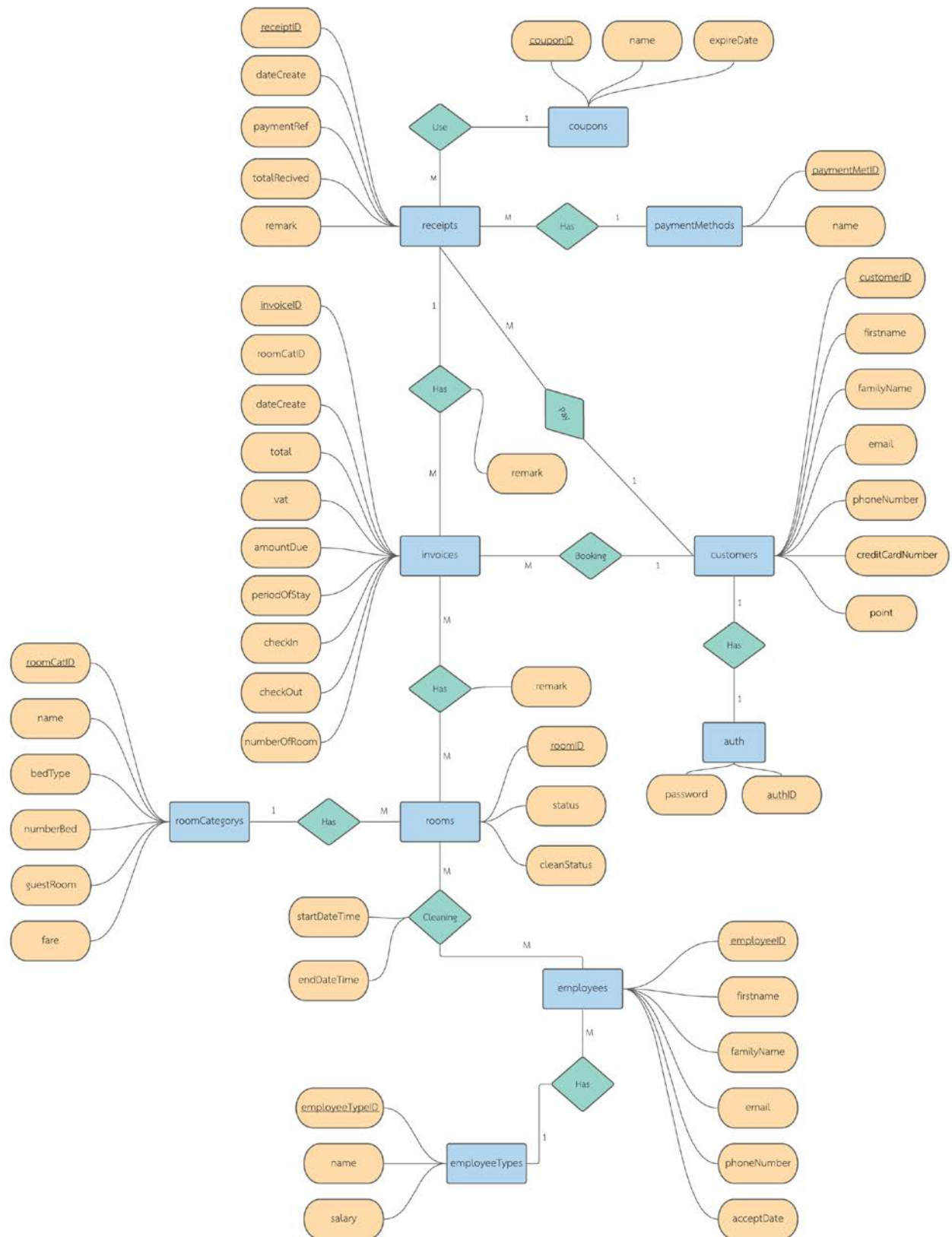


Assumption of the Hotel

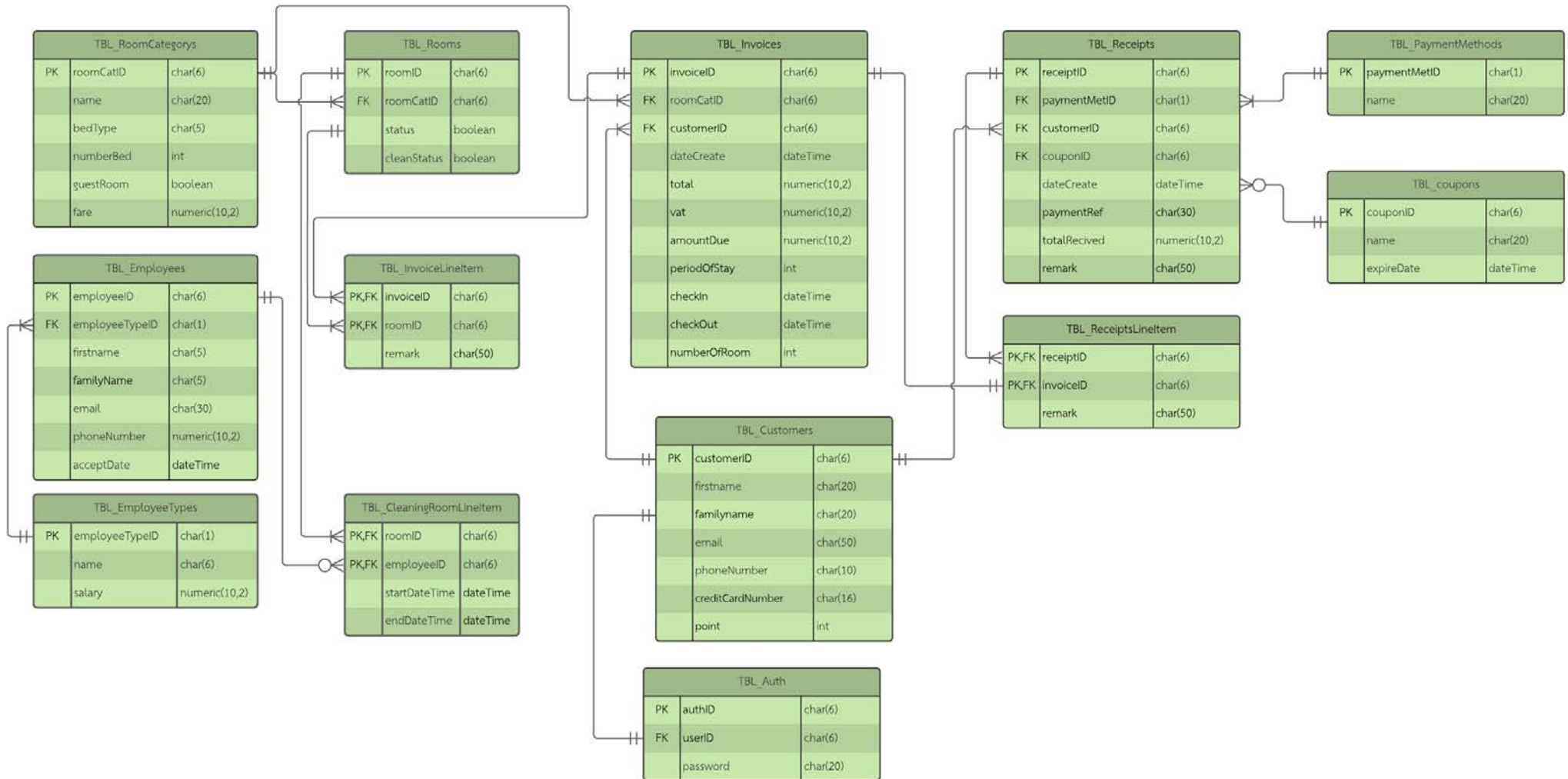
ในระบบการจองโรงแรมที่ได้สร้างขึ้น กำหนดให้ Customer สามารถจองห้องพักในโรงแรมได้เองผ่านทางเว็บไซต์ เมื่อเข้าสู่หน้าแรกของเว็บไซต์ Customer สามารถดูข้อมูล ประเภทของห้อง ราคา และสถานะห้องที่สามารถเข้าพักได้ หาก Customer ต้องการจองห้องเองผ่านทางเว็บไซต์ จำเป็นต้อง register ก่อน โดยข้อมูลเบื้องต้นที่ต้องกรอกในการลงทะเบียนคือ Name Lastname Phone number และ E-mail address หากลงทะเบียนสำเร็จและต้องการทำการจองห้องพัก Customer จำเป็นต้องระบุ room types number of rooms และ period of stay ที่ต้องการเข้าพัก เมื่อคัดเลือกประเภทห้องและจำนวนที่ต้องการจองไป ระบบจะขึ้นยอดรวม total vat และ amount due ที่เป็นยอดเงินที่รวม vat มาแสดงขึ้นให้เห็นก่อนที่จะทำการยืนยันการจอง โดย Customer จะสามารถเลือกชำระเงินได้ 3 ช่องทางคือ cash credit card และ bit coin หาก Customer ที่ไม่สะดวกทำการจองห้องในโรงแรมผ่านทางเว็บไซต์ด้วยตนเองสามารถให้ Salesperson ทำการจองให้ได้ โดย Salesperson จะต้องใช้ข้อมูล user ID ในการเข้าสู่ระบบ และ Salesperson สามารถทำการ print receipt ของคำสั่งซื้อในการจองห้องพักได้ ในส่วนนอกเหนือจาก requirement ที่ได้ทำการเพิ่มคือ Admin สามารถดูสถานะ cleaning room ของห้องพัก และจัดการตารางงานสำหรับ employee ที่เป็นคนทำความสะอาดห้องพัก และจัดการระบบคูปองส่วนลดในการจองห้องพักให้กับ Customer

ER-Diagram (Conceptual Design)



รูปที่ 1 แสดง ER - Diagram ของ Database

Database Schema (Logical Design)



รูปที่ 2 แสดง Schema ของ Database แบบ Crowfoot

Database Description

1. **TBL_RoomCategorys** : สำหรับเก็บข้อมูลประเภทของห้องพักในโรงแรม
 - roomCatID (Primary Key) : รหัสประจำตัว แต่ละประเภทของห้องพัก
 - name : ชื่อประเภทของห้องพัก
 - bedType : ประเภทของเตียงนอนในห้องพัก
 - numberBed : จำนวนของเตียงนอนในห้องพักแต่ละประเภท
 - guestRoom : ห้องรับแขก
 - fare : ราคาต่อคืนของห้องพักแต่ละประเภท
2. **TBL_Rooms** : สำหรับเก็บข้อมูลของห้องพักในโรงแรม
 - roomID (Primary Key) : รหัสประจำตัว แต่ละห้องพัก
 - roomCatID : รหัสประจำตัว แต่ละประเภทของห้องพัก
 - status : สถานะของห้องพัก
 - cleanStatus : สถานการณ์ทำความสะอาดของห้องพัก
3. **TBL_Invoices** : สำหรับเก็บข้อมูลของใบแจ้งหนี้สำหรับการจองห้องพัก
 - invoiceID (Primary Key) : รหัสประจำตัว แต่ละใบแจ้งหนี้
 - customerID : รหัสประจำตัว ของลูกค้าแต่ละคน
 - roomCatID : รหัสประจำตัว แต่ละประเภทของห้องพัก
 - dateCreate : วันที่และเวลาทำการจองห้องพัก
 - total : ยอดเงินรวมค่าห้องพักที่ต้องชำระ
 - vat : ภาษี ณ ที่จ่าย
 - amountDue : ยอดเงินค่าห้องพักรวมภาษี ณ ที่จ่าย ที่ต้องชำระ
 - periodOfStay : จำนวนวันที่และเวลาทำการจองห้องพัก
 - checkIn : วันแรกที่เข้าพัก
 - checkOut : วันสุดท้ายที่เข้าพัก
 - numberOfRoom : จำนวนของห้องพักที่ทำการจอง
4. **TBL_InvoiceLineItem** : สำหรับเก็บข้อมูลของใบแจ้งหนี้แต่ละรายการที่ได้ทำการจองห้องพัก
 - invoiceID (Primary Key) : รหัสประจำตัว แต่ละใบแจ้งหนี้
 - roomID (Primary Key) : รหัสประจำตัว แต่ละห้องพัก
 - remark : รายละเอียดเพิ่มเติม

5. **TBL_Receipts** : สำหรับเก็บข้อมูลของใบเสร็จชำระเงิน
 - receiptID (Primary Key) : รหัสประจำตัว แต่ละใบเสร็จชำระเงิน
 - paymentMetID : รหัสประจำตัว แต่ละประเภทของการชำระเงิน
 - customerID : รหัสประจำตัว ของลูกค้าแต่ละคน
 - couponID : รหัสประจำตัว ของคูปอง
 - dateCreate : วันที่และเวลาออกใบเสร็จชำระเงิน
 - paymentRef : ช่องทางการชำระเงิน
 - totalRecived : ยอดเงินทั้งหมดที่ชำระในใบเสร็จชำระเงิน
 - remark : รายละเอียดเพิ่มเติมในใบเสร็จชำระเงิน

6. **TBL_ReceiptsLineItem** : สำหรับเก็บข้อมูลใบแจ้งหนี้ทั้งหมดที่อยู่ในใบเสร็จชำระเงินแต่ละใบเสร็จชำระเงิน
 - receiptID (Primary Key) : รหัสประจำตัว แต่ละใบเสร็จชำระเงิน
 - invoiceID (Primary Key) : รหัสประจำตัว แต่ละใบแจ้งหนี้
 - remark : รายละเอียดเพิ่มเติม

7. **TBL_PaymentMethods** : สำหรับเก็บข้อมูลของประเภทการชำระเงิน
 - paymentMetID (Primary Key) : รหัสประจำตัว แต่ละประเภทของการชำระเงิน
 - name : ชื่อประเภทการชำระเงิน

8. **TBL_Employees** : สำหรับเก็บข้อมูลของพนักงานทั้งหมด เช่น กลุ่ม Saleperson, Security Guard, Chef และ Cleaner
 - employeeID (Primary Key) : รหัสประจำตัว ของพนักงานแต่ละคน
 - employeeTypeID : รหัสประจำตัว ของประเภทพนักงานแต่ละคน
 - fistname : ชื่อของพนักงานแต่ละคน
 - familyName : ชื่อสกุลของพนักงานแต่ละคน
 - email : อีเมลของพนักงานแต่ละคน
 - phoneNumber : เบอร์โทรของพนักงานแต่ละคน
 - acceptDate : วันที่เข้าทำงานของพนักงานแต่ละคน

9. TBL_EmployeeTypes : สำหรับเก็บข้อมูลประเภทของพนักงาน

- employeeTypeID (Primary Key) : รหัสประจำตัว ของประเภทพนักงานแต่ละคน
- name : ชื่อประเภทของพนักงาน
- salary : เงินเดือนของพนักงานในประเภทนั้น

10. TBL_Customers : สำหรับเก็บข้อมูลของลูกค้า

- customerID (Primary Key) : รหัสประจำตัว ของลูกค้าแต่ละคน
- firstname : ชื่อของลูกค้าแต่ละคน
- familyname : ชื่อสกุลของลูกค้าแต่ละคน
- email : อีเมลของลูกค้าแต่ละคน
- phoneNumber : เบอร์โทรของลูกค้าแต่ละคน
- creditCardNumber : หมายเลขบัตรเครดิตของลูกค้าแต่ละคน
- point : คะแนนเก็บสะสมของลูกค้า

11. TBL_Auth : สำหรับเก็บข้อมูลล็อกอินเข้าสู่ระบบ

- authID (Primary Key) : รหัสประจำตัว ของการล็อกอินเข้าสู่ระบบ
- userID : ชื่อสำหรับการเข้าสู่ระบบ
- password : รหัสสำหรับการเข้าสู่ระบบ

12. TBL_CleaningRoomLineItem : สำหรับเก็บข้อมูลรายชื่อของ Cleaner ในห้องพัก

- roomID (Primary Key) : รหัสประจำตัว ของห้องพักแต่ละห้องพัก
- employeeID (Primary Key) : รหัสประจำตัว ของพนักงานแต่ละคน
- startDateTime : วันและเวลาเข้าทำงาน ของพนักงานแต่ละคน
- endDateTime : วันและเวลาออกทำงาน ของพนักงานแต่ละคน

13. TBL_coupons : สำหรับเก็บข้อมูลประเภทคูปองส่วนลด

- couponID (Primary Key) : รหัสประจำตัว ของประเภทคูปองส่วนลด
- name : ชื่อประเภทคูปองส่วนลด
- expireDate : วันและเวลาหมดอายุของคูปองส่วนลด

Description of constraints that can't explain in ER diagram

1. **Point:** คะแนนเก็บสะสมของลูกค้าจะหมดอายุการใช้งานทุก ๆ สิ้นปี
2. **Invoice:** ใบแจ้งหนี้สำหรับการจองห้องพักจะถูกยกเลิกอัตโนมัติ เมื่อจากการสร้างครบ 24 ชั่วโมง
3. **Coupon:** คูปองส่วนลดในการจองห้องพักของลูกค้า จะหมดอายุตามวันและเวลาที่ระบุไว้เท่านั้น
4. **Coupon:** คูปองส่วนลดในการจองห้องพักของลูกค้า จะไม่สามารถนำกลับมาใช้ซ้ำได้ หากเคยมีการใช้งานแล้ว

Schema

TBL_RoomCategorys

roomCatID	name	bedType	numberBed	guestRoom	fare
-----------	------	---------	-----------	-----------	------

TBL_Rooms

roomID	roomCatID	status	cleanStatus
--------	-----------	--------	-------------

TBL_Customers

customerID	firstname	familyname	email	phoneNumber	creditCardNumber	point
------------	-----------	------------	-------	-------------	------------------	-------

TBL_Auth

authID	userID	password
--------	--------	----------

TBL_Invoices

invoiceID	roomCatID	customerID	dateCreate	total	vat	amountDue	periodOfStay	checkIn	checkOut	numberOfRoom
-----------	-----------	------------	------------	-------	-----	-----------	--------------	---------	----------	--------------

TBL_InvoiceLineItem

invoiceID	roomID	remark
-----------	--------	--------

TBL_EmployeeType

employeeTypeID	name	salay
----------------	------	-------

TBL_Employees

employeeID	employeeTypeID	firstname	familyname	email	phoneNumber	acceptDate
------------	----------------	-----------	------------	-------	-------------	------------

TBL_Receipts

<u>receiptID</u>	paymentMetID	customerID	couponID	dateCreate	paymentRef	totalRecived	remark
------------------	--------------	------------	----------	------------	------------	--------------	--------

TBL_ReceiptsLineItem

<u>receiptID</u>	<u>invoiceID</u>	remark
------------------	------------------	--------

TBL_PaymentMethods

<u>paymentMetID</u>	name
---------------------	------

TBL_coupons

<u>couponID</u>	name	expireDate
-----------------	------	------------

TBL_CleaningRoomLineItem

<u>employeeID</u>	<u>roomID</u>	startDateTime	endDateTime
-------------------	---------------	---------------	-------------

How to normalization the database

1. จาก Requirement ต้องการให้ Customer สามารถดูประเภทของห้อง ราคา และ สถานะของห้องที่สามารถเข้าพักได้ ดังนั้นต้องมี Table roomcategory และ rooms สำหรับเก็บข้อมูลห้องพัก และ room status ที่ยังว่างอยู่ นอกจากนั้นต้องการเก็บข้อมูลของลูกค้าที่ทำการ register เพื่อทำการจองโรงแรมดังนั้นต้องมี Table สำหรับเก็บข้อมูลของ Customer หากลูกค้าต้องการจองห้องพักลูกค้าต้องทำการระบุ ประเภทของห้อง จำนวนของห้องที่ต้องการจอง จำนวนของลูกค้าที่เข้าพัก และจำนวนวันที่ต้องการจองก็สามารถจองได้แล้ว ดังนั้นต้องมี Table สำหรับเก็บใบสั่งซื้อ คือ invoices ต่อมาลูกค้าสามารถเลือกชำระเงินได้ผ่าน 3 ช่องทาง ข้อมูลในส่วนนี้จะถูกเก็บอยู่ใน Table Payment Method และหากลูกค้าทำการชำระเงินเรียบร้อยแล้ว ระบบทำการแสดง receipt เพื่อเป็นหลักฐานในการชำระเงิน โดยข้อมูลในส่วนนี้จะถูกเก็บอยู่ใน Table receipts
2. ในส่วนของ invoice ยังสามารถทำการ normalization แยกออกมาได้เป็น Table Invoice Lineltem เพื่อแสดงแต่ละรายการคำสั่งซื้อของแต่ละ room เพื่อป้องกันการ anomaly ในการ insert update หรือ delete ข้อมูลลงใน database
3. ในส่วนของ receipt ยังสามารถทำการ normalization แยกออกมาได้เป็น Table Receipt Lineltem เพื่อแสดงแต่ละรายการชำระเงินของแต่ละ invoice เพื่อป้องกันการ anomaly ในการ insert update หรือ delete ข้อมูลลงใน database
4. ในส่วนของ Password ที่อยู่ใน Table auth เนื่องจากสาเหตุของความปลอดภัยจึงมีการเก็บข้อมูลในส่วนนี้แยก กับ Table Customer

Example Data

	roomCatID [PK] character (6)	name character (20)	bedType character (20)	numberBed integer	guestRoom boolean	fare numeric (10,2)
1	rc0001	Single	normal bed	1	false	1500.00
2	rc0002	Double	normal bed	2	false	2500.00
3	rc0003	Suite	queen-size bed	1	true	3000.00
4	rc0004	Delux	queen-size bed	1	true	4000.00
5	rc0005	Premier	king-size bed	2	true	5000.00

รูปที่ 3 แสดง ข้อมูลในตาราง TBL_RoomCategorys

	customerID [PK] character (6)	firstname character varying (20)	familyname character varying (20)	email character varying (50)	phoneNumber character (10)	creditCardNumber character (16)	point integer
1	c00001	Somchai	Moon	Somchai@gmail.com	0922638123	1234567891234520	300
2	c00002	Somsee	Metang	Somsee@gmail.com	0922628123	1234567894234520	300
3	c00003	Somna	Meban	Somna@gmail.com	0952638193	-	500
4	c00004	Sompu	Meplane	Sompu@gmail.com	0926638193	-	500
5	c00005	Thanathron	Rangbui	Thanathron@gmail.com	0922638173	-	2
6	c00006	Prawit	Borrowfriend	Prawit@gmail.com	0942668123	4234567891236520	10
7	c00007	Kukai	opop	Kukai@gmail.com	0972678123	7234567895236520	10
8	c00008	Mesawa	Kawasaki	Mesawa@gmail.com	0928638128	2234567532126520	0
9	c00009	Jelly	apple	Jelly@gmail.com	0912638127	3232558532126520	32
10	c00010	Sam	Thailand	Sam@gmail.com	0422638133	-	300
11	c00011	Kati	Aroy	Kati@gmail.com	0857742156	4252662845963512	300
12	c00012	Lina	Jang	Lina@gmail.com	0563254211	4852251263524857	300
13	c00013	Mina	Home	Mina@gmail.com	0565214169	-	0
14	c00014	Natthan	Bhukan	testemail@gmail.com	0922638737	11	100
15	c00015	Natthan22	Bhukan	1232ed2@gmail.com	0922638737	11	100
16	c00016	dsds	Bhukan	12323d@gmail.com	0922638737	11	100
17	c00017	231232	23123	23123@gmail.com	23123	11	100
18	c00018	TestTae	Bhukan	123wdwd@gmail.com	0922638737	11	100
19	c00019	Natthan	Bhukan	231232@gmail.com	32123231	11	100

รูปที่ 4 แสดง ข้อมูลในตาราง TBL_Customers

	authID [PK] character (6)	customerID character (6)	password character varying (20)
1	a00002	c00002	123qweasd
2	a00003	c00003	123qweasd
3	a00004	c00004	123qweasd
4	a00005	c00005	123qweasd
5	a00006	c00006	123qweasd
6	a00007	c00007	123qweasd
7	a00008	c00008	123qweasd
8	a00009	c00009	123qweasd
9	a00010	c00010	123qweasd
10	a00011	c00014	123wqe
11	a00012	c00015	123qwe
12	a00013	c00016	Teat21d
13	a00014	c00017	123231
14	a00015	c00018	123qwe
15	a00016	c00019	123qwe

รูปที่ 5 แสดง ข้อมูลในตาราง TBL_Auth

	roomID [PK] character (6)	roomCatID character (6)	status boolean	cleanStatus boolean
1	r00001	rc0001	false	true
2	r00002	rc0001	false	true
3	r00003	rc0001	false	true
4	r00004	rc0001	false	true
5	r00005	rc0001	false	true
6	r00006	rc0002	false	false
7	r00007	rc0002	false	false
8	r00008	rc0002	false	false
9	r00009	rc0002	true	true
10	r00010	rc0003	false	true
11	r00011	rc0003	false	true
12	r00012	rc0003	false	true
13	r00013	rc0004	false	false
14	r00014	rc0004	true	false
15	r00015	rc0004	false	true
16	r00016	rc0004	true	true
17	r00017	rc0005	false	true
18	r00018	rc0005	true	true
19	r00019	rc0005	true	true
20	r00020	rc0005	false	true
21	r00021	rc0001	true	true
22	r00022	rc0001	true	true
23	r00023	rc0001	true	true
24	r00024	rc0001	true	true
25	r00025	rc0001	true	true
26	r00026	rc0001	true	true
27	r00027	rc0001	true	true
28	r00028	rc0001	true	true
29	r00029	rc0001	true	true
30	r00030	rc0001	true	true
31	r00031	rc0001	true	true
32	r00032	rc0002	true	true
33	r00033	rc0002	true	true
34	r00034	rc0002	true	true
35	r00035	rc0002	true	true
36	r00036	rc0002	true	true
37	r00037	rc0002	true	true
38	r00038	rc0002	true	true
39	r00039	rc0002	true	true
40	r00040	rc0002	true	true
41	r00041	rc0003	true	true
42	r00042	rc0003	true	true
43	r00043	rc0003	true	true
44	r00044	rc0003	true	true
45	r00045	rc0003	true	true
46	r00046	rc0003	true	true
47	r00047	rc0003	true	true
48	r00048	rc0003	true	true
49	r00049	rc0003	true	true
50	r00050	rc0003	true	true
51	r00051	rc0004	true	true
52	r00052	rc0004	true	true
53	r00053	rc0004	true	true
54	r00054	rc0004	true	true
55	r00055	rc0004	true	true
56	r00056	rc0004	true	true
57	r00057	rc0004	true	true
58	r00058	rc0004	true	true
59	r00059	rc0004	true	true
60	r00060	rc0004	true	true
61	r00061	rc0005	true	true
62	r00062	rc0005	true	true
63	r00063	rc0005	true	true
64	r00064	rc0005	true	true
65	r00065	rc0005	true	true
66	r00066	rc0005	true	true
67	r00067	rc0005	true	true
68	r00068	rc0005	true	true
69	r00069	rc0005	true	true
70	r00070	rc0005	true	true

รูปที่ 6 แสดง ข้อมูลในตาราง TBL_Rooms

invoiceID [PK] character (6)	roomCatID character (6)	customerID character (6)	dateCreate timestamp without time zone	total numeric (10,2)	vat numeric (10,2)	amountDue numeric (10,2)	periodOfStay integer	checkin timestamp without time zone	checkout timestamp without time zone	numberOfRoom integer	
1	i00001	rc0003	c00001	2020-11-10 12:00:00	18000.00	1260.00	19260.00	3	2020-11-15 00:00:00	2020-11-18 00:00:00	2
2	i00002	rc0002	c00010	2020-11-10 09:20:00	25000.00	1750.00	26750.00	5	2020-11-17 00:00:00	2020-11-22 00:00:00	2
3	i00003	rc0003	c00005	2020-11-10 18:00:00	12000.00	840.00	12840.00	4	2020-11-15 00:00:00	2020-11-19 00:00:00	1
4	i00004	rc0001	c00009	2020-11-10 07:30:00	30000.00	2100.00	32100.00	4	2020-10-11 00:00:00	2020-11-24 00:00:00	5
5	i00005	rc0005	c00008	2020-11-10 09:40:00	25000.00	1750.00	26750.00	5	2020-11-23 00:00:00	2020-11-28 00:00:00	1
6	i00006	rc0003	c00009	2020-12-14 00:00:00	12000.00	840.00	12840.00	4	2020-12-14 00:00:00	2020-12-18 00:00:00	1

รูปที่ 7 แสดง ข้อมูลในตาราง TBL_Invoices

	invoiceID [PK] character (6)	roomID [PK] character (6)	remark character (50)
1	i00001	r00011	-
2	i00001	r00012	-
3	i00002	r00006	-
4	i00002	r00007	-
5	i00003	r00010	-
6	i00004	r00001	-
7	i00004	r00002	-
8	i00004	r00003	-
9	i00004	r00004	-
10	i00004	r00005	-
11	i00005	r00017	-

รูปที่ 8 แสดง ข้อมูลในตาราง TBL_InvoiceLineItem

	employeeTypeID [PK] character (1)	name character (20)	salary numeric (10,2)
1	1	Saleperson	30000.00
2	2	Security Guard	15000.00
3	3	Chef	40000.00
4	4	Cleaner	20000.00

รูปที่ 9 แสดง ข้อมูลในตาราง TBL_EmployeeType

	employeeID [PK] character (6)	employeeTypeID character (1)	firstname character (20)	familyname character (20)	email character (50)	phoneNumber character (10)	accepteDate date
1	e00001	1	Yee	Yao	Yao@gmail.com ...	0839620821	2010-12-04
2	e00002	1	Su	Lama	Lama@gmail.com ...	0917473165	2010-11-04
3	e00003	2	Jo	Kaka	Jo@gmail.com ...	0725846378	2015-11-05
4	e00004	2	Ake	Dawin	Ake@gmail.com ...	0653416035	2015-11-05
5	e00005	3	Lou	Clarke	mo@tolki.pt ...	0215703898	2015-11-05
6	e00006	3	Georgie	Griffith	aj@ezwijum.na ...	0357213626	2015-11-05
7	e00007	3	Norman	Harrison	bomosvo@sukacis...	0139522289	2015-11-05
8	e00008	3	Jordan	Logan	gogokjo@hopami.a...	0128228238	2010-12-04
9	e00009	3	Joshua	Carpenter	huar@bubuppih.au	0125441400	2010-12-04
10	e00010	3	Michael	Harris	kialowis@ram.nc	0250047613	2013-08-05
11	e00011	3	Olive	Mayer	pinuz@hetovad.ar ...	0175121565	2013-08-05
12	e00012	3	Olga	Holland	zucu@ubu.pm ...	0809081266	2013-08-05
13	e00013	3	Louise	McDaniel	cu@ipibap.cv ...	0125471400	2013-08-12
14	e00014	4	Travis	Oliver	ebuwe@udma.mc	0250147613	2013-08-15
15	e00015	4	Kasja	Wqd	biw@mikam.sj ...	0175121564	2013-08-15
16	e00016	4	Kdwq	dwad	be@emfuku.mm ...	0125441401	2013-08-13

รูปที่ 10 แสดง ข้อมูลในตาราง TBL_Employees

	receiptID [PK] character (6)	customerID character (6)	paymentMedId character (6)	couponID character (6)	dateCreate timestamp without time zone	paymentRef character (30)	totalReceived numeric (10,2)	remark character (50)
1	re0001	c00001	1	co0001	2020-11-10 12:00:00	cash	19260.00	-
2	re0002	c00010	1	co0002	2020-11-10 09:20:00	cash	26750.00	-
3	re0003	c00005	2	co0001	2020-11-10 18:00:00	master card	12840.00	-
4	re0004	c00009	2	co0002	2020-11-11 07:30:00	visa	32100.00	-
5	re0005	c00008	1	co0001	2020-11-11 09:40:00	cash	26750.00	-
6	re0006	c00008	1	co0001	2020-12-11 00:00:00	cash	20000.00	Test

รูปที่ 11 แสดง ข้อมูลในตาราง TBL_Receipts

	receiptID [PK] character (6)	remark character (50)	invoiceID [PK] character (6)
1	re0001	-	i00001
2	re0002	-	i00002
3	re0003	-	i00003
4	re0004	-	i00004
5	re0005	-	i00005

รูปที่ 12 แสดง ข้อมูลในตาราง TBL_ReceiptsLineItem

	paymentMedId [PK] character (6)	name character (15)
1	1	cash
2	2	credit card
3	3	bit coin

รูปที่ 13 แสดง ข้อมูลในตาราง TBL_PaymentMethods

	couponID [PK] character (6)	name character (20)	exipreDate date
1	co0001	50 % off	2020-12-31
2	co0002	25 % off	2020-12-31

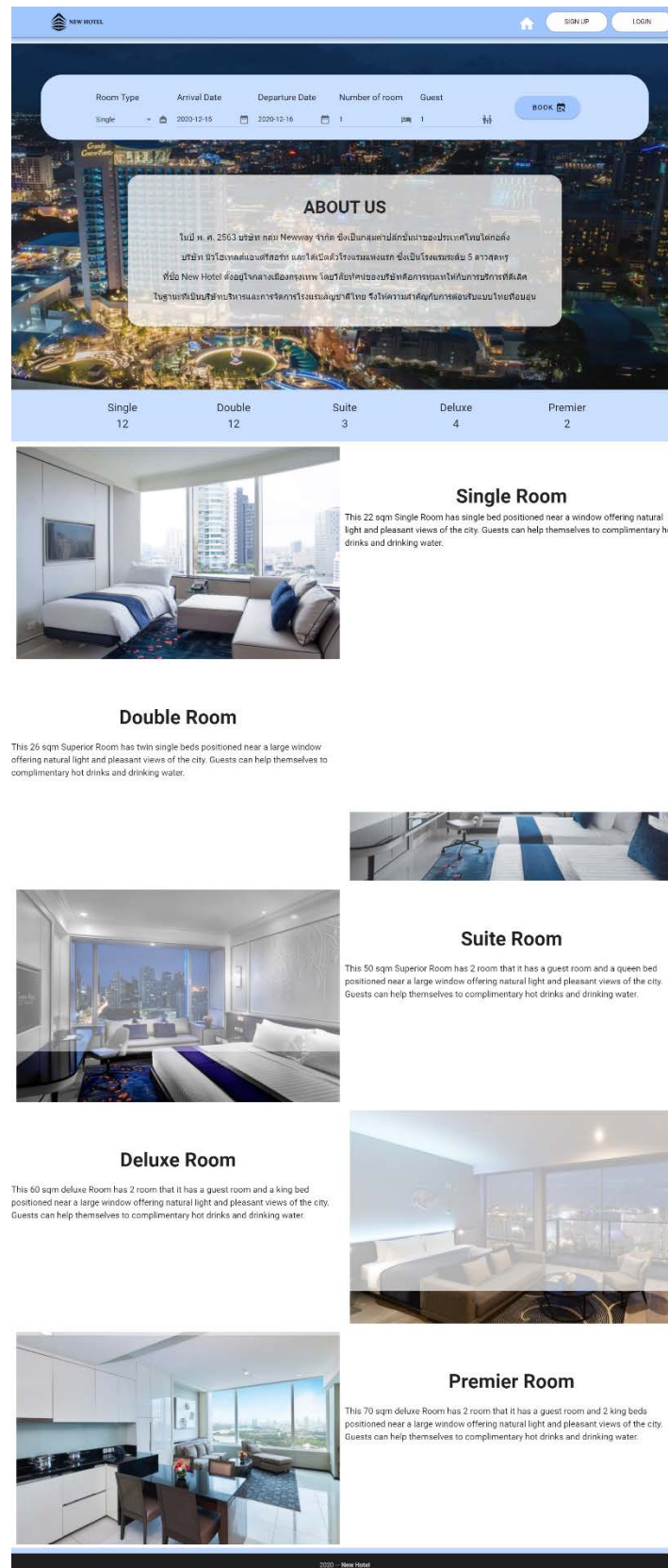
รูปที่ 14 แสดง ข้อมูลในตาราง TBL_Coupons

	employeeID [PK] character (6)	roomID [PK] character (6)	startDateTime timestamp without time zone	endDateTime timestamp without time zone
1	e00014	r00002	2020-10-11 11:00:00	2020-10-11 12:00:00
2	e00014	r00003	2020-10-11 11:00:00	2020-10-11 12:00:00
3	e00015	r00004	2020-11-11 11:00:00	2020-11-11 12:00:00
4	e00015	r00005	2020-11-11 11:00:00	2020-11-11 12:00:00
5	e00016	r00010	2020-12-11 11:00:00	2020-12-11 12:00:00
6	e00016	r00011	2020-12-11 11:00:00	2020-12-11 12:00:00

รูปที่ 15 แสดง ข้อมูลในตาราง TBL_CleaningRoomLineItem

หน้า User Interface

หน้า User Interface ของสำหรับ User ที่ยังไม่ได้ Login



ABOUT US

ในปี พ.ศ. 2563 บริษัท Newway จำกัด จึงเป็นกลุ่มบริษัทชั้นนำของประเทศไทยโดยมี
บริษัท นิวไฮเทคเคอเนชั่นส์ จำกัด เป็นบริษัทแม่ของ Newway จำกัด ซึ่งเป็นบริษัทแม่ของ 5 บริษัท
ที่ New Hotel ตั้งอยู่ในเมืองกรุงเทพฯ โดยมีวัตถุประสงค์ของบริษัทคือการให้บริการที่พักและบริการที่
ในด้านการเงินและการบริการที่สมบูรณ์แบบที่สุด เพื่อให้ลูกค้าสามารถใช้บริการได้อย่างสะดวกสบาย

Single Room

This 22 sqm Single Room has single bed positioned near a window offering natural light and pleasant views of the city. Guests can help themselves to complimentary hot drinks and drinking water.

Double Room

This 26 sqm Superior Room has twin single beds positioned near a large window offering natural light and pleasant views of the city. Guests can help themselves to complimentary hot drinks and drinking water.

Suite Room

This 50 sqm Superior Room has 2 room that it has a guest room and a queen bed positioned near a large window offering natural light and pleasant views of the city. Guests can help themselves to complimentary hot drinks and drinking water.

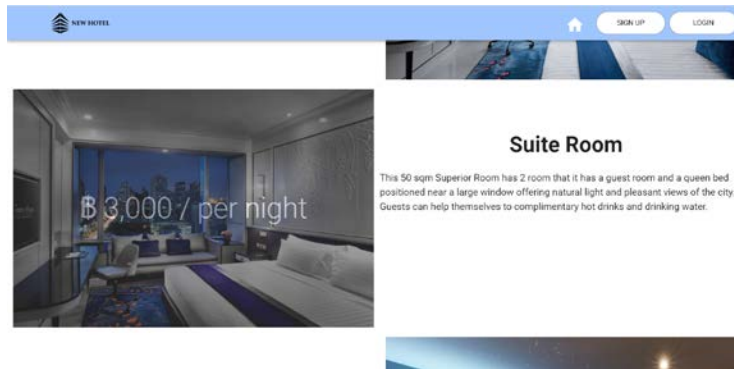
Deluxe Room

This 60 sqm deluxe Room has 2 room that it has a guest room and a king bed positioned near a large window offering natural light and pleasant views of the city. Guests can help themselves to complimentary hot drinks and drinking water.

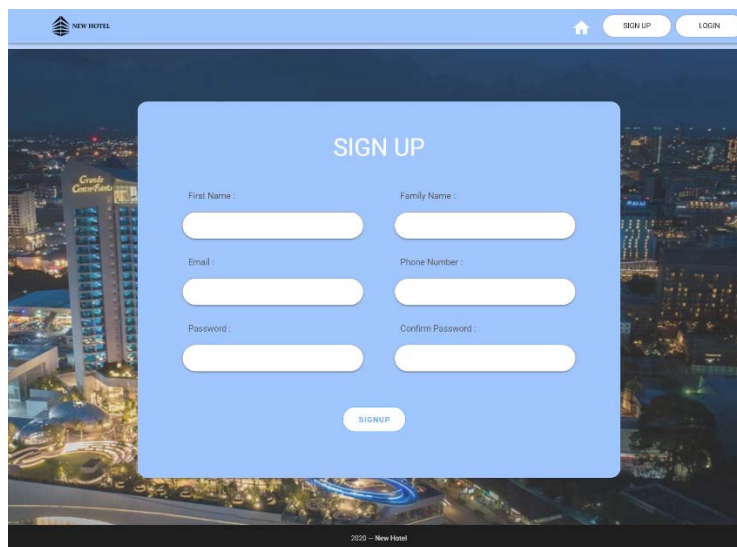
Premier Room

This 70 sqm deluxe Room has 2 room that it has a guest room and 2 king beds positioned near a large window offering natural light and pleasant views of the city. Guests can help themselves to complimentary hot drinks and drinking water.

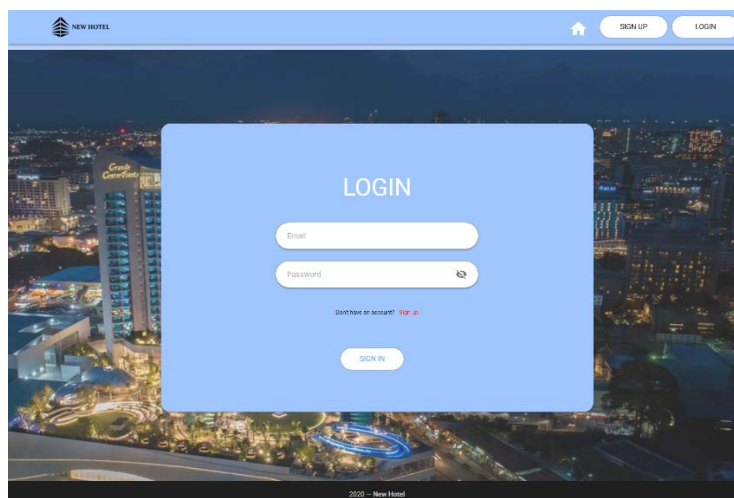
รูปที่ 16 แสดงหน้า Home สำหรับ Guest ที่ยังไม่ทำการเข้าสู่ระบบ



รูปที่ 17 แสดงราคาของแต่ละประเภทห้อง เมื่อนำ cursor ไปชี้

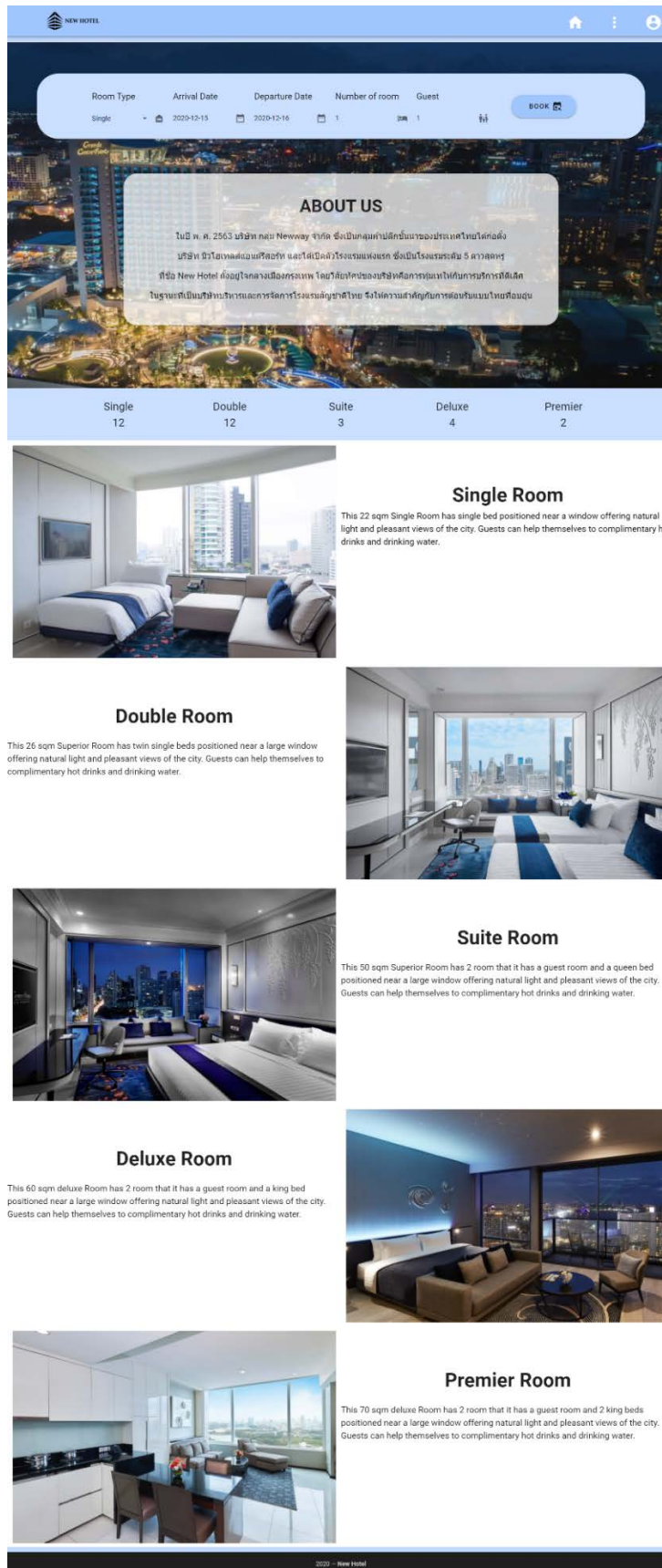


รูปที่ 18 แสดงหน้า Sign up



รูปที่ 19 แสดงหน้า Login

หน้า User Interface ของ ลูกค้าที่ต้องการจองห้องพัก



The screenshot displays the New Hotel website's booking interface. At the top, there's a navigation bar with the hotel logo and icons for home, menu, and user profile. Below this is a search bar with fields for Room Type (Single), Arrival Date (2020-12-15), Departure Date (2020-12-16), Number of room (1), and Guest (1). A 'BOOK' button is visible. The main content area features an 'ABOUT US' section with Thai text describing the hotel's location and amenities. Below this is a table listing room types and their counts: Single (12), Double (12), Suite (3), Deluxe (4), and Premier (2). The bottom section shows detailed views of each room type: Single Room, Double Room, Suite Room, Deluxe Room, and Premier Room, each with a description of its features and amenities.

ABOUT US

ในปี พ.ศ. 2563 บริษัท Newway จำกัด ซึ่งเป็นกลุ่มทุนหลักของธนาคารไทยพาณิชย์
บริษัท นิวโฮเทลแอนด์รีสอร์ท จำกัดได้เปิดตัวโรงแรมแห่งแรก ซึ่งเป็นโรงแรมระดับ 5 ดาวสองฟุ
ที่ชื่อ New Hotel ตั้งอยู่ในใจกลางเมืองกรุงเทพ ใกล้กับท่าอากาศยานสุวรรณภูมิให้บริการที่พัก
ในฐานะที่เป็นบริษัทบริหารจัดการโรงแรมชั้นนำของไทย จึงได้ความสำคัญในการออกแบบเว็บไซต์ด้วย

Room Type	Count
Single	12
Double	12
Suite	3
Deluxe	4
Premier	2

Single Room

This 22 sqm Single Room has single bed positioned near a window offering natural light and pleasant views of the city. Guests can help themselves to complimentary hot drinks and drinking water.

Double Room

This 26 sqm Superior Room has twin single beds positioned near a large window offering natural light and pleasant views of the city. Guests can help themselves to complimentary hot drinks and drinking water.

Suite Room

This 50 sqm Superior Room has 2 room that it has a guest room and a queen bed positioned near a large window offering natural light and pleasant views of the city. Guests can help themselves to complimentary hot drinks and drinking water.

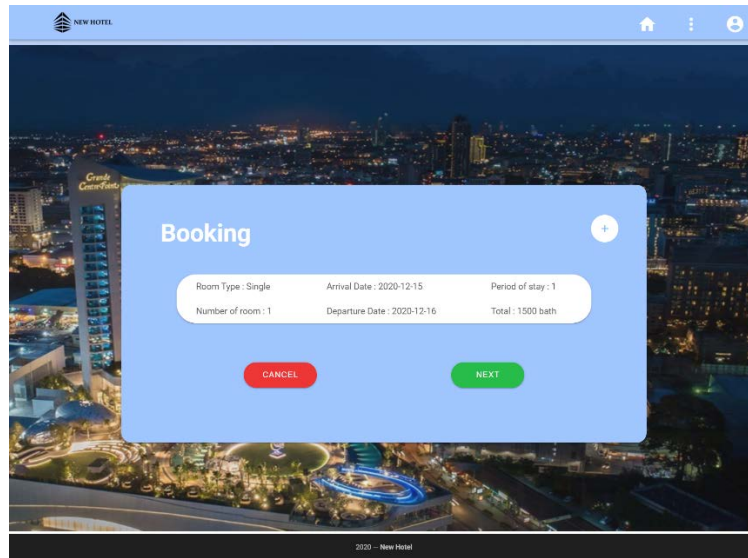
Deluxe Room

This 60 sqm deluxe Room has 2 room that it has a guest room and a king bed positioned near a large window offering natural light and pleasant views of the city. Guests can help themselves to complimentary hot drinks and drinking water.

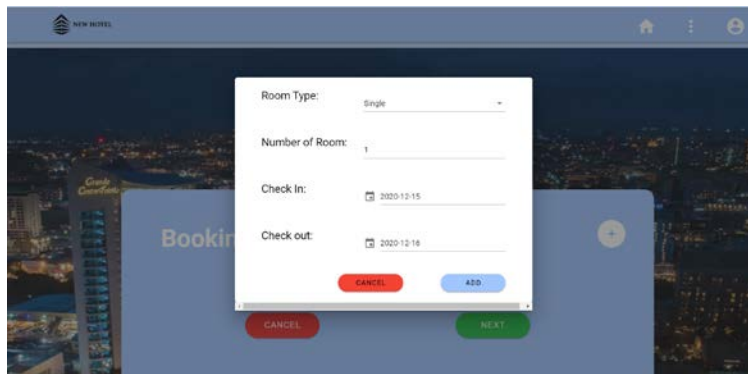
Premier Room

This 70 sqm deluxe Room has 2 room that it has a guest room and 2 king beds positioned near a large window offering natural light and pleasant views of the city. Guests can help themselves to complimentary hot drinks and drinking water.

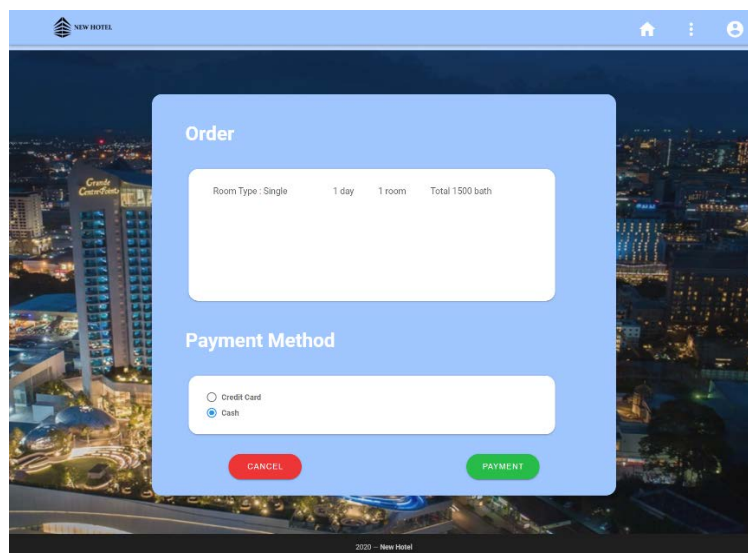
รูปที่ 20 แสดงหน้า Home สำหรับ Guest ที่ทำการเข้าสู่ระบบแล้ว



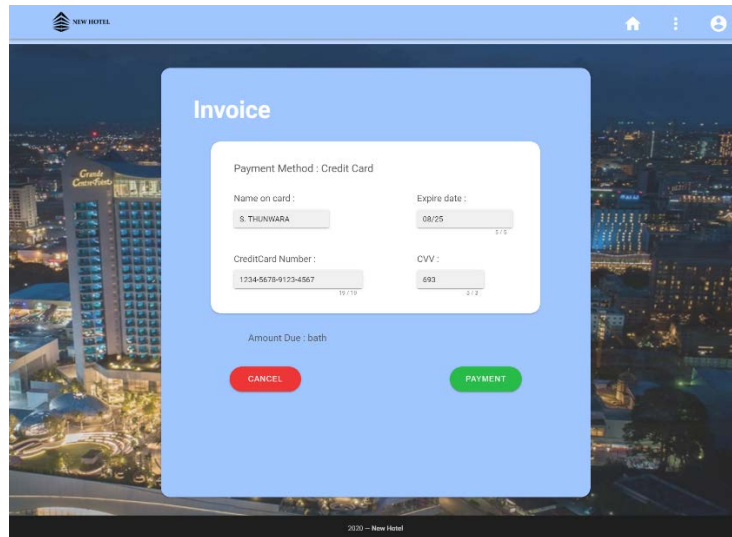
รูปที่ 21 แสดงหน้า Booking ที่มีข้อมูลการจอง



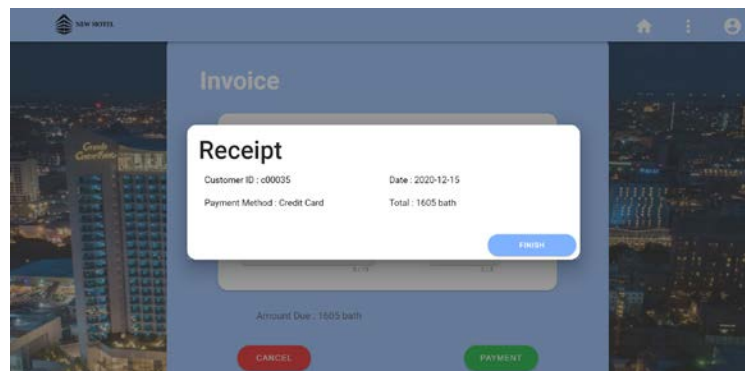
รูปที่ 22 แสดงหน้าต่าง pop-up เพื่อทำการ booking เพิ่ม



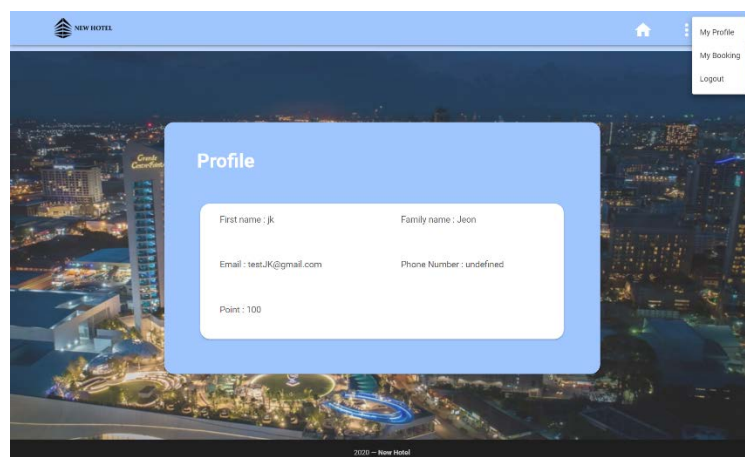
รูปที่ 23 แสดงหน้า Order รวม ที่ทำการ booking และ Payment Method



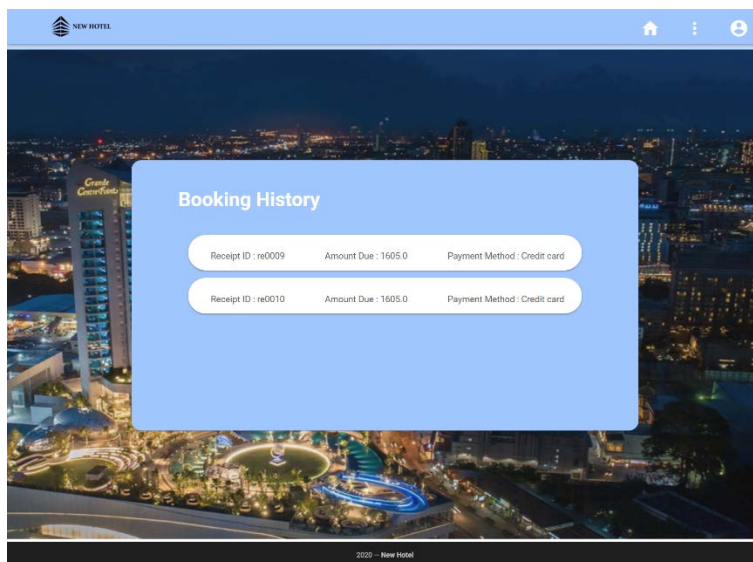
รูปที่ 24 แสดงหน้า Invoice กรณีเลือกชำระเงินด้วย Credit card



รูปที่ 25 แสดงหน้าต่าง pop-up เพื่อแสดง Receipt

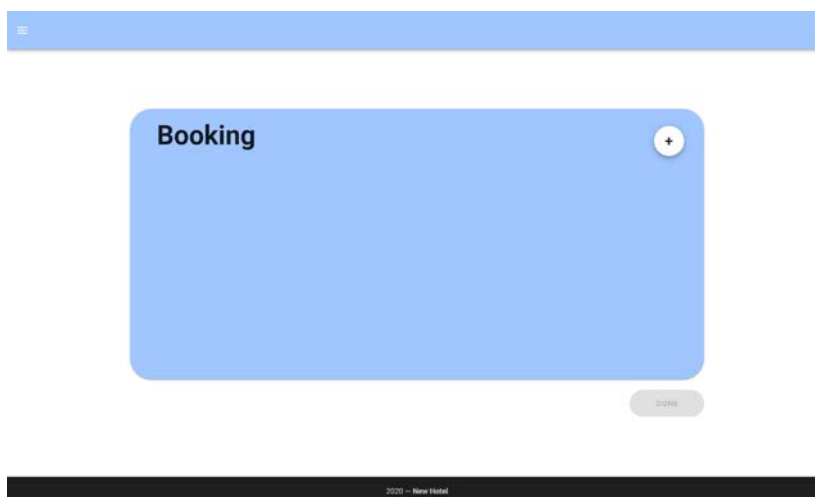


รูปที่ 26 แสดงหน้า Profile

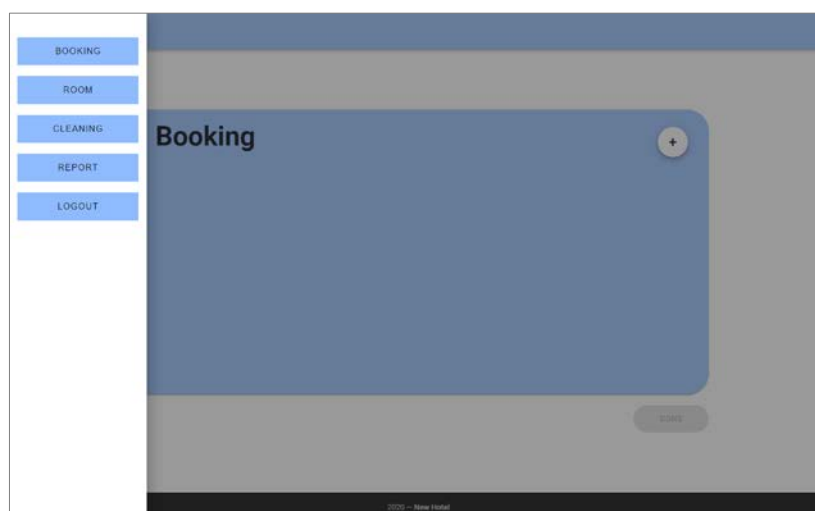


รูปที่ 27 แสดงหน้า Booking History

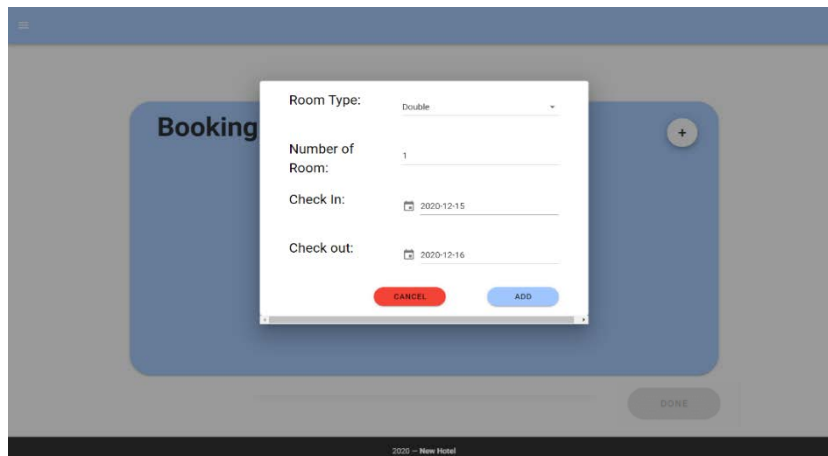
หน้า User Interface ของ Admin



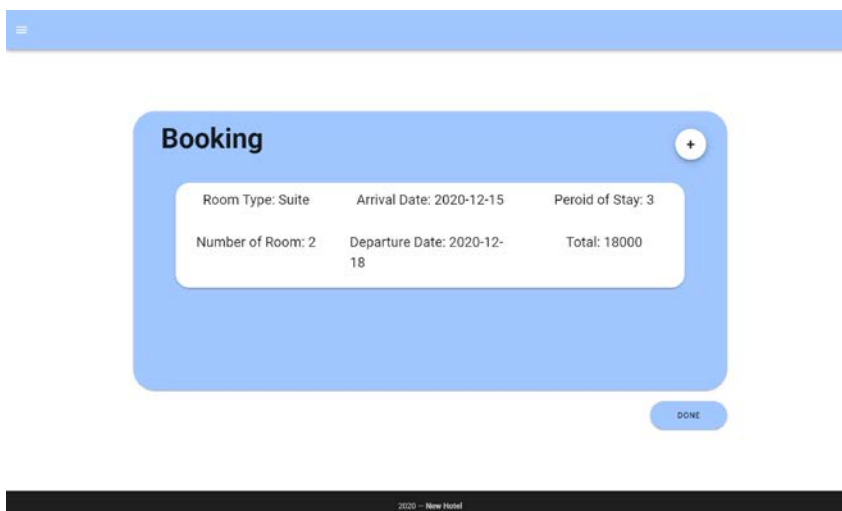
รูปที่ 28 แสดงหน้า Home ของ Admin



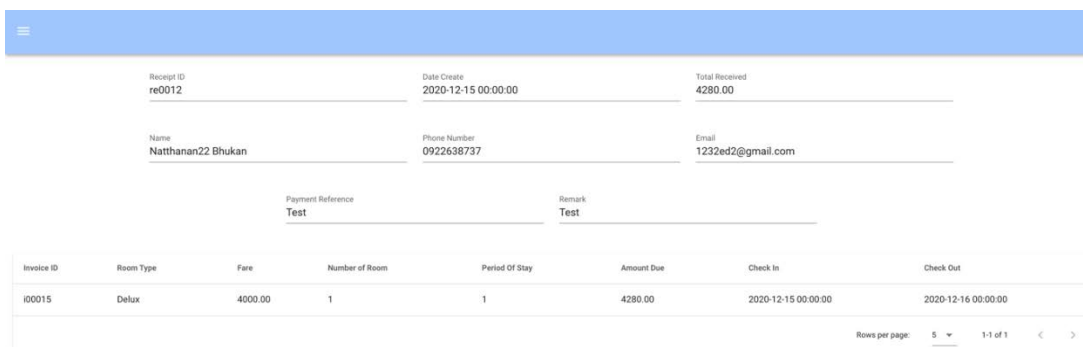
รูปที่ 29 แสดงแถบเมนู ของ Admin



รูปที่ 30 แสดงหน้า Booking เพื่อจองห้องพักของ Admin

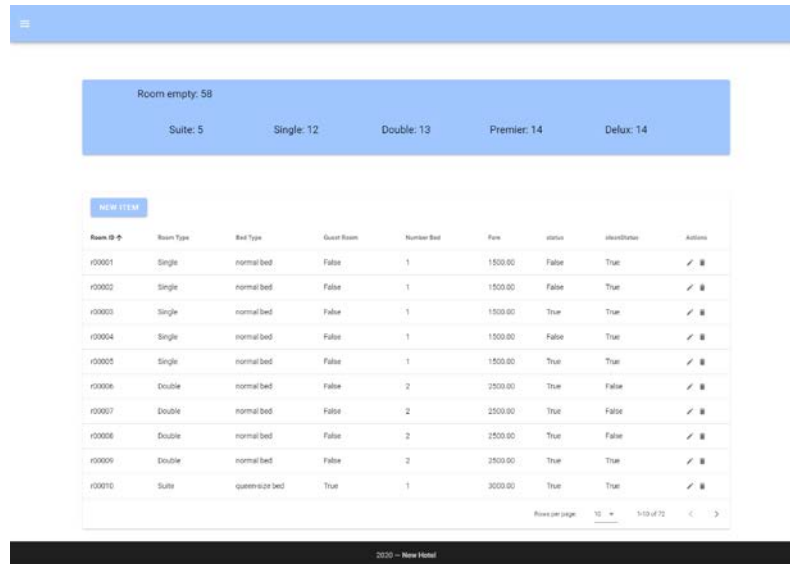


รูปที่ 31 แสดงหน้า Booking เมื่อกดจองห้องเพิ่ม ของ Admin

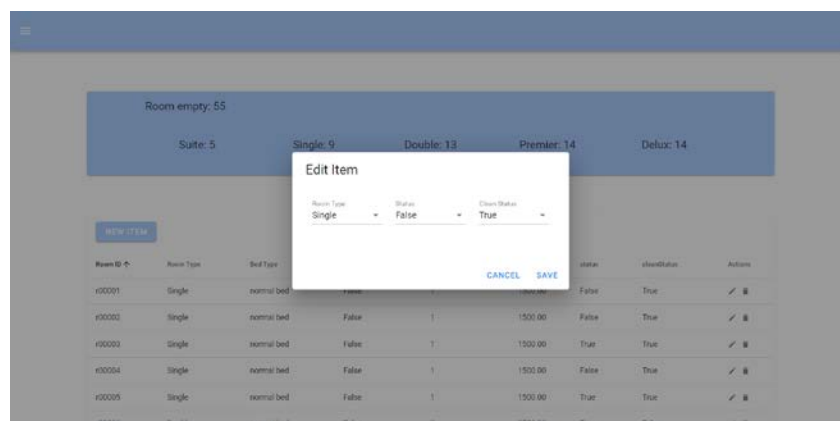


Invoice ID	Room Type	Fare	Number of Room	Period Of Stay	Amount Due	Check In	Check Out
i00015	Delux	4000.00	1	1	4280.00	2020-12-15 00:00:00	2020-12-16 00:00:00

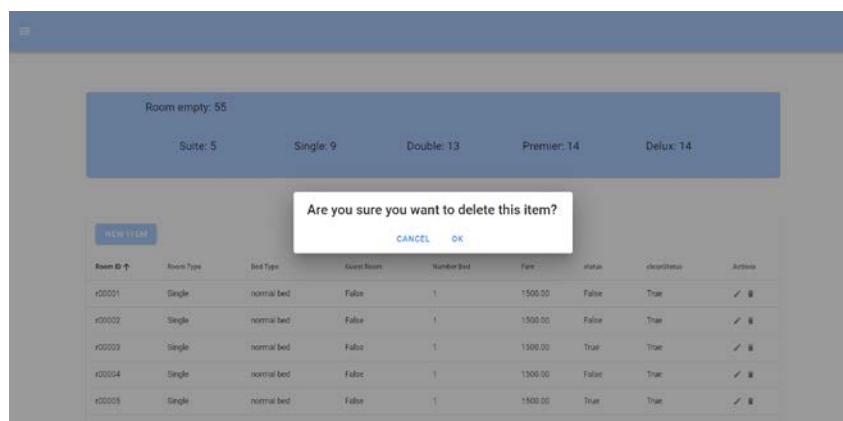
รูปที่ 32 แสดงหน้า Print receipt เมื่อกด print หลังจากทำการ Booking ของ Admin



รูปที่ 33 แสดงหน้า Room เมื่อกดปุ่ม Room จากรูปที่ 29



รูปที่ 34 แสดง Pop up เพื่อแก้ไข เมื่อกดปุ่มรูปดินสอ จากรูปที่ 32



รูปที่ 35 แสดง Pop up เพื่อยืนยันการลบ เมื่อกดปุ่มรูปถังขยะ จากรูปที่ 32

Employee ID	Room ID	Start Date Time	End Date Time
e00014	r00002	Sun, 11 Oct 2020 11:00:00 GMT	Sun, 11 Oct 2020 12:00:00 GMT
e00014	r00003	Sun, 11 Oct 2020 11:00:00 GMT	Sun, 11 Oct 2020 12:00:00 GMT
e00015	r00004	Wed, 11 Nov 2020 11:00:00 GMT	Wed, 11 Nov 2020 12:00:00 GMT
e00015	r00005	Wed, 11 Nov 2020 11:00:00 GMT	Wed, 11 Nov 2020 12:00:00 GMT
e00016	r00010	Fri, 11 Dec 2020 11:00:00 GMT	Fri, 11 Dec 2020 12:00:00 GMT
e00016	r00011	Fri, 11 Dec 2020 11:00:00 GMT	Fri, 11 Dec 2020 12:00:00 GMT

Rows per page: 10 14 of 6

รูปที่ 36 แสดงหน้า Cleaning เมื่อกดปุ่ม Cleaning จากรูปที่ 29

Receipt ID	Date Create	Name	Payment	payment Reference	Remark	Total Received
re0001	2020-11-10 12:00:00	Somchai Moon	cash	cash	-	19260.00
re0005	2020-11-11 09:40:00	Messura Kawasaki	cash	cash	+	26750.00
re0004	2020-11-11 07:30:00	Jelly apple	credit card	visa	-	82100.00
re0003	2020-11-10 18:00:00	Thunathorn Rongbui	credit card	master card	-	12840.00
re0002	2020-11-10 09:20:00	Sam Thailand	cash	cash	-	26750.00
re0008	2020-12-15 00:00:00	Nathanan22 Bhukan	credit card	Cash	Test	52965.00
re0007	2020-12-15 00:00:00	Nathanan22 Bhukan	credit card	Cash	Test	1605.00
re0006	2020-12-15 00:00:00	Nathanan22 Bhukan	credit card	Cash	Test	1605.00
re0010	2020-12-15 00:00:00	jk Jecm	credit card	Cash	Test	1605.00
re0009	2020-12-15 00:00:00	jk Jecm	credit card	Cash	Test	1605.00

Rows per page: 10 112 of 18

รูปที่ 37 แสดงหน้า Receipt เมื่อกดปุ่ม Receipt จากรูปที่ 29

☰
RECEIPT
INVOICE

Invoice ID	Date Create	Check in	Check out	Number of Room	Period Of Stay	Room type	Total	Vat	Amount Due	Name
IK0001	2020-11-10 12:00:00	2020-11-15 00:00:00	2020-11-18 00:00:00	2	3	Suite	18000.00	1260.00	19260.00	Somchai Moon
IK0005	2020-11-10 09:40:00	2020-11-28 00:00:00	2020-11-28 00:00:00	1	5	Premier	25000.00	1750.00	26750.00	Mesana Kawasaki
IK0004	2020-11-10 07:50:00	2020-10-11 00:00:00	2020-11-24 00:00:00	5	4	Single	35000.00	2100.00	32100.00	Jelly apple
IK0003	2020-11-10 18:00:00	2020-11-15 00:00:00	2020-11-19 00:00:00	1	4	Suite	12000.00	840.00	12840.00	Thanatworn Rangbui
IK0002	2020-11-10 09:20:00	2020-11-17 00:00:00	2020-11-22 00:00:00	2	5	Double	25000.00	1750.00	26750.00	Sam Thailand
IK0010	2020-12-15 00:00:00	2020-12-15 00:00:00	2020-12-16 00:00:00	1	1	Single	1500.00	105.00	1605.00	Nattanan22 Bhukan
IK0009	2020-12-15 00:00:00	2020-12-15 00:00:00	2020-12-16 00:00:00	1	1	Single	1500.00	105.00	1605.00	Nattanan22 Bhukan
IK0007	2020-12-15 00:00:00	2020-12-15 00:00:00	2020-12-16 00:00:00	1	1	Single	1500.00	105.00	1605.00	Nattanan22 Bhukan
IK0006	2020-12-15 00:00:00	2020-12-15 00:00:00	2020-12-16 00:00:00	1	1	Single	1500.00	105.00	1605.00	Nattanan22 Bhukan
IK0011	2020-12-15 00:00:00	2020-12-15 00:00:00	2020-12-19 00:00:00	4	4	Suite	48000.00	3360.00	51360.00	Nattanan22 Bhukan

Rows per page: 10 1-10 of 13

รูปที่ 38 แสดงหน้า Invoice เมื่อกดปุ่ม Invoice ด้านบน

SQL command to create table

***** "TBL_RoomCategorys" *****

```
CREATE TABLE "TBL_RoomCategorys"
(
    "roomCatID" char(6) NOT NULL,
    "name"      char(20) NOT NULL,
    "bedType"   char(20) NOT NULL,
    "numberBed" int NOT NULL,
    "guestRoom" boolean NOT NULL,
    "fare"      numeric(10,2) NOT NULL,
    CONSTRAINT "PK_tbl_roomcategorys" PRIMARY KEY ( "roomCatID" )
);
```

***** "TBL_PaymentMedthods"*****

```
CREATE TABLE "TBL_PaymentMedthods"
(
    "paymentMedId" char(6) NOT NULL,
    "name"         char(10) NOT NULL,
    CONSTRAINT "PK_tbl_paymentmedthods" PRIMARY KEY ( "paymentMedId" )
);
```

***** "TBL_EmployeeTypes"*****

```
CREATE TABLE "TBL_EmployeeTypes"
(
    "employeeTypeID" char(1) NOT NULL,
    "name"           char(20) NOT NULL,
    "salary"         numeric(10,2) NOT NULL,
    CONSTRAINT "PK_tbl_employeetypes" PRIMARY KEY ( "employeeTypeID" )
);
```

***** "TBL_Customer"*****

```
CREATE TABLE "TBL_Customer"
(
    "customerID"    char(6) NOT NULL,
    "firstname"     char(20) NOT NULL,
    "familyname"    char(20) NOT NULL,
    "email"         char(50) NOT NULL DEFAULT unique,
    "phoneNumber"   char(10) NULL,
    "creditCardNumber" char(16) NULL,
    "point"         int NULL,
    CONSTRAINT "PK_tbl_customer" PRIMARY KEY ( "customerID" )
);
```

***** "TBL_Coupons"*****

```
CREATE TABLE "TBL_Coupons"
(
    "couponID"    char(6) NOT NULL,
    "name"        char(20) NOT NULL,
    "exipreDate"  timestamp NOT NULL,
    CONSTRAINT "PK_tbl_coupons" PRIMARY KEY ( "couponID" )
);
```

*****"TBL_Rooms"*****

```
CREATE TABLE "TBL_Rooms"
(
    "roomID"      char(6) NOT NULL,
    "roomCatID"   char(6) NOT NULL,
    "status"      boolean NOT NULL,
    "cleanStatus" boolean NOT NULL,
    CONSTRAINT "PK_tbl_rooms" PRIMARY KEY ( "roomID" ),
    CONSTRAINT "FK_61" FOREIGN KEY ( "roomCatID" ) REFERENCES "TBL_RoomCategorys" (
        "roomCatID" )
);
```

```
CREATE INDEX "fkIdx_61" ON "TBL_Rooms"
```

```
(
  "roomCatID"
);
```

```
*****"TBL_Receipts"*****
```

```
CREATE TABLE "TBL_Receipts"
```

```
(
  "receiptID"    char(6) NOT NULL,
  "customerID"   char(6) NOT NULL,
  "paymentMedId" char(6) NOT NULL,
  "couponID"     char(6) NOT NULL,
  "dateCreate"   timestamp NOT NULL,
  "paymentRef"   char(30) NOT NULL,
  "totalReceived" numeric(10,2) NOT NULL,
  "remark"       char(50) NULL,
  CONSTRAINT "PK_tbl_receipts" PRIMARY KEY ( "receiptID" ),
  CONSTRAINT "FK_167" FOREIGN KEY ( "customerID" ) REFERENCES "TBL_Customer" (
    "customerID" ),
  CONSTRAINT "FK_174" FOREIGN KEY ( "paymentMedId" ) REFERENCES
    "TBL_PaymentMedthods" ( "paymentMedId" ),
  CONSTRAINT "FK_182" FOREIGN KEY ( "couponID" ) REFERENCES "TBL_Coupons" ( "couponID" )
);
```

```
CREATE INDEX "fkIdx_167" ON "TBL_Receipts"
```

```
(
  "customerID"
);
```

```
CREATE INDEX "fkIdx_174" ON "TBL_Receipts"
```

```
(
  "paymentMedId"
);
```

```
CREATE INDEX "fkIdx_182" ON "TBL_Receipts"
```

```
( "couponID"
);
```

*****"TBL_Invoices"*****

```
CREATE TABLE "TBL_Invoices"
(
    "invoiceID"    char(6) NOT NULL,
    "roomCatID"    char(6) NOT NULL,
    "customerID"   char(6) NOT NULL,
    "dateCreate"   timestamp NOT NULL,
    "total"        numeric(10,2) NOT NULL,
    "vat"          numeric(10,2) NOT NULL,
    "amountDue"    numeric(10,2) NOT NULL,
    "periodOfStay" int NOT NULL,
    "checkIn"      timestamp NOT NULL,
    "checkOut"     timestamp NOT NULL,
    "numberOfRoom" int NOT NULL,
    CONSTRAINT "PK_tbl_invoices" PRIMARY KEY ( "invoiceID" ),
    CONSTRAINT "FK_76" FOREIGN KEY ( "roomCatID" ) REFERENCES "TBL_RoomCategrys" (
"roomCatID" ),
    CONSTRAINT "FK_97" FOREIGN KEY ( "customerID" ) REFERENCES "TBL_Customer" (
"customerID" )
);
CREATE INDEX "fkIdx_76" ON "TBL_Invoices"
(
    "roomCatID"
);
CREATE INDEX "fkIdx_97" ON "TBL_Invoices"
(
    "customerID"
);
```

*****"TBL_Employees"*****

```
CREATE TABLE "TBL_Employees"
(
    "employeeID"   char(6) NOT NULL,
```

```
"employeeTypeID" char(1) NOT NULL,
"firstname"      char(20) NOT NULL,
"familyname"    char(20) NOT NULL,
"email"         char(50) NOT NULL,
"phoneNumber"   char(10) NOT NULL,
"accepteDate"   char(1) NULL,
CONSTRAINT "PK_tbl_employees" PRIMARY KEY ( "employeeID" ),
CONSTRAINT "FK_16" FOREIGN KEY ( "employeeTypeID" ) REFERENCES "TBL_EmployeeTypes"
( "employeeTypeID" )
);
CREATE INDEX "fkIdx_16" ON "TBL_Employees"
(
  "employeeTypeID"
);
```

*****"TBL_Auth"*****

```
CREATE TABLE "TBL_Auth"
(
  "authID"      char(6) NOT NULL,
  "customerID"  char(6) NOT NULL,
  "password"    varchar(20) NOT NULL,
  CONSTRAINT "PK_tbl_auth" PRIMARY KEY ( "authID" ),
  CONSTRAINT "FK_154" FOREIGN KEY ( "customerID" ) REFERENCES "TBL_Customer" (
    "customerID" )
);
```

```
CREATE INDEX "fkIdx_154" ON "TBL_Auth"
(
  "customerID"
);
```

*****"TBL_ReceiptsLineItem"*****

```
CREATE TABLE "TBL_ReceiptsLineItem"
(
  "receiptID" char(6) NOT NULL,
  "remark" char(50) NULL,
  "invoiceID" char(6) NOT NULL,
  CONSTRAINT "PK_tbl_receiptslineitem" PRIMARY KEY ( "receiptID", "invoiceID" ),
  CONSTRAINT "FK_190" FOREIGN KEY ( "receiptID" ) REFERENCES "TBL_Receipts" ( "receiptID" ),
  CONSTRAINT "FK_193" FOREIGN KEY ( "invoiceID" ) REFERENCES "TBL_Invoices" ( "invoiceID" )
);

CREATE INDEX "fkidx_190" ON "TBL_ReceiptsLineItem"
(
  "receiptID"
);

CREATE INDEX "fkidx_193" ON "TBL_ReceiptsLineItem"
(
  "invoiceID"
);
```

*****"TBL_InvoiceLineItem"*****

```
CREATE TABLE "TBL_InvoiceLineItem"
(
  "invoiceID" char(6) NOT NULL,
  "roomID" char(6) NOT NULL,
  "remark" char(50) NULL,
  CONSTRAINT "PK_tbl_invoicelineitem" PRIMARY KEY ( "invoiceID", "roomID" ),
  CONSTRAINT "FK_81" FOREIGN KEY ( "roomID" ) REFERENCES "TBL_Rooms" ( "roomID" ),
  CONSTRAINT "FK_84" FOREIGN KEY ( "invoiceID" ) REFERENCES "TBL_Invoices" ( "invoiceID" )
);

CREATE INDEX "fkidx_81" ON "TBL_InvoiceLineItem"
(
  "roomID"
);
```


);

CREATE INDEX "fkIdx_84" ON "TBL_InvoiceLineItem"

(

"invoiceID"

);

*****"TBL_CleaningRoomLineItem"*****

CREATE TABLE "TBL_CleaningRoomLineItem"

(

"employeeID" char(6) NOT NULL,

"roomID" char(6) NOT NULL,

"startDateTime" timestamp NOT NULL,

"endDateTime" timestamp NOT NULL,

CONSTRAINT "PK_tbl_cleaningroomlineitem" PRIMARY KEY ("employeeID", "roomID"),

CONSTRAINT "FK_39" FOREIGN KEY ("employeeID") REFERENCES "TBL_Employees" ("employeeID"),

CONSTRAINT "FK_50" FOREIGN KEY ("roomID") REFERENCES "TBL_Rooms" ("roomID")

);

CREATE INDEX "fkIdx_39" ON "TBL_CleaningRoomLineItem"

(

"employeeID"

);

CREATE INDEX "fkIdx_50" ON "TBL_CleaningRoomLineItem"

(

"roomID"

);

How to insert, read, update, delete the table

Model in code from ORM structure

```
import sqlalchemy as sa
from sqlalchemy import Column, ForeignKey
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()

class TBL_Customers(Base):
    __tablename__ = 'TBL_Customer'
    customerID = Column('customerID', sa.String(6), primary_key=True)
    firstname = Column('firstname', sa.String(20))
    familyname = Column('familyname', sa.String(20))
    email = Column('email', sa.String(50))
    phoneNumber = Column('phoneNumber', sa.String(10))
    creditCardNumber = Column('creditCardNumber', sa.String(16))
    point = Column('point', sa.Integer)

class TBL_Auth(Base):
    __tablename__ = 'TBL_Auth'
    authID = Column('authID', sa.String(6), primary_key=True)
    customerID = Column('customerID', ForeignKey('TBL_Customer.customerID'))
    password = Column('password', sa.String(20))

class TBL_Rooms(Base):
    __tablename__ = 'TBL_Rooms'
    roomID = Column('roomID', sa.String(6), primary_key=True)
    roomCatID = Column('roomCatID', ForeignKey('TBL_RoomCategorys.roomCatID'))
    status = Column('status', sa.Boolean)
    cleanStatus = Column('cleanStatus', sa.Boolean)

class TBL_RoomCategorys(Base):
    __tablename__ = 'TBL_RoomCategorys'
    roomCatID = Column('roomCatID', sa.String(6), primary_key=True)
    name = Column('name', sa.String(20))
    bedType = Column('bedType', sa.String(20))
    numberBed = Column('numberBed', sa.Integer)
    guestRoom = Column('guestRoom', sa.Boolean)
    fare = Column('fare', sa.Float(precision=2))

class TBL_Invoices(Base):
    __tablename__ = 'TBL_Invoices'
    invoiceID = Column('invoiceID', sa.String(6), primary_key=True)
    roomCatID = Column('roomCatID', ForeignKey('TBL_RoomCategorys.roomCatID'))
    customerID = Column('customerID', ForeignKey('TBL_Customer.customerID'))
    dateCreate = Column('dateCreate', sa.Date)
    total = Column('total', sa.Float(precision=2))
    vat = Column('vat', sa.Float(precision=2))
    amountDue = Column('amountDue', sa.Float(precision=2))
    periodOfStay = Column('periodOfStay', sa.Integer)
    checkIn = Column('checkIn', sa.Date)
```

```

checkOut = Column('checkOut', sa.Date)
numberOfRoom = Column('numberOfRoom', sa.Integer)

class TBL_InvoiceLineItem(Base):
    __tablename__ = 'TBL_InvoiceLineItem'
    invoiceID = Column('invoiceID', ForeignKey('TBL_Invoices.invoiceID'), primary_key=True)
    roomID = Column('roomID', ForeignKey('TBL_Rooms.roomID'), primary_key=True)
    remark = Column('remark', sa.String(50))

class TBL_Receipts(Base):
    __tablename__ = 'TBL_Receipts'
    receiptID = Column('receiptID', sa.String(6), primary_key=True)
    customerID = Column('customerID', ForeignKey('TBL_Customer.customerID'))
    paymentMedId = Column('paymentMedId', ForeignKey('TBL_PaymentMedthods.paymentMedId'))
    couponID = Column('couponID', ForeignKey('TBL_Coupons.couponID'))
    dateCreate = Column('dateCreate', sa.Date)
    paymentRef = Column('paymentRef', sa.String(30))
    totalReceived = Column('totalReceived', sa.Float(precision=2))
    remark = Column('remark', sa.String(50))

class TBL_ReceiptsLineItem(Base):
    __tablename__ = 'TBL_ReceiptsLineItem'
    receiptID = Column('receiptID', ForeignKey('TBL_Receipts.receiptID'), primary_key=True)
    invoiceID = Column('invoiceID', ForeignKey('TBL_Invoices.invoiceID'), primary_key=True)
    remark = Column('remark', sa.String(50))

class TBL_PaymentMedthods(Base):
    __tablename__ = 'TBL_PaymentMedthods'
    paymentMedId = Column('paymentMedId', sa.String(6), primary_key=True)
    name = Column('name', sa.String(10))

class TBL_Coupons(Base):
    __tablename__ = 'TBL_Coupons'
    couponID = Column('couponID', sa.String(6), primary_key=True)
    name = Column('name', sa.String(10))
    expireDate = Column('expireDate', sa.Date)

class TBL_Employees(Base):
    __tablename__ = 'TBL_Employees'
    employeeID = Column('employeeID', sa.String(6), primary_key=True)
    employeeTypeID = Column('employeeTypeID', ForeignKey('TBL_EmployeeTypes.employeeTypeID'))
    firstname = Column('firstname', sa.String(20))
    familyname = Column('familyname', sa.String(20))
    email = Column('email', sa.String(50))
    phoneNumber = Column('phoneNumber', sa.String(10))
    accepteDate = Column('accepteDate', sa.Date)

class TBL_EmployeeTypes(Base):
    __tablename__ = 'TBL_EmployeeTypes'
    employeeTypeID = Column('employeeTypeID', sa.String(6), primary_key=True)
    name = Column('name', sa.String(20))
    salary = Column('salary', sa.Float(precision=2))
  
```

```
class TBL_CleaningRoomLineItem(Base):
    __tablename__ = 'TBL_CleaningRoomLineItem'
    employeeID = Column('employeeID', ForeignKey('TBL_Employees.employeeID'), primary_key=True)
    roomID = Column('roomID', ForeignKey('TBL_Rooms.roomID'), primary_key=True)
    startDateTime = Column('startDateTime', sa.DateTime)
    endDateTime = Column('endDateTime', sa.DateTime)
```

TBL_Auth

create

```
def create(self, authID, customerID, password):
    auth = TBL_Auth(authID=authID, customerID=customerID, password=password)
    session.add(auth)
    session.commit()
    log = {
        "result": "",
        "msg": "",
        "status": "1"
    }
    return log
```

Read

```
def read(self, authID):
    auth = session.query(TBL_Auth)
    auth = auth.filter(TBL_Auth.authID==authID)
    if auth.scalar() is not None :
        auth = self.serialize(auth.one())
        log = {
            "result": auth,
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "User not found",
            "status": "100"
        }
```

Update

```
def update(self, authID, customerID, password):
    auth = session.query(TBL_Auth)
    auth = auth.filter(TBL_Auth.authID==authID)
    if auth.scalar() is not None :
        auth = auth.one()
        auth.customerID = customerID
```

```

auth.password = password
session.commit()
log = {
    "result": "",
    "msg": "",
    "status": "1"
}
return log
else:
    log = {
        "result": "",
        "msg": "User not found",
        "status": "100"
    }
    return log

```

Delete

```

def delete(self, authID):
    auth = session.query(TBL_Auth)
    auth = auth.filter(TBL_Auth.authID==authID)
    if auth.scalar() is not None :
        session.delete(auth.one())
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "User not found",
            "status": "100"
        }
        return log

```

TBL_Customer

Create

```

def create(self, customerID, firstname, familyname, email, phoneNumber, creditCardNumber, point):
    customer = TBL_Customers(customerID=customerID, firstname=firstname, email=email,
familyname=familyname, phoneNumber=phoneNumber, creditCardNumber=creditCardNumber, point=int(point))
    session.add(customer)
    session.commit()
    log = {
        "result": "",
        "msg": "",

```

```
"status": "1"
}
return log
```

Read

```
def read(self, customerID):
    customer = session.query(TBL_Customers)
    customer = customer.filter(TBL_Customers.customerID==customerID)
    if customer.scalar() is not None :
        customer = self.serialize(customer.one())
        log = {
            "result":customer,
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "User not found",
            "status": "100"
        }
        return log
```

Update

```
def update(self, customerID, firstname, familyname, email, phoneNumber, creditCardNumber, point):
    customer = session.query(TBL_Customers)
    customer = customer.filter(TBL_Customers.customerID==customerID)
    if customer.scalar() is not None :
        customer = customer.one()
        customer.firstname = firstname
        customer.familyname = familyname
        customer.email = email
        customer.phoneNumber = phoneNumber
        customer.creditCardNumber = creditCardNumber
        customer.point = int(point)
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "User not found",
            "status": "100"
        }
        return log
```

return log

Delete

```
def delete(self, customerID):
    customer = session.query(TBL_Customers)
    customer = customer.filter(TBL_Customers.customerID==customerID)
    if customer.scalar() is not None :
        session.delete(customer.one())
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "User not found",
            "status": "100"
        }
        return log
```

TBL_Invoices

Create

```
def create(self, invoiceID, roomCatID, customerID, dateCreate, total, vat, checkIn, checkOut, numberOfRoom):

    # Convert datatype
    dateCreate = datetime.datetime.strptime(dateCreate, '%Y-%m-%d')
    checkIn = datetime.datetime.strptime(checkIn, '%Y-%m-%d')
    checkOut = datetime.datetime.strptime(checkOut, '%Y-%m-%d')
    periodOfStay = checkOut - checkIn
    total = float(total)
    vat = float(vat)
    amountDue = total + vat

    invoice = TBL_Invoices( invoiceID=invoiceID,
                            roomCatID=roomCatID,
                            customerID=customerID,
                            dateCreate=dateCreate.date(),
                            total=total,
                            vat=vat,
                            amountDue=amountDue,
                            periodOfStay=int(periodOfStay.days),
                            checkIn=checkIn.date(),
                            checkOut=checkOut.date(),
                            numberOfRoom=int(numberOfRoom)
                        )
```



```
session.add(invoice)
session.commit()
log = {
    "result": "",
    "msg": "",
    "status": "1"
}
return log
```

Read

```
def read(self, invoiceID):
    invoice = session.query(TBL_Invoices)
    invoice = invoice.filter(TBL_Invoices.invoiceID==invoiceID)
    if invoice.scalar() is not None :
        invoice = self.serialize(invoice.one())
        log = {
            "result": invoice,
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "Not found",
            "status": "100"
        }
        return log
```

Update

```
def update(self, invoiceID, roomCatID, customerID, dateCreate, total, vat, checkIn, checkOut,
numberOfRoom):
    invoice = session.query(TBL_Invoices)
    invoice = invoice.filter(TBL_Invoices.invoiceID==invoiceID)

    # Convert datatype
    dateCreate = datetime.datetime.strptime(dateCreate, '%Y-%m-%d')
    checkIn = datetime.datetime.strptime(checkIn, '%Y-%m-%d')
    checkOut = datetime.datetime.strptime(checkOut, '%Y-%m-%d')
    periodOfStay = checkOut - checkIn
    total = float(total)
    vat = float(vat)
    amountDue = total + vat

    if invoice.scalar() is not None :
        invoice = invoice.one()
        invoice.roomCatID = roomCatID
        invoice.customerID = customerID
        invoice.dateCreate = dateCreate.date()
```

```

invoice.total = total
invoice.vat = vat
invoice.amountDue = amountDue
invoice.periodOfStay = int(periodOfStay.days)
invoice.checkIn = checkIn.date()
invoice.checkOut = checkOut.date()
invoice.numberOfRoom = int(numberOfRoom)
session.commit()
log = {
    "result": "",
    "msg": "",
    "status": "1"
}
return log
else:
    log = {
        "result": "",
        "msg": "Not found",
        "status": "100"
    }
    return log

```

Delete

```

def delete(self, invoiceID):
    invoice = session.query(TBL_Invoices)
    invoice = invoice.filter(TBL_Invoices.invoiceID==invoiceID)
    if invoice.scalar() is not None :
        session.delete(invoice.one())
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "Not found",
            "status": "100"
        }
        return log

```

TBL_InvoiceLineItem

Create

```
def createInvoiceLine(self, invoiceID, roomID, remark):
    invoiceLine = TBL_InvoiceLineItem(invoiceID=invoiceID, roomID=roomID, remark=remark)
    session.add(invoiceLine)
    session.commit()
    log = {
        "result": "",
        "msg": "",
        "status": "1"
    }
    return log
```

Read

```
def readInvoiceLine(self, invoiceID, roomID):
    invoiceLine = session.query(TBL_InvoiceLineItem)
    invoiceLine = invoiceLine.filter(TBL_InvoiceLineItem.invoiceID==invoiceID,
    TBL_InvoiceLineItem.roomID==roomID)
    if invoiceLine.scalar() is not None :
        invoiceLine = self.serializeLine(invoiceLine.one())
        log = {
            "result": invoiceLine,
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "Not found",
            "status": "100"
        }
        return log
```

Update

```
def updateInvoiceLine(self, invoiceID, roomID, remark):
    invoiceLine = session.query(TBL_InvoiceLineItem)
    invoiceLine = invoiceLine.filter(TBL_InvoiceLineItem.invoiceID==invoiceID,
    TBL_InvoiceLineItem.roomID==roomID)

    if invoiceLine.scalar() is not None :
        invoiceLine = invoiceLine.one()
        invoiceLine.remark = remark
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
```

```

    }
    return log
else:
    log = {
        "result": "",
        "msg": "Not found",
        "status": "100"
    }
    return log

```

Delete

```

def deleteInvoiceLine(self, invoiceID, roomID):
    invoiceLine = session.query(TBL_InvoiceLineItem)
    invoiceLine = invoiceLine.filter(TBL_InvoiceLineItem.invoiceID==invoiceID,
TBL_InvoiceLineItem.roomID==roomID)
    if invoiceLine.scalar() is not None :
        session.delete(invoiceLine.one())
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "Not found",
            "status": "100"
        }
        return log

```

TBL_Receipts

Create

```

def create(self, receiptID, customerID, paymentMedId, couponID, dateCreate, paymentRef, totalReceived,
remark):

    # Convert datatype
    dateCreate = datetime.datetime.strptime(dateCreate, '%Y-%m-%d')
    totalReceived = float(totalReceived)

    receipt = TBL_Receipts( receiptID=receiptID,
                            customerID=customerID,
                            paymentMedId=paymentMedId,
                            couponID=couponID,
                            dateCreate=dateCreate.date(),
                            paymentRef=paymentRef,
                            totalReceived=totalReceived,
                            remark=remark

```

```
)

session.add(receipt)
session.commit()
log = {
    "result": "",
    "msg": "",
    "status": "1"
}
return log
```

Read

```
def read(self, receiptID):
    receipt = session.query(TBL_Receipts)
    receipt = receipt.filter(TBL_Receipts.receiptID==receiptID)
    if receipt.scalar() is not None :
        receipt = self.serialize(receipt.one())
        log = {
            "result": receipt,
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "Not found",
            "status": "100"
        }
        return log
```

Update

```
def update(self, receiptID, customerID, paymentMedId, couponID, dateCreate, paymentRef, totalReceived,
remark):
    receipt = session.query(TBL_Receipts)
    receipt = receipt.filter(TBL_Receipts.receiptID==receiptID)

    # Convert datatype
    dateCreate = datetime.datetime.strptime(dateCreate, '%Y-%m-%d')
    totalReceived = float(totalReceived)

    if receipt.scalar() is not None :
        receipt = receipt.one()
        receipt.customerID = customerID
        receipt.paymentMedId = paymentMedId
        receipt.couponID = couponID
        receipt.dateCreate = dateCreate.date()
        receipt.paymentRef = paymentRef
        receipt.totalReceived = totalReceived
```

```

receipt.remark = remark
session.commit()
log = {
    "result": "",
    "msg": "",
    "status": "1"
}
return log
else:
    log = {
        "result": "",
        "msg": "Not found",
        "status": "100"
    }
    return log

```

Delete

```

def delete(self, receiptID):
    receipt = session.query(TBL_Receipts)
    receipt = receipt.filter(TBL_Receipts.receiptID==receiptID)
    if receipt.scalar() is not None :
        session.delete(receipt.one())
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "Not found",
            "status": "100"
        }
        return log

```

TBL_ReceiptLineItem

Create

```

def createReceiptLine(self, receiptID, invoiceID, remark):
    receiptLine = TBL_ReceiptsLineItem(receiptID=receiptID, invoiceID=invoiceID, remark=remark)
    session.add(receiptLine)
    session.commit()
    log = {
        "result": "",
        "msg": "",
        "status": "1"
    }
    return log

```

Read

```
def readReceiptLine(self, receiptID, invoiceID):
    receiptLine = session.query(TBL_ReceiptsLineItem)
    receiptLine = receiptLine.filter(TBL_ReceiptsLineItem.receiptID==receiptID,
TBL_ReceiptsLineItem.invoiceID==invoiceID)
    if receiptLine.scalar() is not None :
        receiptLine = self.serializeLine(receiptLine.one())
        log = {
            "result":receiptLine,
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "Not found",
            "status": "100"
        }
        return log
```

Update

```
def updateReceiptLine(self, receiptID, invoiceID, remark):
    receiptLine = session.query(TBL_ReceiptsLineItem)
    receiptLine = receiptLine.filter(TBL_ReceiptsLineItem.receiptID==receiptID,
TBL_ReceiptsLineItem.invoiceID==invoiceID)

    if receiptLine.scalar() is not None :
        receiptLine = receiptLine.one()
        receiptLine.remark = remark
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "Not found",
            "status": "100"
        }
        return log
```

Delete

```
def deleteReceiptLine(self, receiptID, invoiceID):
    receiptLine = session.query(TBL_ReceiptsLineItem)
    receiptLine = receiptLine.filter(TBL_ReceiptsLineItem.receiptID==receiptID,
TBL_ReceiptsLineItem.invoiceID==invoiceID)
    if receiptLine.scalar() is not None :
        session.delete(receiptLine.one())
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "Not found",
            "status": "100"
        }
        return log
```

TBL_Rooms

Create

```
room = TBL_Rooms(roomID=roomID, roomCatID=roomCatID, status=(status=="1"),
cleanStatus=(cleanStatus=="1"))
session.add(room)
session.commit()
log = {
    "result": "",
    "msg": "",
    "status": "1"
}
return log
```

Read

```
def read(self, roomID):
    room = session.query(TBL_Rooms)
    room = room.filter(TBL_Rooms.roomID==roomID)
    if room.scalar() is not None :
        room = self.serialize(room.one())
        log = {
            "result": room,
            "msg": "",
            "status": "1"
        }
        return log
    else:
```



```
log = {  
    "result": "",  
    "msg": "Not found",  
    "status": "100"  
}  
return log
```

Update

```
def update(self, roomID, roomCatID, status, cleanStatus):  
    room = session.query(TBL_Rooms)  
    room = room.filter(TBL_Rooms.roomID==roomID)  
    if room.scalar() is not None :  
        room = room.one()  
        room.roomCatID = roomCatID  
        room.status = status=="1"  
        room.cleanStatus = cleanStatus=="1"  
        session.commit()  
        log = {  
            "result": "",  
            "msg": "",  
            "status": "1"  
        }  
        return log  
    else:  
        log = {  
            "result": "",  
            "msg": "Not found",  
            "status": "100"  
        }  
        return log
```

Delete

```
def delete(self, roomID):  
    room = session.query(TBL_Rooms)  
    room = room.filter(TBL_Rooms.roomID==roomID)  
    if room.scalar() is not None :  
        session.delete(room.one())  
        session.commit()  
        log = {  
            "result": "",  
            "msg": "",  
            "status": "1"  
        }  
        return log  
    else:  
        log = {  
            "result": "",  
            "msg": "Not found",  
            "status": "100"  
        }
```

```
}
return log
```

TBL_Employee

Create

```
def create(self, employeeID, employeeTypeID, firstname, familyname, email, phoneNumber, accepteDate):
    accepteDate = datetime.datetime.strptime(accepteDate, '%Y-%m-%d')
    employee = TBL_Employees(employeeID=employeeID, employeeTypeID=employeeTypeID,
    firstname=firstname, familyname=familyname, email=email, phoneNumber=phoneNumber,
    accepteDate=accepteDate.date())
    session.add(employee)
    session.commit()
    log = {
        "result": "",
        "msg": "",
        "status": "1"
    }
    return log
```

Read

```
def read(self, employeeID):
    employee = session.query(TBL_Employees)
    employee = employee.filter(TBL_Employees.employeeID==employeeID)
    if employee.scalar() is not None :
        employee = self.serialize(employee.one())
        log = {
            "result": employee,
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "User not found",
            "status": "100"
        }
        return log
```

Update

```
def update(self, employeeID, employeeTypeID, firstname, familyname, email, phoneNumber, accepteDate):
    accepteDate = datetime.datetime.strptime(accepteDate, '%Y-%m-%d')
    employee = session.query(TBL_Employees)
    employee = employee.filter(TBL_Employees.employeeID==employeeID)
    if employee.scalar() is not None :
        employee = employee.one()
        employee.employeeTypeID = employeeTypeID
        employee.firstname = firstname
```

```

employee.familyname = familyname
employee.email = email
employee.phoneNumber = phoneNumber
employee.acceptedDate = acceptedDate.date()
session.commit()
log = {
    "result": "",
    "msg": "",
    "status": "1"
}
return log
else:
    log = {
        "result": "",
        "msg": "User not found",
        "status": "100"
    }
    return log

```

Delete

```

def delete(self, employeeID):
    employee = session.query(TBL_Employees)
    employee = employee.filter(TBL_Employees.employeeID==employeeID)
    if employee.scalar() is not None :
        session.delete(employee.one())
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "User not found",
            "status": "100"
        }
        return log

```

TBL_EmployeeType

Create

```

def createEmployeeType(self, employeeTypeID, name, salary):
    employeeType = TBL_EmployeeTypes(employeeTypeID=employeeTypeID, name=name, salary=salary)
    session.add(employeeType)
    session.commit()
    log = {
        "result": "",
        "msg": "",

```

```
"status": "1"
}
return log
```

Read

```
def readEmployeeType(self, employeeTypeID):
    employeeType = session.query(TBL_EmployeeTypes)
    employeeType = employeeType.filter(TBL_EmployeeTypes.employeeTypeID==employeeTypeID)
    if employeeType.scalar() is not None :
        employeeType = self.serializeEmployeeType(employeeType.one())
        log = {
            "result": employeeType,
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "Not found",
            "status": "100"
        }
        return log
```

Update

```
def updateEmployeeType(self, employeeTypeID, name, salary):
    employeeType = session.query(TBL_EmployeeTypes)
    employeeType = employeeType.filter(TBL_EmployeeTypes.employeeTypeID==employeeTypeID)
    if employeeType.scalar() is not None :
        employeeType = employeeType.one()
        employeeType.name = name
        employeeType.salary = salary
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "Not found",
            "status": "100"
        }
        return log
```

Delete

```
def deleteEmployeeType(self, employeeTypeID):
    employeeType = session.query(TBL_EmployeeTypes)
    employeeType = employeeType.filter(TBL_EmployeeTypes.employeeTypeID==employeeTypeID)
    if employeeType.scalar() is not None :
        session.delete(employeeType.one())
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "Not found",
            "status": "100"
        }
        return log
```

TBL_CleaningRoomLineItem

Create

```
def create(self, employeeID, roomID, startDateTime, endDateTime):
    startDateTime = datetime.datetime.strptime(startDateTime, '%Y-%m-%d %H:%M')
    endDateTime = datetime.datetime.strptime(endDateTime, '%Y-%m-%d %H:%M')
    cleaning = TBL_CleaningRoomLineItem(employeeID=employeeID, roomID=roomID,
startDateTime=startDateTime, endDateTime=endDateTime)
    session.add(cleaning)
    session.commit()
    log = {
        "result": "",
        "msg": "",
        "status": "1"
    }
    return log
```

Read

```
def read(self, employeeID, roomID):
    cleaning = session.query(TBL_CleaningRoomLineItem)
    cleaning = cleaning.filter(TBL_CleaningRoomLineItem.employeeID==employeeID,
TBL_CleaningRoomLineItem.roomID==roomID)
    if cleaning.scalar() is not None :
        cleaning = self.serialize(cleaning.one())
        log = {
            "result": cleaning,
            "msg": "",
            "status": "1"
        }
```

```

    }
    return log
else:
    log = {
        "result": "",
        "msg": "Not found",
        "status": "100"
    }
    return log

```

Update

```

def update(self, employeeID, roomID, startDateTime, endDateTime):
    startDateTime = datetime.datetime.strptime(startDateTime, '%Y-%m-%d %H:%M')
    endDateTime = datetime.datetime.strptime(endDateTime, '%Y-%m-%d %H:%M')
    cleaning = session.query(TBL_CleaningRoomLineItem)
    cleaning = cleaning.filter(TBL_CleaningRoomLineItem.employeeID==employeeID,
TBL_CleaningRoomLineItem.roomID==roomID)
    if cleaning.scalar() is not None :
        cleaning = cleaning.one()
        cleaning.startDateTime = startDateTime
        cleaning.endDateTime = endDateTime
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }
        return log
    else:
        log = {
            "result": "",
            "msg": "User not found",
            "status": "100"
        }
        return log

```

Delete

```

def delete(self, employeeID, roomID):
    cleaning = session.query(TBL_CleaningRoomLineItem)
    cleaning = cleaning.filter(TBL_CleaningRoomLineItem.employeeID==employeeID,
TBL_CleaningRoomLineItem.roomID==roomID)
    if cleaning.scalar() is not None :
        session.delete(cleaning.one())
        session.commit()
        log = {
            "result": "",
            "msg": "",
            "status": "1"
        }

```

```

    }
    return log
else:
    log = {
        "result": "",
        "msg": "User not found",
        "status": "100"
    }
    return log

```

How to queries to view room category, fare and availability

```

def roomSummary(self):
    results = fetch(' SELECT r."roomId", rc."name", rc."bedType",      \
                      rc."numberBed", rc."guestRoom", rc."fare",      \
                      r."status", r."cleanStatus"                      \
                      FROM "TBL_Rooms" r                               \
                      INNER JOIN "TBL_RoomCategorys" rc               \
                      ON r."roomCatID" = rc."roomCatID"')

    rooms = []
    for result in results:
        tempDict = {}
        temp = cursor.fetchone(result)
        tempDict = {
            "roomId": temp[0],
            "roomType": temp[1],
            "bedType": temp[2],
            "numberBed": temp[3],
            "guestRoom": temp[4],
            "fare": temp[5],
            "status": temp[6],
            "cleanStatus": temp[7],
        }
        rooms.append(tempDict)

    log = {
        "result": rooms,
        "msg": "",
        "status": "1"
    }
    return log

```

How to queries to print receipt

```

def showReceiptReportByReceiptID(self, receiptID):
    receipt = []
    receiptID = ""+receiptID+""
    results = fetch('SELECT r."receiptID", r."dateCreate", r."totalReceived",
                      r."remark", r."paymentRef", c."firstname",
                      c."familyname", c."email", c."phoneNumber"

```

```

        FROM "TBL_Receipts" r
        INNER JOIN "TBL_Customer" c
            ON r."customerID" = c."customerID"
        WHERE r."receiptID" = {}'.format(receiptID))

for result in results:
    tempDict = {}
    temp = cursor.fetchone(result)
    tempDict = {
        "receiptID": temp[0],
        "dateCreate": temp[1],
        "totalReceived": temp[2],
        "remark": temp[3],
        "paymentRef": temp[4],
        "firstname": temp[5],
        "familyname": temp[6],
        "email": temp[7],
        "phoneNumber": temp[8],
    }
    receipt.append(tempDict)
log = {
    "result": receipt,
    "msg": "",
    "status": "1"
}
return log

def showReceiptReportByReceiptIDLine(self, receiptID):
    receiptID = ""+receiptID+""
    results = fetch(' SELECT i."invoiceID", rc."name", rc."fare", \
                        i."checkIn", i."checkOut", i."numberOfRoom", \
                        i."periodOfStay", i."amountDue" \
                    FROM "TBL_ReceiptsLineItem" re \
                    INNER JOIN "TBL_Invoices" i \
                        ON re."invoiceID" = i."invoiceID" \
                    INNER JOIN "TBL_Customer" c \
                        ON i."customerID" = c."customerID" \
                    INNER JOIN "TBL_RoomCategories" rc \
                        ON i."roomCatID" = rc."roomCatID" \
                    WHERE re."receiptID" = {}'.format(receiptID))

    receipt = []
    for result in results:
        tempDict = {}
        temp = cursor.fetchone(result)
        tempDict = {
            "invoiceID": temp[0],
            "roomType": temp[1],
            "fare": temp[2],
            "checkIn": temp[3],
            "checkOut": temp[4],
            "numberOfRoom": temp[5],
            "periodOfStay": temp[6],

```



```
        "amountDue": temp[7],  
    }  
    receipt.append(tempDict)  
  
    log = {  
        "result":receipt,  
        "msg": "",  
        "status": "1"  
    }  
    return log
```

Source Code

<https://github.com/RTae/new-hotel>