเก็บข้อมูล ทำให้สามารถจัดการกับข้อมูลปริมาณมากถึง หลักหมื่นระเบียน ได้อย่างมีประสิทธิภาพตามที่กำหนด และมีการบันทึกข้อมูลสำคัญและ timestamp ของทุกกิจกรรมเพื่อใช้ในการตรวจสอบย้อนหลัง (Audit Trail) ได้อย่างละเอียด ระบบมีจุดเด่นด้านความน่าเชื่อถือและความรวดเร็วในการจัดเก็บข้อมูลด้วย Binary File รวมถึงมีระบบควบคุมการออกและคืนคีย์การ์ดที่เคร่งครัด

5.2 ปัญหาและอุปสรรคในการดำเนินงาน

ในระหว่างการพัฒนา ได้พบกับปัญหาและอุปสรรคทางเทคนิคที่สำคัญ ซึ่งได้ดำเนินการแก้ไขและบรรเทาผลกระทบจนสามารถทำให้ระบบทำงานได้ตามข้อกำหนด ปัญหาหลักคือ ความแม่นยำของการจัดการโครงสร้างไบนารี ที่ต้องกำหนด Format String และควบคุม Data Alignment/Padding อย่างแม่นยำ ซึ่งได้แก้ไขโดยใช้ฟังก์ชัน struct.calcsize() และ Endianness Prefix (<) เพื่อกำหนดมาตรฐานโครงสร้างข้อมูลที่ชัดเจน อีกปัญหาหนึ่งคือ การจัดการการเข้ารหัสตัวอักษรภาษาไทย ในแฟ้มไบนารี ซึ่งได้กำหนดให้ใช้ utf-8 เป็นมาตรฐานในการเข้ารหัสและถอดรหัสข้อมูลตลอดทั้งระบบเพื่อป้องกันตัวอักษรผิดเพี้ยน อย่างไรก็ตาม ข้อจำกัดที่ยังคงอยู่คือ ส่วนติดต่อผู้ใช้งาน (CLI) ซึ่งขาดความสะดวกในการใช้งานเมื่อเทียบกับระบบ GUI และมีความเสี่ยงด้านความสมบูรณ์ของข้อมูล เนื่องจากเป็นโครงสร้างแบบไฟล์ที่อาจเกิดความเสียหายหากโปรแกรมปิดตัวลงอย่างผิดปกติ

5.3 ข้อเสนอแนะ

เพื่อเพิ่มประสิทธิภาพ ความปลอดภัย และความสามารถในการใช้งานจริงของระบบยืม – คืนคีย์การ์ดในโรงแรม ควรมีการพัฒนาต่อยอดในอนาคตตามข้อเสนอแนะดังต่อไปนี้:

5.3.1 ควรพิจารณาเปลี่ยนจากการจัดการข้อมูลแบบไฟล์ไบนารีไปใช้ ระบบฐานข้อมูลเชิงสัมพันธ์ (RDBMS) เช่น SQLite, MySQL หรือ PostgreSQL เพื่อเพิ่มประสิทธิภาพในการจัดการความสัมพันธ์

ของข้อมูล (Foreign Key), ความปลอดภัยของข้อมูล, และรองรับการเข้าถึงข้อมูลจากผู้ใช้งานหลายคนพร้อมกัน (Multi-user Access) ซึ่งจะช่วยลดความเสี่ยงด้านความสมบูรณ์ของข้อมูลลงได้

5.3.2 ควรต่อยอดส่วนติดต่อผู้ใช้งานจาก CLI เป็น Desktop Application (GUI) โดยใช้ไลบรารี Python เช่น Tkinter หรือ PyQt หรือพัฒนาเป็น Web Application โดยใช้ Framework Flask หรือ Django เพื่อให้พนักงานสามารถเข้าถึงระบบได้สะดวกและรวดเร็วยิ่งขึ้น รวมถึงสามารถแสดงผลข้อมูลในรูปแบบตารางหรือกราฟิกที่ซับซ้อนได้

5.3.3 ควรพัฒนาระบบให้มีส่วนต่อประสาน (Interface) สำหรับ เชื่อมต่อโดยตรงกับเครื่องอ่าน/เขียนคีย์การ์ด (Card Encoder) เพื่อให้กระบวนการออกและการคืนคีย์การ์ดเป็นไปโดยอัตโนมัติอย่างสมบูรณ์ และลดข้อผิดพลาดจากการบันทึกสถานะด้วยมือ ซึ่งจะช่วยเพิ่มประสิทธิภาพในการปฏิบัติงานให้สูงสุด

5.3.4 ควรเพิ่มระบบ การจัดการบัญชีผู้ใช้ พร้อมการกำหนดสิทธิ์การเข้าถึง (Access Control) ที่มีความปลอดภัยสำหรับพนักงานแต่ละระดับ เพื่อจำกัดการเข้าถึงฟังก์ชันสำคัญ และควรเพิ่มฟังก์ชันการสำรองข้อมูลอัตโนมัติ (Backup) ของแฟ้มข้อมูลทั้งหมดเป็นประจำทุกวัน เพื่อให้สามารถกู้คืนข้อมูลได้ทันทีในกรณีที่เกิดความเสียหาย

5.4 สิ่งที่ผู้จัดทำได้รับในการพัฒนาโครงงาน

โครงงานนี้ได้ก่อให้เกิดประโยชน์อย่างสูงต่อผู้พัฒนา โดยช่วยให้ได้ประยุกต์ใช้ความรู้ด้านวิศวกรรมสารสนเทศอย่างเต็มที่ ทั้งด้าน **ภาษา Python, หลักการเขียนโปรแกรมเชิงวัตถุ (OOP), การวิเคราะห์ระบบ, และการออกแบบฐานข้อมูล** เพื่อสร้างโปรแกรมที่ใช้งานได้จริงตามโจทย์ธุรกิจที่สำคัญคือได้ฝึกฝนทักษะเฉพาะทางด้าน **การจัดการข้อมูลขั้นสูงในระดับต่ำ (Low-Level Data Management)** ผ่านการใช้ Binary File และ Module struct ซึ่งเป็นทักษะที่สำคัญในการจัดการหน่วยความจำและจัดเก็บข้อมูลอย่างมีประสิทธิภาพ รวมถึงการเรียนรู้และแก้ไขปัญหาทางเทคนิคที่ซับซ้อน เช่น ปัญหา **Data Alignment** และ **Encoding** ภาษาไทย นอกจากนี้ยังได้พัฒนาทักษะด้านการวางแผนงาน การแบ่งงาน และการทำงานร่วมกันภายใต้กำหนดเวลา ซึ่งเป็นทักษะการทำงานเป็นทีมและการบริหารโครงการพื้นฐานที่จำเป็นอย่างยิ่ง