

G18 - ระบบจองใช้ห้อง

(ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์

ระบบหลัก : ระบบการจองใช้ห้อง

ระบบย่อย : ระบบบันทึกข้อมูลอุปกรณ์

1 User Requirements และ User Story Requirements

ระบบการจองใช้บริการห้องของบริษัท Room Booking เป็นระบบที่ให้ผู้ให้บริการซึ่งเป็นสมาชิกต้อง Log in เข้าสู่ระบบเพื่อทำการจองห้องต่างๆที่สมาชิกต้องการ เช่น ห้องเรียน ห้องปฏิบัติการ หรือห้องอ่านหนังสือ

นอกจากนี้ยังมีระบบข้อมูลอุปกรณ์เป็นระบบที่บอกรายละเอียดของข้อมูลอุปกรณ์ทั้งหมดมีอะไรบ้างทั้งที่มีให้ในห้องและอุปกรณ์ที่สามารถให้สมาชิกระบบจองใช้ห้องดูอุปกรณ์เสริมได้ เช่น หมายเลขอุปกรณ์

ประเภทของอุปกรณ์รวมถึงหากอุปกรณ์ชำรุดหรือเสียหายโดยมีพนักงานระบบดูแล

User Story(ระบบบันทึกข้อมูลอุปกรณ์)

ในบทบาทของ พนักงานระบบ

ฉันต้องการ บันทึกรายละเอียดข้อมูลอุปกรณ์

เพื่อ

ให้สมาชิกระบบจองใช้ห้องสามารถดูรายละเอียดของอุปกรณ์เพื่อทำการยืมได้

Output หน้าจอ แสดงรายละเอียดของอุปกรณ์

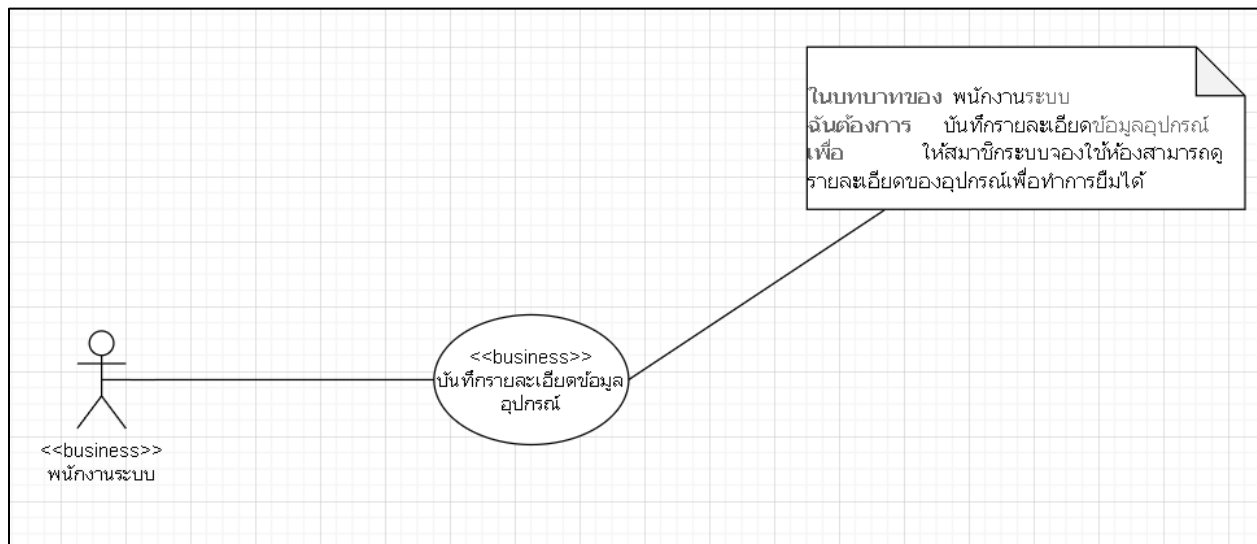
Output ข้อมูล ข้อมูลอุปกรณ์

คำนามที่อาจจะกลายเป็น Entity (ตารางในฐานข้อมูล) ของระบบจองใช้ห้อง

คำนาม	เหตุผล: เกี่ยวข้องกับ User Story นี้หรือไม่
พนักงานระบบ	เกี่ยวข้องโดยตรง เพราะเป็นผู้ใช้ระบบ
สรรพนามของอุปกรณ์	เกี่ยวข้องโดยตรง เพราะเป็นข้อมูลที่ต้องบันทึกเข้าระบบ
ประเภทอุปกรณ์	เกี่ยวข้องโดยตรง เพราะเป็นข้อมูลที่ต้องบันทึกเข้าระบบ

ข้อมูลอุปกรณ์	เกี่ยวข้องกับโดยตรง เพราะเป็นข้อมูลที่ต้องบันทึกเข้าระบบ
---------------	---

2. Business Use Case Diagram (เดี่ยว)



Check List ความถูกต้องของ Business Use Case

1. Business Actor มี <<Business>> กำกับ
2. Business Actor เป็นชื่อบทบาท ไม่ใช่ชื่อคน ไม่เป็นชื่อของบุคคล
3. Business Use Case ที่ <<Business>> กำกับ
4. Business Use Case ขึ้นต้นด้วย คำกริยา
5. คำกริยาของ business Use Case เป็นคำกริยาหลักใน User Story
6. มี Note ที่แสดง User Story อยู่ใน Diagram

ขั้นตอนการเปลี่ยนแปลง Business Use Case Diagram ให้เป็น System Use Case Diagram

พิจารณาประเด็นที่ 1

G18 - ระบบจองใช้ห้อง

(ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิติก

Business Actor “พนักงานระบบ”

สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์ที่ระบบงานได้โดยตรงหรือไม่

ตอบ ได้พนักงานระบบ สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์

เพราะฉะนั้น พนักงานระบบ จะกลายเป็น System Actor ได้

พิจารณาประเด็นที่ 2

Business Use Case “บันทึกรายละเอียดข้อมูลอุปกรณ์” สามารถกลายเป็นกิจกรรมที่เกี่ยวข้องกับระบบคอมพิวเตอร์ได้หรือไม่

ต้องมีขั้นตอนทาง Security มาเกี่ยวข้องหรือไม่

ตอบ Business Use Case “บันทึกรายละเอียดข้อมูลอุปกรณ์” สามารถกลายเป็น กิจกรรมที่เกี่ยวข้องกับระบบคอมพิวเตอร์ได้

และต้องมีขั้นตอนทาง Security (มี Privacy มาเกี่ยวข้อง)

เพราะฉะนั้น Business Use Case “บันทึกรายละเอียดข้อมูลอุปกรณ์” จะกลายเป็น System Use Case มากกว่า 1 Use Case จะ

ประกอบไปด้วย

1. System Use Case สำหรับ “เข้าระบบในฐานะพนักงานระบบ”
2. System Use Case สำหรับ “บันทึกรายละเอียดข้อมูลอุปกรณ์”

พิจารณาประเด็นที่ 3

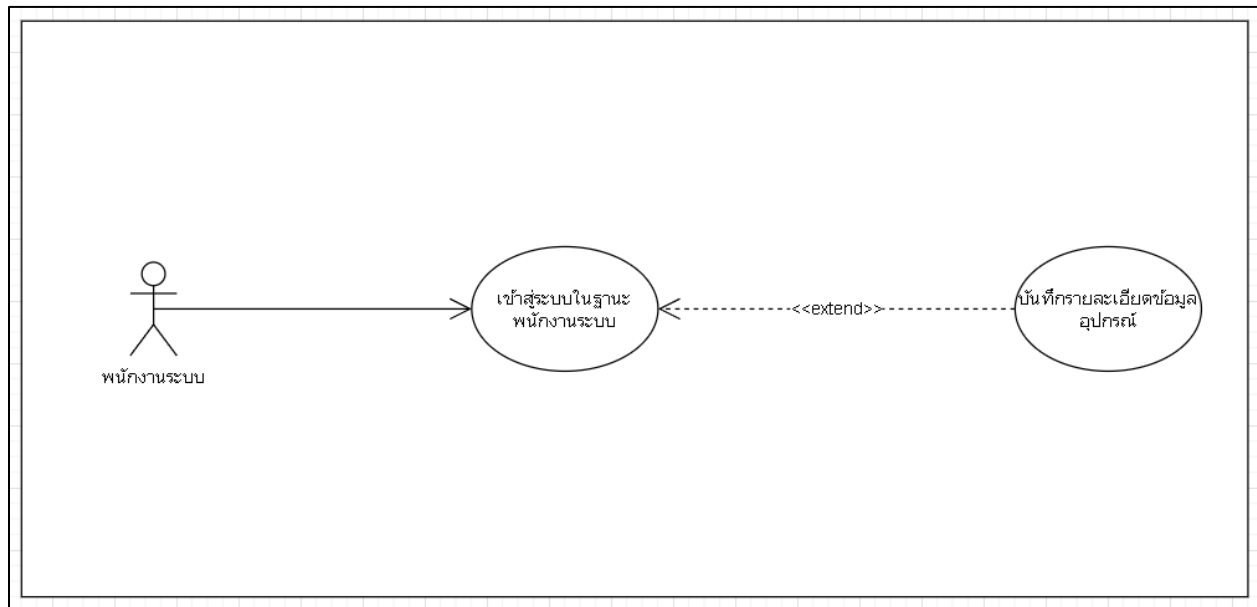
ถ้าพนักงาน “เข้าระบบในฐานะพนักงานระบบ” มาแล้ว

จำเป็นที่ต้อง “บันทึกรายละเอียดข้อมูลอุปกรณ์” ทุกครั้งหรือไม่

ตอบ ไม่

แปลว่า System Use Case “บันทึกรายละเอียดข้อมูลอุปกรณ์” เป็นทางเลือกหลังจากทำ System Use Case “เข้าระบบในฐานะพนักงานระบบ”

3. System Use Case Diagram (เดี่ยว)



Check List ความถูกต้องของ System Use Case

1. System Actor และ System Use Case ต้องไม่มีอะไรกำกับ
2. เส้นโยงจาก System Actor ไปหา System Use Case จะต้องเป็นเส้นทึบ หัวลูกศรปลายเปิด (open) ชี้ไปหา System Use Case
3. System Actor ต้องเป็นบทบาทของคนที่ใช้ระบบจริง ๆ
4. System Use Case ทั้งหมด ต้องขึ้นต้นด้วย คำกริยา
5. ถ้ามีกลไกทาง Security มาเกี่ยว ชื่อ System Use Case จะอยู่ในรูปแบบ “เข้าสู่ระบบในฐานะ<ชื่อบทบาท>” เท่านั้น ใช้คำว่า “เข้าสู่ระบบ” เฉย ๆ ไม่ได้
6. การใช้<--<<extend>>-- ต้องเป็น เส้นประ หัวลูกศร ปลายเปิด ชี้จาก System Use Case ทางเลือกไปยัง System Use Case หลัก
7. การใช้<--<<include>>-- ต้องเป็น เส้นประ หัวลูกศร ปลายเปิด ชี้จาก System Use Case ที่ใหญ่

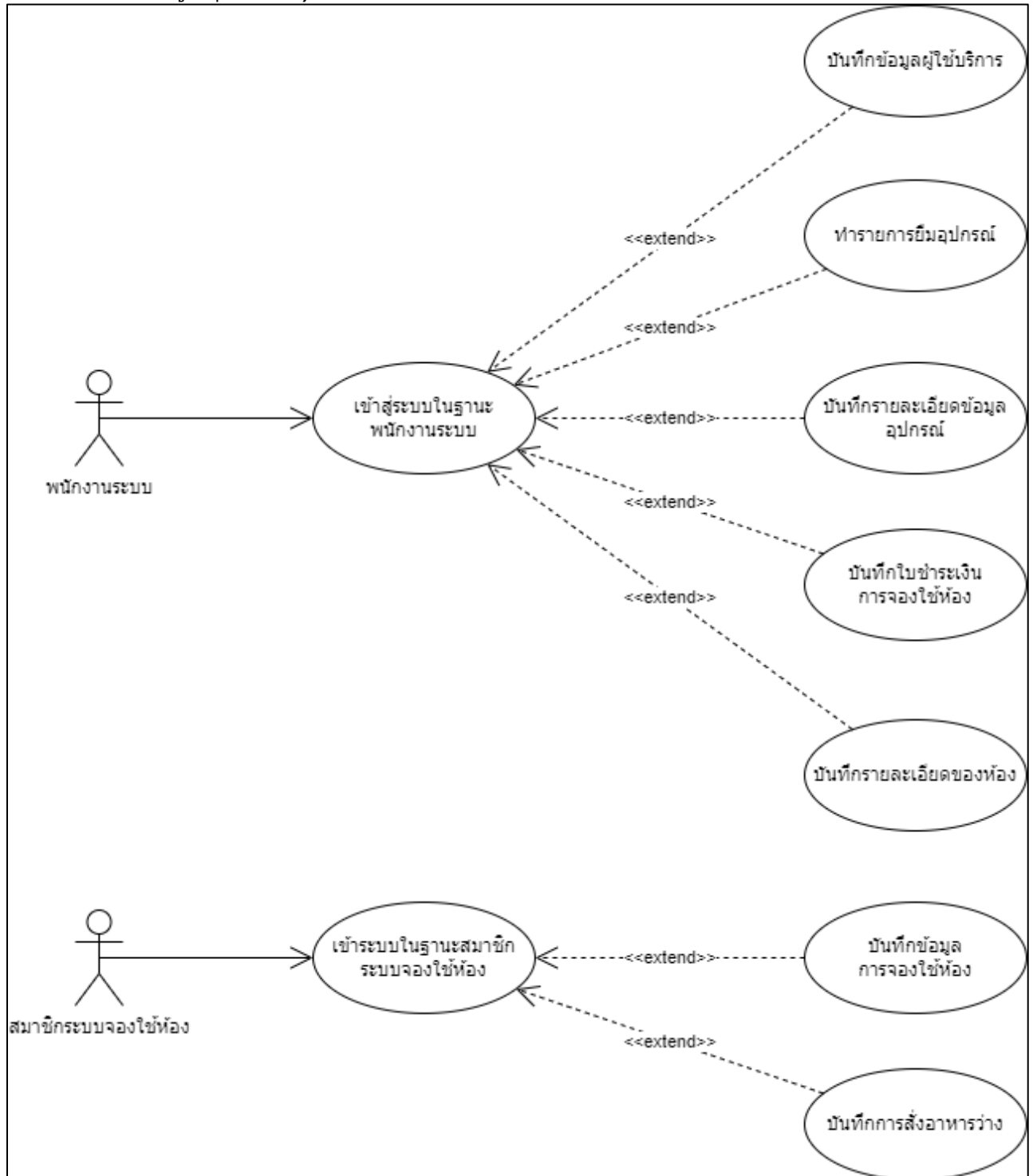
G18 - ระบบจองใช้ห้อง

(ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิത്യ

กว่า ไปยัง System Use Case ที่จะถูกรวมเข้ามา

4. System Use Case Diagram รวมทั้งระบบ



5. User Interface

5.1 การจำลองตัวอย่างตารางข้อมูล (เพื่อใช้เตรียมในการร่าง User Interface, System Activity Diagram และ Class Diagram)

จาก Requirements, จาก Output ข้อมูล และตาราง Entity

เราจะนำมาสมมติเป็น ตารางในฐานข้อมูล โดย

สมมติ Primary Key เป็นตัวเลข รวมทั้งการสมมติ Foreign Key เพื่อเชื่อมโยงข้อมูล

Entity: EMPLOYEE (ตาราง Employee) เพื่อเก็บข้อมูล “พนักงานระบบ”

EMPL OYEE _ID (PK, UINT)	PR EFI X_I D (FK, UIN T)	FIRST _NAM _E (VARC HAR)	LAST_ NAME (VARC HAR)	EMAIL (VARCHAR,U NIQUE)	PASSWORD (VARCHAR,U NIQUE)
0001	1	Arnon	Borribo on	AB@gma il.com	ArBorriboo n78290
0002	4	Chaiw at	Deede n	CD@test .com	ChaiDee32 570
0003	5	Pimch anok	Somsri	aantmo d@gmail .com	Wavekhon suy555

Entity: Category (ตาราง CATEGORY) เพื่อบันทึกข้อมูล “ประเภทอุปกรณ์”

CATEGORY_ID (PK, UINT)	CATEGORY_NAME (VARCHAR)
3001	Electrical appliance
3002	Stationery
3003	Furniture

Entity: Unit (ตาราง UNIT) เพื่อบันทึกข้อมูล “สรรพนามของอุปกรณ์”

UNIT_ID (PK, UINT)	UNIT_NAME (VARCHAR)
4001	Machine
4002	Stick
4003	Chair

Entity: Equipment (ตาราง EQUIPMENT) เพื่อเก็บข้อมูล “อุปกรณ์”

EQUIP MENT _ID (PK, UINT)	EQUIP MENT _NAME (VARC HAR)	AMO UNT (INT)	CATEG ORY _ID (FK, UINT)	UNIT _ID (FK, UINT)	TIME (DAT ETIM E)	EMPLO YEE_ID (FK, UINT)
2001	Projector	10	3001	4001	2021-01-02 10:00	0001
2002	Whiteboard pen	15	3002	4002	2021-01-02 11:00	0002
2003	Chairs	10	3003	4003	2021-01-02 14:00	0003

G18 - ระบบจองใช้ห้อง
(ระบบบันทึกข้อมูลอุปกรณ์)
ตรวจสอบว่าถูกต้องหรือไม่

B6220655 ธนพล ไชยนิทย์

ในบทบาทของ พนักงานระบบ
ต้องการ บันทึกรายละเอียดข้อมูลอุปกรณ์
เพื่อ

ให้สมาชิกระบบจองใช้ห้องสามารถดูรายละเอียดของอุปกรณ์เพื่อทำการยืม
ได้

พนักงานระบบ คนที่ 1 เข้ามาในระบบด้วย email: aanymod@gmail.com
เราสามารถใช้อีเมลระบุตัวตนของพนักงานระบบคนที่ 1 ได้ เพราะ email
มีคุณสมบัติ UNIQUE

จะค้นได้ว่า EMPLOYEE_ID ของพนักงานระบบคนนี้เป็น 0003

จากนั้น เราจะสามารถรู้ได้ว่า ประเภทอุปกรณ์ที่ชื่อ "Furniture"
คือประเภทอุปกรณ์หมายเลข 2003

จากนั้น เราสามารถรู้ได้ว่า Device (อุปกรณ์) ทั้งหมดที่โยงอยู่กับ
ประเภทอุปกรณ์ หมายเลข 2003 เราก็จะสามารถทราบได้ว่า

- อุปกรณ์หมายเลข 2003(Chairs) ประเภท Furniture สรรพนาม
Chair เวลา 14.00 น. วันที่ 2 เดือน 1 ปี 2021

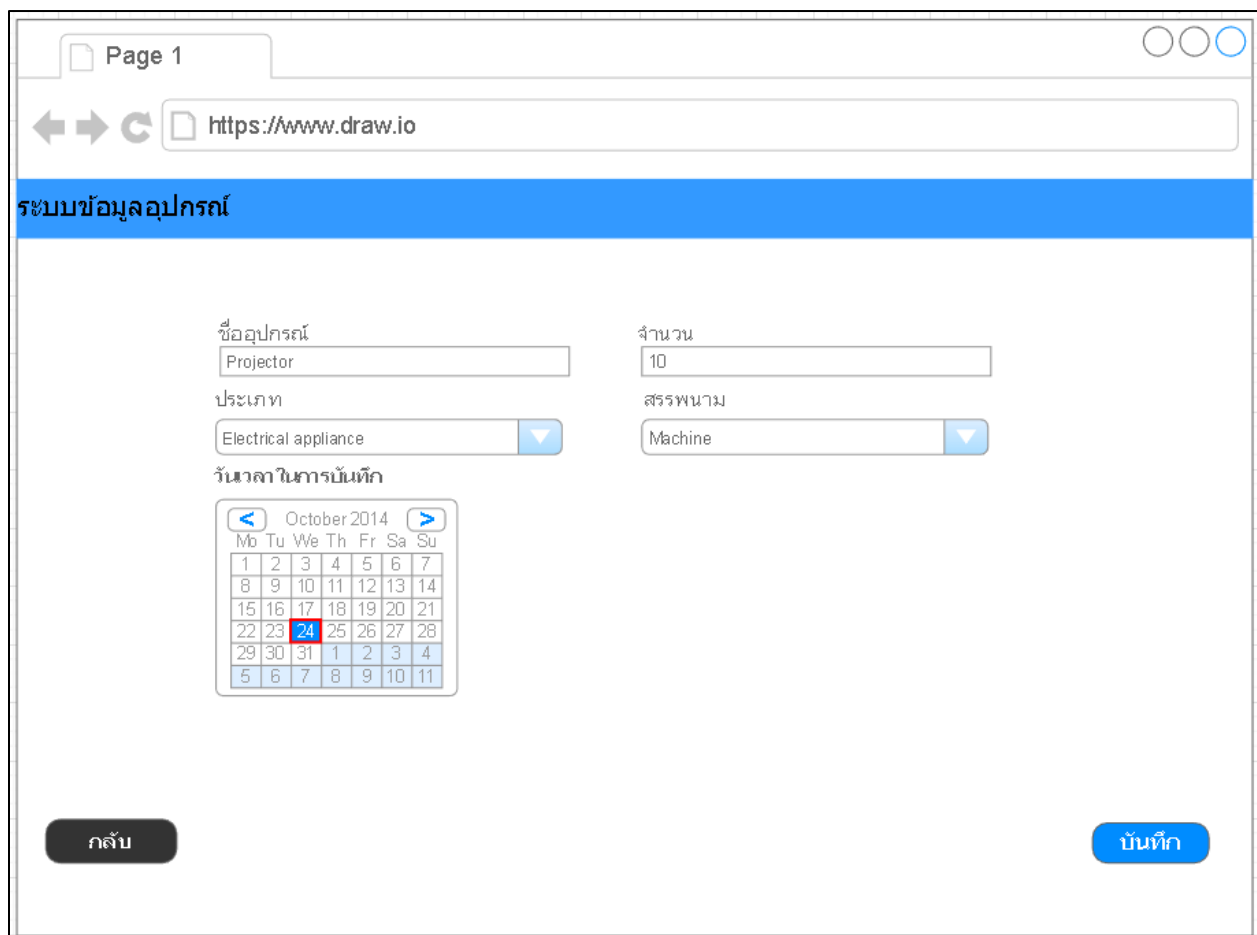
5.2 User Interface

เมื่อไหร่ก็ตามที่มีFK ชี้จาก ตารางหลัก กลับไปหา ตารางสนับสนุน ใช้
ComboBox เป็นตัวโยงข้อมูลใน User

Interface

ส่วน Field ประเภทอื่น ให้สร้างตามประเภทข้อมูล Textbox, Password,
Datetime Picker, Input ตัวเลข

เป็นต้น



Page 1

https://www.draw.io

ระบบข้อมูลอุปกรณ์

ชื่ออุปกรณ์
Projector

จำนวน
10

ประเภท
Electrical appliance

สรรพนาม
Machine

วันเวลาในการบันทึก

October 2014

Mo	Tu	We	Th	Fr	Sa	Su
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

กลับ

บันทึก

6. System Activity Diagram

1 Business Use Case (1 User Story) จะกลายเป็น 1 System Activity Diagram

System Activity Diagram จะเป็นการเล่าการกระทำ (activity) ระหว่าง คน - ระบบ

หลักสำคัญคือจะบอกว่า คนทำอะไร (input) แล้ว ระบบประมวลผลอะไร (process) แล้วระบบตอบอะไรกลับ

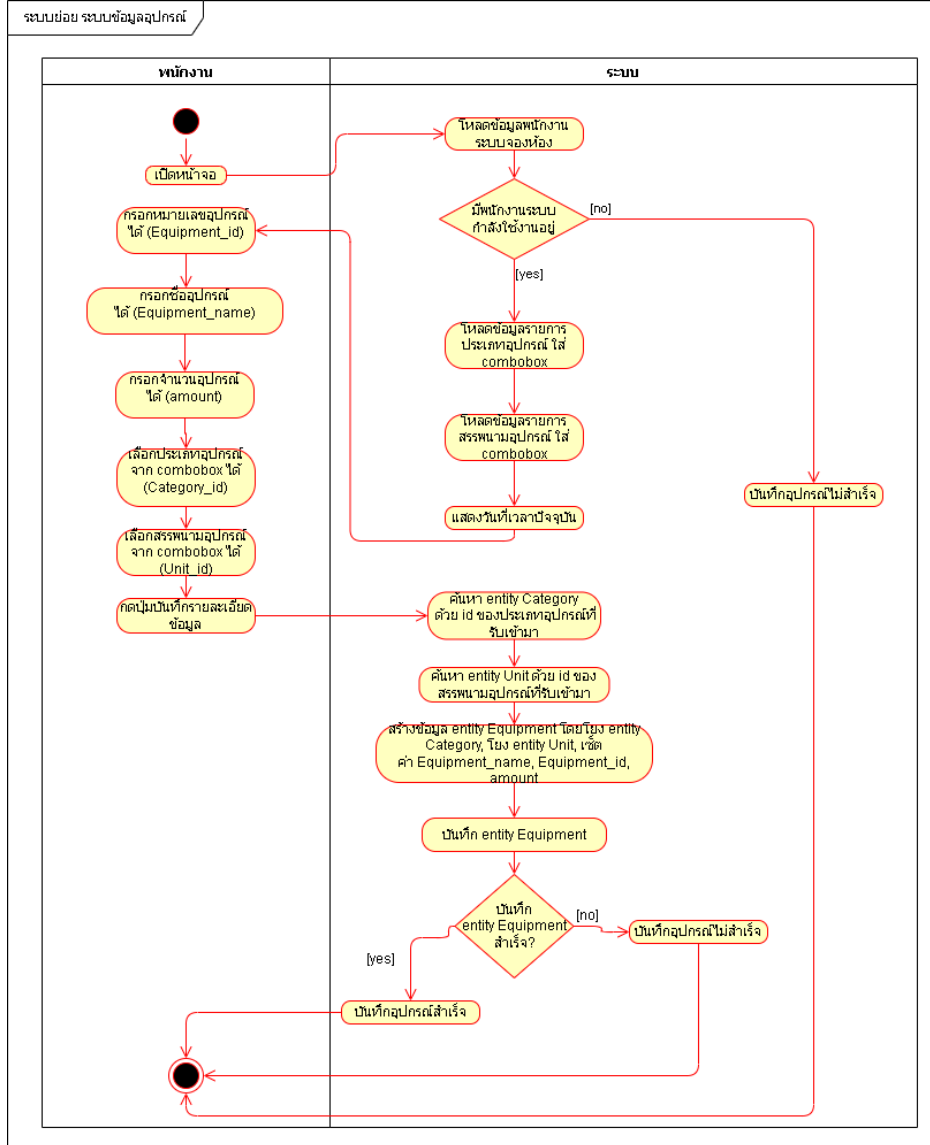
ออกมา (output)

1 System Activity Diagram ต้องจบที่การสร้าง Output ข้อมูลลงในฐานข้อมูล และสร้าง Output

หน้าจอ ให้ตรงกับ User Story ที่เตรียมไว้

G18 - ระบบจองใช้ห้อง (ระบบบันทึกข้อมูลอุปกรณ์)

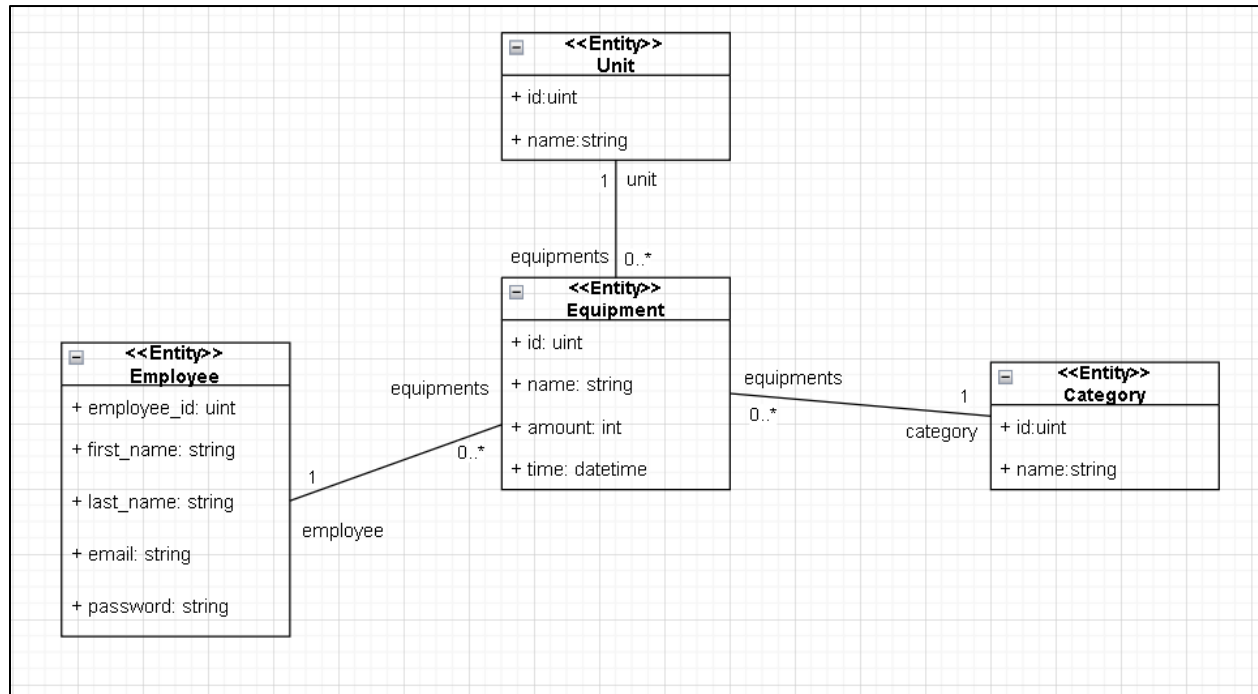
B6220655 ธนพล ไชยนิทย์



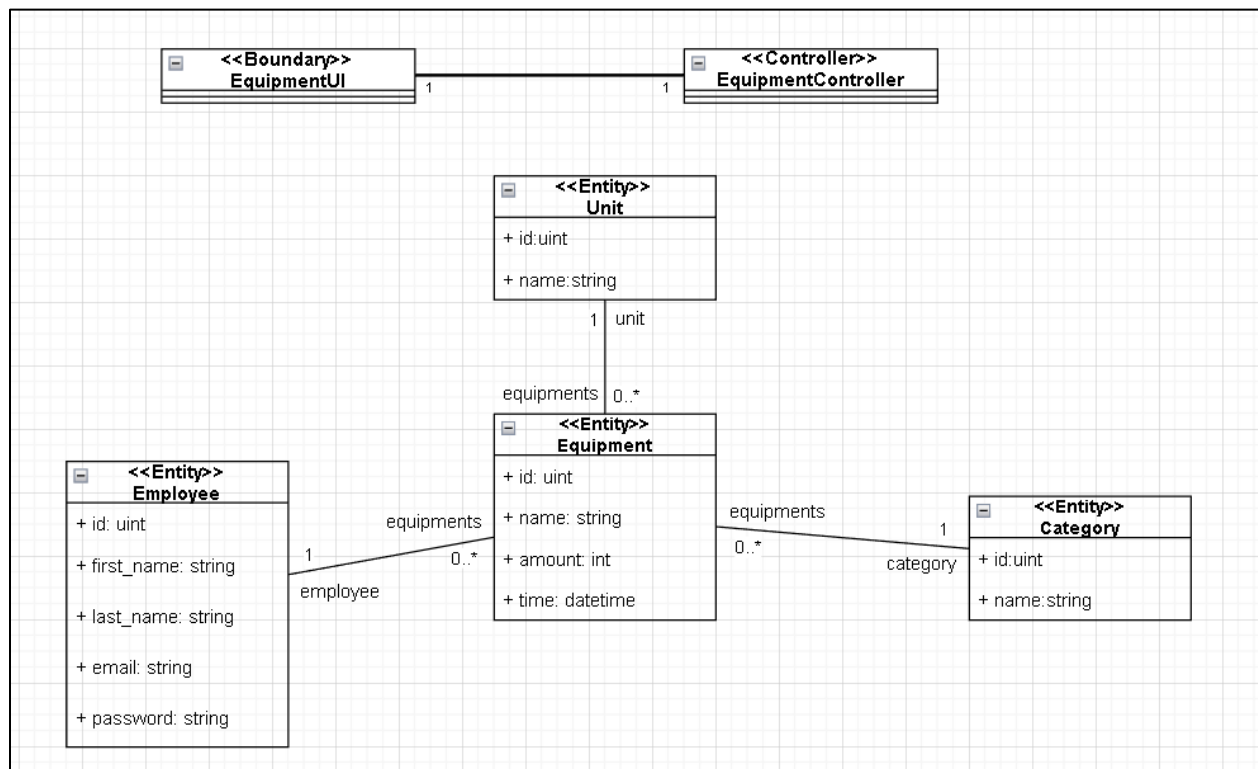
G18 - ระบบจองใช้ห้อง (ระบบบันทึกข้อมูลอุปกรณ์)

7. Class Diagram

B6220655 ธนพล ไชยนิทย์



8. Class Diagram at Design Level



9. การเขียน Communication Diagram

G18 - ระบบจองใช้ห้อง

(ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิิตย์

เราจะใช้ข้อมูลจาก System Activity Diagram เป็นหลัก

วิเคราะห์ เพื่อให้ได้ตารางของการเขียน Communication Diagram
ให้เป็นขั้นตอน

ในการวาด Communication Diagram เราจะมี

เส้นโครง ที่ใช้เชื่อมต่อเพื่อบอกความเกี่ยวข้องของวัตถุในระบบ

เส้นคำสั่ง จะเป็นเส้นที่มีหัวลูกศร วางเรียงกันอยู่บนเส้นโครง เพื่อบอกการ
dispatch message (ส่งคำสั่ง) โดยที่จะเป็นการบอกว่า วัตถุที่อยู่ตรงหัวลูกศร
จะเป็นวัตถุที่ทำตามคำสั่งให้เรา

เราจะวิเคราะห์ระบบ เฉพาะเหตุการณ์หลักของ Use Case ที่ diagram
นี้รับผิดชอบ เป็นการทำงานที่ทำแล้วได้ **Output ข้อมูล** และ **Output หน้าจอ**
ตามที่ระบุไว้ในเอกสาร requirements

จะมีการนำ Class ทั้งหมดที่เคยวิเคราะห์ไว้มาใช้

Boundary Class ทำหน้าที่แทน UI เราจะใช้ชื่อเบื้องต้นตามชื่อ Use Case
หลัก

เช่นในตัวอย่าง จะตั้งชื่อเป็น **EquipmentUI**

วัตถุของ Boundary Class จะส่งข้อมูลที่จำเป็นให้วัตถุของ Control Class

Control Class ทำหน้าที่เป็นตัวประมวลผล

ควบคุมการทำงานการเชื่อมโยงระหว่าง Entity จะตั้งชื่อตาม Use Case หลัก
เช่นในตัวอย่างจะตั้งชื่อเป็น **EquipmentController**

G18 - ระบบจองใช้ห้อง

(ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์

เตรียมแปลงให้แต่ละ Activity ของ System Activity Diagram ให้เป็น Communication Diagram

โดยวิเคราะห์วัตถุที่เกี่ยวข้องกับการทำงานแต่ละขั้นตอน

Activity	เป็นคำสั่ง สำหรับระบบได้ หรือไม่?	ถ้าเป็น, วัตถุที่รับห หน้าที่ทำ งานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
เปิดหน้าจอ	เป็นคำสั่ง เกิดการสั่งงานเพื่อให้หน้า UI เปิด (มี email อยู่ใน UI)	:EquipmentUI	เปิด()
โหลดข้อมูลระบบ พนักงานจองห อง	เป็นคำสั่ง เกิดการสั่งงานให้โหลดข้อมูล พนักงานจองห้องที่ log-in อยู่	:EquipmentController cu :Employee	โหลดข้อมูลพนักงานระบบ(em ail) ค้นหาด้วยอีเมล(email)
โหลดข้อมูลรายการประเภทอุปกรณ์ ใส่ combobox	เป็นคำสั่ง เกิดการสั่งงานให้โหลดข้อมูล ประเภทอุปกรณ์ทั้งหมดมา แสดง	:EquipmentController :Category	เตรียมข้อมูลให้ หน้าจอ() ดึงข้อมูลประเภท อุปกรณ์ทั้งหมด()
โหลดข้อมูลรายการสรรพนามอุปกรณ์ ใส่ combobox	เป็นคำสั่ง เกิดการสั่งงานให้โหลดข้อมูล สรรพนามอุปกรณ์ทั้งหมด มาแสดง	:Unit	ดึงข้อมูลสรรพนามอุปกรณ์ทั้งหมด()
แสดงวันที่เวลาปัจจุบัน	ไม่เป็นคำสั่ง (เป็นผลลัพธ์)	-	-

G18 - ระบบจองใช้ห้อง
(ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์

กรอกหมายเลขอุปกรณ์ได้ (e_id)	ไม่เป็นคำสั่ง (เป็นการใช้งาน UI ของผู้ใช้)	-	-
กรอกชื่ออุปกรณ์ได้ (eq_name)	ไม่เป็นคำสั่ง (เป็นการใช้งาน UI ของผู้ใช้)	-	-
กรอกจำนวนอุปกรณ์ได้ (amount)	ไม่เป็นคำสั่ง (เป็นการใช้งาน UI ของผู้ใช้)	-	-
เลือกประเภทอุปกรณ์จาก combobox ได้ (c_id)	ไม่เป็นคำสั่ง (เป็นการใช้งาน UI ของผู้ใช้)	-	-
เลือกสรรพนามอุปกรณ์จาก combobox ได้ (u_id)	ไม่เป็นคำสั่ง (เป็นการใช้งาน UI ของผู้ใช้)	-	-
กดปุ่มบันทึกรายละเอียดข้อมูล	เป็นคำสั่ง	:EquipmentController	บันทึกรายละเอียดข้อมูล(e_id, eq_name, amount, c_id, u_id)
ค้นหา entity Category ด้วย id ของประเภทอุปกรณ์ที่รับเข้ามา	เป็นคำสั่ง	c :Category	ค้นด้วยไอดีของ entity Category (c_id)
ค้นหา entity Unit ด้วย id ของสรรพนามอุปกรณ์ที่รับเข้ามา	เป็นคำสั่ง	u :Unit	ค้นด้วยไอดีของ entity Unit (u_id)

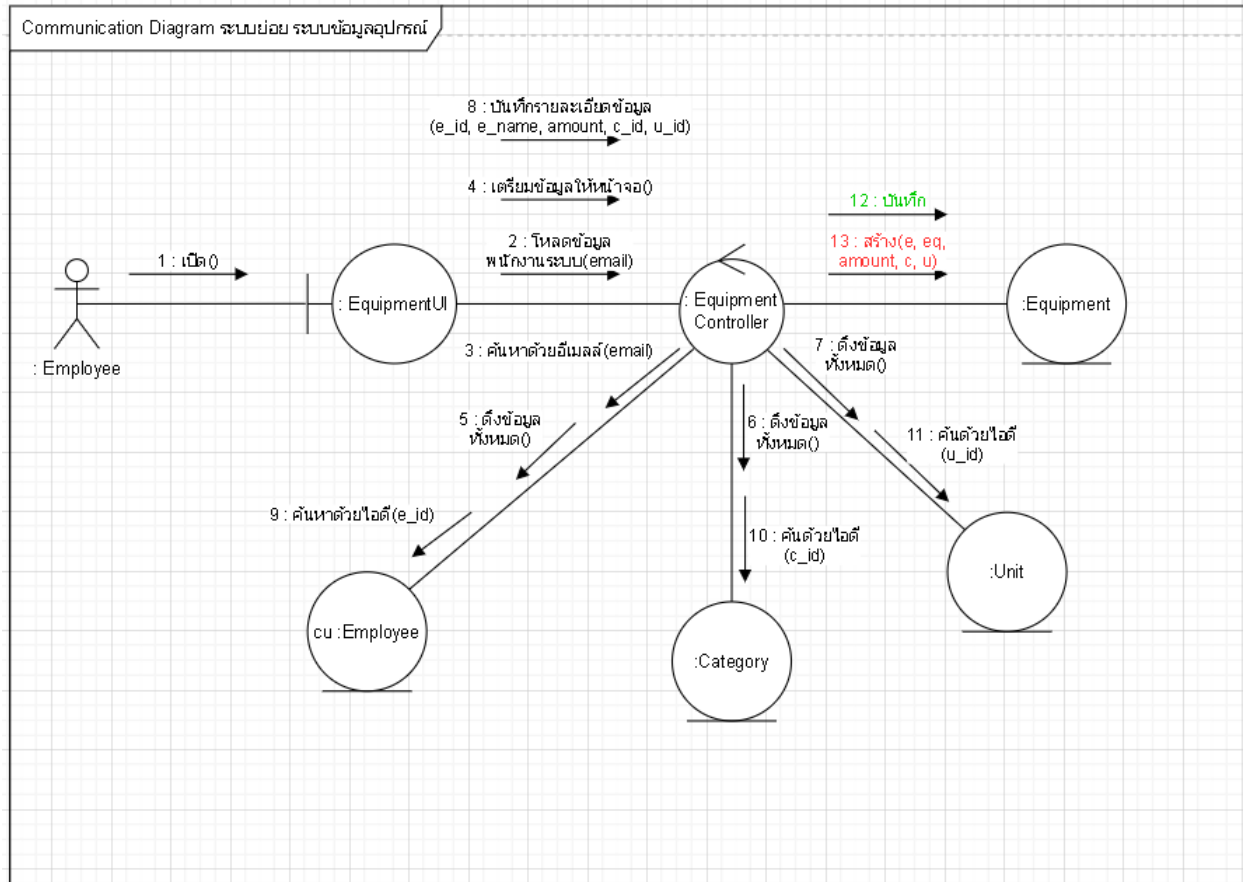
G18 - ระบบจองใช้ห้อง
(ระบบบันทึกข้อมูลอุปกรณ์)

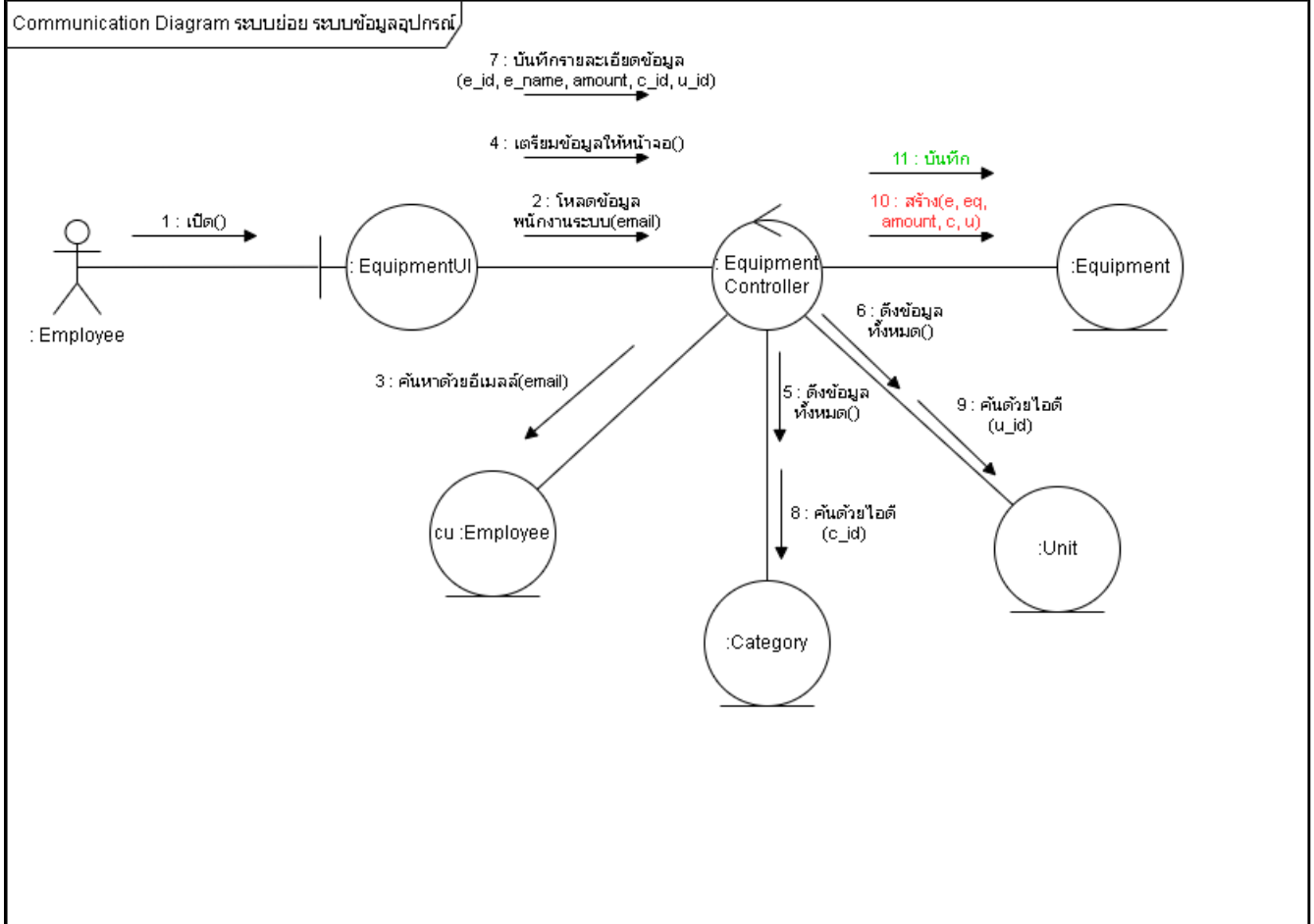
B6220655 ธนพล ไชยนิตย์

สร้างข้อมูล entity Equipment โดยโยง entity Category, โยง entity Unit, เช็ตค่า Equipment_name, Equipment_id, AMOUNT	เป็นคำสั่ง	e :Equipment	สร้าง (e, eq, amount, c, u)
บันทึก entity Equipment	เป็นคำสั่ง	e :Equipment	บันทึก()
บันทึกอุปกรณ์สำเร็จ	ไม่เป็นคำสั่ง	-	-

G18 - ระบบจองใช้ห้อง (ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์





B6220655 ธนพล ไชยนิตย์

ระบบข้อมูลอุปกรณ์	
ระบบบันทึกข้อมูลอุปกรณ์	
Requirements	ระบบการจองให้บริการห้องของบริษัท Room Booking เป็นระบบที่ให้ผู้ใช้บริการซึ่งเป็นสมาชิกต้อง Log in เข้าสู่ระบบเพื่อทำการจองห้องต่างๆที่สามารถต้องการ เช่น ห้องเรียน ห้องปฏิบัติการ หรือห้องอ่านหนังสือ นอกจากนี้ยังมีระบบข้อมูลอุปกรณ์เป็นระบบที่บอกรายละเอียดของข้อมูลอุปกรณ์ทั้งหมดมีอะไรบ้างที่มีอยู่ในห้องและอุปกรณ์ที่สามารถให้สมาชิกระบบจองในห้องอุปกรณ์เสริมได้ เช่น หมายเหตุอุปกรณ์ ประเภทของอุปกรณ์รวมถึงหากอุปกรณ์ชำรุดหรือเสียหายโดยมีพนักงานระบบดูแล
User Story (ระบบบันทึกการจองในห้อง)	ในฐานะสมาชิกของ พนักงานระบบ ฉันต้องการ บันทึกรายละเอียดข้อมูลอุปกรณ์ เพื่อให้สมาชิกระบบจองในห้องสามารถดูรายละเอียดของอุปกรณ์เพื่อทำการจองได้

G18 - ระบบจองใช้ห้อง (ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์

ระบบข้อมูลอุปกรณ์

บันทึกข้อมูลอุปกรณ์

ชื่ออุปกรณ์

จำนวน

ประเภทอุปกรณ์

สรรพทนาม

กรณเลือกประเภทอุปกรณ์

กรณเลือกอุปกรณ์

วันที่และเวลา

10/25/2022

กลับ

บันทึก

ระบบข้อมูลอุปกรณ์

ข้อมูลการบันทึกอุปกรณ์

สร้างข้อมูล

ลำดับ	ชื่ออุปกรณ์	จำนวน	ประเภท	สรรพทนาม	พนักงาน
1	Projector	10	Electricalappliance	Machine	thna srimeang
2	Whiteboardpen	15	Stationery	Stick	sukda mama
3	Chairs	10	Furniture	Chair	thanathon por
4	Projector ASUS	15	Electricalappliance	Machine	thna srimeang

1-4 of 4

G18 - ระบบจองใช้ห้อง (ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์

หน้าแรก

บันทึกข้อมูลอุปกรณ์

ข้อมูลอุปกรณ์

ออกจากระบบ

หน้า

ข้อมูลการมีติดอุปกรณ์

สร้างข้อมูล

ลำดับ	ชื่ออุปกรณ์	จำนวน	ประเภท	สถานที่	พนักงาน
1	Projector	10	Electricalappliance	Machine	thna srimeang
2	Whiteboardpen	15	Stationery	Stick	sukda mama
3	Chairs	10	Furniture	Chair	thanathon por
4	Projector ASUS	15	Electricalappliance	Machine	thna srimeang

1-4 of 4

11. Source Code Back-End และ Front-End

-Source Code Back-End ชื่อไฟล์ main.go

```
package main

import (
    "github.com/thanaponkhanoon/sa-65-example/controller"
    "github.com/thanaponkhanoon/sa-65-example/entity"
    "github.com/thanaponkhanoon/sa-65-example/middlewares"
    "github.com/gin-gonic/gin"
)

const PORT = "8080"

func main() {
    entity.SetupDatabase()
    r := gin.Default()
    r.Use(CORSMiddleware())
    router := r.Group("/")
    {
        router.Use(middlewares.Authorizes())
        {
            // User Routes
            router.GET("/employees", controller.ListEmployees)
            router.GET("/employee/:id", controller.GetEmployee)
            router.PATCH("/employees", controller.UpdateEmployee)
            router.DELETE("/employees/:id", controller.DeleteEmployee)
            router.PATCH("/employee", controller.CreateEmployee)
            // Category Routes
            router.GET("/catagories", controller.ListCategory)
            router.GET("/catagory/:id", controller.GetCategory)
            router.POST("/catagories", controller.CreateCategory)
            router.PATCH("/catagories", controller.UpdateCategory)
            router.DELETE("/catagories/:id", controller.DeleteCategory)
            // Unit Routes
            router.GET("/units", controller.ListUnits)
            router.GET("/unit/:id", controller.GetUnit)
            router.POST("/units", controller.CreateUnit)
            router.PATCH("/units", controller.UpdateUnit)
            router.DELETE("/units/:id", controller.DeleteUnit)
            // Equipment Routes
            router.GET("/equipments", controller.ListEquipments)
            router.GET("/equipment/:id", controller.GetEquipment)
            router.POST("/equipment", controller.CreateEquipment)
            router.PATCH("/equipments", controller.UpdateEquipment)
            router.DELETE("/equipments/:id", controller.DeleteEquipment)
        }
    }
    // Signup User Route
    // r.POST("/signup", controller.CreateUser)
    // login User Route
    r.POST("/login", controller.Login)
    // Run the server go run main.go
    r.Run("localhost: " + PORT)
}
```

G18 - ระบบจองใช้ห้อง (ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์

```
func CORSMiddleware() gin.HandlerFunc {
    return func(c *gin.Context) {
        c.Writer.Header().Set("Access-Control-Allow-Origin", "*")
        c.Writer.Header().Set("Access-Control-Allow-Credentials", "true")
        c.Writer.Header().Set("Access-Control-Allow-Headers", "Content-
Type, Content-Length, Accept-Encoding, X-CSRF-Token, Authorization, accept,
origin, Cache-Control, X-Requested-With")
        c.Writer.Header().Set("Access-Control-Allow-Methods", "POST,
OPTIONS, GET, PUT")
        if c.Request.Method == "OPTIONS" {
            c.AbortWithStatus(204)
            return
        }
        c.Next()
    }
}
```

-Source Code Back-End ใน folder entity ชื่อไฟล์ Equipment.go

```
package entity
```

```
import (
    "time"
```



```
    "gorm.io/gorm"
)

type Employee struct {
    gorm.Model
    FirstName string
    LastName  string
    Email     string `gorm:"uniqueIndex"`
    Password  string

    //1 employee เป็นเจ้าของได้หลาย equipment
    Equipments []Equipment `gorm:"foreignKey:EmployeeID"`
}

type Category struct {
    gorm.Model
    Name string

    Equipments []Equipment `gorm:"foreignKey:CategoryID"`
}

type Unit struct {
    gorm.Model
    Name string

    Equipments []Equipment `gorm:"foreignKey:UnitID"`
}

type Equipment struct {
    gorm.Model
    Time time.Time

    Name      string
    Amount    int

    // CategoryID ทำหน้าที่เป็น FK
    CategoryID *uint
    Category   Category `gorm:"references:id"`

    // UnitID ทำหน้าที่เป็น FK
    UnitID *uint
    Unit   Unit `gorm:"references:id"`

    // EmployeeID ทำหน้าที่เป็น FK
    EmployeeID *uint
    Employee   Employee `gorm:"references:id"`
}
```

**-Source Code Back-End ใน folder entity ชื่อไฟล์ setup.go
เพื่อใช้สร้าง database**

```
package entity

import (
    "time"

    "golang.org/x/crypto/bcrypt"
    "gorm.io/driver/sqlite"
    "gorm.io/gorm"
```

```
)

var db *gorm.DB

func DB() *gorm.DB {
    return db
}

func SetupDatabase() {
    database, err := gorm.Open(sqlite.Open("sa-65.db"), &gorm.Config{})
    if err != nil {
        panic("failed to connect database")
    }

    // Migrate the schema
    database.AutoMigrate(
        &Employee{},
        &Category{},
        &Unit{},
        &Equipment{},
    )

    db = database

    passwordEmployee1, err := bcrypt.GenerateFromPassword([]byte("123456"),
14)
    Employee1 := Employee{
        FirstName: "thna",
        LastName: "srimeang",
        Email: "thana@gmail.com",
        Password: string(passwordEmployee1),
    }
    db.Raw("SELECT * FROM employees WHERE email = ?",
"thanaponkhanoon1123@gmail.com").Scan(&Employee1)
    passwordEmployee2, err :=
bcrypt.GenerateFromPassword([]byte("1234512126"), 14)
    Employee2 := Employee{
        FirstName: "thanathon",
        LastName: "pongpak",
        Email: "thanathon@gmail.com",
        Password: string(passwordEmployee2),
    }
    db.Raw("SELECT * FROM employees WHERE email = ?",
"thanaponkhanoon1123@gmail.com").Scan(&Employee2)
    passwordEmployee3, err := bcrypt.GenerateFromPassword([]byte("123426"),
14)
    Employee3 := Employee{
        FirstName: "sukda",
        LastName: "mama",
        Email: "sakda@gmail.com",
    }
```

```
        Password: string(passwordEmployee3),
    }
    db.Raw("SELECT * FROM employees WHERE email = ?",
"thanaponkhanoon1123@gmail.com").Scan(&Employee3)

    // --- Category Data
    electricalappliance := Category{
        Name: "Electricalappliance",
    }
    db.Model(&Category{}).Create(&electricalappliance)

    stationery := Category{
        Name: "Stationery",
    }
    db.Model(&Category{}).Create(&stationery)

    furniture := Category{
        Name: "Furniture",
    }
    db.Model(&Category{}).Create(&furniture)

    // Unit Data
    machine := Unit{
        Name: "Machine",
    }
    db.Model(&Unit{}).Create(&machine)

    stick := Unit{
        Name: "Stick",
    }
    db.Model(&Unit{}).Create(&stick)

    chair := Unit{
        Name: "Chair",
    }
    db.Model(&Unit{}).Create(&chair)

    // equipment 1
    db.Model(&Equipment{}).Create(&Equipment{
        Name: "Projector",
        Amount: 10,
        Category: electricalappliance,
        Unit: machine,
        Time: time.Now(),
        Employee: Employee1,
    })
    // equipment 2
    db.Model(&Equipment{}).Create(&Equipment{
        Name: "Whiteboardpen",
        Amount: 15,
```

G18 - ระบบจองใช้ห้อง (ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์

```
        Category:    stationery,
        Unit:        stick,
        Time: time.Now(),
        Employee:    Employee3,
    })
    // equipment 3
    db.Model(&Equipment{}).Create(&Equipment{
        Name:        "Chairs",
        Amount:      10,
        Category:    furniture,
        Unit:        chair,
        Time: time.Now(),
        Employee:    Employee2,
    })
}
```

-Source Code Back-End ใน folder controller ชื่อไฟล์ authentication.go

```
package controller

import (
    "net/http"

    "github.com/gin-gonic/gin"
    "github.com/thanaponkhanoon/sa-65-example/service"
    "github.com/thanaponkhanoon/sa-65-example/entity"
    "golang.org/x/crypto/bcrypt"
)
```

```
// LoginPayload login body
type LoginPayload struct {
    Email    string `json:"email"`
    Password string `json:"password"`
}

// SignUpPayload signup body
type SignUpPayload struct {
    Name      string `json:"name"`
    Email     string `json:"email"`
    Password  string `json:"password"`
}

// LoginResponse token response
type LoginResponse struct {
    Token string `json:"token"`
    ID    uint  `json:"id"`
}

// POST /login
func Login(c *gin.Context) {
    var payload LoginPayload
    var employee entity.Employee

    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    // ค้นหา user ด้วย email ที่ผู้ใช้กรอกเข้ามา
    if err := entity.DB().Raw("SELECT * FROM employees WHERE email = ?",
payload.Email).Scan(&employee).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    // ตรวจสอบรหัสผ่าน
    err := bcrypt.CompareHashAndPassword([]byte(employee.Password),
[]byte(payload.Password))
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "password is
incorrect"})
        return
    }

    // กำหนดค่า SecretKey, Issuer และระยะเวลาหมดอายุของ Token สามารถกำหนดเองได้
    // SecretKey ใช้สำหรับการ sign ข้อความเพื่อบอกว่าข้อความมาจากตัวเราแน่นอน
    // Issuer เป็น unique id ที่เอาไว้ระบุตัว client
    // ExpirationHours เป็นเวลาหมดอายุของ token
```

```
    jwtWrapper := service.JwtWrapper{
        SecretKey:      "SvNQpBN8y3qlVrsGAYYWoJJk56LtzFHx",
        Issuer:          "AuthService",
        ExpirationHours: 24,
    }

    signedToken, err := jwtWrapper.GenerateToken(employee.Email)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "error signing
token"})
    }
    return

    tokenResponse := LoginResponse{
        Token: signedToken,
        ID:     employee.ID,
    }

    c.JSON(http.StatusOK, gin.H{"data": tokenResponse})
}

// POST /create
func CreateEmployee(c *gin.Context) {
    var payload SignUpPayload
    var employee entity.Employee

    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    // เข้ารหัสลับรหัสผ่านที่ผู้ใช้กรอกก่อนบันทึกลงฐานข้อมูล
    hashPassword, err :=
bcrypt.GenerateFromPassword([]byte(payload.Password), 14)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "error hashing
password"})
        return
    }

    employee.FirstName = payload.Name
    employee.Email = payload.Email
    employee.Password = string(hashPassword)

    if err := entity.DB().Create(&employee).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusCreated, gin.H{"data": employee})
}
```

}

-Source Code Back-End ใน folder controller ชื่อไฟล์ Equipment.go

```
package controller

import (
    "github.com/thanaponkhanoon/sa-65-example/entity"
    "github.com/gin-gonic/gin"
    "net/http"
)

// POST /equipment
func CreateEquipment(c *gin.Context) {
    var category entity.Category
    var employee entity.Employee
    var unit entity.Unit
    var equipment entity.Equipment
    //ผลลัพธ์ที่ได้จากขั้นตอนที่ 8 จะถูก bind เข้าตัวแปล equipment
    if err := c.ShouldBindJSON(&equipment); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
    }
}
```



```
        return
    }
    //9: ค้นหา Employee ด้วย id
    if tx := entity.DB().Where("id = ?",
equipment.EmployeeID).First(&employee); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "category not found"})
        return
    }
    //10: ค้นหา Category ด้วย id
    if tx := entity.DB().Where("id = ?",
equipment.CategoryID).First(&category); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "category not found"})
        return
    }
    //11: ค้นหา Unit ด้วย id
    if tx := entity.DB().Where("id = ?", equipment.UnitID).First(&unit);
tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "unit not found"})
        return
    }
    //12: สร้าง Equipment
    eq := entity.Equipment{
        Category: category,
        Unit:      unit,
        Employee: employee,
        Time:      equipment.Time,
        Name:      equipment.Name,
        Amount:    equipment.Amount,
    }
    //13: บันทึก
    if err := entity.DB().Create(&eq).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": eq})
}
// GET /equipment/:id
func GetEquipment(c *gin.Context) {
    var equipment entity.Equipment
    id := c.Param("id")
    if err := entity.DB().Raw("SELECT * FROM equipment WHERE id = ?", id).
        Preload("Category").
        Preload("Unit").
        Preload("Employee").
        Find(&equipment).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, gin.H{"data": equipment})
}
```

```
}
// GET /equipments
func ListEquipments(c *gin.Context) {
    var equipments []entity.Equipment
    if err := entity.DB().Raw("SELECT * FROM equipment").
        Preload("Category").
        Preload("Unit").
        Preload("Employee").
        Find(&equipments).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": equipments})
}

// DELETE /equipment/:id
func DeleteEquipment(c *gin.Context) {
    id := c.Param("id")
    if tx := entity.DB().Exec("DELETE FROM equipment WHERE id = ?", id);
    tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "equipment not
found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": id})
}

// PATCH /equipment
func UpdateEquipment(c *gin.Context) {
    var equipment entity.Equipment
    if err := c.ShouldBindJSON(&equipment); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if tx := entity.DB().Where("id = ?", equipment.ID).First(&equipment);
    tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "equipment not
found"})
        return
    }
    if err := entity.DB().Save(&equipment).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": equipment})
}
```

-Source Code Back-End ใน folder controller ชื่อไฟล์ Category.go

```
package controller

import (
    "github.com/thanaponkhanoon/sa-65-example/entity"
    "github.com/gin-gonic/gin"
    "net/http"
)

// POST /categories
func CreateCategory(c *gin.Context) {
    var category entity.Category
    if err := c.ShouldBindJSON(&category); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if err := entity.DB().Create(&category).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusCreated, gin.H{"data": category})
}
```

```
// GET /category/:id
func GetCategory(c *gin.Context) {
    var category entity.Category
    id := c.Param("id")
    if tx := entity.DB().Where("id = ?", id).First(&category);
tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "category not found"})
        return
    }

    c.JSON(http.StatusOK, gin.H{"data": category})
}

// GET /categories
func ListCategory(c *gin.Context) {
    var categories []entity.Category
    if err := entity.DB().Raw("SELECT * FROM
categories").Scan(&categories).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, gin.H{"data": categories})
}

// DELETE /categories/:id
func DeleteCategory(c *gin.Context) {
    id := c.Param("id")
    if tx := entity.DB().Exec("DELETE FROM categories WHERE id = ?", id);
tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "category not found"})
        return
    }

    c.JSON(http.StatusOK, gin.H{"data": id})
}

// PATCH /categories
func UpdateCategory(c *gin.Context) {
    var category entity.Category
    if err := c.ShouldBindJSON(&category); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    if tx := entity.DB().Where("id = ?", category.ID).First(&category);
tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "category not found"})
        return
    }
}
```

```
}

if err := entity.DB().Save(&category).Error; err != nil {
    c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
    return
}

c.JSON(http.StatusOK, gin.H{"data": category})
}
```

-Source Code Back-End ใน folder controller ชื่อไฟล์ Unit.go

```
package controller

import (
    "net/http"
    "github.com/gin-gonic/gin"
    "github.com/thanaponkhanon/sa-65-example/entity"
)
// POST /units
func CreateUnit(c *gin.Context) {
    var unit entity.Unit
    if err := c.ShouldBindJSON(&unit); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if err := entity.DB().Create(&unit).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusCreated, gin.H{"data": unit})
}
// GET /unit/:id
func GetUnit(c *gin.Context) {
    var unit entity.Unit
    id := c.Param("id")
```

G18 - ระบบจองใช้ห้อง (ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์

```
    if tx := entity.DB().Where("id = ?", id).First(&unit); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "unit not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": unit})
}
// GET /units
func ListUnits(c *gin.Context) {
    var units []entity.Unit
    if err := entity.DB().Raw("SELECT * FROM units").Scan(&units).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": units})
}
// DELETE /units/:id
func DeleteUnit(c *gin.Context) {
    id := c.Param("id")
    if tx := entity.DB().Exec("DELETE FROM units WHERE id = ?", id); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "unit not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": id})
}
// PATCH /units
func UpdateUnit(c *gin.Context) {
    var unit entity.Unit
    if err := c.ShouldBindJSON(&unit); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if tx := entity.DB().Where("id = ?", unit.ID).First(&unit); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "unit not found"})
        return
    }
    if err := entity.DB().Save(&unit).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": unit})
}
```

-Source Code Back-End ใน folder controller ชื่อไฟล์ Employee.go

```
package controller

import (
    "github.com/gin-gonic/gin"
    "github.com/thanaponkhanoon/sa-65-example/entity"
    "net/http"
)

// POST /employee
func CreateUser(c *gin.Context) {
    var employee entity.Employee
    if err := c.ShouldBindJSON(&employee); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if err := entity.DB().Create(&employee).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": employee})
}

// GET /employee/:id
func GetEmployee(c *gin.Context) {
    var employee entity.Employee
    id := c.Param("id")
```

G18 - ระบบจองใช้ห้อง (ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์

```
        if err := entity.DB().Raw("SELECT * FROM employees WHERE id = ?",
id).Scan(&employee).Error; err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
            return
        }
        c.JSON(http.StatusOK, gin.H{"data": employee})
    }

// GET /employees
func ListEmployees(c *gin.Context) {
    var employees []entity.Employee
    if err := entity.DB().Raw("SELECT * FROM
employees").Scan(&employees).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": employees})
}

// DELETE /employees/:id
func DeleteEmployee(c *gin.Context) {
    id := c.Param("id")
    if tx := entity.DB().Exec("DELETE FROM employees WHERE id = ?", id);
tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "employee not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": id})
}

// PATCH /employees
func UpdateEmployee(c *gin.Context) {
    var employee entity.Employee
    if err := c.ShouldBindJSON(&employee); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if tx := entity.DB().Where("id = ?", employee.ID).First(&employee);
tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "employee not found"})
        return
    }
    if err := entity.DB().Save(&employee).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": employee})
}
```


-Source Code Back-End ใน folder middlewares ชื่อไฟล์ authorize.go

```
package middlewares

import (
    "net/http"
    "strings"

    "github.com/gin-gonic/gin"
    "github.com/thanaponkhanoon/sa-65-example/service"
)
// validates token
func Authorizes() gin.HandlerFunc {
    return func(c *gin.Context) {
        clientToken := c.Request.Header.Get("Authorization")
        if clientToken == "" {
            c.AbortWithStatusJSON(http.StatusForbidden, gin.H{"error": "No Authorization header provided"})
            return
        }
        extractedToken := strings.Split(clientToken, "Bearer ")
        if len(extractedToken) == 2 {
            clientToken = strings.TrimSpace(extractedToken[1])
        } else {
            c.AbortWithStatusJSON(http.StatusBadRequest, gin.H{"error": "Incorrect Format of Authorization Token"})
            return
        }
        jwtWrapper := service.JwtWrapper{
            SecretKey: "SvNQpBN8y3qlVrsGAYYWoJJk56LtzFHx",
        }
```

```
        Issuer:    "AuthService",
    }
    _, err := jwtWrapper.ValidateToken(clientToken)
    if err != nil {
        c.AbortWithStatusJSON(http.StatusUnauthorized, gin.H{"error":
err.Error()})
        return
    }
    // c.Set("email", claims.Email)
    c.Next()
}
}
```

-Source Code Back-End ใน folder service ชื่อไฟล์ authentication.go

```
package service

import (
    "errors"
    "time"

    jwt "github.com/dgrijalva/jwt-go"
)
// JwtWrapper wraps the signing key and the issuer
type JwtWrapper struct {
    SecretKey    string
    Issuer       string
    ExpirationHours int64
}
// JwtClaim adds email as a claim to the token
type JwtClaim struct {
    Email string
    jwt.StandardClaims
}
// GenerateToken generates a jwt token
func (j *JwtWrapper) GenerateToken(email string) (signedToken string, err
error) {
    claims := &JwtClaim{
        Email: email,
        StandardClaims: jwt.StandardClaims{
            ExpiresAt: time.Now().Local().Add(time.Hour *
time.Duration(j.ExpirationHours)).Unix(),
            Issuer:    j.Issuer,
        },
    }
```

```
    },
    }
    token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
    signedToken, err = token.SignedString([]byte(j.SecretKey))
    if err != nil {
        return
    }
    return
}
// Validate Token validates the jwt token
func (j *JwtWrapper) ValidateToken(signedToken string) (claims *JwtClaim,
err error) {
    token, err := jwt.ParseWithClaims(
        signedToken,
        &JwtClaim{},
        func(token *jwt.Token) (interface{}, error) {
            return []byte(j.SecretKey), nil
        },
    )
    if err != nil {
        return
    }
    claims, ok := token.Claims.(*JwtClaim)
    if !ok {
        err = errors.New("Couldn't parse claims")
        return
    }
    if claims.ExpiresAt < time.Now().Local().Unix() {
        err = errors.New("JWT is expired")
        return
    }
    return
}
```

**-Source Code Front-End ใน folder src แล้วใน folder components
ชื่อไฟล์ EquipmentCreat.tsx**

```
import React, { useEffect, useState } from "react";
import { Link as RouterLink } from "react-router-dom";
import Button from "@mui/material/Button";
import FormControl from "@mui/material/FormControl";
import Container from "@mui/material/Container";
import Paper from "@mui/material/Paper";
import Grid from "@mui/material/Grid";
import Box from "@mui/material/Box";
import Typography from "@mui/material/Typography";
import Divider from "@mui/material/Divider";
import Snackbar from "@mui/material/Snackbar";
import Select, { SelectChangeEvent } from "@mui/material/Select";
import MenuItem from "@mui/material/MenuItem";
import MuiAlert, { AlertProps } from "@mui/material/Alert";
import TextField from "@mui/material/TextField";
import { AdapterDateFns } from "@mui/x-date-pickers/AdapterDateFns";
import { LocalizationProvider } from "@mui/x-date-pickers/LocalizationProvider";
import { DatePicker } from "@mui/x-date-pickers/DatePicker";

import { CategoriesInterface } from "../interfaces/Icategory";
import { EmployeesInterface } from "../interfaces/Iemployees";
import { UnitsInterface } from "../interfaces/Iunit";
import { EquipmentInterface } from "../interfaces/Iequipment";

import {
  GetCategory,
  GetUnit,
  GetEmployeeByUID,
  CreateEquipment,
} from "../services/HttpClientService";
```

G18 - ระบบจองใช้ห้อง (ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์

```
const Alert = React.forwardRef<HTMLDivElement, AlertProps>(function Alert(
  props,
  ref
) {
  return <MuiAlert elevation={6} ref={ref} variant="filled" {...props} />;
});

function EquipmentCreate() {
  const [categories, setCategories] = useState<CategoriesInterface[]>([]);
  const [units, setUnits] = useState<UnitsInterface[]>([]);
  const [employee, setEmployee] = useState<EmployeesInterface>();
  const [equipment, setEquipment] = useState<Partial<EquipmentInterface>>>({
    Time: new Date(),
  });

  const [success, setSuccess] = useState(false);
  const [error, setError] = useState(false);

  const handleClose = (
    event?: React.SyntheticEvent | Event,
    reason?: string
  ) => {
    if (reason === "clickaway") {
      return;
    }
    setSuccess(false);
    setError(false);
  };

  const handleChange = (event: SelectChangeEvent<number>) => {
    const name = event.target.name as keyof typeof equipment;
    setEquipment({
      ...equipment,
      [name]: event.target.value,
    });
  };

  const handleInputChange = (
    event: React.ChangeEvent<{ id?: string; value: any }>
  ) => {
    const id = event.target.id as keyof typeof equipment;
    const { value } = event.target;
    setEquipment({ ...equipment, [id]: value });
  };

  const getCategory = async () => {
    let res = await GetCategory();
    if (res) {
      setCategories(res);
    }
  }
}
```

```
};

const getUnit = async () => {
  let res = await GetUnit();
  if (res) {
    setUnits(res);
  }
};

const getEmployee = async () => {
  let res = await GetEmployeeByUID();
  equipment.EmployeeID = res.ID;
  if (res) {
    setEmployee(res);
  }
};

useEffect(() => {
  getCategory();
  getUnit();
  getEmployee();
}, []);

// console.log(equipment);

const convertType = (data: string | number | undefined) => {
  let val = typeof data === "string" ? parseInt(data) : data;
  return val;
};

async function submit() {
  let data = {
    CategoryID: equipment.CategoryID,
    UnitID: equipment.UnitID,
    Name: equipment.Name,
    Amount: convertType(equipment.Amount),
    EmployeeID: equipment.EmployeeID,
    Time: equipment.Time,
  };
};

let res = await CreateEquipment(data as EquipmentInterface);
if (res) {
  setSuccess(true);
} else {
  setError(true);
}
}

return (
  <Container maxWidth="md">
```

G18 - ระบบจองใช้ห้อง (ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์

```
<Snackbar
  open={success}
  autoHideDuration={3000}
  onClose={handleClose}
  anchorOrigin={{ vertical: "top", horizontal: "center" }}
>
  <Alert onClose={handleClose} severity="success">
    บันทึกข้อมูลสำเร็จ
  </Alert>
</Snackbar>
<Snackbar
  open={error}
  autoHideDuration={6000}
  onClose={handleClose}
  anchorOrigin={{ vertical: "top", horizontal: "center" }}
>
  <Alert onClose={handleClose} severity="error">
    บันทึกข้อมูลไม่สำเร็จ
  </Alert>
</Snackbar>
<Paper>
  <Box
    display="flex"
    sx={{
      marginTop: 2,
    }}
  >
    <Box sx={{ paddingX: 2, paddingY: 1 }}>
      <Typography
        component="h2"
        variant="h6"
        color="primary"
        gutterBottom
      >
        บันทึกข้อมูลอุปกรณ์
      </Typography>
    </Box>
  </Box>
  <Divider />
  <Grid container spacing={3} sx={{ padding: 2 }}>
    <Grid item xs={6}>
      <FormControl fullWidth variant="outlined">
        <p>ชื่ออุปกรณ์</p>
        <TextField
          id="Name"
          value={equipment.Name ?? ""}
          onChange={handleInputChange}
        />
      </FormControl>
    </Grid>
```

```
<Grid item xs={6}>
  <FormControl fullWidth variant="outlined">
    <p>จำนวน</p>
    <TextField
      id="Amount"
      type="number"
      value={equipment.Amount ?? 0}
      onChange={handleInputChange}
    />
  </FormControl>
</Grid>
<Grid item xs={6}>
  <FormControl fullWidth variant="outlined">
    <p>ประเภทอุปกรณ์</p>
    <Select
      value={equipment.CategoryID ?? 0}
      onChange={handleChange}
      inputProps={{
        name: "CategoryID",
      }}
    >
      <MenuItem aria-label="None" value={0}>
        กรุณาเลือกประเภทอุปกรณ์
      </MenuItem>
      {categories.map((item: CategoriesInterface) => (
        <MenuItem value={item.ID} key={item.ID}>
          {item.Name}
        </MenuItem>
      ))}
    </Select>
  </FormControl>
</Grid>
<Grid item xs={6}>
  <FormControl fullWidth variant="outlined">
    <p>สรรพนาม</p>
    <Select
      value={equipment.UnitID ?? 0}
      onChange={handleChange}
      inputProps={{
        name: "UnitID",
      }}
    >
      <MenuItem aria-label="None" value={0}>
        กรุณาเลือกอุปกรณ์
      </MenuItem>
      {units.map((item: UnitsInterface) => (
        <MenuItem value={item.ID} key={item.ID}>
          {item.Name}
        </MenuItem>
      ))}
    </Select>
  </FormControl>
</Grid>
```



```
        </Select>
      </FormControl>
    </Grid>
    <Grid item xs={6}>
      <FormControl fullWidth variant="outlined">
        <p>วันที่และเวลา</p>
        <LocalizationProvider dateAdapter={AdapterDateFns}>
          <DatePicker
            value={equipment.Time}
            onChange={(newValue) => {
              setEquipment({
                ...equipment,
                Time: newValue,
              });
            }}
            renderInput={(params) => <TextField {...params} />}
          />
        </LocalizationProvider>
      </FormControl>
    </Grid>
    <Grid item xs={12}>
      <Button
        component={RouterLink}
        to="/equipment"
        variant="contained"
        color="inherit"
      >
        กลับ
      </Button>
      <Button
        style={{ float: "right" }}
        onClick={submit}
        variant="contained"
        color="primary"
      >
        บันทึก
      </Button>
    </Grid>
  </Grid>
</Paper>
</Container>
);
}
```

```
export default EquipmentCreate;
```

**-Source Code Front-End ใน folder src แล้วใน folder components
ชื่อไฟล์ Equipments.tsx**

```
import React, { useState, useEffect } from "react";
import { Link as RouterLink } from "react-router-dom";
import Typography from "@mui/material/Typography";
import Button from "@mui/material/Button";
import Container from "@mui/material/Container";
import Box from "@mui/material/Box";
import { DataGrid, GridColDef } from "@mui/x-data-grid";
import { EquipmentInterface } from "../interfaces/Iequipment";
import { GetEquipment } from "../services/HttpClientService";
import moment from "moment";

function Equipments() {
  const [equipments, setEquipments] = useState<EquipmentInterface[]>([]);

  useEffect(() => {
    getEquipments();
  }, []);

  const getEquipments = async () => {
    let res = await GetEquipment();
    console.log(res);
    if (res) {
      setEquipments(res);
    }
  };

  const columns: GridColDef[] = [
    { field: "ID", headerName: "ลำดับ", width: 50 },
    {
      field: "Name",
      headerName: "ชื่ออุปกรณ์",
      width: 200,
      valueFormatter: (params) => params.value.Name,
    },
  ],
```

```
{
  field: "Amount",
  headerName: "จำนวน",
  width: 100,
  valueFormatter: (params) => params.value.Amount,
},
{
  field: "Category",
  headerName: "ประเภท",
  width: 250,
  valueGetter: (params) => params.row.Category.Name,
},
{
  field: "Unit",
  headerName: "สรรพนาม",
  width: 150,
  valueGetter: (params) => params.row.Unit.Name,
},
{
  field: "Employee",
  headerName: "พนักงาน",
  width: 250,
  valueGetter: (params) =>
`${params.row.Employee.FirstName} ${params.row.Employee.LastName}`,
},
{ field: "Time", headerName: "วันที่และเวลา", width: 250,
  valueFormatter: (params) => moment(params.value.Time)},
];

return (
  <div>
    <Container maxWidth="md">
      <Box
        display="flex"
        sx={{
          marginTop: 2,
        }}
      >
        <Box flexGrow={1}>
          <Typography
            component="h2"
            variant="h6"
            color="primary"
            gutterBottom
          >
            ข้อมูลการบันทึกอุปกรณ์
          </Typography>
        </Box>
        <Box>
          <Button
```

```
        component={RouterLink}
        to="/equipment/create"
        variant="contained"
        color="primary"
      >
        สร้างข้อมูล
      </Button>
    </Box>
  </Box>
  <div style={{ height: 400, width: "100%", marginTop: "20px" }}>
    <DataGrid
      rows={equipments}
      getRowId={(row) => row.ID}
      columns={columns}
      pageSize={5}
      rowsPerPageOptions={[5]}
    />
  </div>
</Container>
</div>
);
}

export default Equipments;
```

-Source Code Front-End ใน folder src แล้วใน folder components ชื่อไฟล์ home.tsx

```
import Container from "@mui/material/Container";

function Home() {
  return (
    <div>
      <Container sx={{ marginTop: 2 }} maxWidth="md">
        <h2 style={{ textAlign: "center" }}>ระบบบันทึกข้อมูลอุปกรณ์ </h2>
        <h4><u>Requirements</u></h4>
        <p>
          ระบบการจองใช้บริการห้องของบริษัท Room Booking
          เป็นระบบที่ให้ผู้ใช้บริการซึ่งเป็นสมาชิกต้อง Log in เข้าสู่ระบบเพื่อทำการจองห้องต่างๆที่สมาชิกต้องการ เช่น
          ห้องเรียน ห้องปฏิบัติการ หรือห้องอ่านหนังสือ
          นอกจากนี้ยังมีระบบข้อมูลอุปกรณ์เป็นระบบที่บอกรายละเอียดของข้อมูลอุปกรณ์ทั้งหมดมีอะไรบ้างทั้งที่มีให้ในห้อง
          และอุปกรณ์ที่สามารถให้สมาชิกระบบจองใช้ห้องดูอุปกรณ์เสริมได้ เช่น หมายเลขอุปกรณ์
          ประเภทของอุปกรณ์รวมถึงหากอุปกรณ์ชำรุดหรือเสียหายโดยมีพนักงานระบบดูแล

          </p>
          <br />
          <h4><u>User Story</u> (ระบบบันทึกการจองใช้ห้อง)</h4>
          <p>
            <b>ในบทบาทของ</b> พนักงานระบบ<br />
            <b>ฉันต้องการ</b> บันทึกรายละเอียดข้อมูลอุปกรณ์<br />
            <b>เพื่อ</b> ให้สมาชิกระบบจองใช้ห้องสามารถดูรายละเอียดของอุปกรณ์เพื่อทำการยืมได้<b>
          </p>
        </Container>
      </div>
    );
}
export default Home;
```

**-Source Code Front-End ใน folder src แล้วใน folder components
ชื่อไฟล์ Navbar.tsx**

```
import * as React from "react";
import { Link } from "react-router-dom";
import AppBar from "@mui/material/AppBar";

import Box from "@mui/material/Box";
import Toolbar from "@mui/material/Toolbar";
import Typography from "@mui/material/Typography";
import IconButton from "@mui/material/IconButton";
import MenuIcon from "@mui/icons-material/Menu";
import HomeIcon from "@mui/icons-material/Home";
import BrowserUpdatedIcon from '@mui/icons-material/BrowserUpdated';
import WorkHistoryIcon from '@mui/icons-material/WorkHistory';
import LogoutIcon from '@mui/icons-material/Logout';

import Drawer from "@mui/material/Drawer";
import List from "@mui/material/List";
import ListItem from "@mui/material/ListItem";
import ListItemText from "@mui/material/ListItemText";
import ListItemIcon from "@mui/material/ListItemIcon";

const menu = [
  { name: "หน้าแรก", icon: <HomeIcon />, path: "/" },
  { name: "บันทึกข้อมูลอุปกรณ์", icon: <BrowserUpdatedIcon />, path:
"/equipment/create" },
  { name: "ข้อมูลอุปกรณ์", icon: <WorkHistoryIcon />, path: "/equipment" },
];

function Navbar() {
  const [open, setOpen] = React.useState(false);
  const signOut = () => {
    localStorage.clear()
    window.location.href = "/";
  }
  return (
    <Box sx={{ flexGrow: 1 }}>
      <AppBar position="static">
        <Toolbar>
          <IconButton
```

```
        size="large"
        edge="start"
        color="inherit"
        aria-label="menu"
        onClick={() => setOpen(true)}
        sx={{ mr: 2 }}
      >
      <MenuIcon />
    </IconButton>
    <Typography variant="h6" component="div" sx={{ flexGrow: 1 }}>
      ระบบข้อมูลอุปกรณ์
    </Typography>
  </Toolbar>
  <Drawer variant="temporary" open={open} onClose={() =>
setOpen(false)}>
    <List>
      {menu.map((item, index) => (
        <Link
          to={item.path}
          key={item.name}
          style={{ textDecoration: "none", color: "inherit" }}
        >
          <ListItem button>
            <ListItemIcon>{item.icon}</ListItemIcon>
            <ListItemText primary={item.name} />
          </ListItem>
        </Link>
      ))}
      <ListItem button onClick={signOut}>
        <ListItemIcon><LogoutIcon /></ListItemIcon>
        <ListItemText primary="ออกจากระบบ" />
      </ListItem>
    </List>
  </Drawer>
</AppBar>
</Box>

);
}
export default Navbar;
```

**-Source Code Front-End ใน folder src แล้วใน folder components
ชื่อไฟล์ Signin.tsx**

```
import React, { useState } from "react";
import Avatar from "@mui/material/Avatar";
import Button from "@mui/material/Button";
import CssBaseline from "@mui/material/CssBaseline";
import TextField from "@mui/material/TextField";
import FormControlLabel from "@mui/material/FormControlLabel";
import Checkbox from "@mui/material/Checkbox";
import Paper from "@mui/material/Paper";
import Box from "@mui/material/Box";
import Grid from "@mui/material/Grid";
import LockOutlinedIcon from "@mui/icons-material/LockOutlined";
import Typography from "@mui/material/Typography";
import Snackbar from "@mui/material/Snackbar";
import MuiAlert, { AlertProps } from "@mui/material/Alert";
import { createTheme, ThemeProvider } from "@mui/material/styles";
import { SigninInterface } from "../interfaces/ISignin";
import { Login } from "../services/HttpClientService";

const Alert = React.forwardRef<HTMLDivElement, AlertProps>(function Alert(
  props,
  ref
) {
  return <MuiAlert elevation={6} ref={ref} variant="filled" {...props} />;
});

const theme = createTheme();

function SignIn() {
  const [signin, setSignin] = useState<Partial<SigninInterface>>({});
  const [success, setSuccess] = useState(false);
  const [error, setError] = useState(false);

  const handleInputChange = (
    event: React.ChangeEvent<{ id?: string; value: any }>
  ) => {
    const id = event.target.id as keyof typeof signin;
    const { value } = event.target;
    setSignin({ ...signin, [id]: value });
  };
}
```


G18 - ระบบจองใช้ห้อง (ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิทย์

```
const handleClose = (
  event?: React.SyntheticEvent | Event,
  reason?: string
) => {
  if (reason === "clickaway") {
    return;
  }
  setSuccess(false);
  setError(false);
};

const submit = async () => {
  let res = await Login(signin);
  if (res) {
    setSuccess(true);
    setTimeout(() => {
      window.location.reload();
    }, 1000);
  } else {
    setError(true);
  }
};

return (
  <ThemeProvider theme={theme}>
    <Grid container component="main" sx={{ height: "100vh" }}>
      <Snackbar
        open={success}
        autoHideDuration={3000}
        onClose={handleClose}
        anchorOrigin={{ vertical: "top", horizontal: "center" }}
      >
        <Alert onClose={handleClose} severity="success">
          เข้าสู่ระบบสำเร็จ
        </Alert>
      </Snackbar>
      <Snackbar
        open={error}
        autoHideDuration={3000}
        onClose={handleClose}
        anchorOrigin={{ vertical: "top", horizontal: "center" }}
      >
        <Alert onClose={handleClose} severity="error">
          อีเมลหรือรหัสผ่านไม่ถูกต้อง
        </Alert>
      </Snackbar>

      <CssBaseline />
      <Grid
        item
```

```
xs={false}
sm={4}
md={7}
sx={{
  backgroundImage: "url(https://source.unsplash.com/random)",
  backgroundRepeat: "no-repeat",
  backgroundColor: (t) =>
    t.palette.mode === "light"
    ? t.palette.grey[50]
    : t.palette.grey[900],
  backgroundSize: "cover",
  backgroundPosition: "center",
}}
/>
<Grid item xs={12} sm={8} md={5} component={Paper} elevation={6}
square>
  <Box
    sx={{
      my: 8,
      mx: 4,
      display: "flex",
      flexDirection: "column",
      alignItems: "center",
      alignSelf: "center",
    }}
  >
    <Avatar sx={{ m: 1, bgcolor: "secondary.main" }}>
      <LockOutlinedIcon />
    </Avatar>
    <Typography component="h1" variant="h5">
      Sign in
    </Typography>
    <Box sx={{ mt: 1 }}>
      <TextField
        margin="normal"
        required
        fullWidth
        id="Email"
        label="Email Address"
        name="email"
        autoComplete="email"
        autoFocus
        value={signin.Email || ""}
        onChange={handleInputChange}
      />
      <TextField
        margin="normal"
        required
        fullWidth
        name="password"
```

```
        label="Password"
        type="password"
        id="Password"
        autoComplete="current-password"
        value={signin.Password || ""}
        onChange={handleInputChange}
      />
      <FormControllabel
        control={<Checkbox value="remember" color="primary" />}
        label="Remember me"
      />
      <Button
        type="submit"
        fullWidth
        variant="contained"
        sx={{ mt: 3, mb: 2 }}
        onClick={submit}
      >
        Sign In
      </Button>
    </Box>
  </Box>
</Grid>
</Grid>
</ThemeProvider>
);
}

export default SignIn;
```

**-Source Code Front-End ใน folder src แล้วใน folder interfaces
ชื่อไฟล์ Icategory.tsx**

```
export interface CategoriesInterface {  
  ID: number,  
  Name: string,  
}
```

**-Source Code Front-End ใน folder src แล้วใน folder interfaces
ชื่อไฟล์ Iemployees.tsx**

```
export interface EmployeesInterface {  
  ID?: number,  
  FirstName?: string,  
  LastName?: string,  
  Email?: string,  
  Password?: string  
}
```

**-Source Code Front-End ใน folder src แล้วใน folder interfaces
ชื่อไฟล์ Iequipment.tsx**

```
import { CategoriesInterface } from "../Icategory";  
import { UnitsInterface } from "../Iunit";  
import { EmployeesInterface } from "../Iemployees"  
  
export interface EquipmentInterface {  
  ID?: number;  
  Name?: string;  
  Amount?: number;  
  CategoryID?: number;  
  Category?: CategoriesInterface;  
  UnitID?: number;  
  Unit?: UnitsInterface;  
  Time: Date | null;  
  EmployeeID?: number;  
  Employee?: EmployeesInterface  
}
```

**-Source Code Front-End ใน folder src แล้วใน folder interfaces
ชื่อไฟล์ ISignin.tsx**

```
export interface SigninInterface {  
  Email?: string,  
  Password?: string,
```

}

**-Source Code Front-End ใน folder src แล้วใน folder interfaces
ชื่อไฟล์ lunit.tsx**

```
export interface UnitsInterface {  
  ID: number,  
  Name: string,  
}
```

**-Source Code Front-End ใน folder src แล้วใน folder interfaces
ชื่อไฟล์ HttpClientService.tsx**

```
import React from "react";  
import { SigninInterface } from "../interfaces/ISignin";  
import { EmployeesInterface } from "../interfaces/Iemployees";  
import { EquipmentInterface } from "../interfaces/Iequipment";  
  
const apiUrl = "http://localhost:8080";  
  
async function Login(data: SigninInterface) {  
  const requestOptions = {  
    method: "POST",  
    headers: { "Content-Type": "application/json" },  
    body: JSON.stringify(data),  
  };  
  
  let res = await fetch(`${apiUrl}/login`, requestOptions)  
    .then((response) => response.json())  
    .then((res) => {  
      if (res.data) {  
        localStorage.setItem("token", res.data.token);  
        localStorage.setItem("uid", res.data.id);  
        return res.data;  
      } else {  
        return false;  
      }  
    });  
  
  return res;  
}  
  
async function GetCategory() {  
  const requestOptions = {  
    method: "GET",
```

```
        headers: {
            Authorization: `Bearer ${localStorage.getItem("token")}`,
            "Content-Type": "application/json",
        },
    };

    let res = await fetch(`${apiUrl}/catagories`, requestOptions)
    .then((response) => response.json())
    .then((res) => {
        if (res.data) {
            return res.data;
        } else {
            return false;
        }
    });

    return res;
}

async function GetUnit() {
    const requestOptions = {
        method: "GET",
        headers: {
            Authorization: `Bearer ${localStorage.getItem("token")}`,
            "Content-Type": "application/json",
        },
    };

    let res = await fetch(`${apiUrl}/units`, requestOptions)
    .then((response) => response.json())
    .then((res) => {
        if (res.data) {
            return res.data;
        } else {
            return false;
        }
    });

    return res;
}

async function GetEmployeeByUID() {
    const requestOptions = {
        method: "GET",
        headers: {
            Authorization: `Bearer ${localStorage.getItem("token")}`,
            "Content-Type": "application/json",
        },
    };
};
```

```
    let res = await
fetch(`${apiUrl}/employee/${localStorage.getItem("uid")}`, requestOptions)
    .then((response) => response.json())
    .then((res) => {
        if (res.data) {
            return res.data;
        } else {
            return false;
        }
    });

    return res;
}

async function CreateEquipment(data: EquipmentInterface) {
    const requestOptions = {
        method: "POST",
        headers: {
            Authorization: `Bearer ${localStorage.getItem("token")}`,
            "Content-Type": "application/json",
        },
        body: JSON.stringify(data),
    };

    let res = await fetch(`${apiUrl}/equipment`, requestOptions)
        .then((response) => response.json())
        .then((res) => {
            if (res.data) {
                return res.data;
            } else {
                return false;
            }
        });

    return res;
}

async function GetEquipment() {
    const requestOptions = {
        method: "GET",
        headers: {
            Authorization: `Bearer ${localStorage.getItem("token")}`,
            "Content-Type": "application/json",
        },
    };

    let res = await fetch(`${apiUrl}/equipments`, requestOptions)
        .then((response) => response.json())
        .then((res) => {
            if (res.data) {
```

G18 - ระบบจองใช้ห้อง
(ระบบบันทึกข้อมูลอุปกรณ์)

B6220655 ธนพล ไชยนิติก

```
        return res.data;
    } else {
        return false;
    }
});

return res;
}

export {
    Login,
    GetCategory,
    GetUnit,
    GetEmployeeByUID,
    CreateEquipment,
    GetEquipment,
};
```