



## ระบบร้านขายเกมออนไลน์

Game Shop Online System

โดย

นายณัฐทักษ์ดนาย	ແຫຍງກາຮະໂທກ	รหັສນັກສຶກສາ B6611613	ກລຸ່ມທີ 2
นายกิตตินันท์	ປ່ຈຍໂຄຄາ	รหັສນັກສຶກສາ B6618643	ກລຸ່ມທີ 2
นายทองนรินทร์	ແຢັ້ມຄວີ	รหັສນັກສຶກສາ B6627713	ກລຸ່ມທີ 2
นายปุณณพัฒน์	ເກົ່າຂອມ	รหັສນັກສຶກສາ B6649273	ກລຸ່ມທີ 2
นายณัฐนันท์	ຈັນທຽບສຸຮິຍາ	รหັສນັກສຶກສາ B6641054	ກລຸ່ມທີ 2

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ ดร.ຄมศัลล์ ครีวิสุทธิ์

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา ENG23 3031 การวิเคราะห์และออกแบบระบบ

หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนาครี

ประจำภาคการศึกษาที่ 1 ปีการศึกษา 2568

## ระบบร้านขายเกมออนไลน์

ระบบร้านขายเกมออนไลน์ เป็นระบบที่ให้บริการสำหรับผู้ใช้งานสามารถทำการเข้าสู่ระบบ Login เพื่อ เลือกซื้อ เกมต่าง ๆ ที่เปิดขายบนแพลตฟอร์ม โดยระบบจะจัดแสดงข้อมูลผ่าน ระบบจัดการข้อมูลเกม ที่ รวมถึงชื่อเกม ประเภทเกม ค่ายผู้ผลิต ราคา คำอธิบาย และภาพตัวอย่าง เพื่ออำนวยความสะดวกให้กับ ผู้ใช้งาน ระบบมี ระบบ workshop ที่ผู้ใช้งานจะสามารถนำมودเกมที่ผู้ใช้งานสร้างสรรค์เองมาอัปโหลดลงบนระบบของเราเพื่อเผยแพร่ มอดของตัวเองและระบบยังสามารถให้ผู้ใช้คนอื่นๆสามารถดาวน์โหลดไปใช้ได้แล้วก็ยังสามารถมาให้คะแนน มอดได้ด้วย โดยผู้ใช้งานสามารถเลือกซื้อเกมผ่าน ระบบจัดการการชำระเงิน พร้อมกับการประมวลผลคำสั่งซื้อ ระบบจัดการโปรโมชั่น สามารถเพิ่มโปรโมชั่นตามเทคโนโลยีได้เมื่อมีความผิดพลาดหรือความไม่พึง พ่อใจในการซื้อ ระบบมี ระบบการจัดการคืนเงิน ที่ให้ผู้ใช้งานส่งคำขอคืนเงิน พร้อมตรวจสอบเงื่อนไขตามที่ ระบบกำหนด หากผู้ใช้ ต้องการเสนอให้ร้านนำเกมที่ยังไม่มีเข้ามาจำหน่าย สามารถใช้งานระบบ - รีเควสเกมสำหรับการขาย ซึ่งสมาชิก สามารถพิมพ์ชื่อเกมและรายละเอียดเกมที่ต้องการ พร้อมระบบโหวตจาก ผู้ใช้คนอื่น เพื่อช่วยให้ผู้ดูแลพิจารณา เพื่อสร้างประสบการณ์ที่ดียิ่งขึ้น สมาชิกสามารถแสดงความคิดเห็นและให้คะแนนเกมผ่าน ระบบรีวิวและให้ คะแนน ซึ่งรองรับการให้คะแนนใน ระดับ 1-5 ดาว พร้อมข้อความรีวิวและระบบนี้จะเชื่อมกับหน้าเกมโดยตรง ระบบยังมีระบบคอมมูนิตี้- ขนาดเล็กที่เปิดให้ผู้ใช้งานตั้งกระทู้ถาม-ตอบ และเปลี่ยนประสบการณ์ และพูดคุย เกี่ยวกับเกม พร้อม ความสามารถในการแสดงความคิดเห็น กดถูกใจ ระบบมี ระบบจัดการสิทธิผู้ใช้งาน ที่ สามารถ กำหนด ระดับสิทธิของผู้ใช้งานทั่วไป ผู้ดูแลระบบ และพนักงาน ใน การเข้าถึงฟังก์ชันต่าง ๆ ภายใต้ระบบ นอกจากนี้ยัง มี ระบบรายงานปัญหา ที่ให้สมาชิกสามารถแจ้งปัญหาการใช้งาน เช่น ปัญหาในการซื้อเกม ระบบ ขัดข้อง ซึ่ง จะถูกส่งต่อให้แอดมินทำการตรวจสอบและดำเนินการต่อไป

## ระบบที่เกิดขึ้นจาก ระบบฐานข่ายเกมออนไลน์

- ระบบย่อย ระบบปรีเคสเกมสำหรับการขาย
- ระบบย่อย ระบบจัดการข้อมูลเกม
- ระบบย่อย ระบบสร้างคอนเทนต์เกมใน workshop
- ระบบย่อย ระบบจัดการสิทธิผู้ใช้งาน
- ระบบย่อย ระบบจัดการการชำระเงิน
- ระบบย่อย ระบบคอมมูนิตี้ขนาดเล็ก
- ระบบย่อย ระบบบริวิวและให้คะแนน
- ระบบย่อย ระบบจัดการโปรโมชัน
- ระบบย่อย ระบบยืนยันคำร้องขอคืนเงิน
- ระบบย่อย ระบบรายงานปัญหา

B6611613 นายณัฐพงษ์ดันนัย แหยงกระโทก

ระบบหลัก ระบบร้านขายเกมออนไลน์

ระบบย่อย ระบบบริโภคเกมสำหรับการขาย

ระบบซื้อ-ขายเกม เป็นระบบที่ให้ลูกค้า (Customer) สามารถเข้ามาซื้อเกมผ่านระบบของเราราได้ โดยสามารถทำการลงทะเบียน เข้ามาซื้อเกมโดยลูกค้าสามารถรีโภคเกมที่อยากรับน้ำมายในแต่ละเดือนผ่านระบบบริโภคเกมสำหรับการขาย โดยจะให้กรอกชื่อเกม หมวดหมู่ของเกม วันที่รีโภค และเหตุผลที่อยากได้เกมนั้น เพื่อที่ระบบการขายจะได้เห็นความต้องการลูกค้าแล้วนำเกมที่ลูกค้าการจำแนกมาขายได้

### User Story ระบบบริโภคเกมสำหรับการขาย

ในบทบาทของ (As a) ผู้ดูแลระบบ

ฉันต้องการ (I want to) ข้อมูลจำนวนรีโภคเกมจากผู้ใช้

เพื่อ (So that) จะสามารถใช้ข้อมูลนั้นวิเคราะห์ว่าเกมใดควรวางขายในระบบ

ในบทบาทของ (As a) ลูกค้า

ฉันต้องการ (I want to) ส่งรีโภคเกมที่อยากรับน้ำมาย

เพื่อที่ (So that) สามารถซื้อเกมที่ฉันสนใจอนาคตได้

Output บนหน้าจอ

- ลูกค้าจะสามารถรีโภคเกม(Create)จากที่ต้องการได้ จากนั้นระบบจะดึงข้อมูล(Read)มาแสดงเกมที่มีการรีโภคและจำนวนรีโภคของแต่ละเกมที่ลูกค้าเลือกมา

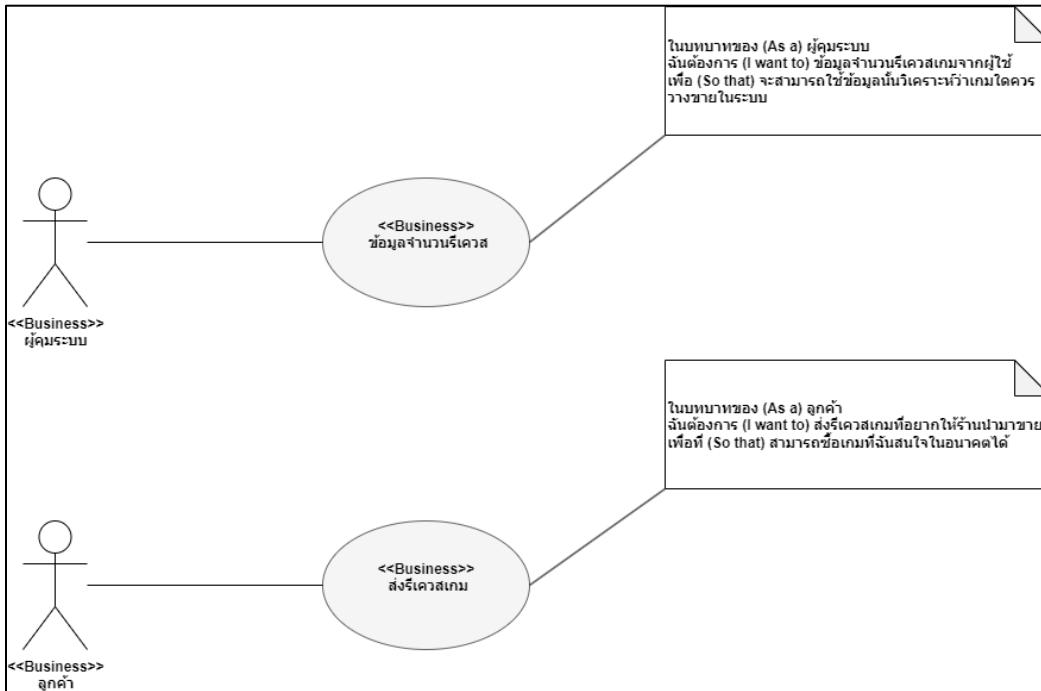
Output ของข้อมูล

- ระบบจะทำการเพิ่มข้อมูลของผู้ใช้ที่รีโภคเกมมา เข้าในฐานข้อมูลของเกมที่มีการรีโภค

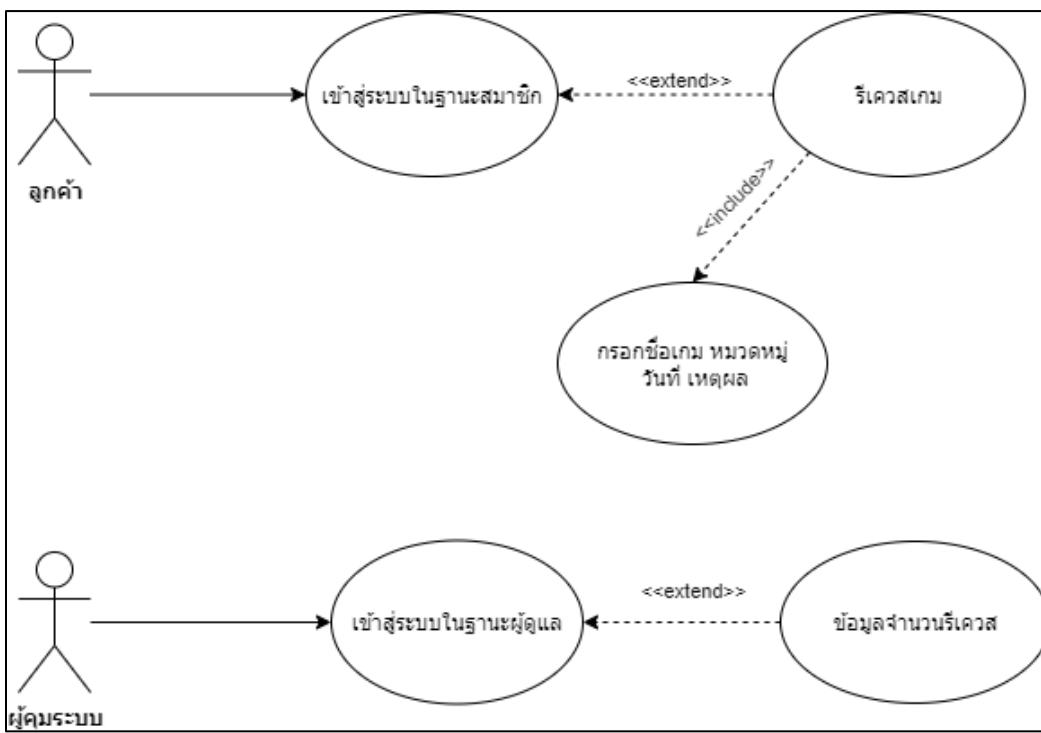
## คำนำที่อาจจะถูกยกมาเป็นตารางในฐานข้อมูลของระบบ

คำนำ	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
ลูกค้า	เกี่ยวข้องโดยตรง เนื่องจากผู้ใช้งานจะเป็นคนสร้างรีเควสต์ขึ้นมา เกี่ยวข้องกับจำนวนลูกค้าที่รีเควสต์
เกม	เกี่ยวข้องโดยตรง กับจำนวนลูกค้าที่รีเควสเกมนั้นๆ
หมวดหมู่	เกี่ยวข้องโดยตรง กับเกม
วันที่	ไม่เกี่ยวข้องโดยตรง
เหตุผล	เกี่ยวข้องโดยตรง เนื่องจากลูกค้าอาจมีเหตุผลในการรีเควสเกมนั้นหรือไม่ก็ได้
จำนวน	เกี่ยวข้องโดยตรง เนื่องจากในเกมเดียวก็มีจำนวนลูกค้าที่รีเควสเกมนั้นต่างกัน ซึ่งเป็นตัวตัดสินในการเลือกเกมมาวางขาย
รีเควสเกม	เกี่ยวข้องโดยตรง เนื่องจากผู้ใช้งานจะเป็นคนสร้างรีเควสเกม

## Business Use Case Diagram (แบบเดี่ยว)



## System use Case Diagram (แบบเดี่ยว)



### Checklist: อธิบาย

- การรีเคสเกมลูกค้าจำเป็นต้องล็อกอินในฐานะสมาชิกก่อน
- ในระหว่างที่รีเคสเกมจำเป็นต้องกรอกรายละเอียดเกมที่รีเคสไป
- ผู้ดูระบบจะเข้าถึงข้อมูลได้ต้องล็อกอินเข้าสู่ระบบก่อน

### Checklist: System Use Case Diagram

1. System Actor และ System Use Case ต้องไม่มีอะไรกำกับ
2. เส้นโยงจาก System Actor ไปหา System Use Case จะต้องเป็นเส้นทึบ หัวลูกศรปลายเปิดซึ่งไปหา System Use Case
3. System Actor ต้องเป็นบทบาทของคนที่ใช้ระบบจริง ๆ
4. System Use Case ต้องเป็นคำที่แสดงพฤติกรรม
5. ถ้ามีกลไกทาง Security มาเกี่ยว ชื่อ System Use Case จะอยู่ในรูปแบบ “เข้าระบบในฐาน <Actor>” เท่านั้น
6. การใช้ <--<<extend>>--> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ทางเลือกไปยัง System Use Case หลัก
7. การใช้ <--<<include>>--> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ที่ใหญ่กว่า ไปยัง System Use Case ที่จะถูกรวบเข้ามา

## ลูกค้า (customer User)

- ชื่อ Entity: Customer
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Username: เก็บในรูปแบบ varchar ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Password: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Email: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - First Name: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Last Name: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Birthday: เก็บในรูปแบบ date ไม่สามารถเป็นค่า Null ได้

ID custom er INTEGER NOT NULL, PK	USERNA ME VERCHAR NOT NULL, Unique	PASSWORD VARCHAR NOT NULL	EMAIL VARCHAR NOT NULL	FIRST_NA ME VARCHAR NOT NULL	LAST_NA ME VARCHAR NOT NULL	BIRTHD AY DATE NOT NULL
1001	Meber01	Encrypt(1234 56)	demo@gmail.co m	SA	67	19-12- 2000
1002	Meber02	Encrypt(1234 56)	demo01@gmail.c om	SE	67	19-12- 2000

## รีเควสต์(request)

- ชื่อ Entity: request
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Reason: เก็บในรูปแบบ varchar ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Date: เก็บในรูปแบบ date ไม่สามารถเป็นค่า Null ได้
  - User\_id: เป็น Foreign Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Game\_id: เป็น Foreign Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้

ID <b>Request</b> INTEGER NOT NULL, PK	Reason VARCHAR NOT NULL	Date DATE NOT NULL	User_id INTEGER FK	Game_id INTEGER FK
1001	Love it	19-12- 2000	1	1
1002	scary	19-12- 2000	2	2

## UI Design

### Create Request

ชื่อ\*

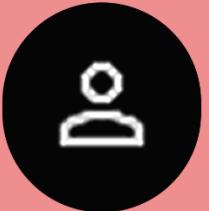
กรุณากรอกชื่อเกنم

姓氏\*

กรุณากรอก姓氏

วันที่\*

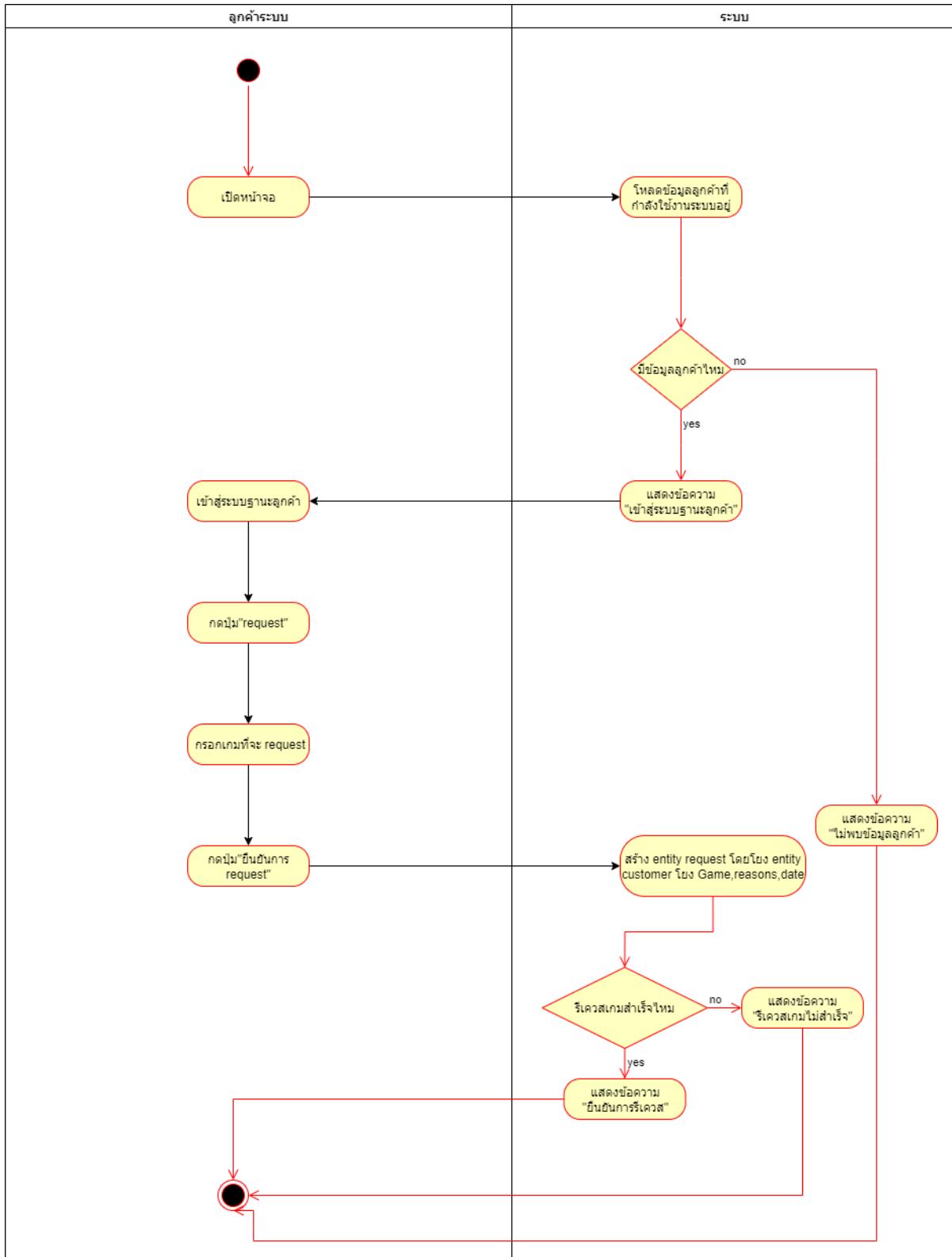
กรุณากรอกวันที่

 Nattakdanai Yangkratok





## Activity Diagram



B6611613 นายณัฐก์ชุดนัย แหยงกระโทก

ระบบหลัก ระบบร้านขายเกมออนไลน์

ระบบย่อย ระบบจัดการเกม

ข้อมูลเกมมีบทบาทสำคัญในการทำให้ลูกค้าสามารถเข้าถึงข้อมูลที่จำเป็นต่อการตัดสินใจซื้อเกมอย่างมีประสิทธิภาพ โดยรายละเอียดของเกมที่ถูกจัดเก็บและแสดงผลจะช่วยให้ลูกค้าเข้าใจเนื้อหา แนวเกม และความต้องการด้านเทคนิคของเกมนั้น ๆ ได้ดียิ่งขึ้น เช่น การแสดง สเปกขั้นต่ำของคอมพิวเตอร์ จะช่วยให้ลูกค้าทราบว่า เครื่องของตนสามารถเล่นเกมได้หรือไม่ ซึ่งช่วยลดปัญหาหลังการขายและเพิ่มความพึงพอใจให้กับลูกค้า

ส่วนบทบาทผู้ควบคุมจะสามารถจัดการข้อมูลเกมเพื่อแสดงรายละเอียดเกม สเปกขั้นต่ำของคอมพิวเตอร์ รายวัลของเกม และรีวิวของลูกค้าผ่านระบบรีวิวและให้คะแนน ที่เก็บข้อมูลรีวิว ชื่อเกมที่รีวิว คะแนน ความคิดเห็น วันที่รีวิว

## User Story ระบบจัดการเกม

ในบทบาทของ (As a) แอดมิน

ฉันต้องการ (I want to) จัดการเกม

เพื่อ (So that) เพิ่มเกมเข้าสู่ระบบ

## Output บนหน้าจอ

- ผู้ควบคุมระบบจะสามารถกดแก้ไขข้อมูลเกม -> ผู้ควบคุมจะสามารถกดเพิ่ม, แก้ไข หรือ ลบ ข้อมูลเกมนั้นๆได้

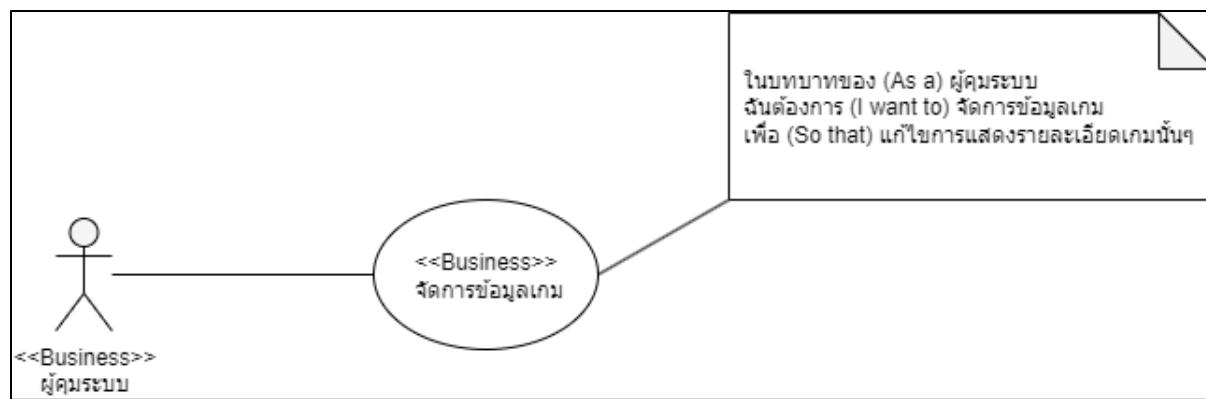
## Output ของข้อมูล

- ระบบจะดึงข้อมูล(Read)รายละเอียดของเกมนั้นๆ มาเพื่อให้ผู้ควบคุมระบบสามารถแก้ไขรายละเอียด (update) ของเกมนั้นๆ

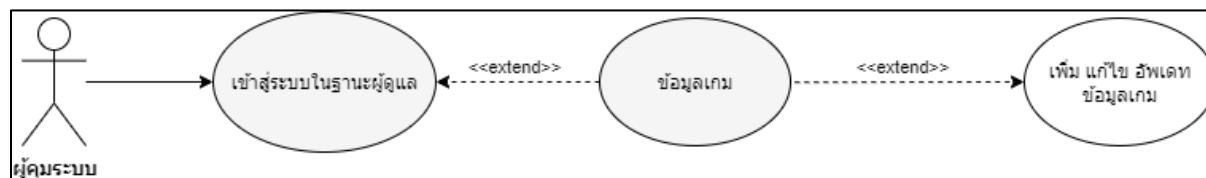
## คำนำที่อาจจะกล้ายมาเป็นตารางในฐานข้อมูลของระบบ

คำนำ	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
ที่เก็บข้อมูลรีวิว	เกี่ยวข้องโดยตรง เกี่ยวข้องผู้ใช้งานที่สร้างรีวิวเกมขึ้นมา
ชื่อเกมที่รีวิว	เกี่ยวข้องโดยตรง เนื่องจากผู้ใช้งานแต่ละคนจะมีชื่อที่รีวิวเกมนั้นๆ
คะแนน	เกี่ยวข้องโดยตรง กับผู้ใช้งานที่ให้คะแนนเกมนั้นๆ
ความคิดเห็น	เกี่ยวข้องโดยตรง กับผู้ใช้งานที่มีข้อมูลความคิดเห็นของผู้ใช้งานนั้นๆ
วันที่รีวิว	เกี่ยวข้องโดยตรง เพื่อบอกวันเวลาที่ผู้ใช้งานนั้นรีวิว
สเปกขั้นต่ำของคอมพิวเตอร์	เกี่ยวข้องโดยตรง เพื่อบอกสเปกขั้นต่ำของเกมนั้นๆ
รางวัล	เกี่ยวข้องโดยตรง กับเกมที่มีรางวัล
ข้อมูลเกม	เกี่ยวข้องโดยตรง กับข้อมูลเกมนั้นๆ

## Business Use Case Diagram (แบบเดี่ยว)



## System use Case Diagram (แบบเดี่ยว)



### Checklist: อธิบาย

- ผู้ดูแลระบบสามารถเข้าถึงข้อมูลเกมได้ฝ่ายเดียว
- การเข้าถึงข้อมูลเกมนั้นไม่จำเป็นที่จะต้องแก้ไขข้อมูลเกมก็ได้

### Checklist: System Use Case Diagram

1. System Actor และ System Use Case ต้องไม่มีอะไรกำกับ
2. เส้นโยงจาก System Actor ไปหา System Use Case จะต้องเป็นเส้นทึบ หัวลูกศรปลายเปิดซึ่งไปหา System Use Case
3. System Actor ต้องเป็นบทบาทของคนที่ใช้ระบบจริง ๆ
4. System Use Case ต้องเป็นคำที่แสดงพฤติกรรม
5. ถ้ามีกลไกทาง Security มาเกี่ยว ชื่อ System Use Case จะอยู่ในรูปแบบ “เข้าระบบในฐานะ <Actor>” เช่นนั้น
6. การใช้ <--<<extend>>---> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ทางเลือกไปยัง System Use Case หลัก
7. การใช้ <--<<include>>---> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ที่ใหญ่กว่า ไปยัง System Use Case ที่จะถูกรวบเข้ามา

## ผู้ดูแล (Admin User)

- ชื่อ Entity: Admin
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Username: เก็บในรูปแบบ varchar ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Password: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Email: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - First Name: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Last Name: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Birthday: เก็บในรูปแบบ date ไม่สามารถเป็นค่า Null ได้

ID admin INTEGE R NOT NULL, PK	USERNA ME VERCHAR NOT NULL, Unique	PASSWORD VARCHAR NOT NULL	EMAIL VARCHAR NOT NULL	FIRST_NA ME VARCHAR NOT NULL	LAST_NA ME VARCHAR NOT NULL	BIRTHD AY DATE NOT NULL
1001	Meber01	Encrypt(1234 56)	demo@gmail.com	SA	67	19-12- 2000
1002	Meber02	Encrypt(1234 56)	demo01@gmail.c om	SE	67	19-12- 2000

## เกม (Game)

- ชื่อ Entity: game
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - ID customer: เป็น Foreign Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Game: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Review: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Date: เก็บในรูปแบบ date ไม่สามารถเป็นค่า Null ได้
  - Minimum Spec: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Reward: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้

ID Game INTEGER NOT NULL, PK	Game VARCHAR	C_id INTEGER FK	Date DATE NOT NULL	M_id INTEGER FK	Img_src VARCHAR	Status VARCHAR	Agerating Integer
1001	Kf2	1	19- 12- 2000	1	fasf	pending	16
1002	Db	2	19- 12- 2000	2	asfasf	pending	20

## หมวดหมู่ (categories)

- ชื่อ Entity: categories
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - title: เก็บในรูปแบบ varchar UNIQUE

ID Categories	Title
INTEGER NOT NULL, PK	VARCHAR UNIQUE
1001	Fps
1002	horror

## สเปคขั้นต่ำ(minimum spec)

- ชื่อ Entity: **minimum spec**
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - OS: เก็บในรูปแบบ varchar
  - Processor: : เก็บในรูปแบบ varchar
  - Memory: : เก็บในรูปแบบ varchar
  - Graphics: : เก็บในรูปแบบ varchar
  - Storage: : เก็บในรูปแบบ varchar

ID Minimum spec INTEGER NOT NULL, PK	OS VARCHAR	Processor VARCHAR	Memory VARCHAR	Graphics VARCHAR	Storage VARCHAR
1001	Window10	I5 10300	32 gb	Gtx 1090	12 gb
1002	Window 7	I7 11900	16 gb	Rtx 2020	200 mb

UI Design

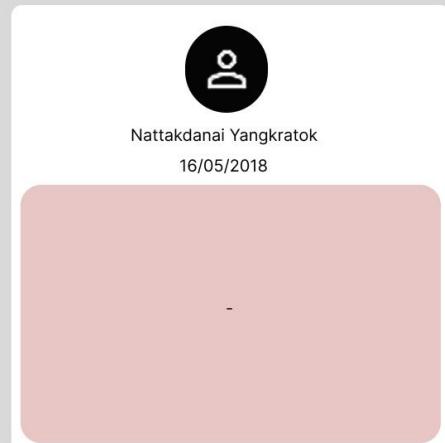
## Edit

ចំណែក\*

គេលេប\*

- /10

រឿង\*



Nattakdanai Yangkratok

16/05/2018



Minimum Spec\*

OS\*

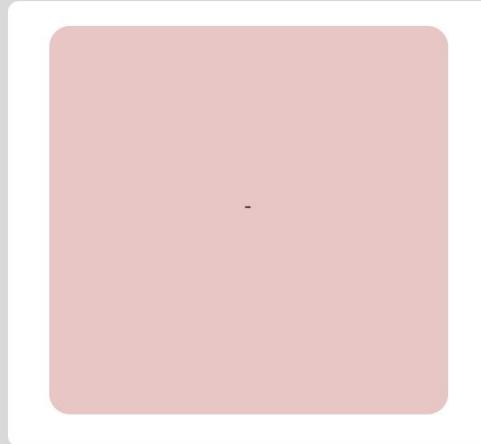
Processor\*

Memory\*

Graphics\*

Storage\*

Reward\*

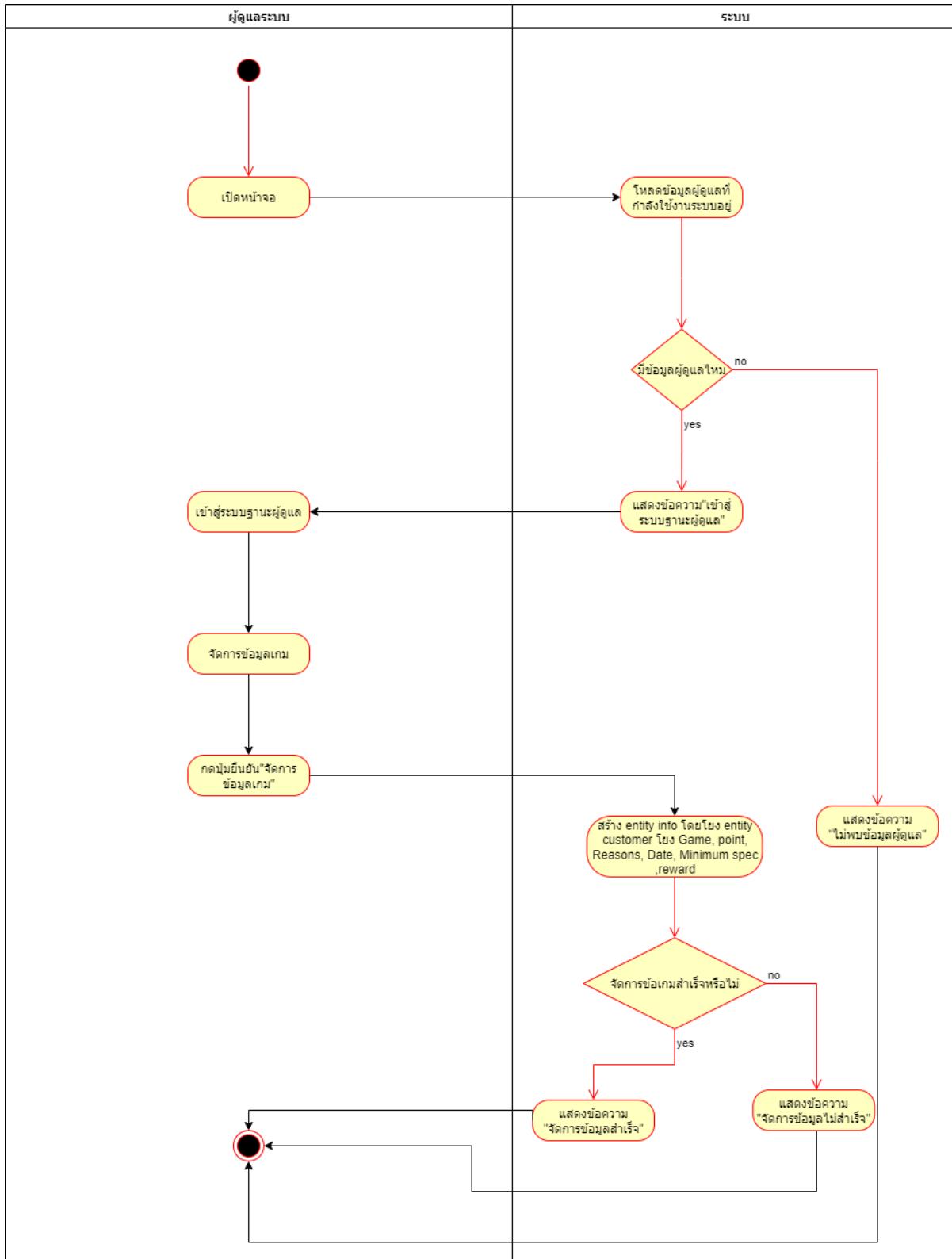


Jack Dorson

16/05/2018

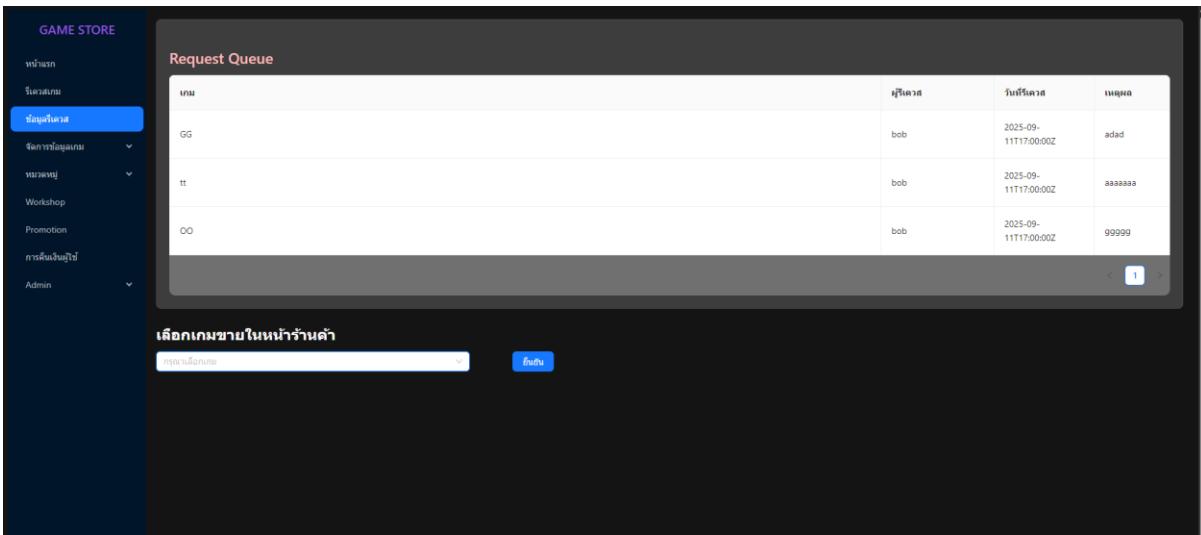
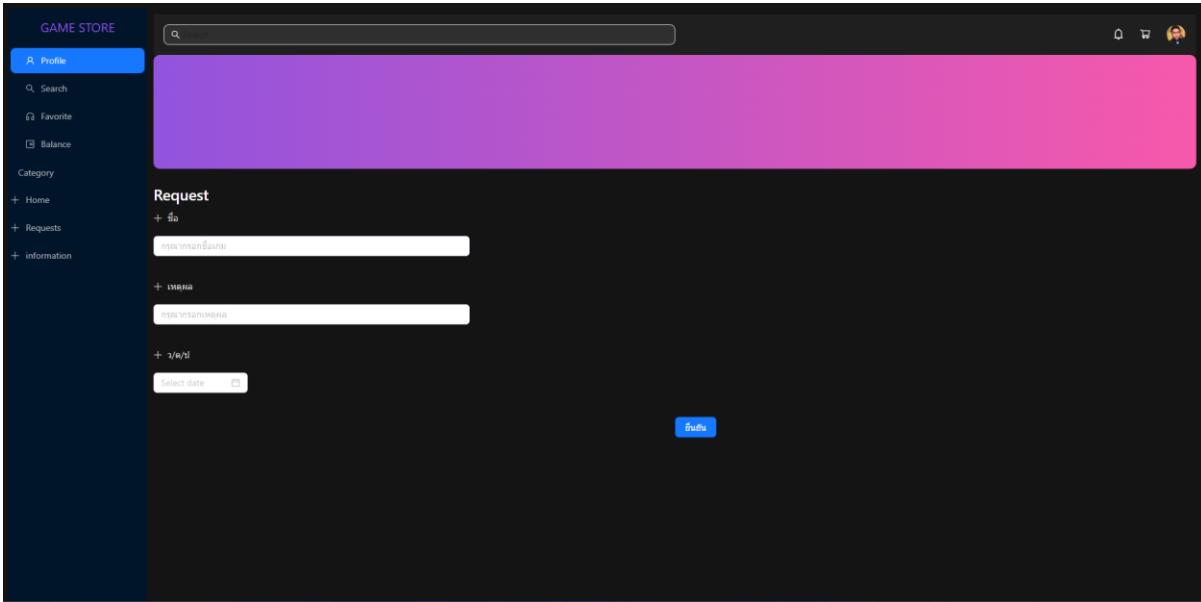
ឲ្យបាយបាន

## Activity Diagram



## Frontend

### ระบบบริเวชส



```

import { Layout, Space, Button, Select } from 'antd';
import Navbar from "../components/Navbar";
import { Typography, Input, DatePicker, Result} from "antd";
import { Col, Row } from 'antd';
import { Link } from 'react-router-dom';
import axios from 'axios';
import type { Game } from './interfaces';
import { useState, useEffect} from 'react';
import { useAuth } from '../context/AuthContext';
import {
  PlusOutlined ,
} from '@ant-design/icons';
const base_url = 'http://localhost:8088'
const { Title } = Typography;

const Request = () => {
  const[game, Setgame] = useState<Game[]>([])
  const[gameid,SetgameID] = useState<number | null>(null);
  const[reason, SetReason]=useState("")
  const [date, setDate] = useState<string | null>(null);
  const { id } = useAuth(); //ค้นใช้งานระบบ

  async function CreateRequest() {
    try {
      const response = await axios.post(`${base_url}/new-request`, {
        reason: reason,
        release_date: date,
        user: id,
        game: gameid,
      });
      console.log("เพิ่มรีเควสสำเร็จ:", response.data)
    } catch(err) {
      console.log("add request error",err)
    }
  }
  async function GetGame() {
    try {
      const response = await axios.get(`${base_url}/game`)
      Setgame(response.data)
      console.log(response.data)
    } catch(err) {
      console.log('get game error',err)
    }
  }
  useEffect(() =>{
    GetGame()
  }, [])
}

const pendingGames = game ? game.filter(g => g.status === "pending") : []

```

```

return(<div style={{background: '#141414', flex: 1 , minHeight: '100vh'}}>{(pendingGames.length != 0) ? (
  <Layout style={{flex: 1}}>
    <Layout style={{ background: '#141414', flex: 1 , minHeight: '100vh'}}>
      <Navbar />
      <div style={{ padding: '10px', flex: 1}}>
        <div style={{ background: 'linear-gradient(90deg, #9254de 0%, #f759ab 100%)', height: 180, borderRadius: 10, marginBottom: 24 }}>
      </div></div>
      <Title level={3} style={{ color: 'white' }}>Request</Title> /* title ต้อง import Typography*/
      <Space><PlusOutlined style={{background: '#141414', color: '#d6d6d6ff'}}><text style={{ color: '#d6d6d6ff' }}>ចូល</text></Space>
      <br />
      <br />
      <Select placeholder="ក្រុណាកើតកែវ" style={{ width: 500}} options={pendingGames.map(c => ({ value: c.ID, label: c.game_name }))} value={gameid ?? undefined} onChange={(val, option) => {console.log("onChange val:", val);console.log("onChange option:", option);SetgameID(val);}}/>
      <br />
      <br />
      <br />
      <Space><PlusOutlined style={{background: '#141414', color: '#d6d6d6ff'}}><text style={{ color: '#d6d6d6ff' }}>ពេតអត់</text></Space>
      <br />
      <br />
      <Input placeholder="ក្រុណាករកពេតអត់" style={{ width: 500}} value={reason} onChange={(e) => SetReason(e.target.value)}/>
      <br />
      <br />
      <br />
      <Space><PlusOutlined style={{background: '#141414', color: '#d6d6d6ff'}}><text style={{ color: '#d6d6d6ff' }}>ឱ្យទៅ</text></Space>
      <br />
      <br />
      <DatePicker value={date} onChange={(value) => setDate(value)}/>
      <br />
      <br />
      <br />
      <br />
      <Row>
        <Col offset={12}><Button type="primary" onClick={() => {CreateRequest()}}>បើនយោង</Button></Col>
      </Row>
    </div>
  </Layout>
)>(<Result style={{ flex: 1, background:'#313131ff', justifyContent: "left", minHeight:'100vh', alignItems: "baseline", minWidth:'180vh'}} status={"404"} title={{<div style={{color:'#ffffffff'}}>"404"</div>}} subTitle={{<div style={{color:'#ffffffff'}}>"Sorry, request does not exist.maybe not have game is pending."</div>}} extra={[<Link to="/home"><Button type="primary">Back Home</Button></Link>]}>
);
};

export default Request;

```

```

import { Card, Table, Typography, Select, Button, message } from "antd";
import { Col, Row, Result } from 'antd';
import type { ColumnsType } from "antd/es/table";
import { Link } from "react-router-dom";
import axios from "axios";
import { useState, useEffect } from "react";
import type { Request } from "../interfaces/Request";
import type { Game } from "./interfaces";
import { useAuth } from "../context/AuthContext";

const base_url = "http://localhost:8088";
const { Title } = Typography;

type Row = {
  id: number;
  game_name: string;
  reason: string;
  user: string;
  date: string; // แสดงเป็นข้อความพอ
};

type useronline = {
  ID: number
  role_id: number
  role: {title: string};
}

const columns: ColumnsType<Row> = [
  {
    title: "ເລີມ",
    dataIndex: "game_name",
  },
  { title: "ຜູ້ໃຊ້ເວົາສ", dataIndex: "user", width: 180, ellipsis: true },
  { title: "ວັນທີໃຊ້ເວົາສ", dataIndex: "date", width: 160 },
  {
    title: "ເຫດຜລ",
    dataIndex: "reason",
    width: 120,
  },
];

export default function Requestinfo() {
  const [Requestinfo, SetRequestinfo] = useState<Request[]>([]);
  const [game, Setgame] = useState<Game[]>([]);
  const [gameid, SetgameID] = useState<number | null>(null);
  const { id } = useAuth(); //ຄົນໃຊ້ຈານຮະບບ
  const [useron, Setuseron] = useState<useronline | null>(null)
}

```

```

async function UpdateGame(id: number, data: { status?: string }) {
  try {
    const response = await axios.put(`${base_url}/update-game/${id}`, data);
    console.log("อัปเดตสำเร็จ:", response.data);
    message.success("อัปเดตสถานะเกมสำเร็จ!");
    return response.data;
  } catch (err) {
    console.error("update error:", err);
    message.error("เกิดข้อผิดพลาดในการอัปเดตเกม");
  }
}

async function GetRequest() {
  try {
    const response = await axios.get(`${base_url}/request`);
    SetRequestinfo(response.data);
    console.log(response.data);
  } catch (err) {
    console.log("get request error", err);
  }
}

async function GetUserbyid(id: number) { //เก็บเป็น object ไม่ใช่ array
  try {
    const response = await axios.get(`${base_url}/users/${id}`);
    console.log("ดึงข้อมูลผู้ใช้สำเร็จ", response.data);
    Setuseron(response.data)
  } catch (err) {
    console.error("get user error:", err);
    message.error("เกิดข้อผิดพลาดในการดึงผู้ใช้");
  }
}

useEffect(() => {
  GetUserbyid(Number(id));
}, [id]);

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

async function GetGame() {
  try {
    const response = await axios.get(`${base_url}/game`)
    Setgame(response.data)
    console.log(response.data)
  } catch(err) {
    console.log('get game error',err)
  }
}

```

```

}

useEffect(() =>{
  GetGame()
}, [])

const pendingGames = game ? game.filter(g => g.status === "pending") : [];
const useronline = useron?role?.title === "Admin"

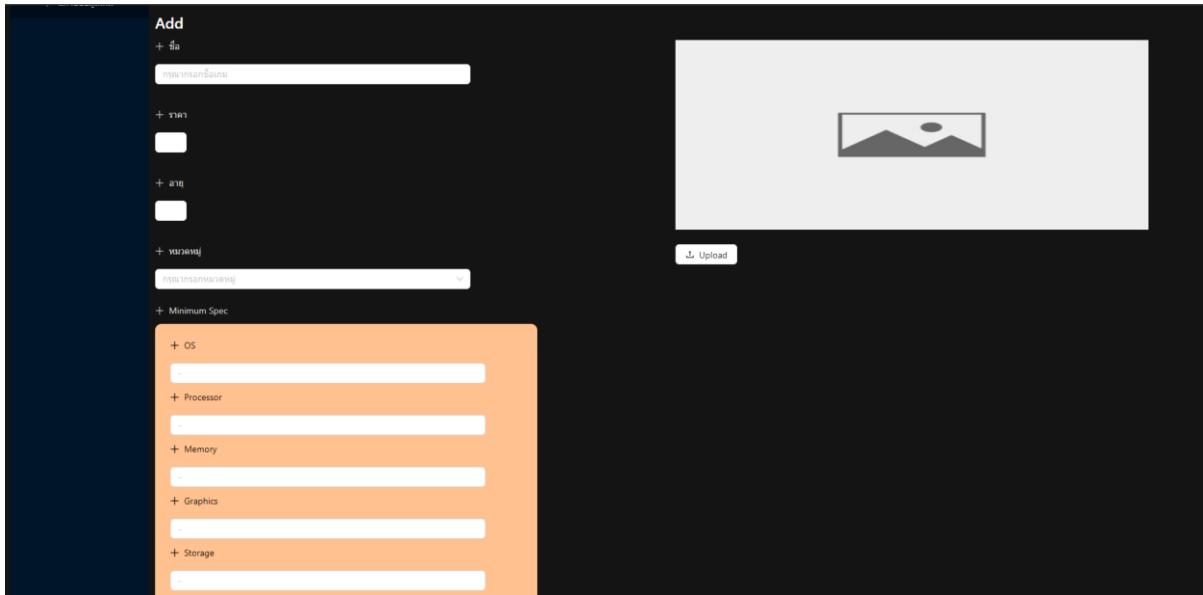
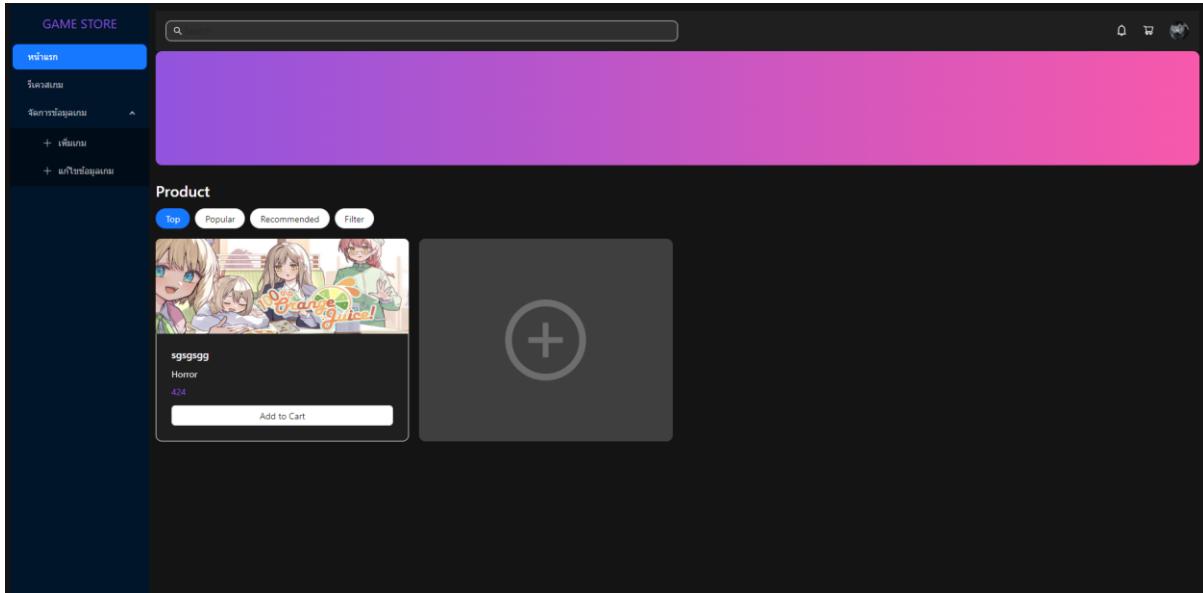
return (<div><div>{useronline ? (
<div style={{ padding: 16, background:'#141414', minHeight:'100vh'}}>
<Card
  bodyStyle={{ padding: 20 }}
  style={{
    border: "1px solid rgba(255,255,255,0.08)",
    background: "#3d3d3dff",
  }}
>
<Title level={3} style={{ marginBottom: 12, color: "#fdbcbcff" }}>
  Request Queue
</Title>
<Table<Row>
  rowKey="id"
  columns={columns}
  dataSource={Requestinfo.map((c) => ({
    id: c.ID,
    game_name: c.game_obj.game_name,
    reason: c.reason,
    user: c.user_obj.username,
    date: c.release_date,
  }))}
  pagination={{ pageSize: 8, showSizeChanger: false }}
  bordered
  style={{ background: "#707070ff", borderRadius: 10 }}
/>
</Card>

<Title level={3} style={{ color: "white" }}>
  เลือกเกมขายในหน้าร้านค้า
</Title>
<Row>
<Select
  placeholder="กรุณาเลือกเกม"
  style={{ width: 500 }}
  options={pendingGames.map((c) => ({
    value: c.ID,
    label: c.game_name,
  }))}
  value={gameid ?? undefined}
  onChange={(val) => SetgameID(val)}
)

```

```
/>
<Col offset={1}>
  <Button
    type="primary"
    onClick={() => {
      if (gameid !== null) {
        UpdateGame(gameid, { status: "approve" });
      } else {
        message.warning("ກຽມນາເລືອກເກມກ່ອນ");
      }
    }}
  >
  ຍືນຍັນ
</Button>
</Col>
</Row>
</div> : (<Result style={{ flex: 1, background:'#313131ff', justifyContent: "left", minHeight:'100vh', alignItems: "baseline", minWidth:'180vh'}} status="error" title={<div style={{color:'#ffffffff'}}>"block user"</div>} subTitle={<div style={{color:'#ffffffff'}}>"Sorry, you dont have admin role."</div>} extra={[<Link to="/home"><Button type="primary">Back Home</Button></Link>]}>)
  );
}
```

## ระบบจัดการข้อมูลเกม



## Page add

```
import { Select, Layout } from 'antd';
import Navbar from "../../components/Navbar";
import { Typography, } from "antd";
import { Col, Row, Space } from 'antd';
import {Input, Button} from 'antd';
import { Upload } from "antd";
import type { UploadFile } from "antd/es/upload/interface";
import {
  PlusOutlined,
  UploadOutlined,
} from '@ant-design/icons';
const { Title } = Typography;
import { useState,useEffect } from 'react';
import axios from 'axios';

const base_url = 'http://localhost:8088'
const Add = () => {
  interface Category {
    ID: number;
    title: string;
  }
  const [categories, setCategories] = useState<Category[]>([]);
  const [categoryId, setCategoryId] = useState<number | null>(null);
  const [gameName, setGameName] = useState("");
  const [price, setPrice] = useState<number | null>(null);
  const [age, setAge] = useState<number | null>(null);
  const [imgurl, setImageurl] = useState("https://staticvecteezy.com/system/resources/thumbnails/004/141/669/small_2x/no-photo-or-blank-image-icon-loading-images-or-missing-image-mark-image-not-available-or-image-coming-soon-sign-simple-nature-silhouette-in-frame-isolated-illustration-vector.jpg");
  const [os, setOs] = useState("");
  const [m_id, setM_id] = useState<number>(1);
  const [processor, setProcessor] = useState("");
  const [memory, setMemory] = useState("");
  const [graphics, setGraphics] = useState("");
  const [storage, setStorage] = useState("");
  /*
  const handleChange = ({ fileList }: { fileList: UploadFile[] }) => {
    if (fileList.length > 0) {
      const file = fileList[0].originFileObj as File | undefined;
      if (file) {
        const url = URL.createObjectURL(file); // แปลงเป็น URL
        setImageurl(url);
      }
    } else {
      setImageurl(""); // ถ้าลบไฟล์ออกก็เปลี่ยน state
    }
  };*/
}
```

```

const handleChange = ({ fileList }: { fileList: UploadFile[] }) => {
  if (fileList.length === 0) {
    setImageurl("");
    return;
  }
  const file = fileList[0].originFileObj as File | undefined;
  if (!file) return;

  const reader = new FileReader();
  reader.onload = () => {
    const dataUrl = reader.result as string; // "data:image/png;base64,..."
    setImageurl(dataUrl); // ✅ เก็บไว้ส่งไป /new-game
  };
  reader.onerror = (e) => console.error("FileReader error:", e);
  reader.readAsDataURL(file);
};

async function GetCategories() {
  try {
    const response = await axios.get(`${base_url}/categories`)
    setcategories(response.data)
    console.log(response.data)
  } catch(err) {
    console.log('get categories error',err)
  }
}
useEffect(() =>{
  GetCategories()
}, []);

async function AddGame() {
  try {
    const response = await axios.post(`${base_url}/new-game`, {
      game_name: gameName,
      base_price: price,
      age_rating: age,
      img_src: imgurl,
      minimum_spec_id: m_id,
      categories_id: categoryId,
    });
    console.log("เพิ่มเกมสำเร็จ:", response.data)
    setM_id(m_id+1)
  } catch(err) {
    console.log("add game error",err)
  }
}

async function AddMinimumSpec() {
  try {
    const response = await axios.post(`${base_url}/new-minimumspec`, {

```

```

        os: os,
        processor: processor,
        memory: memory,
        graphics:graphics,
        storage:storage,
    });
    console.log("เพิ่มเกมสเปค:", response.data)
} catch(err) {
    console.log("add spec error",err)
}
}

return(
<Layout>
<Layout style={{ background: '#141414', flex: 1 , minHeight: '100vh'}}>
<div style={{ padding: '10px' }}>
<Navbar />
<div style={{ background: 'linear-gradient(90deg, #9254de 0%, #f759ab 100%)', height: 180, borderRadius: 10, marginBottom: 24 }}></div>
<Title level={3} style={{ color: 'white' }}>Add</Title> /* title ต้อง import Typography*/
<Row style={{marginBottom: 24 }}>
<Col span={12}>
<Space><PlusOutlined style={{background: '#141414', color: '#d6d6d6ff'}}/><text style={{ color: '#d6d6d6ff' }}>ชื่อ
</text></Space>
<br />
<br />
<Input placeholder="กรุณากรอกชื่อเกม" style={{ width: 500 }} value={gameName} onChange={(e) =>
setGameName(e.target.value)}/>
<br />
<br />
<br />
<Space><PlusOutlined style={{background: '#141414', color: '#d6d6d6ff'}}/><text style={{ color: '#d6d6d6ff' }}>ราคา
</text></Space>
<br />
<br />
<Input style={{ width: 50 }} onChange={(e) => setPrice(Number(e.target.value))}/>
<br />
<br />
<br />
<Space><PlusOutlined style={{background: '#141414', color: '#d6d6d6ff'}}/><text style={{ color: '#d6d6d6ff' }}>อายุ
</text></Space>
<br />
<br />
<Input style={{ width: 50 }} onChange={(e) => setAge(Number(e.target.value))}/>
<br />
<br />
<br />
<Space><PlusOutlined style={{background: '#141414', color: '#d6d6d6ff'}}/><text style={{ color: '#d6d6d6ff' }}>หมายเหตุ
</text></Space>
<br />

```

```

<br />
<Select placeholder="ກູ້ນາກອກໝາດໜີ" style={{ width: 500 }} options={categories.map(c => ({ value: c.ID, label: c.title }))}>
value={categoryId ?? undefined} onChange={(val, option) => {console.log("onChange val:", val);console.log("onChange option:", option);setCategoryId(val);}}/>
</Col>
<Col span={12}>
<img src={imgurl} width={"750px"} height={"300px"} />
<br />
<br />
<Upload onChange={(handleChange)} maxCount={1} beforeUpload={() => false} accept="image/*/*">ເນື່ອປົວໂລດຈິງ ກັບເກີບໃນ
state*</><Button icon={<UploadOutlined />}>Upload</Button></Upload>
</Col>
</Row>
<Row>
<Col span={12}><Space><PlusOutlined style={{background: '#141414', color: '#d6d6d6ff'}}/><text style={{ color: '#d6d6d6ff' }}>Minimum Spec</text></Space></Col>
</Row>
<Row style={{marginTop: 12}}>
<Col>
<Layout style={{ background: '#ffc18fff', height: 500, borderRadius: 10, marginBottom: 24}} >
<Row style={{marginLeft: 24, marginTop: 24}}>
<Col span={12}><Space><PlusOutlined style={{background: '#ffc18fff', color: '#000000ff'}}/><text style={{ color: '#000000ff' }}>OS</text></Space></Col>
<br />
<br />
<Input placeholder="-" style={{ width: 500}} value={os} onChange={(e) => setOs(e.target.value)}/>
</Row>
<Row style={{marginLeft: 24, marginTop: 12}}>
<Col span={12}><Space><PlusOutlined style={{background: '#ffc18fff', color: '#000000ff'}}/><text style={{ color: '#000000ff' }}>Processor</text></Space></Col>
<br />
<br />
<Input placeholder="-" style={{ width: 500}} value={processor} onChange={(e) => setProcessor(e.target.value)}/>
</Row>
<Row style={{marginLeft: 24, marginTop: 12}}>
<Col span={12}><Space><PlusOutlined style={{background: '#ffc18fff', color: '#000000ff'}}/><text style={{ color: '#000000ff' }}>Memory</text></Space></Col>
<br />
<br />
<Input placeholder="-" style={{ width: 500}} value={memory} onChange={(e) => setMemory(e.target.value)}/>
</Row>
<Row style={{marginLeft: 24, marginTop: 12}}>
<Col span={12}><Space><PlusOutlined style={{background: '#ffc18fff', color: '#000000ff'}}/><text style={{ color: '#000000ff' }}>Graphics</text></Space></Col>
<br />
<br />
<Input placeholder="-" style={{ width: 500}} value={graphics} onChange={(e) => setGraphics(e.target.value)}/>
</Row>
<Row style={{marginLeft: 24, marginBottom: 24, marginTop: 12}}>

```

```

<Col span={12}><Space><PlusOutlined style={{background: '#ffc18fff', color: '#000000ff'}}/><text style={{ color: '#000000ff' }}>Storage</text></Space></Col>
      <br />
      <br />
      <Input placeholder="-" style={{ width: 500}} value={storage} onChange={(e) => setStorage(e.target.value)}/>
    </Row>
  </Layout>

</Col>
</Row>
<Row style={{marginTop: 12}}>
  <Col offset={23}><Button type="primary" onClick={() => {
    AddMinimumSpec();
    AddGame();
  }}>ບັນທຶນ</Button></Col>
</Row>
</div>
</Layout>
</Layout>
);

}

export default Add;

```

## Page Home

```

import { Row, Col } from 'antd';
import AddProductCard from './AddProductCard';
import { Link } from 'react-router-dom';
import { Card, Button } from 'antd';
import { useState, useEffect } from 'react';
import axios from 'axios';

const base_url = 'http://localhost:8088'

const ProductGrid = () => {
  interface Game {
    ID: number;
    game_name: string;
    key_id: number;
    categories: {ID: number, title: string};
  }

```

```

        release_date: string;
        base_price: number;
        img_src: string;
        age_rating: number;
        status: string;
        minimum_spec_id: number;
    }

    // แปลง img_src ให้เป็น absolute URL เสมอ
    const resolveImgUrl = (src?: string) => {
        if (!src) return "";
        if (src.startsWith("blob:")) return "";
        if (src.startsWith("data:image/")) return src;
        // ถ้าเป็น blob: เดิม จะมักใช้ไม้ได้ในหน้าอื่น — ควรแก้ผึ่ง backend ให้ส่ง URL ดาวร
        if (src.startsWith("http://") || src.startsWith("https://") || src.startsWith("blob:")) {
            return src;
        }
        // ถ้า backend ส่งเป็น "uploads/xxx.jpg" หรือ "/uploads/xxx.jpg"
        const clean = src.startsWith("/") ? src.slice(1) : src;
        return `${base_url}/${clean}`;
    };

    const[game, Setgame] = useState<Game>([])
    async function GetGame() {
        try {
            const response = await axios.get(`${base_url}/game`)
            Setgame(response.data)
            console.log(response.data)
        } catch(err) {
            console.log('get game error',err)
        }
    }
    useEffect(() =>{
        GetGame()
    }, [])
}

return (
    <Row gutter={[16, 16]}>
        {game.map((c) => (
            <Col xs={24} sm={12} md={8} lg={6} >
                <Card
                    style={{ background: '#1f1f1f', color: 'white', borderRadius: 10 }}
                    cover={<img src={resolveImgUrl(c.img_src)} style={{height: 150}}/>}
                >
                    <Card.Meta title={<div style={{color: '#ffffff'}>{c.game_name}</div>} description={<div style={{color: '#ffffff'}>{c.categories.title}</div>}}
                    <div style={{ marginTop: 10, color: '#9254de' }}>{c.base_price}</div>
                    <Button block style={{ marginTop: 10 }}>Add to Cart</Button>
                </Card>
        ))
    )
)

```

```
</Col>
})
}

<Col xs={24} sm={12} md={8} lg={6}>
  <Link to={'/information/Add'}>
    <AddProductCard />
  </Link>
</Col>
</Row>
);

};

export default ProductGrid;
```

## Backend

### Entity user

```
package entity

import (
    "time"

    "gorm.io/gorm"
)

type User struct {
    gorm.Model
    Username string
    Password string
    Email    string
    First_name string
    Last_name string
    Birthday time.Time
    RoleRefer uint
    Requests []Request `gorm:"foreignKey:UserRefer"`
}
```

### Entity request

```
package entity

import (
    "time"

    "gorm.io/gorm"
)

type Request struct {
    gorm.Model
    Reason   string
    Date     time.Time
    UserRefer uint `gorm:"not null;index:idx_user_game,unique"`
    GameRefer uint `gorm:"not null;index:idx_user_game,unique"`
}
```

## Entity games

```
package entity

import (
    "time"

    "gorm.io/gorm"
)

type Game struct {
    gorm.Model
    GameName     string    `json:"game_name" gorm:"type:varchar(512)"`  

    KeyGameID    uint      `json:"key_id"``  

    CategoriesID int       `json:"categories_id"``  

    Categories   Categories `json:"categories" gorm:"foreignkey:CategoriesID"``  

    Date         time.Time `json:"release_date" gorm:"autoCreateTime"``  

    Baseprice    int       `json:"base_price"``  

    ImgSrc       string    `json:"img_src" gorm:"type:varchar(512)"``  

    AgeRating    int       `json:"age_rating"``  

    Status       string    `json:"status" gorm:"type:varchar(20);default:'pending';not null;index"``  

    Minimum_specID uint     `json:"minimum_spec_id"``  

    Requests     []Request `gorm:"foreignKey:GameRefer"``  

    Market       Market    `json:"market"``  

}

// Hook function ໄວ້ຄັ້ງສ້າງເກມເສົ່າງແລ້ວ keygame ຈະເຈນເອງ
func (g *Game) AfterCreate(tx *gorm.DB) (err error) {
    kg := KeyGame{}
    if err = tx.Create(&kg).Error; err != nil {
        return err
    }
    return tx.Model(g).Update("key_game_id", kg.ID).Error
}
```

## Entity keygame

```
package entity

import (
    "gorm.io/gorm"
)

type KeyGame struct {
    gorm.Model
    Key string `json:"key" gorm:"uniqueIndex; type:text; default:(lower(hex(randomblob(16))))"`
    Game Game `json:"game"`
}
```

## Entity categories

```
package entity

import (
    "gorm.io/gorm"
)

type Categories struct {
    gorm.Model
    Title string `json:"title" gorm:"unique"`
}
```

## Entity role

```
package entity

import (
    "gorm.io/gorm"
)

type Role struct {
    gorm.Model
    Title     string `json:"title" gorm:"unique"`
    Description string `json:"description" gorm:"unique"`
    Users     []User `gorm:"foreignKey:RoleRefer"`
}
```

```
}
```

## Entity market

```
package entity

import (
    "gorm.io/gorm"
)

type Market struct {
    gorm.Model
    GameID uint
}
```

## Entity minimum\_spec

```
package entity

import (
    "gorm.io/gorm"
)

type MinimumSpec struct {
    gorm.Model
    OS      string `json:"os" gorm:"type:varchar(20)"`  

    Processor string `json:"processor" gorm:"type:varchar(20)"`  

    Memory   string `json:"memory" gorm:"type:varchar(20)"`  

    Graphics  string `json:"graphics" gorm:"type:varchar(20)"`  

    Storage   string `json:"storage" gorm:"type:varchar(20)"`  

    Game     Game
}
```

## Controller

### Entity categories

```
package controller

import (
    "net/http"

    "G13.com/backend/configs"
    "G13.com/backend/entity"
    "github.com/gin-gonic/gin"
)

func FindCategories(c *gin.Context) {
    var categories []entity.Categories
    if err := configs.DB().Raw("SELECT * FROM categories").Find(&categories).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, categories)
}
```

### Entity game

```
package controller

import (
    "net/http"

    "G13.com/backend/configs"
    "G13.com/backend/entity"
    "github.com/gin-gonic/gin"
)

func FindGames(c *gin.Context) {
    var games []entity.Game
    if err := configs.DB().
        Preload("Categories").
        Find(&games).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
```

```

c.JSON(http.StatusOK, games)
}

func CreateGame(c *gin.Context) {
    var input struct {
        GameName      string `json:"game_name" binding:"required"`
        BasePrice     int    `json:"base_price" binding:"required"`
        AgeRating    int    `json:"age_rating"`
        ImgSrc       string `json:"img_src"`
        Minimum_spec_id int   `json:"minimum_spec_id" binding:"required"`
        CategoriesID int   `json:"categories_id" binding:"required"`
    }
    if err := c.ShouldBindJSON(&input); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    games := entity.Game{
        GameName:      input.GameName,
        Baseprice:     input.BasePrice,
        AgeRating:    input.AgeRating,
        ImgSrc:       input.ImgSrc,
        Minimum_specID: uint(input.Minimum_spec_id),
        CategoriesID: input.CategoriesID,
    }
    if err := configs.DB().Create(&games).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, games)
}

```

## Entity keygame

```

package controller

import (
    "net/http"

    "G13.com/backend/configs"
    "G13.com/backend/entity"
    "github.com/gin-gonic/gin"
)

```

```

func CreateKeyGame(c *gin.Context) {
    var keygame entity.KeyGame
    if err := c.ShouldBindJSON(&keygame); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    // สร้าง record ในมล DB
    if err := configs.DB().Create(&keygame).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    // ส่งข้อมูล category ที่เพิ่งสร้างกลับไปให้ frontend
    c.JSON(http.StatusOK, keygame)
}

func FindKeyGame(c *gin.Context) {
    var keygame []entity.KeyGame
    if err := configs.DB().Preload("Game").Find(&keygame).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, keygame)
}

func DeleteKeyGameById(c *gin.Context) {
    id := c.Param("id")
    if tx := configs.DB().Exec("DELETE FROM KeyGame WHERE id = ?", id); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "id not found"})
        return
    }

    c.JSON(http.StatusOK, gin.H{"message": "deleted succesful"})
}

```

## Entity minimumspec

```

package controller

import (
    "net/http"

    "G13.com/backend/configs"
    "G13.com/backend/entity"
)

```

```
"github.com/gin-gonic/gin"
)

func FindMinimumSpec(c *gin.Context) {
    var minimum_specs []entity.MinimumSpec
    if err := configs.DB().Preload("Game").Find(&minimum_specs).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, minimum_specs)
}

func CreateMinimumSpec(c *gin.Context) {
    var minimum_specs entity.MinimumSpec
    if err := c.ShouldBindJSON(&minimum_specs); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

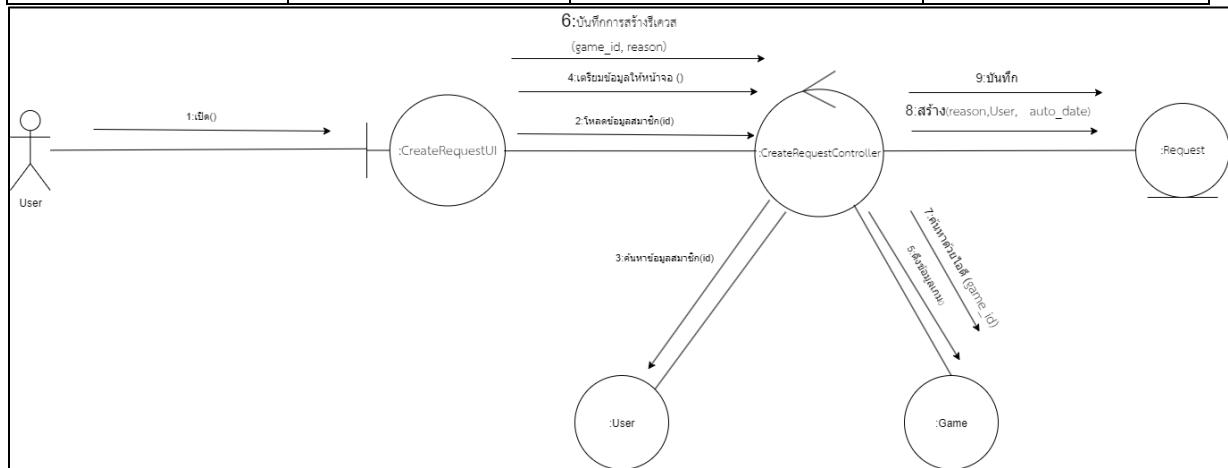
    if err := configs.DB().Create(&minimum_specs).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, minimum_specs)
}
```

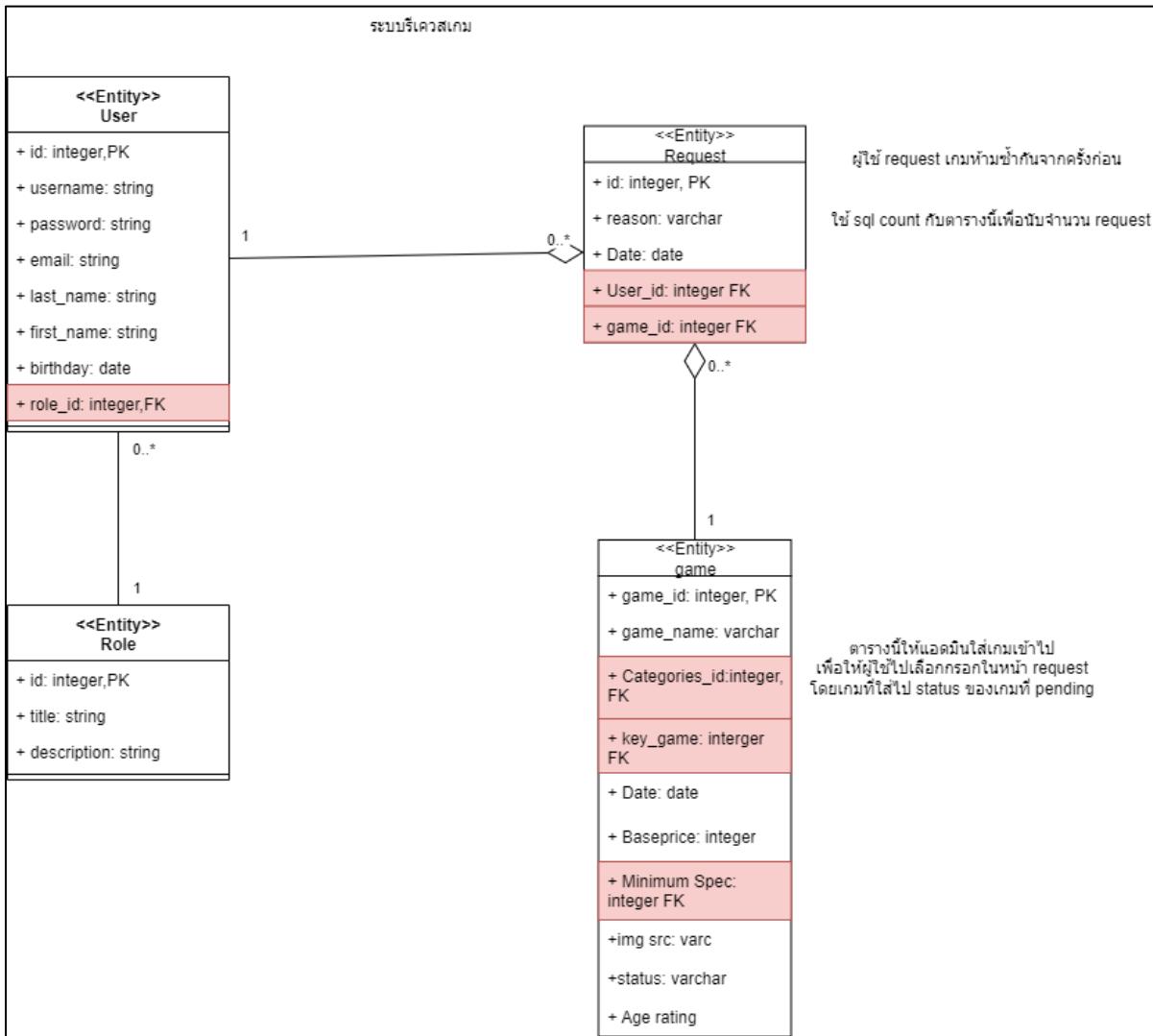
## Communication diagram(ระบบปรีเคสเกม)

Activity	เป็นคำสั่งสำหรับระบบได้หรือไม่	ถ้าเป็น ,วัตถุที่รับหน้าที่ทำงานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click "เปิดหน้าจอ"	เป็นคำสั่ง สั่งงานเพื่อให้หน้า UI เปิด	:CreateRequestUI	เปิด ()
โหลดข้อมูลผู้ใช้ที่กำลังใช้งานระบบอยู่	เป็นคำสั่ง เกิดการสั่งงานให้โหลดข้อมูลสมาชิกที่ login อยู่	:CreateRequestController U:User	โหลดข้อมูลสมาชิก (id) ค้นหาด้วยไอดี (id)
โหลดข้อมูลรายการเกม	เป็นคำสั่ง เกิดการสั่งงานให้โหลดข้อมูลเกมที่มีสถานะ pending	:CreateRequestController :game	เตรียมข้อมูลให้หน้าจอ () ดึงข้อมูลเกม()
แสดงหน้าจอสำหรับสร้าง "Request"	ไม่เป็นคำสั่ง	-	-
กดเลือกเกม (ได้game_id)	ไม่เป็นคำสั่ง	-	-
กรอกเหตุผล (ได้reason)	ไม่เป็นคำสั่ง	-	-
กดปุ่ม "Request"	เป็นคำสั่ง ระบบทำการจัดเก็บข้อมูล Request	:CreateRequestController	บันทึกการสร้างรีเควส (game_id, reason)
ค้นหา entity game ด้วย game_id ของ game ที่รับเข้ามา	เป็นคำสั่ง	:game	ค้นหาด้วยไอดี (game_id)
สร้างข้อมูล entity Request โดยโยง entity User เช็ต ค่า reason	เป็นคำสั่ง	:Request	สร้าง(reason,User, auto_date)
บันทึก entity Request	เป็นคำสั่ง	:Request	บันทึก

แสดงข้อความ "สร้างรีเควสต์สำเร็จ"	เป็นคำสั่ง	-	-
-----------------------------------	------------	---	---



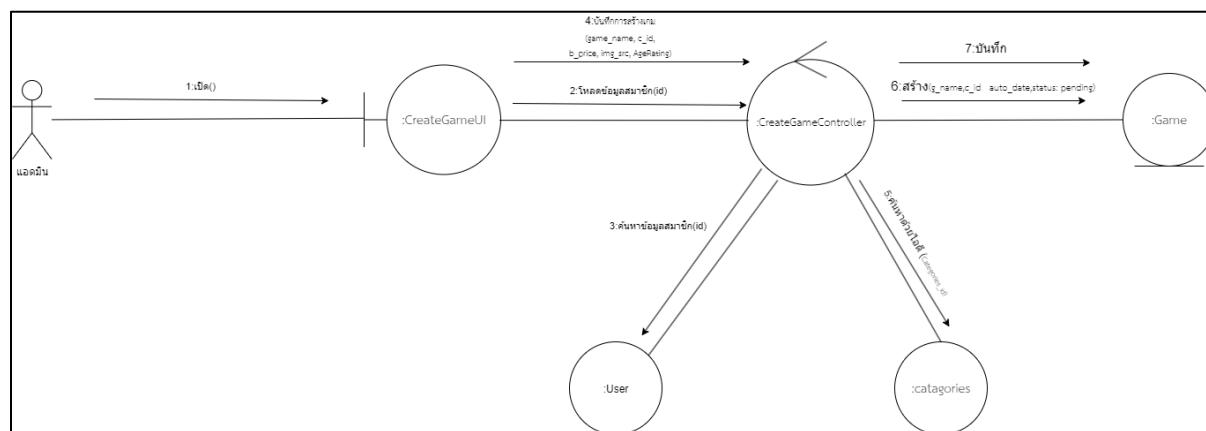
## Class Diagram at Design Level(ระบบบรีเคสเกม)



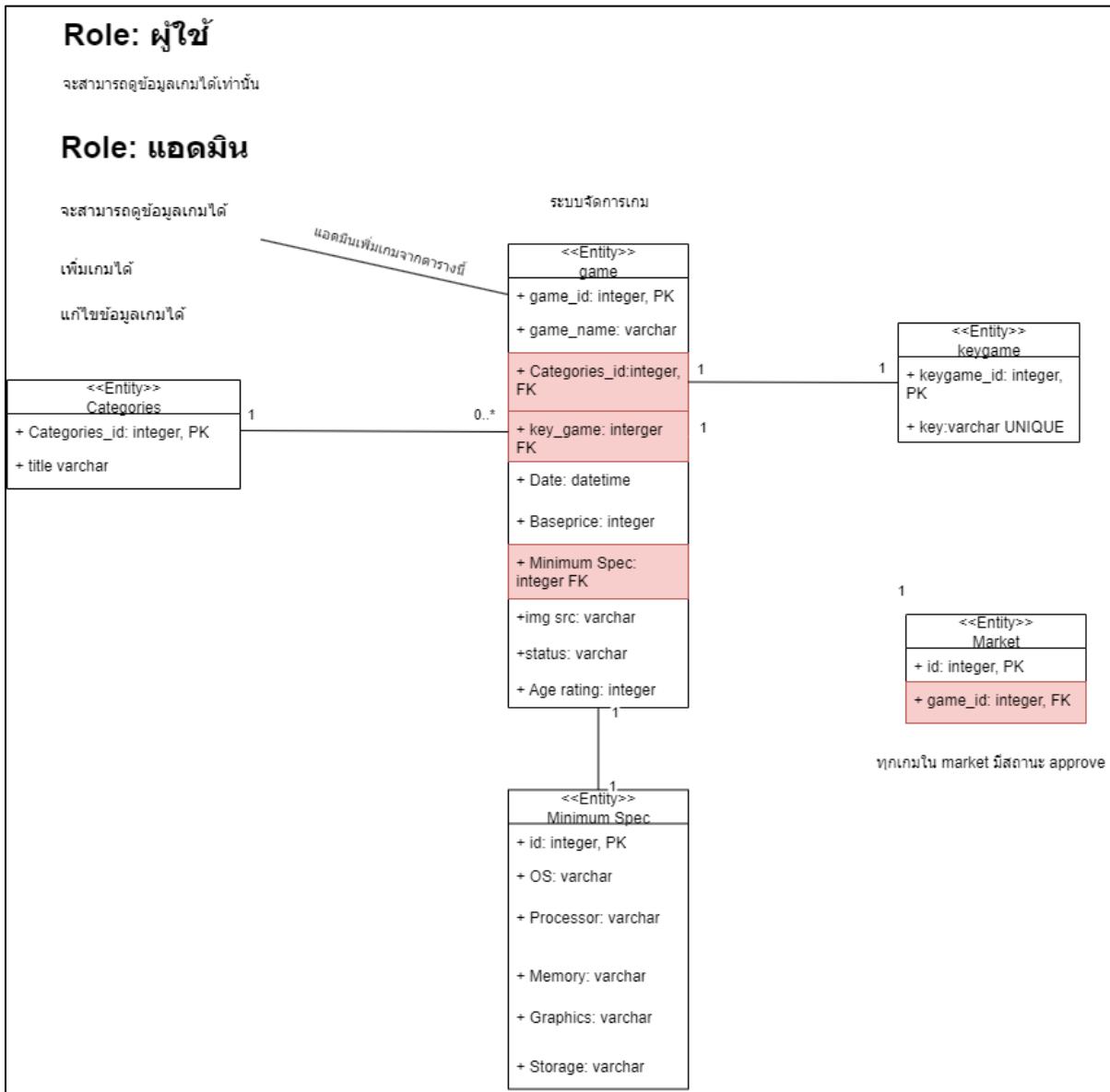
Communication diagram(ระบบจัดการเกม) (สร้างเกม)

Activity	เป็นคำสั่งสำหรับระบบได้หรือไม่	ถ้าเป็น ,วัตถุที่รับหน้าที่ทำงานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click "เปิดหน้าจอ"	เป็นคำสั่ง สั่งงานเพื่อให้หน้า UI เปิด	:CreateGameUI	เปิด ()
โหลดข้อมูลผู้ใช้ที่กำลังใช้งานระบบอยู่	เป็นคำสั่ง เกิดการสั่งงานให้โหลดข้อมูลสมาชิกที่ login อยู่	:CreateGameController	โหลดข้อมูลสมาชิก (id)
		U:User	ค้นหาด้วยไอดี (id)
แสดงหน้าจอสำหรับสร้าง "game"	ไม่เป็นคำสั่ง	-	-
กรอกชื่อเกม (ได้ game_name)	ไม่เป็นคำสั่ง	-	-
เลือกหมวดหมู่ (ได้ c_id)	ไม่เป็นคำสั่ง	-	-
กรอกราคา (ได้ b_price)	ไม่เป็นคำสั่ง	-	-
กรอกภาพเกม (ได้ img_src)	ไม่เป็นคำสั่ง	-	-
กรอกอายุที่แนะนำ (ได้ AgeRating)	ไม่เป็นคำสั่ง	-	-
กดปุ่ม "สร้างเกม"	เป็นคำสั่งระบบทำการจัดเก็บข้อมูล Game	:CreateGameController	บันทึกการสร้างเกม (game_name, c_id, b_price, img_src, AgeRating)
ค้นหา entity catagories ด้วย Categories_id ของ catagories ที่รับเข้ามา	เป็นคำสั่ง	:catagories	ค้นหาด้วยไอดี (Categories_id)

สร้างข้อมูล entity Game โดยโยงโดยโยง entity catagories เช็ตค่า game_name, b_price, img_src , AgeRating	เป็นคำสั่ง	:Game	สร้าง( g_name,c_id auto_date,status: pending)
บันทึก entity Game	เป็นคำสั่งโดยเกมมีสถานะ pending	:Game	บันทึก
แสดงข้อความ "สร้าง เกมสำเร็จ"	เป็นคำสั่ง	-	-



## Class Diagram at Design Level(ระบบจัดการเกม) (สร้างเกม)



B6639273 นายปุณณพัฒน์ เกษหอม

## ระบบหลัก ระบบร้านขายเกมออนไลน์

### ระบบย่อย ระบบจัดการสิทธิผู้ใช้งาน

ระบบร้านขายเกมออนไลน์ **ผู้ใช้งาน (User)** จะต้องลงทะเบียนก่อนจึงจะเข้าใช้งานได้ เมื่อลงทะเบียนเสร็จก็จะต้องทำการ Log in เข้ามาใช้งาน เพื่อซื้อเกมต่างๆ และเมื่อเลือกเกมที่อยากซื้อได้แล้วก็จะต้องทำการชำระเงิน เมื่อชำระเงินสำเร็จทางเราก็จะส่งคีย์เกมให้ลูกค้า เมื่อลูกค้าเล่นเกมแล้วทางเราก็อย่างให้มารีวิวและให้คะแนนเกมโดยมีคะแนนตั้งแต่ 1-5 คะแนน เพื่อเชิญประසบการณ์ให้ลูกค้าคนอื่นๆ ได้มาพิจารณาดูว่าเกมนั้นเป็นอย่างไรบ้าง โดยในระบบของเราจะมีผู้ใช้งานหลายประเภท เช่น ลูกค้า พนักงาน ผู้ดูแลระบบ(Admin) ก็เลยจำเป็นต้องมีระบบจัดการสิทธิผู้ใช้งานเพื่อแยกสิทธิการใช้งานและจัดการสิทธิการใช้งานของผู้ใช้แต่ละประเภท โดยแอดมินจะสามารถ **create, update Role** ของผู้ใช้ได้ และก็สามารถ **delete Role** ได้ โดย Role ต่างๆ จะถูกกำหนดสิทธิให้ว่าสามารถทำอะไรได้ หรือเข้าถึงอะไรได้บ้าง

### User Story ระบบจัดการสิทธิผู้ใช้งาน

ในบทบาทของ (As a) **ผู้ดูแลระบบ (Admin)**

ฉันต้องการ (I want to) มอบหมายสิทธิการเข้าถึงต่างๆ ให้ผู้ใช้

เพื่อ (So that) ให้ผู้ใช้แต่ละคนสามารถเข้าถึงเฉพาะสิ่งที่เขาได้รับอนุญาตเท่านั้น

## Output บนหน้าจอ

- แออดมินกดเพิ่ม Role -> ตั้งชื่อ Role และกำหนดสิทธิให้ Role นั้นๆ -> แออดมินกดเลือกผู้ใช้ -> มอบหมาย Role ให้ผู้ใช้ จากนั้นระบบจะบันทึกข้อมูลสิทธิการเข้าถึงของผู้ใช้ และระบบจะแสดง indicator บางอย่างที่แสดงให้เห็นว่าระบบได้บันทึกข้อมูลเรียบร้อยแล้ว

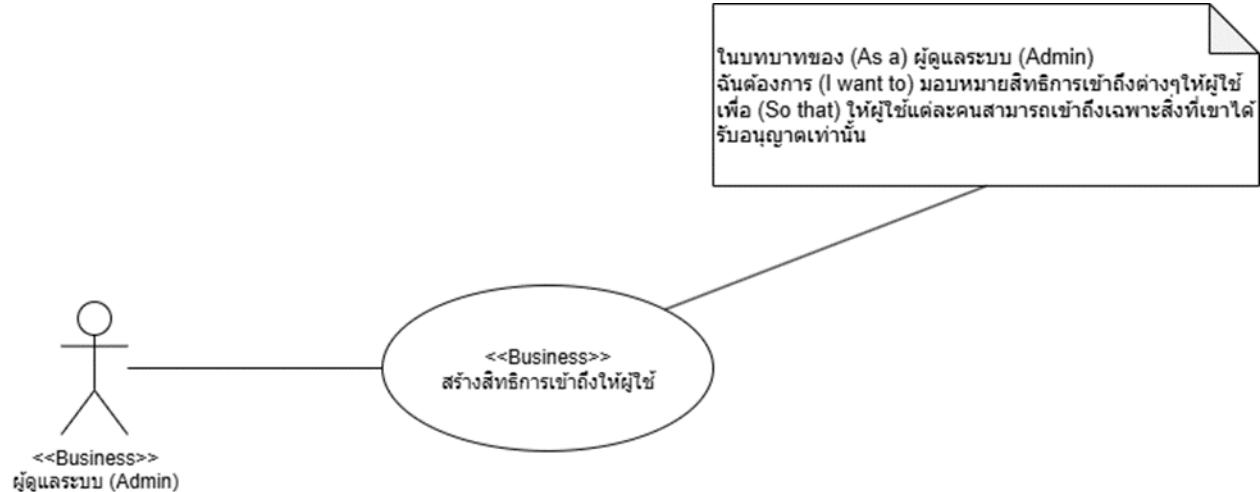
## Output ของข้อมูล

- ระบบสร้าง Role ใหม่ พร้อมบันทึกสิทธิที่ Role นั้นมี -> ระบบจะเรียกข้อมูลผู้ใช้มาเพื่อให้แออดมิน กำหนด Role ให้ผู้ใช้ -> ระบบจะบันทึกข้อมูลและอัพเดทสิทธิของผู้ใช้ในฐานข้อมูล

คำนามที่อาจจะกล้ายมาเป็นตารางในฐานข้อมูลของระบบ

คำนาม	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
ผู้ใช้งาน (User)	เกี่ยวข้องโดยตรง เนื่องจากผู้ใช้งานเป็นคนได้รับ Role และสิทธิ์ต่างๆ
เกม	ไม่เกี่ยวข้องโดยตรง
การชำระเงิน	ไม่เกี่ยวข้องโดยตรง
คีย์เกม	ไม่เกี่ยวข้องโดยตรง
คะแนน	ไม่เกี่ยวข้องโดยตรง
Role	เกี่ยวข้องโดยตรง เนื่องจากใช้รวมชุดสิทธิ์ต่างๆ เพื่อจะได้กำหนดให้ผู้ใช้งานได้ง่ายขึ้น
สิทธิ	เกี่ยวข้องโดยตรง เนื่องจากเป็นสิ่งที่กำหนดความสามารถของผู้ใช้งาน

## Business Use Case Diagram (แบบเดี่ยว)



### Checklist: Business Use Case Diagram

Business Actor มี <<Business>> กำกับ

Business Actor เป็นชื่อบทบาทเท่านั้น ไม่ใช่ชื่อคน ไม่เป็นชื่อบุคคล

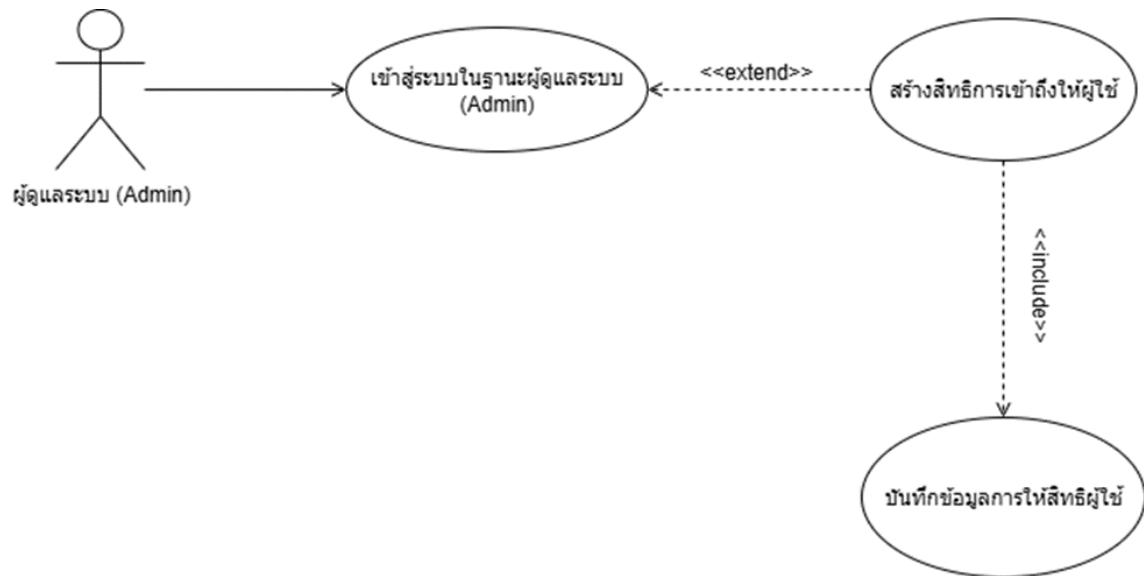
Business Use Case มี <<Business>> กำกับ

Business Use Case ขึ้นต้นด้วยคำที่แสดงพฤติกรรม “คำกริยา”

พฤติกรรมของ Business Use Case ต้องมาจากการ User Story

มี Note ที่แสดง User Story อยู่ใน Diagram

## System use Case Diagram (แบบเดี่ยว)



### ขั้นตอนการแปลง Business Use Case Diagram ให้เป็น System Use Case Diagram

#### พิจารณาประเด็นที่ 1

Business Actor "ผู้ดูแลระบบ (Admin)" สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์ที่ระบบงานได้โดยตรงหรือไม่

ตอบ ได้ ผู้ดูแลระบบ สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์

เพราะฉะนั้น ผู้ดูแลระบบ (Admin) จะถูกจัดเป็น System Actor ได้

## พิจารณาประเด็นที่ 2

Business Use Case “สร้างสิทธิการเข้าถึงให้ผู้ใช้” สามารถถูกถ่ายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบ

คอมพิวเตอร์ได้หรือไม่ต้องมีขั้นตอนทาง Security มาเกี่ยวหรือไม่

ตอบ Business Use Case “สร้างสิทธิการเข้าถึงให้ผู้ใช้” สามารถถูกถ่ายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบ

คอมพิวเตอร์ได้และต้องการขั้นตอนทาง Security (มี Privacy มาเกี่ยวข้อง)

เพราะฉะนั้น Business Use Case “สร้างสิทธิการเข้าถึงให้ผู้ใช้” จะถูกถ่ายเป็น System Use Case มากกว่า 1

Use Case

จะประกอบไปด้วย

- 1 System Use Case สำหรับ “เข้าระบบในฐานะผู้ดูแลระบบ (Admin)”
- 2 System Use Case สำหรับ “สร้าง Playlist” แต่กิจกรรมของระบบ จะยังไม่ครบตาม Requirements ที่ต้องการ
- 3 System Use Case สำหรับ “บันทึกข้อมูลการให้สิทธิผู้ใช้”
  - System Use Case สำหรับ “เข้าระบบในฐานะผู้ดูแลระบบ (Admin)”
  - System Use Case สำหรับ “สร้างสิทธิการเข้าถึงให้ผู้ใช้”
  - System Use Case สำหรับ “บันทึกข้อมูลการให้สิทธิผู้ใช้”

### พิจารณาประเด็นที่ 3

ถ้าสมาชิก “เข้าระบบในฐานะผู้ดูแลระบบ (Admin)” มาแล้วจำเป็นที่ต้อง “สร้างสิทธิการเข้าถึงให้ผู้ใช้” ทุกครั้งหรือไม่

ตอบ ไม่

แปลว่า System Use Case “สร้างสิทธิการเข้าถึงให้ผู้ใช้” เป็นกิจกรรมทางเลือกหลังจากทำ System Use Case “เข้าระบบในฐานะผู้ดูแลระบบ (Admin)”

### พิจารณาประเด็นที่ 4

ถ้าสมาชิก “เข้าระบบในฐานะผู้ดูแลระบบ” มาแล้วจำเป็นต้อง “บันทึกข้อมูลการให้สิทธิผู้ใช้” ทุกครั้งหรือไม่

ตอบ ไม่ และเป็น Use Case ที่ไม่เกี่ยวข้องกัน

### พิจารณาประเด็นที่ 5

ถ้าสมาชิก “สร้างสิทธิการเข้าถึงให้ผู้ใช้” แล้ว

จำเป็นต้อง “บันทึกข้อมูลการให้สิทธิผู้ใช้” ทุกครั้งหรือไม่

ตอบ ใช่ จำเป็น

แปลว่า System Use Case “สร้างสิทธิการเข้าถึงให้ผู้ใช้” จะต้องรวมขั้นตอนของ System Use Case

“บันทึกข้อมูลการให้สิทธิผู้ใช้” ไว้ด้วยเสมอ

### Checklist: System Use Case Diagram

8. System Actor และ System Use Case ต้องไม่มีอะไรรากกับ
9. เส้นโยงจาก System Actor ไปหา System Use Case จะต้องเป็นเส้นทิบ หัวลูกศรปลายเปิดซึ่งไปหา System Use Case
10. System Actor ต้องเป็นบทบาทของคนที่ใช้ระบบจริง ๆ
11. System Use Case ต้องเป็นคำที่แสดงพฤติกรรม
12. ถ้ามีกลไกทาง Security มาเกี่ยว ซึ่ง System Use Case จะอยู่ในรูปแบบ “เข้าระบบในฐานะ <Actor>” เท่านั้น
13. การใช้ <--<<extend>>--> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ทางเดียวไปยัง System Use Case หลัก
14. การใช้ <--<<include>>--> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ที่ใหญ่กว่า ไปยัง System Use Case ที่จะถูกรวบเข้ามา

### Checklist สำหรับการทวนสอบความถูกต้องของ Requirements

1. งาน (ระบบย่อยของแต่ละคน) ต้องไม่ซ้ำกับเพื่อนในกลุ่ม
2. งาน (ระบบย่อยของแต่ละคน) ต้องสอดคล้องกับ Business Domain ของระบบหลัก
3. ระบบย่อยของแต่ละคน ต้องเชื่อมโยงและต่อเนื่องกับระบบย่อยของคนอื่นๆ ในทีม ไม่สามารถเป็นงานเดียวที่ไม่เกี่ยวข้องกับใครเลยได้
4. Entity (ตาราง) ที่ออกแบบ ต้องประกอบด้วย Entity หลัก 1 ตาราง และมีความสัมพันธ์กับ Entity อื่นๆ อย่างน้อย 3 ตาราง กล่าวคือ ต้องมีความสัมพันธ์ (Relation) ระหว่างตารางอย่างน้อย 3 เส้น
5. ใน Entity หลัก ต้องมีคอลัมน์ (ฟิลด์) ที่ใช้เก็บข้อมูลซึ่งหมายความตามลักษณะของระบบย่อยที่ออกแบบ

การจำลองตัวอย่างตารางและข้อมูล (เพื่อใช้ในการเตรียมร่าง User Interface, System Activity Diagram และ Class Diagram)

จาก Requirements, จาก ข้อมูล Output และ Entity ที่ได้ออกแบบไว้เราจะนำมาสมมติเป็นตารางในฐานข้อมูล โดยกำหนด Primary Key เป็นตัวเลขรวมถึงการสมมติ Foreign Key เพื่อเข้มข้นข้อมูลระหว่างตาราง ซึ่งจะช่วยให้สามารถออกแบบและวิเคราะห์ระบบได้อย่างมีโครงสร้างและชัดเจน

## วิเคราะห์ Entity ที่เกี่ยวข้อง

### ผู้ใช้งาน (User)

- ชื่อ Entity: User
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Username: เก็บในรูปแบบ varchar ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Password: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Email: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - First Name: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Last Name: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Birthday: เก็บในรูปแบบ date ไม่สามารถเป็นค่า Null ได้

ID INTEGE R NOT NULL, PK	USERNAM E VARCHAR NOT NULL, Unique	PASSWORD VARCHAR NOT NULL	EMAIL VARCHAR NOT NULL	FIRST_NA ME VARCHAR NOT NULL	LAST_NA ME VARCHAR NOT NULL	BIRTHDA Y DATE NOT NULL
1001	User01	Encrypt(12345 6)	user01@gmail.co m	SA	68	31-12- 2004
1002	User02	Encrypt(12345 6)	user02@gmail.co m	SE	69	25-12- 2005

## Role

- ชื่อ Entity: Role
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบของ integer ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Title: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Description: เก็บรูปแบบของ varchar และไม่สามารถเป็นค่า Null ได้

ID INTEGER NOT NULL, PK	TITLE VARCHAR NOT NULL	DESCRIPTION VARCHAR NULL
2001	Admin	เป็นผู้ดูแลระบบ มีสิทธิการจัดการทั้งหมด
2002	Staff	พนักงาน คอยตรวจสอบการโอนเงินเป็นต้น
2003	Member	ผู้ใช้งานไป

## สิทธิ

- ชื่อ Entity: Permission
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบของ integer ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Title: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Description: เก็บรูปแบบของ varchar และไม่สามารถเป็นค่า Null ได้
  - Role: เก็บในรูปแบบความสัมพันธ์แบบ Many-to-Many โดยมีการสร้างของตารางดังนี้
    - Permission\_ID: เป็น Primary Key เก็บในรูปแบบของ integer และต้องไม่เป็นค่า Null
    - Role\_ID: เป็น Primary Key เก็บในรูปแบบของ integer และต้องไม่เป็นค่า Null

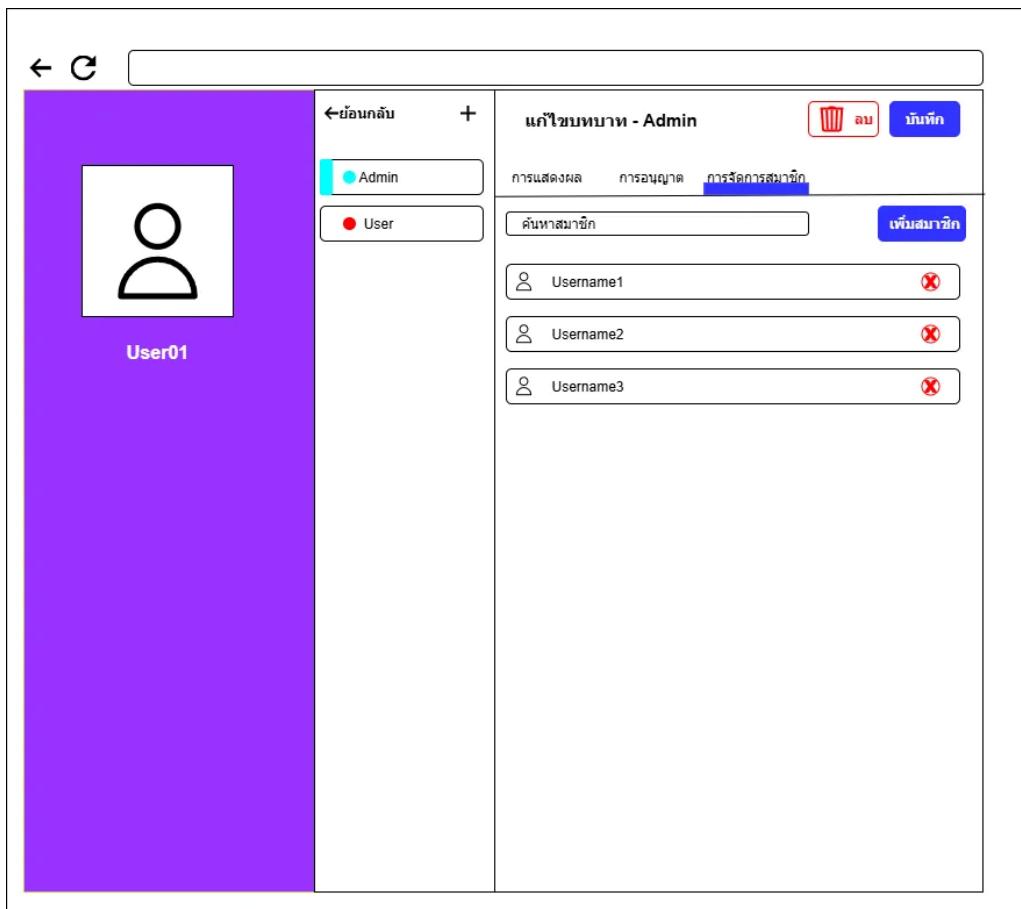
ID INTEGER NOT NULL, PK	TITLE VARCHAR NOT NULL	DESCRIPTION VARCHAR NULL
3001	ViewUser	ดูข้อมูลผู้ใช้งาน
3002	EditUser	แก้ไขข้อมูลผู้ใช้งาน
3001	DeleteUser	ลบผู้ใช้งาน
3004	ManageGame	จัดการข้อมูลเกม เช่น แก้ไขเกม หรือ ลบเกมออกจากร้าน
3005	BuyGame	ซื้อเกมได้

PERMISSION_ID	ROLE_ID
INTEGER	INTEGER
NOT NULL, FK	NOT NULL, FK
3001	2001
3002	2001
3003	2001
3004	2001
3005	2001
3002	2003
3005	2003

## หลักการออกแบบ User Interface

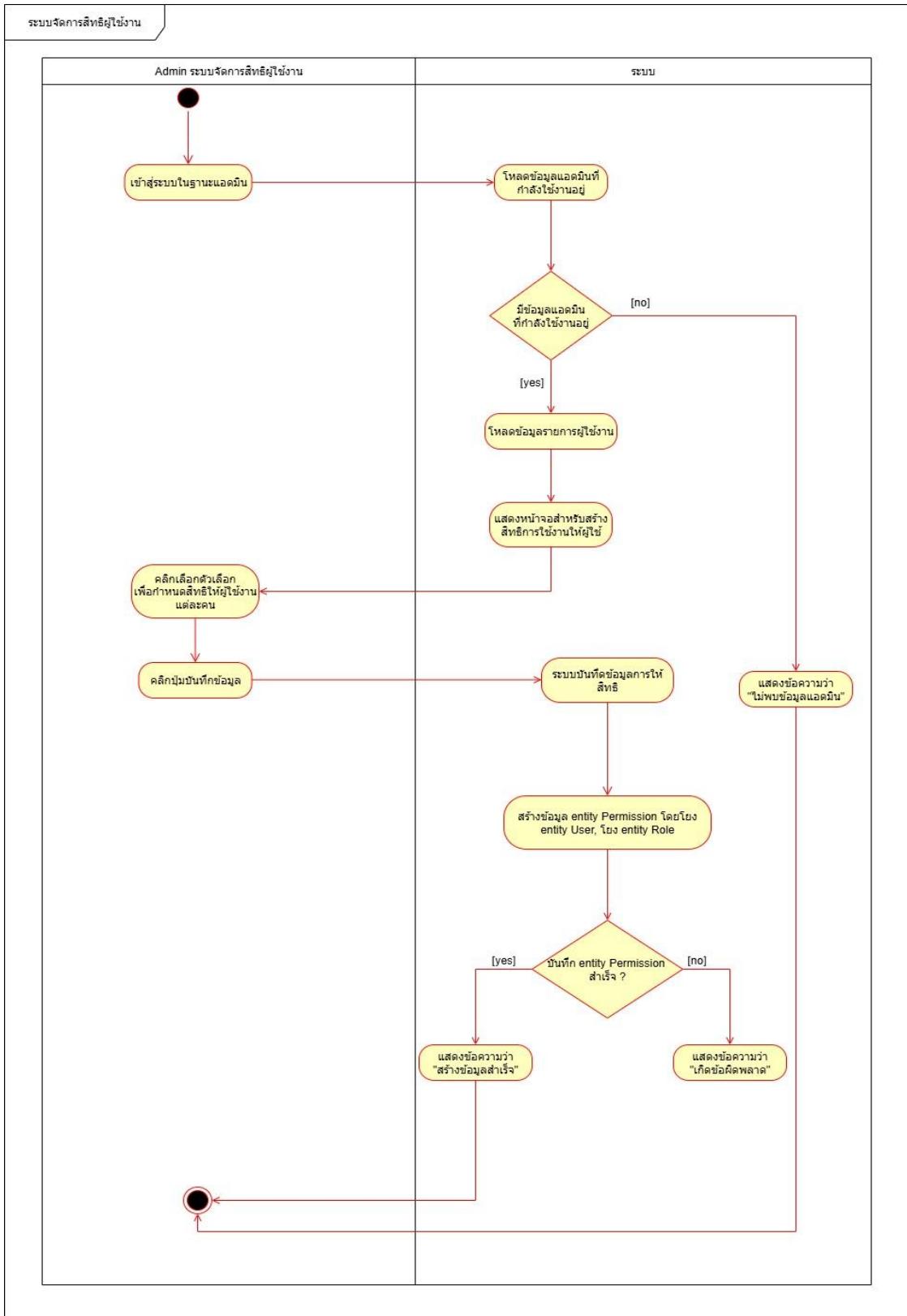
- เมื่อได้กีตามที่มี Foreign Key (FK) ซึ่งจาก ตารางหลักกลับไปยัง ตารางสนับสนุน  
ให้ใช้ ComboBox เป็นตัวเลือกเพื่อเชื่อมโยงข้อมูลใน User Interface
- สำหรับ Field ประเภทอื่นๆ ให้สร้างตามประเภทของข้อมูล เช่น
  - Textbox สำหรับข้อความทั่วไป
  - Password สำหรับข้อมูลรหัสผ่าน
  - Datetime Picker สำหรับเลือกวันและเวลา
  - Numeric Input สำหรับป้อนตัวเลข

## UI Design

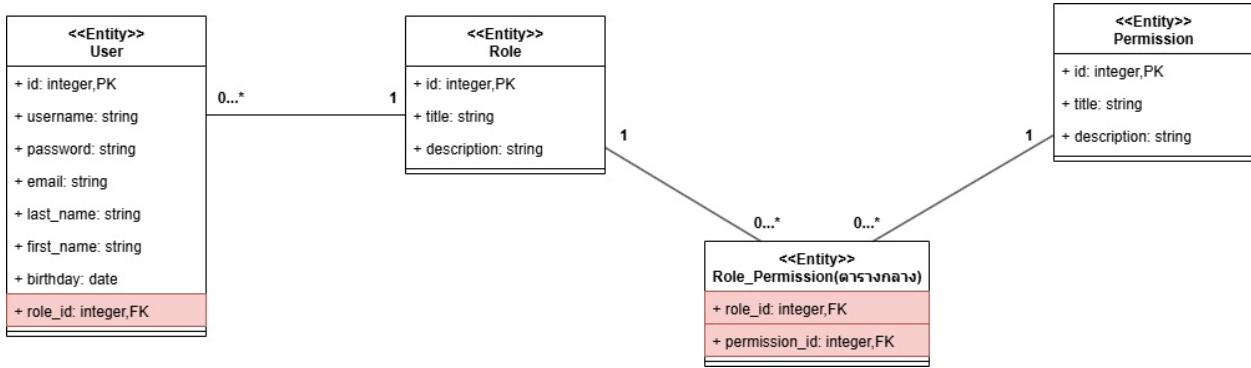


The UI design mockup illustrates a user management interface. On the left, a purple sidebar features a user icon and the text "User01". The main area has a header with a back arrow, a refresh icon, and a search bar. Below the header is a sidebar with a "ย้อนกลับ" button, a "+" button, and two radio buttons: "Admin" (selected) and "User". The main content area is titled "แก้ไขข้อมูลหน้า - Admin" and includes three tabs: "การแสดงผล", "การอนญาต", and "การรับรองสมาชิก" (selected). It contains a "ค้นหาสมาชิก" input field and three entries: "Username1", "Username2", and "Username3", each with a red "X" icon to its right. A blue "เพิ่มสมาชิก" button is located next to the search field.

## Activity Diagram

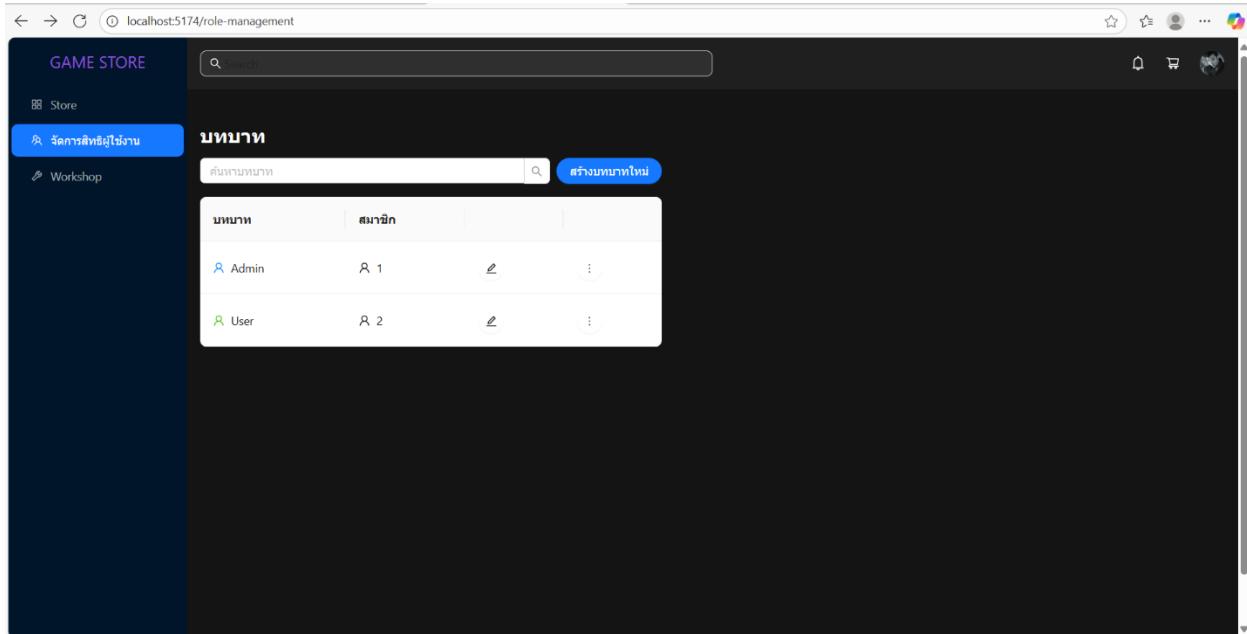


## Class Diagram at Analysis Level



## UI

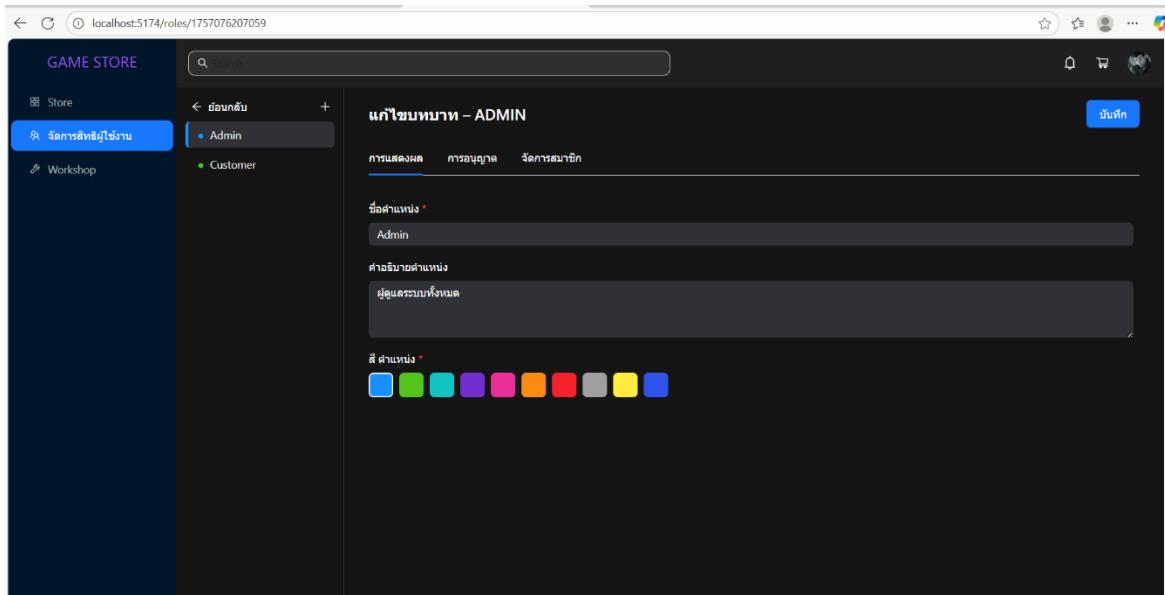
### RoleManagement Page



The screenshot shows a web application interface titled "GAME STORE". On the left sidebar, there are links for "Store", "จัดการสิทธิ์ในร้าน" (Manage Store Rights), and "Workshop". The main content area has a search bar at the top. Below it, a table lists roles:

บทบาท	ผู้ดูแล	ผู้ใช้งาน	⋮
Admin	ผู้ดูแล	ผู้ใช้งาน	⋮
User	ผู้ดูแล	ผู้ใช้งาน	⋮

### RoleEdit Page



The screenshot shows the "แก้ไขบทบาท – ADMIN" (Edit Role – Admin) page. On the left sidebar, there are links for "Store", "จัดการสิทธิ์ในร้าน" (Manage Store Rights), and "Workshop". The main content area shows the "Admin" role selected in a sidebar menu. The main form fields include:

- ชื่อผู้ดูแล \*: Admin
- ค่าอินบ็อกซ์เดฟолต์: ผู้ดูแลและรับบล็อกเมด
- สี ลักษณะ \*: A color palette with various colored squares.

## SourceCode

### RoleManagement Page

```
import React, { useState } from "react";
import {
  Button,
  Input,
  Table,
  Typography,
  Space,
  Card,
  Dropdown,
  Popconfirm,
  message,
} from "antd";
import { UserOutlined, EditOutlined, MoreOutlined } from "@ant-design/icons";
import { useNavigate } from "react-router-dom";

const { Title } = Typography;

const RoleManagement: React.FC = () => {
  const navigate = useNavigate();

  const [roles, setRoles] = useState([
    {
      key: "1",
      role: "Admin",
      members: 1,
      icon: <UserOutlined style={{ color: "#1890ff" }} />,
    },
    {
      key: "2",
      role: "User",
    },
  ]);

  return (
    <Table
      columns={columns}
      dataSource={roles}
      bordered
      rowKey="key"
      pagination={false}
    >
      <Table.Column title="Role" dataIndex="role" width="15%" />
      <Table.Column title="Members" dataIndex="members" width="15%" />
      <Table.Column title="Actions" dataIndex="icon" width="70%" />
    </Table>
  );
}

export default RoleManagement;
```

```
        members: 2,  
        icon: <UserOutlined style={{ color: "#52c41a" }} />,  
      },  
    ]);  
  
const handleDeleteRole = (key: string) => {  
  setRoles((prev) => prev.filter((r) => r.key !== key));  
  message.success("ลบบทบาทเรียบร้อยแล้ว");  
};  
  
const handleAddRole = () => {  
  const newRole = {  
    key: Date.now().toString(),  
    role: "ตำแหน่งใหม่",  
    members: 0,  
    icon: <UserOutlined style={{ color: "#faad14" }} />,  
  };  
  
  setRoles((prev) => {  
    const everyone = prev.find((r) => r.key === "everyone");  
    const others = prev.filter((r) => r.key !== "everyone");  
    return [...others, newRole, everyone!];  
  });  
  
  navigate('/roles/${newRole.key}');  
};  
  
const columns = [  
  {  
    title: "บทบาท",  
    dataIndex: "role",  
    key: "role",  
    render: (text: string, record: any) => (  
      <Space>  
      {record.icon}  
    ),  
  },  
];
```

```
{text}
</Space>
),
},
{
  title: "สมาชิก",
  dataIndex: "members",
  key: "members",
  render: (count: number) => (
    <Space>
      <UserOutlined />
      {count}
    </Space>
  ),
},
{
  title: "",
  key: "edit",
  render: (_: any, record: any) => (
    <Button
      shape="circle"
      icon={<EditOutlined />}
      style={{ border: "none" }}
      onClick={() => navigate('/roles/${record.key}')}>
    />
  ),
},
{
  title: "",
  key: "actions",
  render: (_: any, record: any) => {
    // 🔒 ห้ามลบ role @everyone
    if (record.key === "everyone") return null;

  const items = [
```

```

{
  key: "delete",
  label: (
    <Popconfirm
      title="ຄູນແນ່ໃຈຫຼືອ່າມໍທີ່ຈະລົບບົກບາຫຼື?"
      okText="ລົບ"
      cancelText="ຍົກເລີກ"
      onConfirm={() => handleDeleteRole(record.key)}
    >
      <span style={{ color: "red" }}> ລົບ</span>
    </Popconfirm>
  ),
},
];
}

return (
  <Dropdown menu={{ items }} trigger={['click']}>
    <Button
      shape="circle"
      icon={<MoreOutlined />}
      style={{ border: "none" }}
    />
  </Dropdown>
);
},
];
};

return (
  <div style={{ background: "#141414", minHeight: "100vh" }}>
    <div style={{ padding: "16px", maxWidth: "600px" }}>
      <Title level={3} style={{ color: "white" }}>
        ບົກບາຫຼື
      </Title>
    </div>
  </div>
);
}

```

```

<Space style={{ marginBottom: 16 }}>
  <Input.Search placeholder="ค้นหาบทบาท" style={{ width: 430 }} />
  <Button type="primary" shape="round" onClick={handleAddRole}>
    สร้างบทบาทใหม่
  </Button>
</Space>
<Table
  columns={columns}
  dataSource={roles}
  pagination={false}
  style={{
    background: "#1f1f1f",
    borderRadius: 8,
    overflow: "hidden",
  }}
/>
</div>
</div>
);
};

export default RoleManagement;

```

### RoleEdit Page

```

// frontend/src/pages/role/RoleEdit.tsx
import React, { useEffect, useMemo, useState } from "react";
import {
  Typography,
  Input,
  Tabs,
  Button,
  Row,
  Col,
  Switch,

```

```
Modal,
List,
Checkbox,
message,
Spin,
Popconfirm,
Tooltip,
ConfigProvider,
theme,
} from "antd";
import {
  ArrowLeftOutlined,
  PlusOutlined,
  DeleteOutlined,
  CloseOutlined,
  CheckOutlined,
} from "@ant-design/icons";
import { useParams, useNavigate } from "react-router-dom";
import axios from "axios";

const { Title, Text } = Typography;
const API_URL = "http://localhost:8088";

const colorPalette = [
  "#1890ff", "#52c41a", "#13c2c2", "#722ed1", "#eb2f96",
  "#fa8c16", "#f5222d", "#a0a0a0", "#ffec3d", "#2f54eb",
];

interface Permission { ID: number; title: string; description?: string; }
interface User { ID: number; username: string; role_id: number; }
interface RolePermission { ID: number; permission_id: number; }
interface Role {
  ID: number;
  title: string; description?: string; color?: string;
  users?: User[]; role_permissions?: RolePermission[];
}
```

```

}

const RoleEdit: React.FC = () => {
  const { id } = useParams();
  const navigate = useNavigate();
  const roleIdNum = useMemo(() => (id ? Number(id) : undefined), [id]);

  const [messageApi, contextHolder] = message.useMessage();

  // Left: roles
  const [roles, setRoles] = useState<Role[]>([]);
  const [rolesLoading, setRolesLoading] = useState(false);

  // Right: detail
  const [loadingDetail, setLoadingDetail] = useState(false);
  const [roleName, setRoleName] = useState("");
  const [roleDescription, setRoleDescription] = useState("");
  const [color, setColor] = useState("#1890ff");
  const [members, setMembers] = useState<User[]>([]);
  const [activeTab, setActiveTab] = useState("display");

  // Permissions
  const [permissions, setPermissions] = useState<Permission[]>([]);
  const [permissionStates, setPermissionStates] = useState<Record<number, boolean>>({});
  const [rolePermissionMap, setRolePermissionMap] = useState<Record<number, number>>({});
  const [permissionSearch, setPermissionSearch] = useState("");
  const [bulkLoading, setBulkLoading] = useState(false); // ◆ สถานะโหลดเวลาทำ Select All / Clear All

  // Users
  const [allUsers, setAllUsers] = useState<User[]>([]);

  // Search states
  const [memberSearch, setMemberSearch] = useState(""); // ในแท็บ “จัดการสมาชิก”
  const [addSearchText, setAddSearchText] = useState(""); // ในโมดูล “เพิ่มสมาชิก”

```

```
// Modal add members
const [isModalVisible, setIsModalVisible] = useState(false);
const [selectedMembers, setSelectedMembers] = useState<number[]>([]);
const [addingMembers, setAddingMembers] = useState(false);

// Deleting role
const [deleting, setDeleting] = useState(false);

// default role (User) to demote when removing
const defaultRoleId = useMemo(
() => roles.find((r) => (r.title || "").toLowerCase() === "user")?.ID,
[roles]
);

// ----- fetch -----
const fetchRoles = async () => {
    setRolesLoading(true);
    try {
        const res = await axios.get<Role[]>(`${API_URL}/roles`);
        setRoles(res.data || []);
        if (!roleIdNum && res.data?.length) {
            navigate(`/roles/${res.data[0].ID}`, { replace: true });
        }
    } catch {
        messageApi.error("โหลดรายการบทบาทไม่สำเร็จ");
    } finally {
        setRolesLoading(false);
    }
};

const fetchRoleDetail = async (rid: number) => {
    setLoadingDetail(true);
    try {
        const [roleRes, permRes, usersRes] = await Promise.all([

```

```

axios.get<Role>(`${API_URL}/roles/${rid}`),
permissions.length
? Promise.resolve({ data: permissions })
: axios.get<Permission[]>(`${API_URL}/permissions`),
  axios.get<User[]>(`${API_URL}/users`),
);

const role = roleRes.data;
setRoleName(role.title || "");
setRoleDescription(role.description || "");
setColor(role.color || colorPalette[0]);
setMembers(role.users || []);

const map: Record<number, number> = {};
(role.role_permissions || []).forEach((rp) => {
  map[rp.permission_id] = rp.ID;
});
setRolePermissionMap(map);

const perms = (permRes as any).data as Permission[];
setPermissions(perms);
const state: Record<number, boolean> = {};
perms.forEach((p) => (state[p.ID] = map[p.ID] !== undefined));
setPermissionStates(state);

setAllUsers(usersRes.data || []);
} catch {
  messageApi.error("ໂທລດຂໍ້ມູນບໍາຫາທີ່ໄສ່ເສົ້າເຮັດ");
} finally {
  setLoadingDetail(false);
}
};

useEffect(() => { fetchRoles(); /* eslint-disable-next-line */ }, []);

```

```
useEffect(() => { if (roleIdNum) fetchRoleDetail(roleIdNum); /* eslint-disable-next-line */ },  
[roleIdNum]);  
  
useEffect(() => {  
    if (!permissions.length) return;  
    setPermissionStates(() => {  
        const next: Record<number, boolean> = {};  
        permissions.forEach((p) => {  
            next[p.ID] = rolePermissionMap[p.ID] !== undefined;  
        });  
        return next;  
    });  
}, [rolePermissionMap, permissions]);  
  
// ----- actions -----  
const addRole = async () => {  
    try {  
        const res = await axios.post<Role>(`${API_URL}/roles`, {  
            title: "New Role",  
            description: "",  
            color: "#a0a0a0",  
        });  
        messageApi.success("สร้างบทบาทใหม่เรียบร้อย");  
        await fetchRoles();  
        navigate(`/roles/${res.data.ID}`);  
    } catch {  
        messageApi.error("สร้างบทบาทไม่สำเร็จ");  
    }  
};  
  
const deleteRole = async () => {  
    if (!roleIdNum) return;  
    try {  
        setDeleting(true);  
        const key = "deleteRole";
```

```
messageApi.open({ key, type: "loading", content: "ກຳລັງລົບທຳທາທ...", duration: 0 });

await axios.delete(` ${API_URL}/roles/${roleIdNum}`);

const rolesRes = await axios.get<Role[]>(` ${API_URL}/roles`);
const fresh = rolesRes.data || [];
setRoles(fresh);

messageApi.open({ key, type: "success", content: "ລົບທຳທາທເຮືອບ້ອຍ", duration: 1.6 });

if (fresh.length) {
    const oldIdx = roles.findIndex((r) => r.ID === roleIdNum);
    const nextIdx = Math.min(Math.max(oldIdx - 1, 0), fresh.length - 1);
    const nextId = fresh[nextIdx].ID;
    if (nextId) {
        navigate(`/roles/${nextId}`, { replace: true });
    }
} else {
    setRoleName("");
    setRoleDescription("");
    setColor(colorPalette[0]);
    setMembers([]);
    setPermissionStates({});
    setRolePermissionMap({});
}

} catch (err: any) {
    messageApi.open({
        type: "error",
        content: err?.response?.data?.error || "ລົບທຳທາທໄມ້ສໍາເລັດ",
        duration: 2.4,
    });
} finally {
    setDeleting(false);
}
};
```

```
const updateRole = async () => {
  if (!roleIdNum) return;
  try {
    await axios.patch(`.${API_URL}/roles/${roleIdNum}`, {
      title: roleName,
      description: roleDescription,
      color,
    });
    messageApi.success("บันทึกบทบาทเรียบร้อย");
    fetchRoles();
  } catch {
    messageApi.error("บันทึกบทบาทไม่สำเร็จ");
  }
};

const togglePermission = async (pid: number, checked: boolean) => {
  if (!roleIdNum) return;
  try {
    if (checked) {
      const res = await axios.post(`.${API_URL}/rolepermissions`, {
        role_id: roleIdNum,
        permission_id: pid,
      });
      setRolePermissionMap({ ...rolePermissionMap, [pid]: (res.data as any).ID });
      setPermissionStates({ ...permissionStates, [pid]: true });
    } else {
      const rplD = rolePermissionMap[pid];
      if (!rplD) return;
      await axios.delete(`.${API_URL}/rolepermissions/${rplD}`);
      const newMap = { ...rolePermissionMap };
      delete newMap[pid];
      setRolePermissionMap(newMap);
      setPermissionStates({ ...permissionStates, [pid]: false });
    }
  }
};
```

```

} catch {
    messageApi.error("ອັບເດຕສີທີ່ມີສຳເຮົາ");
}

};

// ◆ ໜ້າຍເຫຼືອສໍາຫຼັບເລືອກ/ລ້າງ ທີ່ໜີ້ຈຸດ (ທຳມານຜົກກາຣົນໜາ)
const enablePermissions = async (pids: number[]) => {
    if (!roleIdNum) return;
    // ເນັ້າມຕ້ວທີ່ຍັງໄປດ້ອຍໆ
    const toCreate = pids.filter((pid) => !permissionStates[pid]);
    if (!toCreate.length) return;
    const results = await Promise.all(
        toCreate.map((pid) =>
            axios.post(`${API_URL}/rolepermissions`, {
                role_id: roleIdNum,
                permission_id: pid,
            })
        )
    );
    // ອັບເດຕ map/state ທີ່ເດືອກ
    const newMap = { ...rolePermissionMap };
    const newState = { ...permissionStates };
    results.forEach((res, idx) => {
        const pid = toCreate[idx];
        const id = (res.data as any).ID;
        newMap[pid] = id;
        newState[pid] = true;
    });
    setRolePermissionMap(newMap);
    setPermissionStates(newState);
};

const disablePermissions = async (pids: number[]) => {
    if (!roleIdNum) return;
    // ເນັ້າມຕ້ວທີ່ເປີດອຍໆ
}

```

```

const toDelete = pids.filter((pid) => !permissionStates[pid]);
if (!toDelete.length) return;
await Promise.all(
  toDelete.map((pid) => {
    const rpID = rolePermissionMap[pid];
    if (!rpID) return Promise.resolve(null as any);
    return axios.delete(` ${API_URL}/rolepermissions/${rpID}`);
  })
);
const newMap = { ...rolePermissionMap };
const newState = { ...permissionStates };
toDelete.forEach((pid) => {
  delete newMap[pid];
  newState[pid] = false;
});
setRolePermissionMap(newMap);
setPermissionStates(newState);
};

const handleSelectAll = async () => {
  try {
    setBulkLoading(true);
    const ids = filteredPermissions.map((p) => p.ID); // ตามผลการค้นหา
    await enablePermissions(ids);
    messageApi.success("เปิดสิทธิ์ทั้งหมดตามที่แสดงแล้ว");
  } catch {
    messageApi.error("เปิดสิทธิ์ทั้งหมดไม่สำเร็จ");
  } finally {
    setBulkLoading(false);
  }
};

const handleClearAll = async () => {
  try {
    setBulkLoading(true);

```

```

const ids = filteredPermissions.map((p) => p.ID); // ตามผลการค้นหา
await disablePermissions(ids);
messageApi.success("ปิดสิทธิ์ทั้งหมดตามที่แสดงแล้ว");
} catch {
  messageApi.error("ปิดสิทธิ์ทั้งหมดไม่สำเร็จ");
} finally {
  setBulkLoading(false);
}
};

// remove member (demote to default role)
const removeMember = async (uid: number, username: string) => {
  if (!roleIDNum) return;
  if (!defaultRoleID || defaultRoleID === roleIDNum) {
    messageApi.error("ไม่พบ/ไม่สามารถยกย้ายไปบทบาทพื้นฐาน");
    return;
  }
  try {
    await axios.patch(`/${API_URL}/users/${uid}/role`, { role_id: defaultRoleID });
    const res = await axios.get<Role>(`${API_URL}/roles/${roleIDNum}`);
    setMembers(res.data.users || []);
    messageApi.success(`นำ ${username} ออกจากบทบาทเรียบร้อย`);
  } catch {
    messageApi.error("นำสมาชิกออกไม่สำเร็จ");
  }
};

// ----- modal add members -----
const showModal = () => {
  setIsModalVisible(true);
  setSelectedMembers([]);
  setAddSearchText("");
};

const handleCheckboxToggle = (uid: number) => {

```

```

setSelectedMembers((prev) =>
  prev.includes(uid) ? prev.filter((x) => x !== uid) : [...prev, uid]
);
};

const handleOk = async () => {
  if (!roleIdNum || selectedMembers.length === 0) {
    setIsModalVisible(false);
    return;
  }
  try {
    setAddingMembers(true);
    await Promise.all(
      selectedMembers.map((uid) =>
        axios.patch(`${API_URL}/users/${uid}/role`, { role_id: roleIdNum })
      )
    );
    const res = await axios.get<Role>(`${API_URL}/roles/${roleIdNum}`);
    setMembers(res.data.users || []);
    setIsModalVisible(false);
    setSelectedMembers([]);
    messageApi.success("เพิ่มสมาชิกเข้าบทบาทเรียบร้อย");
  } catch {
    messageApi.error("เพิ่มสมาชิกไม่สำเร็จ");
  } finally {
    setAddingMembers(false);
  }
};

const handleCancel = () => setIsModalVisible(false);

// ----- filters -----
const filteredUsers = allUsers.filter(
  (u) =>
    u.username.toLowerCase().includes(addSearchText.toLowerCase()) &&

```

```

!members.some((m) => m.ID === u.ID)
);

const filteredMembersTab = useMemo(
() =>
members.filter((m) =>
  m.username.toLowerCase().includes(memberSearch.toLowerCase())
),
[members, memberSearch]
);

const filteredPermissions = permissions.filter((p) =>
(p.title || "").toLowerCase().includes(permissionSearch || "").toLowerCase()
);

// ◆ ความสูงของลิสต์เพาธ์ส่วน
const scrollAreaHeight = "calc(100vh - 260px)";

return (
<ConfigProvider
theme={{
algorithm: theme.darkAlgorithm,
token: {
colorBgBase: "#141414",
colorBgContainer: "#1f1f1f",
colorText: "#ffffff",
colorTextSecondary: "#aaaaaa",
colorBorder: "#303030",
colorPrimary: "#1677ff",
borderRadius: 8,
},
}}
>
{contextHolder}

```

```
/* ไม่ต้อง CSS ภายนอก — ใช้เฉพาะ inline style */

<div
  className="role-page"
  style={{ background: "#141414", height: "100vh", flex: 1, overflow: "hidden", display: "flex" }}
>
  /* Left */
  <div style={{ width: 220, padding: 12, borderRight: "1px solid #333", display: "flex",
    flexDirection: "column" }}>
    <div style={{ marginBottom: 8, display: "flex", justifyContent: "space-between", alignItems:
      "center", color: "white" }}>
      <Button
        type="text"
        size="small"
        icon={<ArrowLeftOutlined />}
        onClick={() => navigate("/roles", { replace: true })}
        style={{ color: "white" }}>
        >
        ย้อนกลับ
      </Button>
      <Button type="text" size="small" icon={<PlusOutlined />} onClick={addRole} style={{ color:
        "white" }} title="สร้างบทบาทใหม่" />
    </div>

    <div style={{ flex: 1, overflowY: "auto" }}>
      <Spin spinning={rolesLoading}>
        {roles.map((role) => {
          const isActive = role.ID === roleIdNum;
          return (
            <div
              key={role.ID}
              onClick={() => navigate(`/roles/${role.ID}`)}
              style={{
                display: "flex", alignItems: "center", padding: "8px 12px", marginBottom: 4,
                borderRadius: 6, cursor: "pointer", color: "white",
                background: isActive ? "#2f3136" : "transparent",
              }
            >
              <span>{role.name}</span>
              <span>{role.description}</span>
            </div>
          );
        })
      </Spin>
    </div>
  </div>

```

```
borderLeft: isActive ? `4px solid ${role.color} || "#1890ff"} : "4px solid transparent",
    transition: "all 0.2s",
}
onMouseEnter={(e) => { if (!isActive) (e.currentTarget.style.background = "#1f1f1f"); }}
onMouseLeave={(e) => { if (!isActive) (e.currentTarget.style.background =
"transparent"); }}
>
<span style={{ color: role.color || "#1890ff", marginRight: 8 }}>●</span>
<span style={{ overflow: "hidden", textOverflow: "ellipsis", whiteSpace: "nowrap" }}>
    {role.title}
</span>
</div>
);
)}
</Spin>
</div>
</div>

/* Right */
<div
style={{
flex: 1,
padding: "16px 32px",
boxSizing: "border-box",
maxWidth: "1000px",
margin: "0 auto",
display: "flex",
flexDirection: "column",
height: "100%",
overflow: "hidden",
}}
>
/* Header (fixed) */
<div style={{ display: "flex", marginBottom: 16, alignItems: "center", gap: 16 }}>
    <Title level={4} style={{ color: "white", margin: 0 }}>
```

```

    ແກ້ໄຂບ່ານທຳມາດ – {roleName ? roleName.toUpperCase() : "LOADING"}
    </Title>
    <div style={{ marginLeft: "auto", display: "flex", gap: 8 }}>
        <Popconfirm title="ຕ້ອງກາລຸບບ່ານນີ້ຫຼືອ່ານີ້?" okText="ລົບ" cancelText="ຍົກເລີກ"
        onConfirm={deleteRole}>
            <Button danger icon={<DeleteOutlined />} loading={deleting}>ລົບ</Button>
        </Popconfirm>
        <Button type="primary" onClick={updateRole}>ບັນທຶກ</Button>
    </div>
</div>

/* Tabs */
<div style={{ flex: 1, minHeight: 0, overflow: "hidden" }}>
    <Spin spinning={loadingDetail}>
        <Tabs
            activeKey={activeTab}
            onChange={setActiveTab}
            items={[
                {
                    key: "display",
                    label: <Text style={{ color: "white" }}>ກາຮແສດງຜລ</Text>,
                    children: (
                        <div style={{ maxWidth: 700, height: scrollAreaHeight, overflowY: "auto",
                        paddingRight: 8 }}>
                            <div style={{ marginBottom: 16 }}>
                                <Text style={{ color: "white" }}>
                                    ຂໍອຕໍມາແໜ່ງ <Text type="danger">*</Text>
                                </Text>
                                <Input value={roleName} onChange={(e) => setRoleName(e.target.value)}
                                style={{ marginTop: 8 }} />
                            </div>
                            <div style={{ marginBottom: 16 }}>
                                <Text style={{ color: "white" }}>ຄໍາອະນຸມາຍຕໍມາແໜ່ງ</Text>
                            </div>
                        </div>
                    )
                }
            ]>
        </Tabs>
    </Spin>
</div>

```

```

<Input.TextArea value={roleDescription} onChange={(e) =>
setRoleDescription(e.target.value)} rows={3} style={{ marginTop: 8 }} />

</div>

<div>
<Text style={{ color: "white" }}>
ສື່ຕຳແໜ່ງ <Text type="danger">*</Text>
</Text>
<Row gutter={[8, 8]} style={{ marginTop: 8 }}>
{colorPalette.map((c) => (
<Col key={c}>
<div
onClick={() => setColor(c)}
style={{
width: 32, height: 32, borderRadius: 6, cursor: "pointer",
background: c, border: color === c ? "2px solid #fff" : "2px solid
transparent",
}}
/>
</Col>
))}
</Row>
</div>
</div>
),
},
{
key: "permissions",
label: <Text style={{ color: "white" }}>ກາຮອນໜູາຕ</Text>,
children: (
<div style={{ maxWidth: 700 }}>
{/* ແກ້ວຄວບຄຸມຄົນທາ + ປຸ່ມເລືອກ/ລ້າງທັງໝົດ */}
<div style={{ display: "flex", gap: 8, alignItems: "center", marginBottom: 12,
flexWrap: "wrap" }}>
<Input.Search

```

```

placeholder="ค้นหาสิทธิ์"
value={permissionSearch}
onChange={(e) => setPermissionSearch(e.target.value)}
style={{ flex: "1 1 260px", minWidth: 220 }}
allowClear
/>
<Button
  onClick={handleSelectAll}
  loading={bulkLoading}
  disabled={bulkLoading || filteredPermissions.length === 0}
>
  เลือกทั้งหมด (ตามที่แสดง)
</Button>
<Button
  onClick={handleClearAll}
  loading={bulkLoading}
  disabled={bulkLoading || filteredPermissions.length === 0}
>
  ถ้าทั้งหมด (ตามที่แสดง)
</Button>
</div>

```

```

/* ◆ เลื่อนเฉพาะรายการสิทธิ์ */
<div style={{ height: scrollAreaHeight, overflowY: "auto", paddingRight: 8 }}>
  {filteredPermissions.map((perm) => (
    <div
      key={perm.ID}
      style={{
        display: "flex",
        justifyContent: "space-between",
        alignItems: "center",
        padding: "10px 0",
        borderBottom: "1px solid #333",
        color: "white",
      }}

```

```

>
<div style={{ paddingRight: 16 }}>
  <div style={{ fontWeight: 600 }}>{perm.title}</div>
  {perm.description} && (
    <div style={{ color: "#aaa", fontSize: 12 }}>{perm.description}</div>
  )
</div>
<Switch
  checked={!permissionStates[perm.ID]}
  onChange={(checked) => togglePermission(perm.ID, checked)}
/>
</div>
))}

{!filteredPermissions.length && (
  <div style={{ color: "#aaa" }}>ມີພະສິຫຼື່ງທີ່ຄົນໜາ</div>
)}

</div>
</div>

),
},
{
  key: "members",
  label: <Text style={{ color: "white" }}>ຈັດກາຮມາຊີກ</Text>,
  children: (
    <div style={{ height: scrollAreaHeight, overflowY: "auto", paddingRight: 8 }}>
      <div style={{ marginBottom: 16, display: "flex", justifyContent: "space-between", gap: 12 }}>
        <Input
          placeholder="ຄົນໜາສມາຊີກ"
          value={memberSearch}
          onChange={(e) => setMemberSearch(e.target.value)}
          style={{ width: "70%" }}
        />
        <Button type="primary" onClick={showModal}>ເພີ່ມສມາຊີກ</Button>
      </div>
    </div>
  ),
}

```

```

{filteredMembersTab.length === 0 ? (
  <div style={{ color: "#aaa", textAlign: "center" }}>
    'ไม่พบสมาชิก{memberSearch ? "ตามคำค้น" : ""}{"
  {!memberSearch && (
    <a style={{ color: "#1890ff" }} onClick={showModal}>
      เพิ่มสมาชิกให้กับทบทวนนี้
    </a>
  )}}
</div>
) : (
  <List
    dataSource={filteredMembersTab}
    renderItem={(m) => (
      <List.Item
        actions={[
          roleIdNum !== defaultRoleId
            ? [
              <Tooltip title="ลบสมาชิกออก" key="remove">
                <Popconfirm
                  title={`ยืนยันนำ ${m.username} ออกจากบทบาทนี้?`}
                  okText="ลบ"
                  cancelText="ยกเลิก"
                  onConfirm={() => removeMember(m.ID, m.username)}
                >
                  <Button danger size="small" shape="circle" icon={<CloseOutlined />} />
                </Popconfirm>
              </Tooltip>,
            ]
            : []
        }
      >
        <span style={{ color: "white" }}>{m.username}</span>
      </List.Item>
    )}
  )
)

```

```

        )}
      />
    )}
  </div>
),
},
]}
/>
</Spin>
</div>

/* Modal: เพิ่มสมาชิก */
<Modal
  title={<span style={{ color: "black" }}>เพิ่มสมาชิก</span>}
  open={isModalVisible}
  onOk={handleOk}
  onCancel={handleCancel}
  okText="เพิ่ม"
  cancelText="ยกเลิก"
  confirmLoading={addingMembers}
>
  <Input.Search
    placeholder="ค้นหาสมาชิก"
    value={addSearchText}
    onChange={(e) => setAddSearchText(e.target.value)}
    style={{ marginBottom: 16 }}
  />
  <div style={{ maxHeight: 360, overflowY: "auto" }}>
    <List
      dataSource={filteredUsers}
      renderItem={(user) => {
        const selected = selectedMembers.includes(user.ID);
        return (
          <List.Item
            key={user.ID}

```

```
        onClick={() => handleCheckboxToggle(user.ID)}
```

```
        style={{
            cursor: "pointer",
            borderRadius: 6,
            marginBottom: 6,
            padding: "10px 12px",
            backgroundColor: selected ? "#e6f4ff" : "transparent",
            border: selected ? "1px solid #91caff" : "1px solid transparent",
            transition: "background-color 0.15s, border-color 0.15s",
        }}
        actions={[
            selected ? (
                <CheckOutlined key="checked" style={{ color: "#1677ff" }} />
            ) : null,
        ]}
    >
    <Checkbox
        checked={selected}
        onClick={(e) => {
            e.stopPropagation();
            handleCheckboxToggle(user.ID);
        }}
        style={{ marginRight: 8 }}
    />
    <span style={{ color: "#000" }}>{user.username}</span>
</List.Item>
);
}
/>
</div>
</Modal>
</div>
</div>
</ConfigProvider>
);
```

```
};
```

```
export default RoleEdit;
```

## Entity

```
//user.go
package entity

import (
    "time"

    "gorm.io/gorm"
)

type User struct {
    gorm.Model
    Username string `gorm:"uniqueIndex" json:"username"`
    Password string `json:"password"`
    Email    string `gorm:"uniqueIndex" json:"email"`
    FirstName string `json:"first_name"`
    LastName  string `json:"last_name"`
    Birthday  time.Time `json:"birthday"`

    RoleID uint `json:"role_id"`
    Role   *Role `gorm:"foreignKey:RoleID" json:"role"`
}

//role.go
package entity

import "gorm.io/gorm"

type Role struct {
    gorm.Model
    Title    string `json:"title"`
    Description string `json:"description"`

    RolePermissions []RolePermission `gorm:"foreignKey:RoleID" json:"role_permissions"`
}
```

```
    Users []User      `gorm:"foreignKey:RoleID" json:"users"`
}

//permission.go
package entity

import "gorm.io/gorm"

type Permission struct {
    gorm.Model
    Title     string `json:"title"`
    Description string `json:"description"`

    RolePermissions []RolePermission `gorm:"foreignKey:PermissionID" json:"role_permissions"`
}

//RolePermission.go
package entity

import "gorm.io/gorm"

type RolePermission struct {
    gorm.Model

    RoleID     uint     `json:"role_id"`
    Role       *Role    `gorm:"foreignKey:RoleID" json:"role"`

    PermissionID uint     `json:"permission_id"`
    Permission  *Permission `gorm:"foreignKey:PermissionID" json:"permission"`
}
```

## Controller

```
auth.go
package controllers

import (
    "net/http"

    "github.com/gin-gonic/gin"
    "gorm.io/gorm"

    "gameshop-backend/services"
)

var DB *gorm.DB

func InitDB(db *gorm.DB) {
    DB = db
}

type RegisterInput struct {
    Username string `json:"username" binding:"required"`
    Email    string `json:"email" binding:"required,email"`
    Password string `json:"password" binding:"required"`
}

type LoginInput struct {
    Email    string `json:"email" binding:"required,email"`
    Password string `json:"password" binding:"required"`
}

// POST /auth/register
func Register(c *gin.Context) {
    var input RegisterInput
    if err := c.ShouldBindJSON(&input); err != nil {
```

```
c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
return
}

if err := services.RegisterUser(DB, input.Username, input.Email, input.Password); err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}

c.JSON(http.StatusCreated, gin.H{"message": "Registered successfully"})
}

// POST /auth/login
func Login(c *gin.Context) {
    var input LoginInput
    if err := c.ShouldBindJSON(&input); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    token, err := services.LoginUser(DB, input.Email, input.Password)
    if err != nil {
        c.JSON(http.StatusUnauthorized, gin.H{"error": err.Error()})
        return
    }

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

user.go
package controllers

import (
    "net/http"
```

```
"gameshop-backend/configs"
"gameshop-backend/entity/role"
"github.com/gin-gonic/gin"
)

// GET /users
func GetUsers(c *gin.Context) {
    var users []entity.User
    if err := configs.DB().Find(&users).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, users)
}

// GET /users/:id
func GetUserById(c *gin.Context) {
    id := c.Param("id")
    var user entity.User
    if tx := configs.DB().Where("id = ?", id).First(&user); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "user not found"})
        return
    }
    c.JSON(http.StatusOK, user)
}

// POST /users
func CreateUser(c *gin.Context) {
    var user entity.User
    if err := c.ShouldBindJSON(&user); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if err := configs.DB().Create(&user).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    }
}
```

```
        return
    }

    c.JSON(http.StatusCreated, user)
}

// PATCH /users/:id
func UpdateUser(c *gin.Context) {
    id := c.Param("id")
    var user entity.User
    if tx := configs.DB().Where("id = ?", id).First(&user); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "user not found"})
        return
    }

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

    if err := configs.DB().Model(&user).Updates(input).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, user)
}

// DELETE /users/:id
func DeleteUser(c *gin.Context) {
    id := c.Param("id")
    if tx := configs.DB().Exec("DELETE FROM users WHERE id = ?", id); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "user not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "deleted successfully"})
}
```

```
}

role.go
package controllers

import (
    "net/http"

    "gameshop-backend/configs"
    "gameshop-backend/entity/role"
    "github.com/gin-gonic/gin"
)

// GET /roles
func GetRoles(c *gin.Context) {
    var roles []entity.Role
    if err := configs.DB().Find(&roles).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, roles)
}

// GET /roles/:id
func GetRoleById(c *gin.Context) {
    id := c.Param("id")
    var role entity.Role
    if tx := configs.DB().Where("id = ?", id).First(&role); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "role not found"})
        return
    }
    c.JSON(http.StatusOK, role)
}

// POST /roles
```

```
func CreateRole(c *gin.Context) {
    var role entity.Role

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

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

    c.JSON(http.StatusCreated, role)
}

// PATCH /roles/:id
func UpdateRole(c *gin.Context) {
    id := c.Param("id")
    var role entity.Role

    if tx := configs.DB().Where("id = ?", id).First(&role); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "role not found"})
        return
    }

    var input entity.Role

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

    if err := configs.DB().Model(&role).Updates(input).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, role)
}
```

```

// DELETE /roles/:id

func DeleteRole(c *gin.Context) {
    id := c.Param("id")

    if tx := configs.DB().Exec("DELETE FROM roles WHERE id = ?", id); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "role not found"})
        return
    }

    c.JSON(http.StatusOK, gin.H{"message": "deleted successfully"})
}

permission.go

package controllers

import (
    "net/http"

    "gameshop-backend/configs"
    "gameshop-backend/entity/role"
    "github.com/gin-gonic/gin"
)

// GET /permissions

func GetPermissions(c *gin.Context) {
    var permissions []entity.Permission

    if err := configs.DB().Find(&permissions).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, permissions)
}

// GET /permissions/:id

func GetPermissionById(c *gin.Context) {
    id := c.Param("id")

    var permission entity.Permission

```

```

if tx := configs.DB().Where("id = ?", id).First(&permission); tx.RowsAffected == 0 {
    c.JSON(http.StatusNotFound, gin.H{"error": "permission not found"})
    return
}
c.JSON(http.StatusOK, permission)
}

// POST /permissions
func CreatePermission(c *gin.Context) {
    var permission entity.Permission
    if err := c.ShouldBindJSON(&permission); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if err := configs.DB().Create(&permission).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusCreated, permission)
}

// PATCH /permissions/:id
func UpdatePermission(c *gin.Context) {
    id := c.Param("id")
    var permission entity.Permission
    if tx := configs.DB().Where("id = ?", id).First(&permission); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "permission not found"})
        return
    }

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

```

```

if err := configs.DB().Model(&permission).Updates(input).Error; err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}
c.JSON(http.StatusOK, permission)
}

// DELETE /permissions/:id
func DeletePermission(c *gin.Context) {
    id := c.Param("id")
    if tx := configs.DB().Exec("DELETE FROM permissions WHERE id = ?", id); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "permission not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "deleted successfully"})
}

role_permission.go
package controllers

import (
    "net/http"

    "gameshop-backend/configs"
    "gameshop-backend/entity/role"
    "github.com/gin-gonic/gin"
)

// GET /rolepermissions
func GetRolePermissions(c *gin.Context) {
    var rolePermissions []entity.RolePermission
    if err := configs.DB().Find(&rolePermissions).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
}

```

```
        }

        c.JSON(http.StatusOK, rolePermissions)
    }

    // GET /rolepermissions/:id
    func GetRolePermissionById(c *gin.Context) {
        id := c.Param("id")

        var rolePermission entity.RolePermission

        if tx := configs.DB().Where("id = ?", id).First(&rolePermission); tx.RowsAffected == 0 {
            c.JSON(http.StatusNotFound, gin.H{"error": "role_permission not found"})
            return
        }

        c.JSON(http.StatusOK, rolePermission)
    }

    // POST /rolepermissions
    func CreateRolePermission(c *gin.Context) {
        var rolePermission entity.RolePermission

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

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

        c.JSON(http.StatusCreated, rolePermission)
    }

    // PATCH /rolepermissions/:id
    func UpdateRolePermission(c *gin.Context) {
        id := c.Param("id")

        var rolePermission entity.RolePermission

        if tx := configs.DB().Where("id = ?", id).First(&rolePermission); tx.RowsAffected == 0 {
            c.JSON(http.StatusNotFound, gin.H{"error": "role_permission not found"})
        }
    }
}
```

```
        return
    }

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

    if err := configs.DB().Model(&rolePermission).Updates(input).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, rolePermission)
}

// DELETE /rolepermissions/:id
func DeleteRolePermission(c *gin.Context) {
    id := c.Param("id")
    if tx := configs.DB().Exec("DELETE FROM role_permissions WHERE id = ?", id); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "role_permission not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "deleted successfully"})
}
```

## วิเคราะห์ Communication Diagram

เราจะใช้ข้อมูลจาก System Activity Diagram เป็นหลัก

ในการวิเคราะห์เพื่อให้ได้ตารางสำหรับการเขียน Communication Diagram เป็นขั้นตอน ในการวัด

Communication Diagram เราจะมี

เส้นโครง ที่ใช้เชื่อมต่อ เพื่อบอกความเกี่ยวข้องของวัตถุในระบบ

เส้นคำสั่ง จะเป็นเส้นที่มีหัวลูกศร วางเรียงกันอยู่บนเส้นโครง เพื่อบอกการส่งข้อความ (dispatch message) โดยที่หัวลูกศรจะชี้ไปที่วัตถุซึ่งทำงานตามคำสั่งนั้นๆ

เราจะวิเคราะห์ระบบเฉพาะเหตุการณ์หลักของ Use Case ที่ diagram นี้รับผิดชอบ เป็นเส้นทางการทำงานที่เมื่อทำแล้วจะได้ Output ของข้อมูลและ Output ของหน้าจอ ตามที่ระบุไว้ในเอกสาร requirements

จะมีการนำ Class ทั้งหมดที่เคยวิเคราะห์ไว้มาใช้โดย

Boundary Class ทำหน้าที่แทน UI และเราจะใช้ชื่อเบื้องต้นตามชื่อ Use Case หลัก เช่น ในที่นี้จะตั้งชื่อเป็น CreatePermissionUI

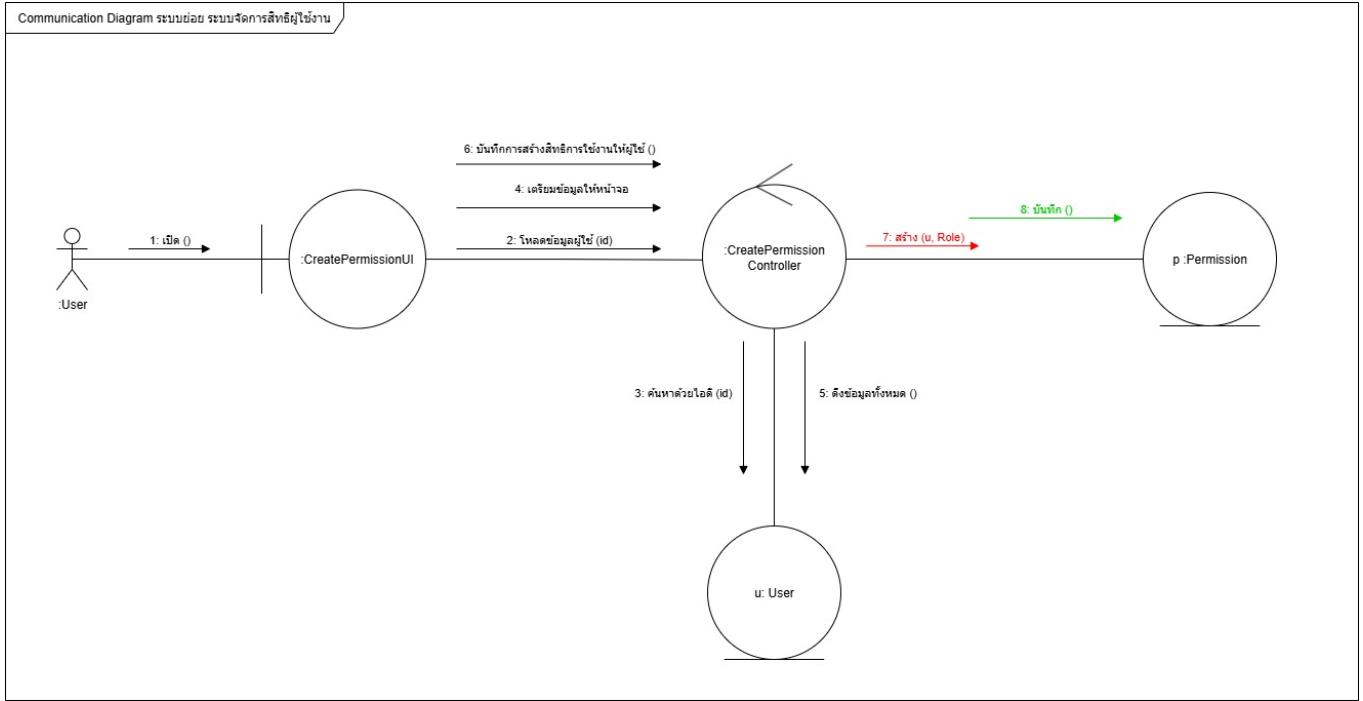
วัตถุของ Boundary Class จะส่งข้อมูลที่จำเป็นให้กับวัตถุของ Control Class

Control Class ทำหน้าที่เป็นตัวประมวลผล ควบคุมการทำงาน และเชื่อมโยงระหว่าง Entity โดยจะตั้งชื่อตาม Use Case หลัก เช่น ในที่นี้จะตั้งชื่อเป็น CreatePermissionController

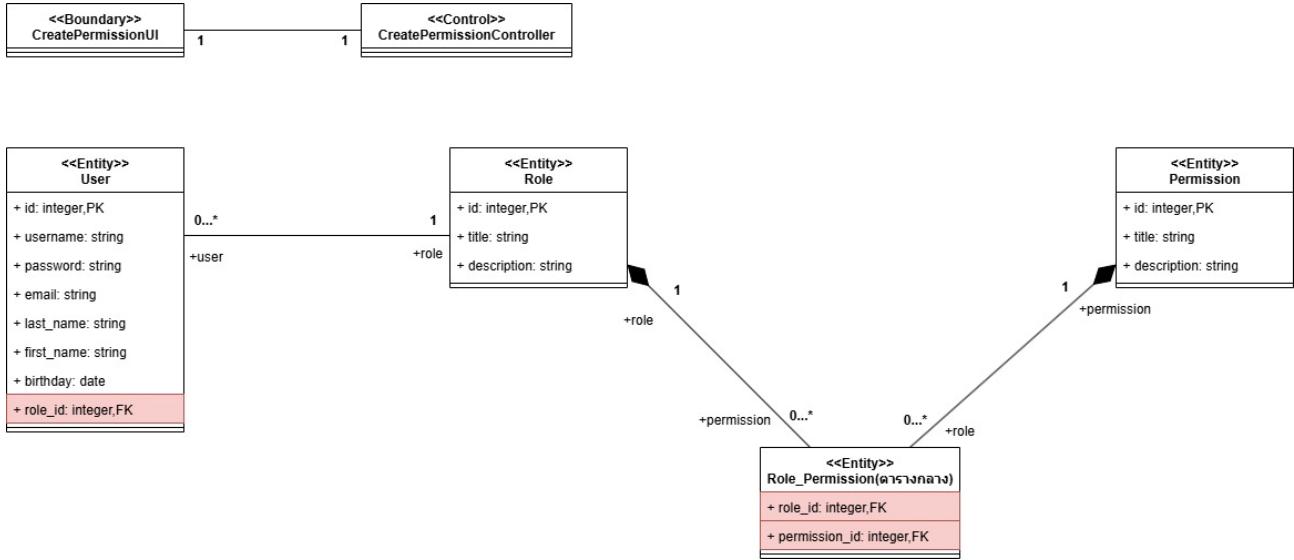
เตรียมแปลงแต่ละ Activity ของ System Activity Diagram ให้เป็น Communication Diagram โดยวิเคราะห์ให้วัตถุที่เกี่ยวข้องกับการทำงานในแต่ละขั้นตอน

Activity	เป็นคำสั่งสำหรับระบบได้ หรือไม่	ถ้าเป็น, วัตถุที่รับหน้าที่ ทำงานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click “เปิดหน้าจอ”	เป็นคำสั่ง สั่งงานเพื่อให้หน้า UI เปิด	:CreatePermissionUI	เปิด ()
เข้าสู่ระบบในฐานะ แอดมิน	ไม่เป็นคำสั่ง	-	-
โหลดข้อมูลแอดมินที่ กำลังใช้งานอยู่	เป็นคำสั่ง เกิดการสั่งงานให้โหลดข้อมูล ผู้ใช้ Role แอดมิน ที่ login อยู่	:CreatePermissionController	โหลดข้อมูลผู้ใช้ (id)
		u : User	ค้นหาด้วย (id)
โหลดข้อมูลรายการ ผู้ใช้งาน	เป็นคำสั่ง เกิดการสั่งงานให้โหลดข้อมูล รายการผู้ใช้งานทั้งหมด	:CreatePermissionController	เตรียมข้อมูลให้หน้าจอ ()
		u : User	ดึงข้อมูลทั้งหมด ()
แสดงหน้าจอสำหรับ สร้างสิทธิการใช้งาน ให้ผู้ใช้	ไม่เป็นคำสั่ง	-	-
Click เลือกตัวเลือก เพื่อกำหนดสิทธิให้ ผู้ใช้งานแต่ละคน	ไม่เป็นคำสั่ง	-	-
Click ปุ่มบันทึก ข้อมูล	เป็นคำสั่ง ระบบจะทำการจัดเก็บข้อมูล การกำหนดสิทธิ	:CreatePermissionController	บันทึกการสร้างสิทธิการใช้งานให้ผู้ใช้ (permission)
ระบบบันทึกการให้ สิทธิ	ไม่เป็นคำสั่ง	-	-
สร้างข้อมูล entity Permission โดยโยง entity User, โยง entity Role	เป็นคำสั่ง	b: Permission	สร้าง (u, Role)
บันทึก entity Permission	เป็นคำสั่ง	b: Permission	บันทึก ()

แสดงข้อความว่า “สร้างข้อมูลสำเร็จ”	ไม่เป็นคำสั่ง	-	-
---------------------------------------	---------------	---	---



## Class Diagram at Design Level



B6639273 นายปุณณพัฒน์ เกษหอม

ระบบหลัก ระบบร้านขายเกมออนไลน์

ระบบย่อย ระบบสร้างคอนเทนต์เกมใน workshop

ระบบบริการขายเกมออนไลน์ ผู้ใช้งาน (User) จะต้องลงทะเบียนก่อนจึงจะเข้าใช้งานได้ เมื่อลงทะเบียนเสร็จก็จะต้องทำการ Log in เข้ามาใช้งาน เพื่อซื้อเกมต่างๆ และเมื่อเลือกเกมที่อยากซื้อได้แล้วก็จะต้องทำการชำระเงิน เมื่อชำระเงินสำเร็จทางเราจะจะส่งคีย์เกมให้ลูกค้า

ส่วนระบบสร้างคอนเทนต์เกมใน Workshop เป็นระบบที่เปิดให้ผู้ใช้งาน (User) สามารถสร้างและเผยแพร่เนื้อหา เสิร์ฟเวอร์เกม หรือที่เรียกว่า Mod (Modification) ได้ โดยผู้ใช้งานจะต้องทำการ ลงทะเบียน (Register) และ เข้าสู่ระบบ (Log in) ก่อนจึงจะสามารถใช้งานระบบ Workshop ได้ เมื่อเข้าสู่ระบบเรียบร้อยแล้ว ผู้ใช้งานสามารถเลือกเกมที่ตนเองเป็นเจ้าของ (Owner\_Game) เพื่อเริ่มต้นสร้าง Mod ได้ หลังจากนั้นผู้ใช้งานจะสามารถทำการ อัปโหลด Mod (Upload Mod) ขึ้นไปยัง Workshop ได้ โดยจะต้องกรอกข้อมูลรายละเอียดต่างๆ ของ Mod เช่น ชื่อ, คำอธิบาย และอัปโหลดไฟล์ที่เกี่ยวข้องกับ Mod นั้น ผู้ใช้งานสามารถแก้ไข (Update) ข้อมูลต่างๆ ของ Mod ได้ แล้วก็สามารถลบ (Delete) Mod นั้นออกได้เมื่อไม่ต้องการ Mod นั้นแล้วหรือไม่ต้องการเผยแพร่ Mod นั้นแล้ว เพื่อให้ Mod ที่อัปโหลดไปนั้นสามารถค้นหาและเข้าถึงได้ง่ายขึ้น ระบบจะเปิดให้ผู้ใช้งานเลือก Tags (ป้ายกำกับ) ที่เกี่ยวข้องกับ Mod ของตนเอง โดย 1 Mod สามารถมีได้หลาย Tags

นอกจากนี้ เมื่อมีผู้ใช้งานคนอื่น ๆ ดาวน์โหลดและทดลองใช้ Mod แล้ว พวกราสามารถกลับมา ให้คะแนน (Rating) และเขียน รีวิว เพื่อแสดงความคิดเห็นเกี่ยวกับ Mod นั้น ๆ ได้ โดยสามารถให้คะแนนได้ตั้งแต่ 1-5 คะแนน ซึ่งจะช่วยสร้างความน่าเชื่อถือและส่งเสริมให้ผู้อื่นตัดสินใจดาวน์โหลด Mod ได้ง่ายขึ้น

## User Story ระบบสร้างคอนเทนต์เกมใน workshop

ในบทบาทของ (As a) ผู้ใช้งาน (User)

ฉันต้องการ (I want to) อัปโหลดมอดเกมลงใน workshop

เพื่อ (So that) เพยแพร์มอดเกมที่ฉันสร้างให้ผู้ใช้งานคนอื่นๆได้อ่านได้

### Output บนหน้าจอ

- ผู้ใช้งาน (User) กดอัปโหลดมอดเกม -> กรอกรายละเอียดข้อมูลของมอดเกม -> เพิ่มไฟล์ของมอดเกม  
-> กดบันทึกข้อมูล จากนั้นระบบจะทำการบันทึกข้อมูล และระบบจะแสดง indicator บางอย่างที่แสดงให้เห็นว่าได้บันทึกข้อมูลเรียบร้อยแล้ว

### Output ของข้อมูล

- ระบบจะอัปโหลดมอดเกมลงใน workshop -> ระบบจะบันทึกข้อมูลของมอดเกมที่เราอัปโหลดลงในฐานข้อมูล

## คำนามที่อาจจะถูกนำมาเป็นตารางในฐานข้อมูลของระบบ

คำนาม	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
ผู้ใช้งาน (User)	เกี่ยวข้องโดยตรง เนื่องจากเป็นสิ่งที่แอดมินต้องสร้าง, อัปเดต หรือ ลบ
เกม	เกี่ยวข้องโดยตรง เนื่องจากต้องอัปโหลดมอดูลของเกมนั้นๆ
Mod	เกี่ยวข้องโดยตรง เนื่องจากเป็นสิ่งที่จะอัปโหลดลง workshop
เกมที่ตนเองเป็นเจ้าของ (Owner_game)	เกี่ยวข้องโดยตรง เนื่องจากก่อนจะอัปโหลดมอดของเกมนั้นๆได้ จำเป็นต้องมีเกมนั้นก่อน
คะแนน	เกี่ยวข้องโดยตรง เนื่องจากระบบสามารถให้คะแนนได้
Tags	เกี่ยวข้องโดยตรง เนื่องจากเป็นป้ายกำกับให้ผู้ใช้งานค้นหาได้ง่าย
การชำระเงิน	ไม่เกี่ยวข้องโดยตรง
คีย์เกม	ไม่เกี่ยวข้องโดยตรง

## Business Use Case Diagram (แบบเดี่ยว)



### Checklist: Business Use Case Diagram

Business Actor มี <<Business>> กำกับ

Business Actor เป็นชื่อบทบาทเท่านั้น ไม่ใช่ชื่อคน ไม่เป็นชื่อบุคคล

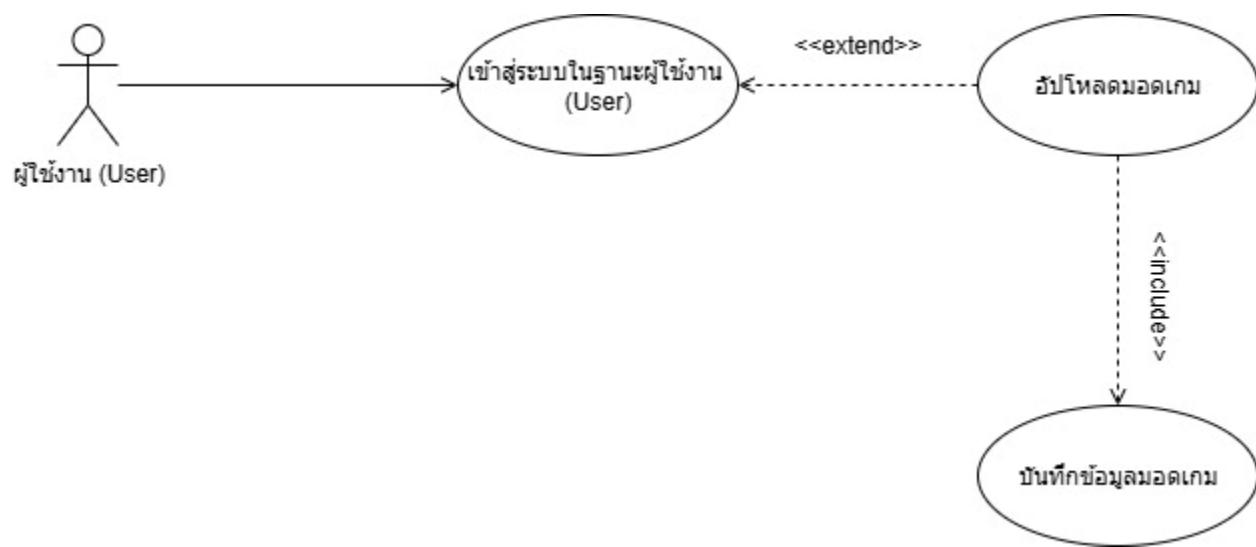
Business Use Case มี <<Business>> กำกับ

Business Use Case ขึ้นต้นด้วยคำที่แสดงพฤติกรรม “คำกริยา”

พฤติกรรมของ Business Use Case ต้องมาจาก User Story

มี Note ที่แสดง User Story อญญาติใน Diagram

## System Use Case Diagram (แบบเดี่ยว)



### ขั้นตอนการแปลง Business Use Case Diagram ให้เป็น System Use Case Diagram

#### พิจารณาประเด็นที่ 1

Business Actor "ผู้ใช้งาน (User)" สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์ที่ระบบงานได้โดยตรงหรือไม่

ตอบ ได้ ผู้ใช้งาน สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์

เพราะฉะนั้น ผู้ใช้งาน (User) จะถูกจัดเป็น System Actor ได้

## พิจารณาประเด็นที่ 2

Business Use Case “อัปโหลดมอดเกม” สามารถถูกลายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบ

คอมพิวเตอร์ได้หรือไม่ต้องมีขั้นตอนทาง Security มาเกี่ยวหรือไม่

ตอบ Business Use Case “อัปโหลดมอดเกม” สามารถถูกลายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบ

คอมพิวเตอร์ได้และต้องการขั้นตอนทาง Security (มี Privacy มาเกี่ยวข้อง)

เพราะฉะนั้น Business Use Case “อัปโหลดมอดเกม” จะถูกลายเป็น System Use Case มากกว่า 1

Use Case

จะประกอบไปด้วย

- 1 System Use Case สำหรับ “เข้าระบบในฐานะผู้ใช้งาน (User)”
- 2 System Use Case สำหรับ “อัปโหลดมอดเกม” แต่กิจกรรมของระบบ จะยังไม่ครบตาม Requirements ที่ต้องการ
- 3 System Use Case สำหรับ “บันทึกข้อมูลมอดเกม”
  - System Use Case สำหรับ “เข้าระบบในฐานะผู้ใช้งาน (User)”
  - System Use Case สำหรับ “อัปโหลดมอดเกม”
  - System Use Case สำหรับ “บันทึกข้อมูลมอดเกม”

## พิจารณาประเด็นที่ 3

ถ้าสมาชิก “เข้าระบบในฐานะผู้ใช้งาน (User)” มาแล้วจำเป็นที่ต้อง “อัปโหลดมอดเกม” ทุกครั้งหรือไม่

ตอบ ไม่

แปลว่า System Use Case “อัปโหลดมอดเกม” เป็นกิจกรรมทางเลือกหลังจากทำ System Use Case

“เข้าระบบในฐานะผู้ใช้งาน (User)”

## พิจารณาประเด็นที่ 4

ถ้าสมาชิก “เข้าระบบในฐานะผู้ใช้งาน” มาแล้วจำเป็นต้อง “บันทึกข้อมูลอดเกม” ทุกครั้งหรือไม่  
ตอบ ไม่ และเป็น Use Case ที่ไม่เกี่ยวข้องกัน

## พิจารณาประเด็นที่ 5

ถ้าสมาชิก “อัปโหลดมอดเกม” และ<sup>จะ</sup>  
จำเป็นต้อง “บันทึกข้อมูลมอดเกม” ทุกครั้งหรือไม่  
ตอบ ใช่ จำเป็น

แปลว่า System Use Case “อัปโหลดมอดเกม” จะต้องรวมขึ้นตอนของ System Use Case  
“บันทึกข้อมูลมอดเกม” ไว้ด้วยเสมอ

### Checklist: System Use Case Diagram

- 1 System Actor และ System Use Case ต้องไม่มีอะไรกำกับ
- 2 เส้นโยงจาก System Actor ไปหา System Use Case จะต้องเป็นเส้นทึบ หัวลูกศรปลายเปิดซึ่งไปหา System Use Case
- 3 System Actor ต้องเป็นบทบาทของคนที่ใช้ระบบจริง ๆ
- 4 System Use Case ต้องเป็นคำที่แสดงพฤติกรรม
- 5 ถ้ามีกลไกทาง Security มาเกี่ยว ชื่อ System Use Case จะอยู่ในรูปแบบ “เข้าระบบในฐานะ <Actor>” เท่านั้น
- 6 การใช้ <--<<extend>>--> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ทางเดียวไปยัง System Use Case หลัก

7 การใช้ <--<<include>>--> ต้องเป็นส่วนประ ทวัลุกศรปลายเปิด ซึ่งจาก System Use Case ที่ใหญ่กว่า ไปยัง System Use Case ที่จะถูกรวบเข้ามา

### Checklist สำหรับการทวนสอบความถูกต้องของ Requirements

- 1 งาน (ระบบย่อยของแต่ละคน) ต้องไม่ซ้ำกับเพื่อนในกลุ่ม
- 2 งาน (ระบบย่อยของแต่ละคน) ต้องสอดคล้องกับ Business Domain ของระบบหลัก
- 3 ระบบย่อยของแต่ละคน ต้องเชื่อมโยงและต่อเนื่องกับระบบย่อยของคนอื่นๆ ในทีม ไม่สามารถเป็นงานเดียวที่ไม่เกี่ยวข้องกับใครเลยได้
- 4 Entity (ตาราง) ที่ออกแบบ ต้องประกอบด้วย Entity หลัก 1 ตาราง และมีความสัมพันธ์กับ Entity อื่นๆ อย่างน้อย 3 ตาราง กล่าวคือ ต้องมีความสัมพันธ์ (Relation) ระหว่างตารางอย่างน้อย 3 เส้น
- 5 ใน Entity หลัก ต้องมีคอลัมน์ (ฟิลด์) ที่ใช้เก็บข้อมูลซึ่งหมายความตามลักษณะของระบบย่อยที่ออกแบบ

### การจำลองตัวอย่างตารางและข้อมูล (เพื่อใช้ในการเตรียมร่าง User Interface, System Activity Diagram และ Class Diagram)

จาก Requirements, จาก ข้อมูล Output และ Entity ที่ได้ออกแบบไว้เราจะนำมาสมมติเป็นตารางในฐานข้อมูล โดยกำหนด Primary Key เป็นตัวเลขรวมถึงการสมมติ Foreign Key เพื่อเชื่อมโยงข้อมูลระหว่างตาราง ซึ่งจะช่วยให้สามารถออกแบบและวิเคราะห์ระบบได้อย่างมีโครงสร้างและชัดเจน

## วิเคราะห์ Entity ที่เกี่ยวข้อง

### ผู้ใช้งาน (User)

- ชื่อ Entity: User
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Username: เก็บในรูปแบบ varchar ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Password: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Email: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - First Name: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Last Name: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Birthday: เก็บในรูปแบบ date ไม่สามารถเป็นค่า Null ได้

ID INTEG ER NOT NULL, PK	USERNA ME VERCHAR NOT NULL, NULL, Unique	PASSWORD VARCHAR NOT NULL	EMAIL VARCHAR NOT NULL	FIRST_NA ME VARCHAR NOT NULL	LAST_NA ME VARCHAR NOT NULL	BIRTHDAY DATE NOT NULL
1001	User01	Encrypt(1234 56)	Demo01@gmai.lc om	SA	68	10-08-2000
1002	User02	Encrypt(1234 56)	demo02@gmai.lc om	SE	69	25-12-2000

## เกม

- ชื่อ Entity: Game
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Title: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Description: เก็บในรูปแบบ varchar สามารถเป็นค่า Null ได้

ID INTEGER NOT NULL, PK	TITLE VARCHAR NOT NULL	DESCRIPTION VARCHAR NULL
2001	Counterstrike	เกมประเภท fps ยิงปืน ต่อสู้ ผู้เล่นหลายคน

## เกมที่ตนของเป็นเจ้าของ (Owner\_game)

- ชื่อ Entity: Owner\_Game
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Purchase\_date: เก็บในรูปแบบ date ไม่สามารถเป็นค่า Null ได้
  - status: เก็บในรูปแบบ varchar ไม่สามารถเป็นค่า Null ได้
  - User\_ID: เก็บในรูปแบบ integer ไม่สามารถเป็นค่า Null ได้
  - Game\_ID: เก็บในรูปแบบ integer ไม่สามารถเป็นค่า Null ได้

ID INTEGER NOT NULL, <span style="background-color: cyan;">PK</span>	PURCHASE_DATE DATE NOT NULL	STATUS VARCHAR NOT NULL	USER_ID DATE NOT NULL, <span style="background-color: cyan;">FK</span>	GAME_ID INTEGER NOT NULL, <span style="background-color: cyan;">FK</span>
3001	10-10-2568	เป็นเจ้าของเกม	<span style="color: red;">1001</span>	<span style="color: red;">2001</span>

## Mod

- ชื่อ Entity: Mod
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - title: เก็บในรูปแบบ date ไม่สามารถเป็นค่า Null ได้
  - description: เก็บในรูปแบบ varchar ไม่สามารถเป็นค่า Null ได้
  - upload\_date: เก็บในรูปแบบ date ไม่สามารถเป็นค่า Null ได้
  - file\_path: เก็บในรูปแบบ varchar ไม่สามารถเป็นค่า Null ได้
  - status: เก็บในรูปแบบ varchar ไม่สามารถเป็นค่า Null ได้
  - User\_ID: เก็บในรูปแบบ integer ไม่สามารถเป็นค่า Null ได้
  - Gamr\_ID: เก็บในรูปแบบ integer ไม่สามารถเป็นค่า Null ได้
  - Tags: เก็บในรูปแบบความสัมพันธ์แบบ Many-to-Many โดยมีการสร้างของตารางดังนี้
    - Mod\_ID: เป็น Primary Key เก็บในรูปแบบของ integer และต้องไม่เป็นค่า Null
    - Tags\_ID: เป็น Primary Key เก็บในรูปแบบของ integer และต้องไม่เป็นค่า Null

ID INTEG ER NOT NULL, <b>PK</b>	TITLE VARCH AR NOT NULL	DESCRIPTI ON VARCHAR NOT NULL	UPLOAD_D ATE DATE NOT NULL	FILE_PATH VARCHAR NOT NULL	STATU S VARCH AR NOTNU LL	USER_ ID INTEG ER NOT NULL, <b>FK</b>	GAME_ ID INTEGE R NOT NULL, <b>FK</b>
4001	funfun	เป็นมอดเกม ที่ให้ผู้เล่นได้ เล่นค่าน ใหม่ๆ	20-10-2568	C:\User\acer\Docu ments	Publis ed	1001	2001

MOD_ID INTEGER NOT NULL, <b>FK</b>	TAGS_ID INTEGER NOT NULL, FK
4001	5001
4001	5002

## Tags

- ชื่อ Entity: Tags
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Title: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้

ID INTEGER NOT NULL, PK	TITLE VARCHAR NOT NULL
5001	Map
5002	Weapon

## คะแนน

- ชื่อ Entity: ModRating
- ข้อมูลที่ต้องจัดเก็บ
  - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - rated\_date: เก็บในรูปแบบ date ไม่สามารถเป็นค่า Null ได้
  - review: เก็บในรูปแบบ varchar สามารถเป็นค่า Null ได้
  - rating: เก็บในรูปแบบ integer สามารถเป็นค่า Null ได้
  - User\_ID: เก็บในรูปแบบ integer ไม่สามารถเป็นค่า Null ได้
  - Mod\_ID: เก็บในรูปแบบ integer ไม่สามารถเป็นค่า Null ได้

ID INTEGER NOT NULL, PK	RATED_DATE DATE NOT NULL	REVIEW VARCHAR NULL	RATING INTEGER NULL	USER_ID DATE NOT NULL, FK	GAME_ID INTEGER NOT NULL, FK
6001	30-10-2568	เป็นมอดที่เยี่ยมไป เลย	5	1001	2001

## หลักการออกแบบ User Interface

- เมื่อได้กีตามที่มี Foreign Key (FK) ซึ่งจาก ตารางหลักกลับไปยัง ตารางสนับสนุน  
ให้ใช้ ComboBox เป็นตัวเลือกเพื่อเชื่อมโยงข้อมูลใน User Interface
- สำหรับ Field ประเภทอื่นๆ ให้สร้างตามประเภทของข้อมูล เช่น
  - Textbox สำหรับข้อความทั่วไป
  - Password สำหรับข้อมูลรหัสผ่าน
  - Datetime Picker สำหรับเลือกวันและเวลา
  - Numeric Input สำหรับป้อนตัวเลข

## UI Design

The wireframe illustrates a mobile application interface for uploading game mods. The screen is divided into two main sections: a sidebar on the left and a main content area on the right.

**Left Sidebar:** A purple vertical bar containing a user icon and the text "User01".

**Right Content Area:**

- Upload Game Mod Section:** Contains fields for "ชื่อ\*" (Name\*) and "กรุณากรอกชื่อ Mod" (Please enter Mod name).
  - Icon: Person icon
  - Text: "ชื่อ\*กรุณากรอกชื่อ Mod"
- Select your mods file Section:** Contains a "Select" button and a file input field.
  - Icon: Select icon
  - Text: "Select your mods file"
  - Buttons: "เลือกไฟล์" (Select file)
- Add a description Section:** Contains a text input field.
  - Text: "Add a description"
  - Text: "ใส่คำอธิบาย" (Enter description)
- Bottom Right:** A blue "Upload" button.

## การเตรียม System Activity Diagram

- 1 Business Use Case (1 User Story) จะถูกแปลงเป็น 1 System Activity Diagram
- System Activity Diagram คือ การบรรยายลำดับของกิจกรรม (Activity) ระหว่าง ผู้ใช้ (คน) กับ ระบบ

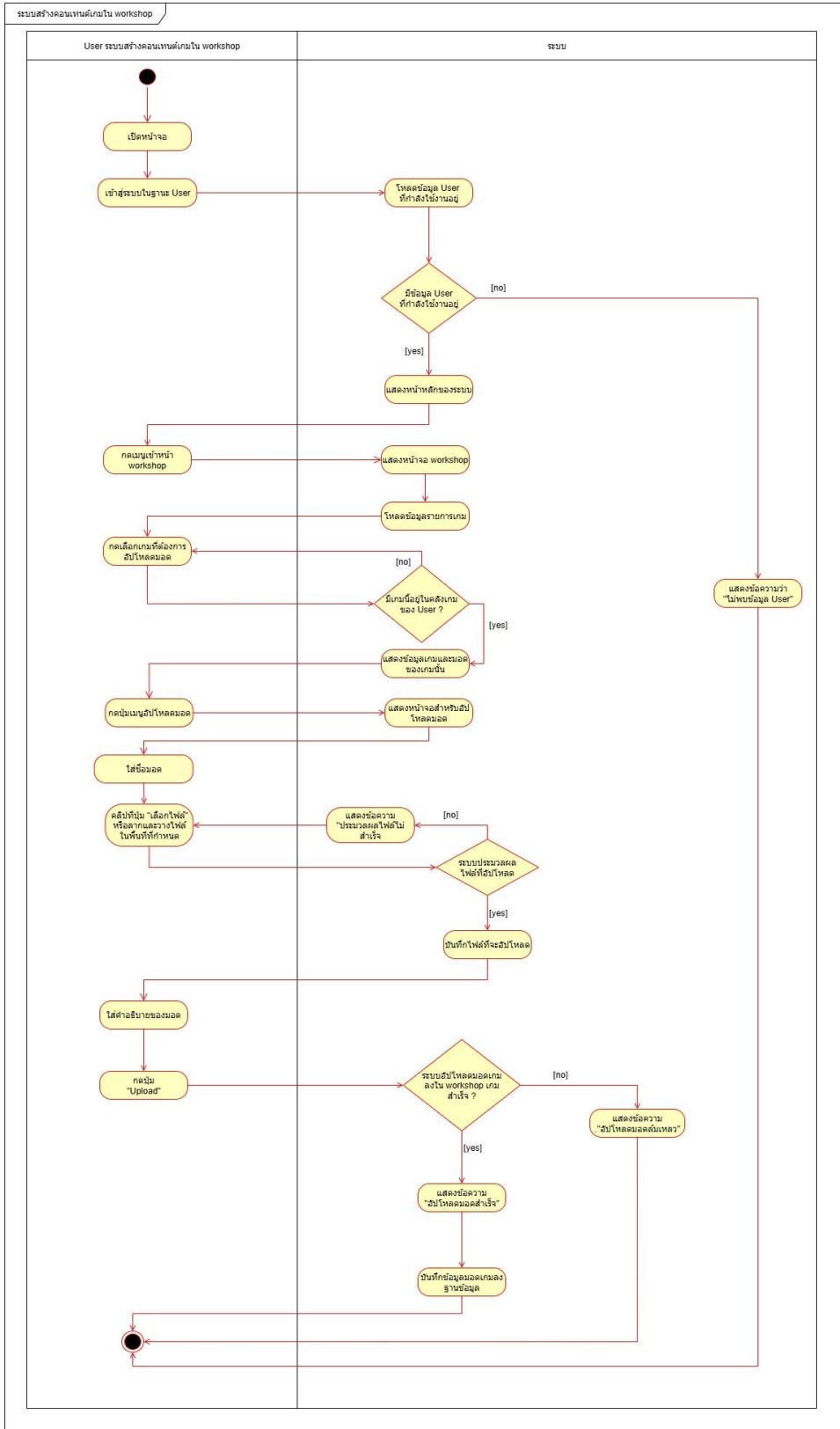
## หลักสำคัญ ของการออกแบบ System Activity Diagram ได้แก่

- ระบุว่า ผู้ใช้ทำอะไร (Input)
- ระบบประมวลผลอะไร (Process)
- ระบบตอบกลับอะไร (Output)

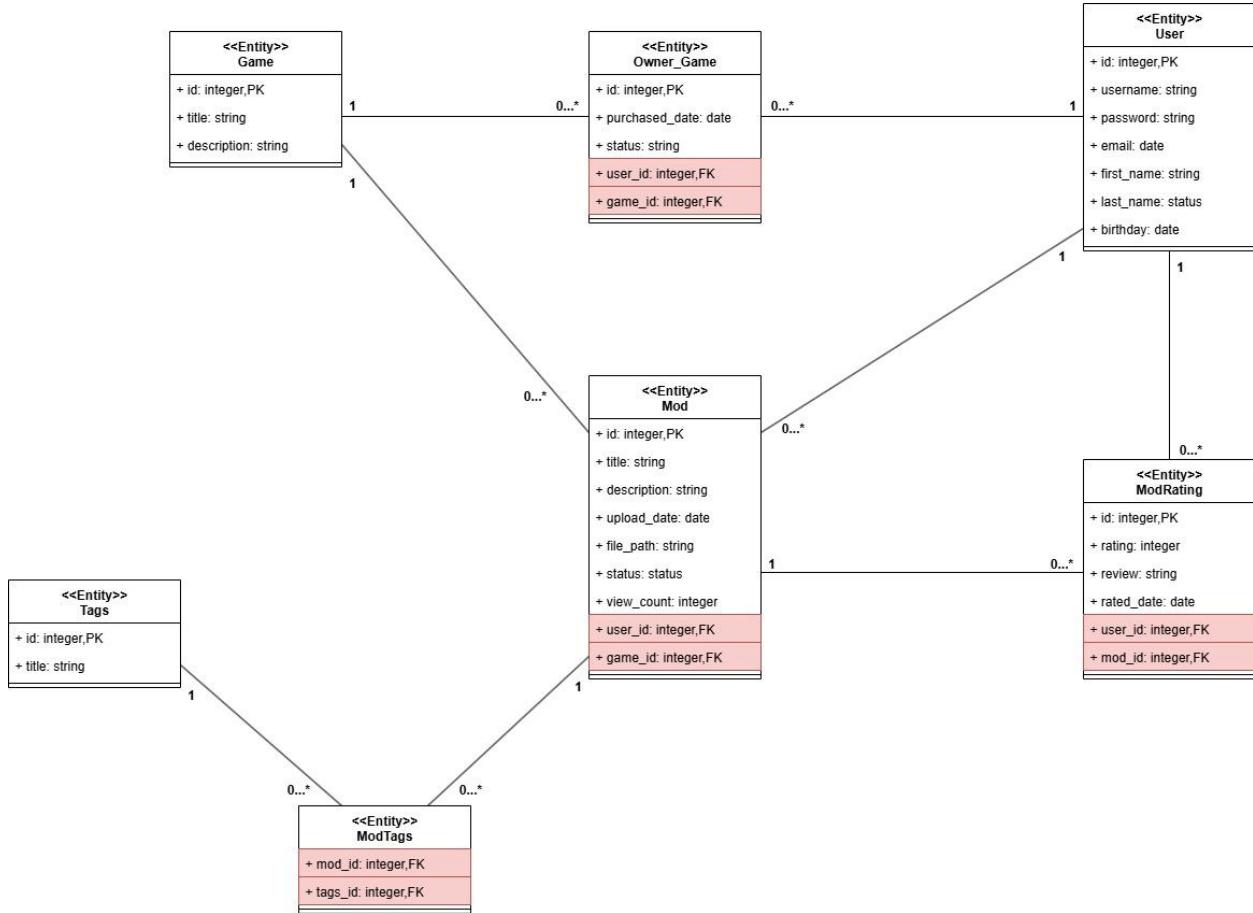
## System Activity Diagram ต้องสื่นสุดที่การสร้าง Output ดังนี้

1. บันทึกข้อมูลลงใน ฐานข้อมูล
2. แสดงผลข้อมูลผ่าน หน้าจอ (User Interface) โดยต้องสอดคล้องกับ User Story ที่กำหนดไว้

## Activity Diagram

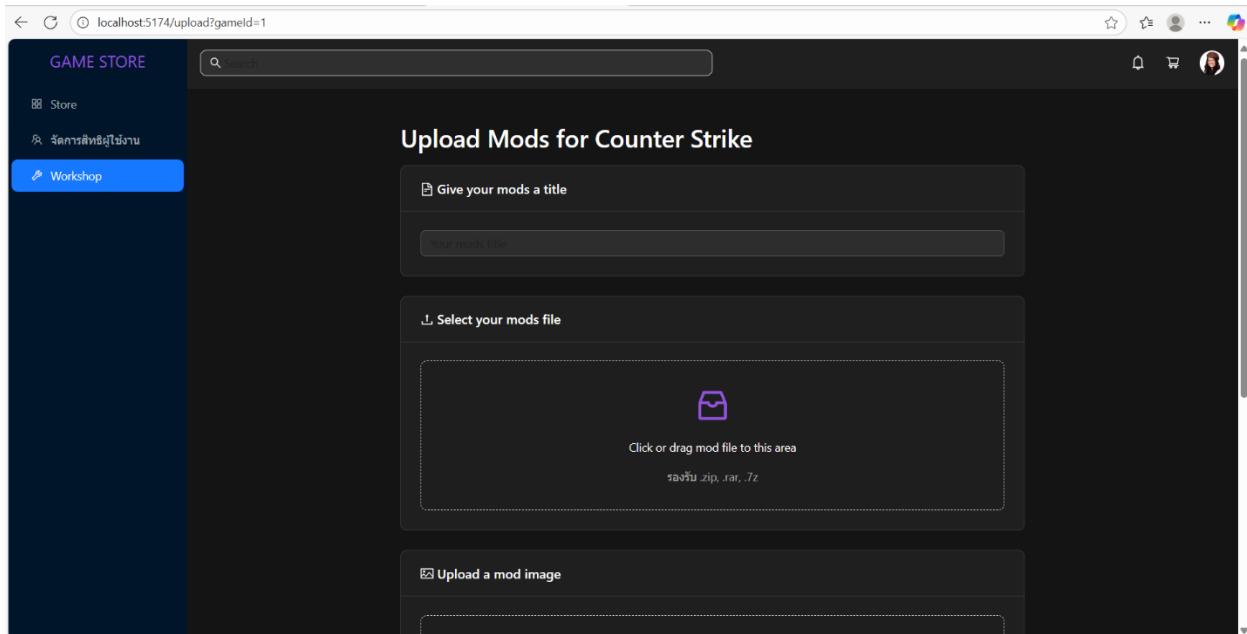


## Class Diagram at Analysis Level

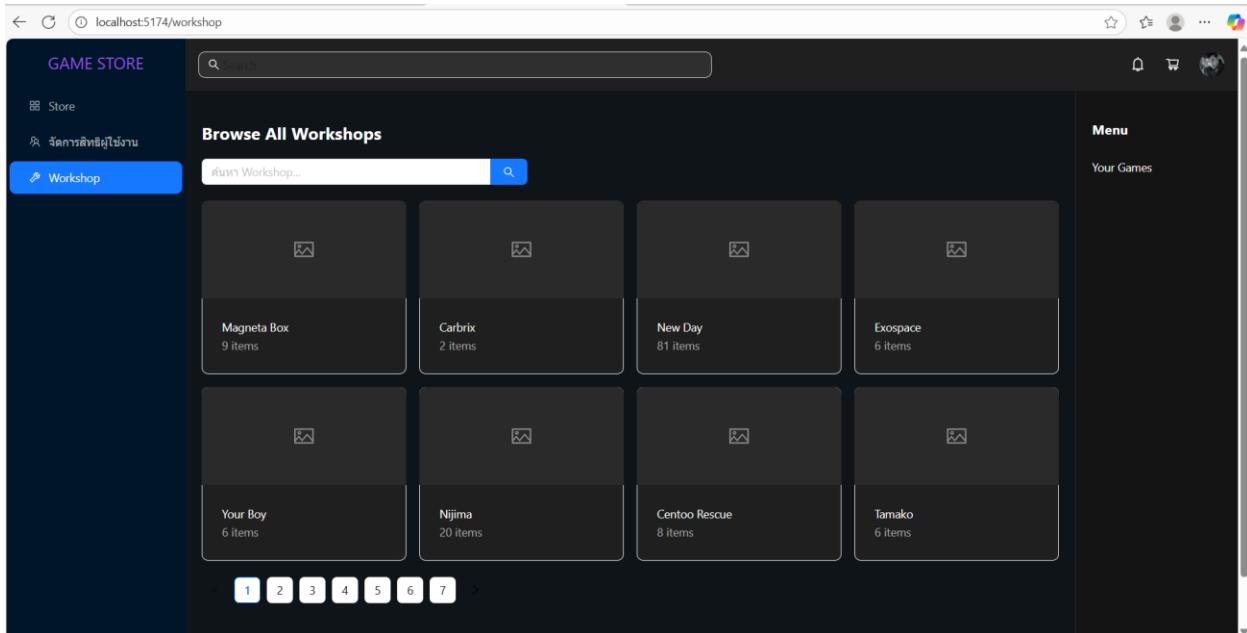


## UI

### WorkshopUpload Page



### WorkshopMain Page



## WorkshopDetail Page

The screenshot shows a web browser window with the URL `localhost:5174/workshop/1`. The page has a dark theme with a sidebar on the left labeled "GAME STORE" containing "Store", "จัดการสิ่งที่คุณชื่นชอบ", and "Workshop". The main content area displays a banner for "Magna Box" and a list titled "Magna Box – 9 items" showing 1–6 of 6 entries. The mods listed are "Weapons Course" by Asmisint, "CS:CTF Double Cross" by CS:CTF, "CS:CTF 2Fort" by CS:CTF, and "Mocha" by Bevster. A search bar and a "Search mods..." button are visible. On the right, there's a sidebar with "SHOW:" dropdowns for "All" and "Your Favorites", and a blue "Upload Mod" button.

## ModDetail Page

The screenshot shows a web browser window with the URL `localhost:5174/mod/m1`. The layout is similar to the WorkshopDetail page. The sidebar shows "Workshop" is selected. The main content features a large image of a game map titled "Weapons Course". Below the image, a "Download" button is present. A placeholder text "This is a placeholder description for the mod. In the future, you can load real mod details from the backend." is displayed. A "Comments" section with a text input field and "Add Comment" button follows. To the right, a sidebar provides details about the creator: "Creator: Asmisint" and "Uploaded: 2025-09-05". It also shows statistics: "0 Unique Visitors" and "0 Current Downloads". A "Rate this Mod" section includes a 5-star rating and "Your Rating: 4 / 5".

## SourceCode

```
WorkshopUpload Page
// src/pages/Workshop/UploadPage.tsx

import React, { useState, useEffect, useRef } from "react";
import { Typography, Card, Input, Button, Upload, message, Space } from "antd";
import {
  UploadOutlined,
  FileTextOutlined,
  PictureOutlined,
  InboxOutlined,
  CheckCircleFilled,
} from "@ant-design/icons";
import { useSearchParams, useNavigate } from "react-router-dom";
import { getGame, getMod, createMod, updateMod, listUserGames } from "../../services/workshop";
import type { Game } from "../../interfaces";
import { useAuth } from "../../context/AuthContext";

const { Title } = Typography;
const { Dragger } = Upload;

const Ok: React.FC<{ ok: boolean }> = ({ ok }) => (
  <span style={{ color: ok ? "#52c41a" : "#ff4d4f", fontWeight: 600 }}>
    {ok ? " ✓ " : " ✗ "}
  </span>
);

const UploadPage: React.FC = () => {
  const navigate = useNavigate();
  const [searchParams] = useSearchParams();
  const gameldParam = searchParams.get("gameld");
  const modIdParam = searchParams.get("modId");
  const gameld = gameldParam ? Number(gameldParam) : undefined;
  const modId = modIdParam ? Number(modIdParam) : undefined;
```

```

const { id: userId, token } = useAuth() as { id?: number; token?: string };

const [game, setGame] = useState<Game | null>(null);
const isEditing = gameId !== undefined;

// เช็คความเป็นเจ้าของเกม
const [checkingOwn, setCheckingOwn] = useState(false);
const [ownsThisGame, setOwnsThisGame] = useState(false);
const [myUserGameId, setMyUserGameId] = useState<number | undefined>(undefined);

useEffect(() => {
  if (gameId) {
    getGame(gameId)
      .then(setGame)
      .catch((e) => {
        console.error("[getGame] failed:", e);
        message.error(e?.message || "โหลดข้อมูลเกมไม่สำเร็จ");
      });
  }
}, [gameId]);

// โหลด user-games ของผู้ใช้ ⇒ ใช้ยืนยันว่าเป็นเจ้าของเกมนี้ไหม
useEffect(() => {
  if (!userId || !gameId) {
    setCheckingOwn(false);
    setOwnsThisGame(false);
    setMyUserGameId(undefined);
    return;
  }
  setCheckingOwn(true);
  listUserGames(userId)
    .then((rows: any[]) => {
      const match = (rows ?? []).find(
        (r: any) => Number(r.game_id ?? r.gameId ?? r.GameID) === Number(gameId)
      );
    });
});

```

```

);
setOwnsThisGame(!!match);
const ugid = match
? Number(match.user_game_id ?? match.UserGameID ?? match.id ?? match.ID)
: undefined;
setMyUserGameId(Number.isFinite(ugid) ? ugid : undefined);
})
.catch((e) => {
console.warn("listUserGames failed:", e);
setOwnsThisGame(false);
setMyUserGameId(undefined);
})
.finally(() => setCheckingOwn(false));
}, [userId, gameId]);

useEffect(() => {
if (modId) {
getMod(modId)
.then((m: any) => {
setModTitle(m?.title ?? "");
setModDescription(m?.description ?? "");
const ugid = m?.user_game_id ?? m?.userGameId ?? m?.UserGameID ?? undefined;
setUserGameId(typeof ugid === "number" ? ugid : (ugid != null ? Number(ugid) : undefined));
})
.catch((e) => {
console.error("[getMod] failed:", e);
message.error(e?.message || "ໂທລດຂໍ້ມູນມີອຳໄມສໍາເລັງ");
});
}
}, [modId]);

// Form states
const [modTitle, setModTitle] = useState("");
const [modDescription, setModDescription] = useState("");
const [modFile, setModFile] = useState<File | null>(null);

```

```
const [modImage, setModImage] = useState<File | null>(null);
const [imagePreview, setImagePreview] = useState<string>("");
const [uploading, setUploading] = useState(false);

// success banner
const [showSuccess, setShowSuccess] = useState(false);
const redirectTimer = useRef<number | undefined>(undefined);

// (compat) สำหรับแก้ไข
const [userGameId, setUserGameId] = useState<number | undefined>(undefined);

// --- file handlers ---
const handleModFile = (file: File) => {
  setModFile(file);
  message.success(`.${file.name} selected`);
  return false;
};

const handleModFileChange: React.ComponentProps<typeof Upload>["onChange"] = (info) => {
  const f = info.file?.originFileObj as File | undefined;
  if (f) setModFile(f);
};

const handleModFileRemove = () => setModFile(null);

const handleModImage = (file: File) => {
  setModImage(file);
  setImagePreview(URL.createObjectURL(file));
  message.success(`.${file.name} selected`);
  return false;
};

const handleModImageChange: React.ComponentProps<typeof Upload>["onChange"] = (info) => {
  const f = info.file?.originFileObj as File | undefined;
  if (f) {
    setModImage(f);
    setImagePreview(URL.createObjectURL(f));
  }
}
```

```
};

const handleModImageRemove = () => {
    setModImage(null);
    setImagePreview("");
};

const handleUpload = async () => {
    const toastKey = "modUpload";
    try {
        if (!gameId) {
            message.error("ไม่พบรหัสเกม");
            return;
        }
        if (!userId) {
            message.error("กรุณาเข้าสู่ระบบก่อนอัปโหลดมือด");
            return;
        }
        if (checkingOwn) {
            message.loading({ content: "กำลังตรวจสอบสิทธิ...", duration: 1 });
            return;
        }
        if (!ownsThisGame) {
            message.error("ต้องเป็นเจ้าของเกมนี้ก่อนจึงจะอัปโหลดมือดได้");
            return;
        }
        if (!modTitle.trim()) {
            message.error("กรุณาใส่ชื่อมือด");
            return;
        }
        if (!token) {
            message.error("คุณไม่มีสิทธิ์อัปโหลด (token ไม่พบ) — กรุณาเข้าสู่ระบบใหม่");
            return;
        }
    }
```

```
if (!isEditing) {
    if (!modFile) {
        message.error("กรุณาเลือกไฟล์มือด");
        return;
    }

    setUploading(true);
    message.open({ key: toastKey, type: "loading", content: "กำลังอัปโหลด...", duration: 0 });

    const fd = new FormData();
    fd.append("title", modTitle);
    fd.append("description", modDescription);
    fd.append("game_id", String(gameld));
    if (myUserGameld != null) fd.append("user_game_id", String(myUserGameld));
    fd.append("file", modFile);
    if (modImage) fd.append("image", modImage);

    await createMod(fd, token, userId);
    message.open({ key: toastKey, type: "success", content: "อัปโหลดเรียบร้อยแล้ว!", duration: 0.8 });

    setShowSuccess(true);
    window.scrollTo({ top: 0, behavior: "smooth" });
    redirectTimer.current = window.setTimeout(() => {
        navigate(`/workshop/${gameld}`, { replace: true });
    }, 1400);

    setModTitle("");
    setModDescription("");
    setModFile(null);
    setModImage(null);
    setImagePreview("");
} else {
    setUploading(true);
    message.open({ key: toastKey, type: "loading", content: "กำลังบันทึก...", duration: 0 });
```

```
const payload: any = { title: modTitle, description: modDescription, game_id: gameId };
if (userGameId != null) payload.user_game_id = userGameId;

await updateMod(modId!, payload, token, userId);

message.open({ key: toastKey, type: "success", content: "บันทึกเรียบร้อยแล้ว!", duration: 0.8 });

setShowSuccess(true);
window.scrollTo({ top: 0, behavior: "smooth" });
redirectTimer.current = window.setTimeout(() => {
  navigate(`/workshop/${gameId}`, { replace: true });
}, 1200);
}

} catch (err: any) {
  console.error("[Upload] failed:", err);
  message.open({
    key: "modUpload",
    type: "error",
    content: err?.message || (isEditing ? "แก้ไขไม่สำเร็จ" : "อัปโหลดไม่สำเร็จ"),
    duration: 2.5,
  });
} finally {
  setUploading(false);
}
};

useEffect(() => {
  return () => {
    if (redirectTimer.current) {
      window.clearTimeout(redirectTimer.current);
    }
  };
}, []);
```

```
const disableSubmit = uploading || checkingOwn || !ownsThisGame;

const checks = {
  gameld: !!gameld,
  title: !!modTitle.trim(),
  fileReady: isEditing ? true : !!modFile,
  owns: ownsThisGame && !checkingOwn,
};

useEffect(() => {
  console.log("[Form check]", {
    gameld,
    title: checks.title,
    modFile: !!modFile,
    isEditing,
    ownsThisGame,
    checkingOwn,
    myUserGameld,
  });
}, [gameld, modTitle, modFile, isEditing, ownsThisGame, checkingOwn, myUserGameld]);

return (
<div style={{ background: "#141414", minHeight: "100vh", flex: 1, position: "relative" }}>
  {showSuccess && (
    <div
      style={{
        position: "fixed",
        zIndex: 1000,
        top: 24,
        left: "50%",
        transform: "translateX(-50%)",
        background: "linear-gradient(135deg, #1f2937 0%, #111827 100%)",
        border: "1px solid #2b3a42",
        color: "#e5e7eb",
        padding: "10px 16px",
      }})
  )
</div>
);
```

```

borderRadius: 10,
boxShadow: "0 10px 30px rgba(0,0,0,0.35)",
)}

>
<Space size="middle">
  <CheckCircleFilled style={{ color: "#52c41a", fontSize: 18 }} />
  <span style={{ fontWeight: 600 }}>
    {isEditing ? "บันทึกมือดเรียบร้อย" : "อัปโหลดมือดเรียบร้อย"}
  </span>
  <span style={{ color: "#9aa4ad" }}>กำลังกลับไปยังหน้า Workshop...</span>
</Space>
</div>
)}

<div style={{ padding: "16px", maxWidth: "800px", margin: "0 auto" }}>
  <Title level={2} style={{ color: "white" }}>
    {game
      ? `${isEditing ? "Edit" : "Upload"} Mods for ${game.name ?? ""}`}
      : "Upload Game Mods"
  </Title>

  {!checkingOwn && !ownsThisGame && (
    <Card
      style={{ background: "#1f1f1f", borderColor: "#404040", marginBottom: 16 }}
      headStyle={{ color: "white" }}
    >
      <div style={{ color: "#ff7875" }}>
        คุณต้องเป็นเจ้าของเกมนี้ก่อนจึงจะอัปโหลดมือได้
      </div>
    </Card>
  )}
<Card
  title={
```

```
<span style={{ color: "white" }}>
  <FileTextOutlined /> Mod Information
</span>
}

style={{
  background: "#1f1f1f",
  marginBottom: "24px",
  borderRadius: 8,
  borderColor: "#404040",
}}
headStyle={{ color: "white", borderBottomColor: "#404040" }}

>
<Input
  placeholder="Your mods title"
  value={modTitle}
  onChange={(e) => setModTitle(e.target.value)}
  style={{
    background: "#2d2d2d",
    color: "white",
    borderColor: "#595959",
  }}
/>
</Card>

{!isEditing && (
  <>
  <Card
    title={
      <span style={{ color: "white" }}>
        <UploadOutlined /> Select your mods file
      </span>
    }
    style={{
      background: "#1f1f1f",
      marginBottom: "24px",
    }}
  </Card>
)}
```

```
borderRadius: 8,  
borderColor: "#404040",  
}  
headStyle={{ color: "white", borderBottomColor: "#404040" }}  
>  
<Dragger  
beforeUpload={handleModFile}  
onChange={handleModFileChange}  
onRemove={handleModFileRemove}  
showUploadList={!modFile}  
multiple={false}  
accept=".zip,.rar,.7z"  
maxCount={1}  
disabled={!ownsThisGame || checkingOwn}  
>  
<p className="ant-upload-drag-icon">  
<InboxOutlined style={{ color: "#9254de" }} />  
</p>  
<p style={{ color: "white" }}>Click or drag mod file to this area</p>  
<p style={{ color: "#aaa" }}>รองรับ .zip, .rar, .7z</p>  
</Dragger>  
{modFile && (  
  <p style={{ color: "white", marginTop: 8 }}>  
    Selected: {modFile.name}  
  </p>  
)  
</Card>  
  
<Card  
title={  
  <span style={{ color: "white" }}>  
    <PictureOutlined /> Upload a mod image (optional)  
  </span>  
}  
style={{
```

```
background: "#1f1f1f",
marginBottom: "24px",
borderRadius: 8,
borderColor: "#404040",
}}
headStyle={{ color: "white", borderBottomColor: "#404040" }}
>
<Dragger
beforeUpload={handleModlImage}
onChange={handleModlImageChange}
onRemove={handleModlImageRemove}
showUploadList={!modlImage}
accept="image/*"
maxCount={1}
multiple={false}
disabled={!ownsThisGame || checkingOwn}
>
<p className="ant-upload-drag-icon">
<InboxOutlined style={{ color: "#52c41a" }} />
</p>
<p style={{ color: "white" }}>Click or drag image to this area</p>
<p style={{ color: "#aaa" }}>ຮອງຮັບ .jpg, .png, .gif</p>
</Dragger>
{imagePreview && (
<img
src={imagePreview}
alt="Preview"
style={{ marginTop: 12, maxHeight: 200, borderRadius: 6 }}
/>
)}
</Card>
</>
)}

<Card
```

```
title={

  <span style={{ color: "white" }}>
    <FileTextOutlined /> Add a description
  </span>
}

style={{

  background: "#1f1f1f",
  marginBottom: "24px",
  borderRadius: 8,
  borderColor: "#404040",
}

headStyle={{ color: "white", borderBottomColor: "#404040" }}

>

<Input.TextArea

  rows={4}

  placeholder="Use this space to describe your mods or what was involved in making it."
  value={modDescription}
  onChange={(e) => setModDescription(e.target.value)}

  style={{

    background: "#2d2d2d",
    color: "white",
    borderColor: "#595959",
  }}
/>

</Card>

<Card

  title={{<span style={{ color: "white" }}>Checklist</span>}}
  style={{

    background: "#1f1f1f",
    marginBottom: 16,
    borderRadius: 8,
    borderColor: "#404040",
  }}

  headStyle={{ color: "white", borderBottomColor: "#404040" }}
```

```

>
<ul style={{ margin: 0, paddingLeft: 18, color: "white", lineHeight: 1.9 }}>
  <li><Ok ok={!!gameId} /> มีรหัสเกมใน URL (เช่น <code>?gameId=123</code>)</li>
  <li><Ok ok={!!modTitle.trim()} /> กรอกชื่อเม็ด</li>
  &{isEditing && (
    <li><Ok ok={!!modFile} /> เลือกไฟล์เม็ด (.zip/.rar/.7z)</li>
  )}
  <li><Ok ok={!!ownsThisGame && !checkingOwn} /> เป็นเจ้าของเกมนี้</li>
</ul>
</Card>

<div style={{ textAlign: "right", position: "relative", zIndex: 1 }}>
  <Button
    type="primary"
    size="large"
    loading={uploading}
    disabled={uploading || checkingOwn || !ownsThisGame}
    style={{ background: "#9254de", borderColor: "#9254de" }}
    onClick={handleUpload}
  >
    {uploading
      ? isEditing
        ? "กำลังบันทึก..."
        : "กำลังอัปโหลด..."
      : isEditing
        ? "Save"
        : "Upload"}
  </Button>
</div>
</div>
</div>
);
};

```

```
export default UploadPage;
```

### WorkshopMain Page

```
import React, { useState } from "react";
import {
  Card,
  Row,
  Col,
  Typography,
  Pagination,
  Input,
  Layout,
  List,
} from "antd";
import { PictureOutlined } from "@ant-design/icons";
import { useNavigate } from "react-router-dom";

const { Content, Sider } = Layout;
const { Text } = Typography;
const { Search } = Input;

interface WorkshopItem {
  id: string;
  title: string;
  items: number;
  image?: string;
}

const WorkshopUI: React.FC = () => {
  const [search, setSearch] = useState("");
  const navigate = useNavigate();

  const workshops: WorkshopItem[] = [
    { id: "1", title: "Magneta Box", items: 9, image: "" },
  ]
```

```

{ id: "2", title: "Carbrix", items: 2, image: "" },
{ id: "3", title: "New Day", items: 81, image: "" },
{ id: "4", title: "Exospace", items: 6, image: "" },
{ id: "5", title: "Your Boy", items: 6, image: "" },
{ id: "6", title: "Nijima", items: 20, image: "" },
{ id: "7", title: "Centoo Rescue", items: 8, image: "" },
{ id: "8", title: "Tamako", items: 6, image: "" },
];

const filtered = workshops.filter((w) =>
  w.title.toLowerCase().includes(search.toLowerCase())
);

return (
  <Layout style={{ background: "#0f1419", minHeight: "100vh" }}>
    <Content style={{ padding: "20px" }}>
      <h2 style={{ color: "white" }}>Browse All Workshops</h2>

      <div style={{ marginBottom: 20 }}>
        <Search
          placeholder="ค้นหา Workshop..."
          allowClear
          enterButton
          onSearch={(value) => setSearch(value)}
          style={{ maxWidth: 400 }}
        />
      </div>

      <Row gutter={[16, 16]}>
        {filtered.map((w) => (
          <Col xs={24} sm={12} md={8} lg={6} key={w.id}>
            <Card
              hoverable
              onClick={() => navigate(`~/workshop/${w.id}`, { state: w })} // ⚡️ ส่ง object ไปเลย
              style={{ background: "#1f1f1f", borderRadius: 8 }}
            </Card>
          </Col>
        ))
      </Row>
    </Content>
  </Layout>
)

```

```
cover={

w.image ? (
    <img
        alt={w.title}
        src={w.image}
        style={{ height: 120, objectFit: "cover" }}
    />
) : (
    <div
        style={{
            height: 120,
            background: "#2a2a2a",
            display: "flex",
            justifyContent: "center",
            alignItems: "center",
            color: "#888",
            fontSize: 24,
        }}
    >
        <PictureOutlined />
    </div>
)
}

>
<Text style={{ color: "white" }}>{w.title}</Text>
<br />
<Text type="secondary" style={{ color: "#aaa" }}>
    {w.items} items
</Text>
</Card>
</Col>
))}

</Row>

<div style={{ marginTop: 20, textAlign: "center" }}>
```

```
<Pagination defaultCurrent={1} total={50} pageSize={8} />
</div>
</Content>

/* Sidebar */
<Sider
  width={200}
  style={{
    background: "#141414",
    padding: "20px",
    borderLeft: "1px solid #2a2a2a",
  }}
>
  <h3 style={{ color: "white" }}>Menu</h3>
  <List
    dataSource={[{ name: "Your Games", path: "/your-games" }]}
    renderItem={({item}) =>
      <List.Item
        style={{
          color: "white",
          cursor: "pointer",
          border: "none",
          padding: "8px 0",
          transition: "color 0.2s",
        }}
        onMouseEnter={(e) =>
          ((e.currentTarget.style.color = "#40a9ff"))}
        }
        onMouseLeave={(e) =>
          ((e.currentTarget.style.color = "white"))}
        }
        onClick={() => (window.location.href = item.path)}
      >
        {item.name}
      </List.Item>
    }
  }
</Sider>
```

```
        )}
      />
    </Sider>
  </Layout>
);
};

export default WorkshopUI;
```

### WorkshopDetail Page

```
import React, { useState } from "react";
import { useLocation, useNavigate } from "react-router-dom";
import {
  Layout,
  Typography,
  Input,
  Row,
  Col,
  Card,
  List,
  Button,
  message,
} from "antd";
import { PictureOutlined } from "@ant-design/icons";

const { Content, Sider, Header } = Layout;
const { Title, Text } = Typography;
const { Search } = Input;

interface WorkshopItem {
  id: string;
  title: string;
  items: number;
  image?: string;
}
```

```

interface ModItem {
  id: string;
  title: string;
  author: string;
}

const WorkshopDetail: React.FC = () => {
  const location = useLocation();
  const navigate = useNavigate();
  const workshop = location.state as WorkshopItem;

  // mock data mods
  const mods: ModItem[] = [
    { id: "m1", title: "Weapons Course", author: "Asmisint" },
    { id: "m2", title: "CS:CTF Double Cross", author: "CS:CTF" },
    { id: "m3", title: "CS:CTF 2Fort", author: "CS:CTF" },
    { id: "m4", title: "Mocha", author: "Bevster" },
    { id: "m5", title: "CS:CTF Turbine", author: "CS:CTF" },
    { id: "m6", title: "1v1_the_desert_pit", author: "MMArezech_" },
  ];

  // state สำหรับ search
  const [searchText, setSearchText] = useState("");
  const [filteredMods, setFilteredMods] = useState<ModItem[]>(mods);

  // mock: เกมที่ user มี
  const userGames = ["1", "3", "6"];

  const handleUpload = () => {
    if (userGames.includes(workshop.id)) {
      navigate(`/upload?gameId=${workshop.id}`);
    } else {
      message.error("คุณไม่มีเกมนี้ ไม่สามารถอัปโหลดมื้อต่อได้");
    }
  }
}

```

```
};

const handleSearch = (value: string) => {
  setSearchText(value);
  if (!value) {
    setFilteredMods(mods);
  } else {
    const lower = value.toLowerCase();
    const filtered = mods.filter(
      (m) =>
        m.title.toLowerCase().includes(lower) ||
        m.author.toLowerCase().includes(lower)
    );
    setFilteredMods(filtered);
  }
};

return (
  <Layout style={{ background: "#0f1419", minHeight: "100vh" }}>
    {/* Header Banner */}
    <Header
      style={{
        background: "#1f1f1f",
        padding: 0,
        height: 200,
        display: "flex",
        justifyContent: "center",
        alignItems: "center",
      }}
    >
    {workshop.image ? (
      <img
        src={workshop.image}
        alt={workshop.title}
        style={{ width: "100%", height: "100%", objectFit: "cover" }}
    
```

```
/>
) : (
<div
style={{
width: "90%",
height: "100%",
background: "#2a2a2a",
display: "flex",
justifyContent: "center",
alignItems: "center",
color: "#888",
fontSize: 20,
}}
>
<PictureOutlined style={{ fontSize: 40, marginRight: 10 }} />
Banner for {workshop.title}
</div>
)}
</Header>

<Layout>
/* Content */
<Content style={{ padding: "20px" }}>
<Title level={3} style={{ color: "white" }}>
{workshop.title} – {workshop.items} items
</Title>
<Text style={{ color: "#aaa" }}>
Showing 1–{filteredMods.length} of {mods.length} entries
</Text>

/* Search */
<div
style={{
marginTop: 20,
marginBottom: 20,
```

```
        display: "flex",
        gap: "10px",
        flexWrap: "wrap",
    }}
>
<Search
    placeholder="Search mods..."
    allowClear
    style={{ maxWidth: 250 }}
    value={searchText}
    onChange={(e) => handleSearch(e.target.value)}
    onSearch={handleSearch}
/>
</div>

/* Grid Mods */
<Row gutter={[16, 16]}>
{filteredMods.map((mod) => (
    <Col xs={24} sm={12} md={8} lg={6} key={mod.id}>
        <Card
            hoverable
            style={{ background: "#1f1f1f", borderRadius: 8 }}
            cover={
                <div
                    style={{
                        height: 120,
                        background: "#2a2a2a",
                        display: "flex",
                        justifyContent: "center",
                        alignItems: "center",
                        color: "#888",
                        fontSize: 24,
                    }}
                >
                    <PictureOutlined />
                </div>
            }
        </Card>
    </Col>
))}</Row>
```

```

        </div>
    }
    onClick={() =>
      navigate(`/mod/${mod.id}`, { state: mod })
    }
  >
  <Text style={{ color: "white" }}>{mod.title}</Text>
  <br />
  <Text type="secondary" style={{ color: "#aaa" }}>
    by {mod.author}
  </Text>
</Card>
</Col>
))}

{filteredMods.length === 0 && (
  <Col span={24} style={{ textAlign: "center", color: "#aaa" }}>
    No mods found.
  </Col>
)}
</Row>
</Content>

/* Sidebar */
<Sider
  width={220}
  style={{
    background: "#141414",
    padding: "20px",
    borderLeft: "1px solid #2a2a2a",
  }}
>
<h3 style={{ color: "white" }}>SHOW:</h3>
<List
  dataSource={[

```

```
{ name: "All", path: "#" },  
 { name: "Your Favorites", path: "#" },  
 ]}  
  
renderItem={(item) => (  
  <List.Item  
    style={{  
      color: "white",  
      cursor: "pointer",  
      border: "none",  
      padding: "8px 0",  
      transition: "color 0.2s",  
    }}  
    onMouseEnter={(e) =>  
      ((e.currentTarget.style.color = "#40a9ff"))  
    }  
    onMouseLeave={(e) =>  
      ((e.currentTarget.style.color = "white"))  
    }  
  >  
  {item.name}  
  </List.Item>  
)}  
>  
  
<Button  
  type="primary"  
  block  
  style={{ marginTop: 20 }}  
  onClick={handleUpload}  
>  
  Upload Mod  
</Button>  
</Sider>  
</Layout>  
</Layout>
```

```
};

};

export default WorkshopDetail;

ModDetail Page

import React, { useState, useEffect } from "react";
import { useLocation, useNavigate } from "react-router-dom";
import {
  Layout,
  Typography,
  Button,
  Card,
  Divider,
  Space,
  List,
  Input,
  Rate,
} from "antd";
import {
  UserOutlined,
  CalendarOutlined,
  DownloadOutlined,
  ShareAltOutlined,
  ArrowLeftOutlined,
} from "@ant-design/icons";

// import รูปจาก src/assets
import defaultModImage from "../assets/mod1.jpg";

const { Content, Header } = Layout;
const { Title, Text, Paragraph } = Typography;
const { TextArea } = Input;

interface ModItem {
```

```
id: string;
title: string;
author: string;
image?: string;
description?: string;
date?: string;
downloads?: number; // ถ้าจะเก็บรวม
views?: number; // จำนวนผู้เข้าชม (Unique Visitors)
subscribers?: number; // จำนวน Subscribers
}

interface CommentItem {
  author: string;
  content: string;
  datetime: string;
}

const ModDetail: React.FC = () => {
  const location = useLocation();
  const navigate = useNavigate();
  const mod = location.state as ModItem;

  const [comments, setComments] = useState<CommentItem>([
    {
      author: "Player1",
      content: "This mod is awesome! 🔥",
      datetime: "2025-09-05 12:30",
    },
    {
      author: "Player2",
      content: "Can you update for the latest version?",
      datetime: "2025-09-05 13:10",
    },
  ]);
}
```

```
const [newComment, setNewComment] = useState("");

// ⭐ Rating states
const [rating, setRating] = useState<number>(0); // คะแนนที่ user ให้
const [averageRating, setAverageRating] = useState<number>(4.2); // mock ค่าเฉลี่ย
const [ratingCount, setRatingCount] = useState<number>(125); // mock จำนวนโหวต

// mock fetch จาก API
useEffect(() => {
    const fetchRating = async () => {
        // TODO: แก้เป็น fetch("/api/mods/:id/rating")
        const data = { avg: 4.2, count: 125 };
        setAverageRating(data.avg);
        setRatingCount(data.count);
    };
    fetchRating();
}, []);

const handleAddComment = () => {
    if (!newComment.trim()) return;

    const newItem: CommentItem = {
        author: "You",
        content: newComment,
        datetime: new Date().toLocaleString(),
    };

    setComments([newItem, ...comments]);
    setNewComment("");
};

const handleRateChange = async (value: number) => {
    setRating(value);

    // TODO: ส่งค่าไป backend
}
```

```
// fetch(`/api/mods/${mod.id}/rate`, { method: "POST", body: JSON.stringify({ value }) })
console.log("User rated:", value);

// mock update ค่าเฉลี่ย
const newCount = ratingCount + 1;
const newAvg = (averageRating * ratingCount + value) / newCount;
setRatingCount(newCount);
setAverageRating(newAvg);
};

if (!mod) {
  return (
    <Layout style={{ background: "#0f1419", minHeight: "100vh" }}>
      <Content style={{ padding: "20px", color: "white" }}>
        <Title level={3} style={{ color: "white" }}>
          ✗ Mod not found
        </Title>
        <Button
          type="primary"
          icon={<ArrowLeftOutlined />}
          onClick={() => navigate(-1)}
        >
          Back
        </Button>
      </Content>
    </Layout>
  );
}

return (
  <Layout style={{ background: "#0f1419", minHeight: "100vh" }}>
    /* Banner */
    <Header
      style={{
        background: "#1f1f1f",
        color: "white",
        padding: "10px 0",
      }}>
      <h1>ModDB</h1>
    </Header>
    <Content style={{ padding: "20px" }}>
      <h2>Mod Details</h2>
      <Row style={{ gap: "20px" }}>
        <Col>
          <Image alt="Placeholder image for mod icon" style={{ width: "100px", height: "100px", border: "1px solid #ccc", border-radius: "50%", objectFit: "cover" }}></Image>
```

```
padding: 0,
height: 250,
display: "flex",
justifyContent: "center",
alignItems: "center",
}}
>
<img
src={mod.image || defaultModImage}
alt={mod.title}
style={{ width: "100%", height: "100%", objectFit: "cover" }}
/>
</Header>

/* Main Content */
<Content style={{ padding: "20px" }}>
<div style={{ display: "flex", gap: "20px", alignItems: "flex-start" }}>
  /* Left: Details */
  <div style={{ flex: 3 }}>
    <Title level={2} style={{ color: "white" }}>
      {mod.title}
    </Title>
    <Space>
      <Button type="primary">Download</Button>
    </Space>
  </div>
  <Divider style={{ borderColor: "#333" }} />
  <Paragraph style={{ color: "white" }}>
    {mod.description ||
      "This is a placeholder description for the mod. In the future, you can load real mod
      details from the backend."}
  </Paragraph>
</div>
</Content>
```

```
/* Comments Section */
<Divider style={{ borderColor: "#333" }} />
<Title level={4} style={{ color: "white" }}>
    Comments
</Title>

<TextArea
    rows={3}
    value={newComment}
    onChange={(e) => setNewComment(e.target.value)}
    placeholder="Write a comment..."
    style={{ marginBottom: "10px" }}
/>
<Button type="primary" onClick={handleAddComment}>
    Add Comment
</Button>

<List
    dataSource={comments}
    style={{ marginTop: 20 }}
    renderItem={(item) => (
        <List.Item style={{ borderBottom: "1px solid #333" }}>
            <Card
                style={{
                    width: "100%",
                    background: "#1f1f1f",
                    color: "white",
                }}
                bodyStyle={{ padding: "10px" }}
            >
                <Text strong style={{ color: "#4dabf7" }}>
                    {item.author}
                </Text>
                <Paragraph style={{ color: "white", margin: "5px 0" }}>
                    {item.content}
                </Paragraph>
            </Card>
        </List.Item>
    )}
</List>
```

```
</Paragraph>
<Text type="secondary" style={{ color: "#D3D3D3", fontSize: "12px" }}>
  {item.datetime}
</Text>
</Card>
</List.Item>
)}
/>
</div>

/* Right: Sidebar */
<div style={{ flex: 1 }}>
  <Card
    style={{
      background: 'linear-gradient(90deg, #9254de 0%, #f759ab 100%)',
      color: "white",
      borderRadius: 8,
    }}
  >
    <Title level={5} style={{ color: "white" }}>
      <UserOutlined /> Creator: {mod.author}
    </Title>
    <Title level={5} style={{ color: "white" }}>
      <CalendarOutlined /> Uploaded: {mod.date || "2025-09-05"}
    </Title>

    <Divider style={{ borderColor: "#333" }} />

    /* Stats */
    <div style={{ marginBottom: "10px" }}>
      <Text strong style={{ color: "#4dabf7" }}>
        {mod.views?.toLocaleString() || 0}
      </Text>{" "}
      <Text style={{ color: "white" }}>Unique Visitors</Text>
      <br />
    </div>
  </Card>
</div>
```

```

<Text strong style={{ color: "#4dabf7" }}>
  {mod.subscribers?.toLocaleString() || 0}
</Text>" "}
<Text style={{ color: "white" }}>Current Downloads</Text>
</div>

<Divider style={{ borderColor: "#D" }} />

/* ⭐ Rating Section */

<div style={{ marginBottom: "10px" }}>
  <Title level={5} style={{ color: "white", marginBottom: 5 }}>
    Rate this Mod
  </Title>
  <Rate value={rating} onChange={handleRateChange} />
  <div style={{ marginTop: 5 }}>
    <Text style={{ color: "white" }}>
      {rating > 0 ? `Your Rating: ${rating} / 5` : "No rating yet"}
    </Text>
  </div>

<Divider style={{ borderColor: "#333" }} />

<Text style={{ color: "white" }}>Average Rating:</Text>
<div>
  <Rate disabled allowHalf value={averageRating} />
  <Text style={{ color: "white", marginLeft: 8 }}>
    {averageRating.toFixed(1)} / 5 ({ratingCount.toLocaleString()} ratings)
  </Text>
</div>
</div>
</Card>
</div>
</div>
</Content>
</Layout>

```

```
    );  
};  
  
export default ModDetail;
```

## Entity

```
//gameUser.go
package entity

import (
    "time"

    "gorm.io/gorm"
)

type GameUser struct {
    gorm.Model
    Username string `json:"username"`
    Password string `json:"password"`
    Email    string `json:"email"`
    FirstName string `json:"first_name"`
    LastName string `json:"last_name"`
    Birthday time.Time `json:"birthday"`

    OwnerGames []OwnerGame `gorm:"foreignKey:GameUserID" json:"owner_games"`
    Mods      []Mod       `gorm:"foreignKey:GameUserID" json:"mods"`
    ModRatings []ModRating `gorm:"foreignKey:GameUserID" json:"mod_ratings"`
}

//game.go
package entity

import "gorm.io/gorm"

type Game struct {
    gorm.Model
    Title    string `json:"title"`
    Description string `json:"description"
```

```
OwnerGames []OwnerGame `gorm:"foreignKey:GameID" json:"owner_games"`
Mods     []Mod      `gorm:"foreignKey:GameID" json:"mods"`
}

//mod.go
package entity

import (
    "time"

    "gorm.io/gorm"
)

type Mod struct {
    gorm.Model

    Title     string   `json:"title"`
    Description string   `json:"description"`
    UploadDate time.Time `json:"upload_date"`
    FilePath   string   `json:"file_path"`
    Status     string   `json:"status"`

    GameUserID uint     `json:"game_user_id"`
    GameUser   *GameUser `gorm:"foreignKey:GameUserID" json:"game_user"`

    GameID uint     `json:"game_id"`
    Game   *Game    `gorm:"foreignKey:GameID" json:"game"`

    ModTags []ModTag `gorm:"foreignKey:ModID" json:"mod_tags"`
}
```

```
//modrating.go
package entity

import (
    "time"

    "gorm.io/gorm"
)

type ModRating struct {
    gorm.Model

    Rating     string `json:"rating"`
    Review     string `json:"review"`
    PurchaseDate time.Time `json:"purchase_date"`

    GameUserID uint `json:"game_user_id"`
    GameUser   *GameUser `gorm:"foreignKey:GameUserID" json:"game_user"`

    ModID uint `json:"mod_id"`
    Mod   *Mod `gorm:"foreignKey:ModID" json:"mod"`
}

//modtags.go
package entity

import "gorm.io/gorm"

type ModTag struct {
    gorm.Model

    ModID uint `json:"mod_id"`
    Mod   *Mod `gorm:"foreignKey:ModID" json:"mod"`

    TagID uint `json:"tag_id"`
}
```

```
    Tag *Tag `gorm:"foreignKey:TagID" json:"tag"`
}

//ownergame.go
package entity

import (
    "time"

    "gorm.io/gorm"
)

type OwnerGame struct {
    gorm.Model

    PurchaseDate time.Time `json:"purchase_date"`
    Status       string   `json:"status"`

    GameUserID uint     `json:"game_user_id"`
    GameUser   *GameUser `gorm:"foreignKey:GameUserID" json:"game_user"`

    GameID uint `json:"game_id"`
    Game   *Game `gorm:"foreignKey:GameID" json:"game"`
}

//tags.go
package entity

import "gorm.io/gorm"

type Tag struct {
    gorm.Model
    Title string `json:"title"`

    ModTags []ModTag `gorm:"foreignKey:TagID" json:"mod_tags"`
}
```

## Controller

```
game.go
package controllers

import (
    "net/http"

    "gameshop-backend/configs"
    "gameshop-backend/entity/workshop"
    "github.com/gin-gonic/gin"
)

// GET /games
func GetGames(c *gin.Context) {
    var games []entity.Game
    if err := configs.DB().Find(&games).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, games)
}

// GET /games/:id
func GetGameById(c *gin.Context) {
    id := c.Param("id")
    var game entity.Game
    if tx := configs.DB().Where("id = ?", id).First(&game); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "game not found"})
        return
    }
    c.JSON(http.StatusOK, game)
}

// POST /games
```

```
func CreateGame(c *gin.Context) {
    var game entity.Game
    if err := c.ShouldBindJSON(&game); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if err := configs.DB().Create(&game).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusCreated, game)
}

// PATCH /games/:id
func UpdateGame(c *gin.Context) {
    id := c.Param("id")
    var game entity.Game
    if tx := configs.DB().Where("id = ?", id).First(&game); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "game not found"})
        return
    }

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

    if err := configs.DB().Model(&game).Updates(input).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, game)
}
```

```

// DELETE /games/:id
func DeleteGame(c *gin.Context) {
    id := c.Param("id")
    if tx := configs.DB().Exec("DELETE FROM games WHERE id = ?", id); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "game not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "deleted successfully"})
}

mod.go
package controllers

import (
    "net/http"

    "gameshop-backend/configs"
    "gameshop-backend/entity/workshop"
    "github.com/gin-gonic/gin"
)

// GET /mods
func GetMods(c *gin.Context) {
    var mods []entity.Mod
    if err := configs.DB().Find(&mods).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, mods)
}

// GET /mods/:id
func GetModById(c *gin.Context) {
    id := c.Param("id")
    var mod entity.Mod

```

```
if tx := configs.DB().Where("id = ?", id).First(&mod); tx.RowsAffected == 0 {
    c.JSON(http.StatusNotFound, gin.H{"error": "mod not found"})
    return
}
c.JSON(http.StatusOK, mod)
}

// POST /mods
func CreateMod(c *gin.Context) {
    var mod entity.Mod
    if err := c.ShouldBindJSON(&mod); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if err := configs.DB().Create(&mod).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusCreated, mod)
}

// PATCH /mods/:id
func UpdateMod(c *gin.Context) {
    id := c.Param("id")
    var mod entity.Mod
    if tx := configs.DB().Where("id = ?", id).First(&mod); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "mod not found"})
        return
    }

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

```

if err := configs.DB().Model(&mod).Updates(input).Error; err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}
c.JSON(http.StatusOK, mod)
}

// DELETE /mods/:id
func DeleteMod(c *gin.Context) {
    id := c.Param("id")
    if tx := configs.DB().Exec("DELETE FROM mods WHERE id = ?", id); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "mod not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "deleted successfully"})
}

```

### **moderating.go**

package controllers

```

import (
    "net/http"

    "gameshop-backend/configs"
    "gameshop-backend/entity/workshop"
    "github.com/gin-gonic/gin"
)

// GET /modratings
func GetModRatings(c *gin.Context) {
    var ratings []entity.ModRating
    if err := configs.DB().Find(&ratings).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }

```

```
}

c.JSON(http.StatusOK, ratings)

}

// GET /modratings/:id
func GetModRatingById(c *gin.Context) {
    id := c.Param("id")
    var rating entity.ModRating
    if tx := configs.DB().Where("id = ?", id).First(&rating); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "rating not found"})
        return
    }
    c.JSON(http.StatusOK, rating)
}

// POST /modratings
func CreateModRating(c *gin.Context) {
    var rating entity.ModRating
    if err := c.ShouldBindJSON(&rating); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if err := configs.DB().Create(&rating).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusCreated, rating)
}

// PATCH /modratings/:id
func UpdateModRating(c *gin.Context) {
    id := c.Param("id")
    var rating entity.ModRating
    if tx := configs.DB().Where("id = ?", id).First(&rating); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "rating not found"})
    }
}
```

```

        return
    }

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

    if err := configs.DB().Model(&rating).Updates(input).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, rating)
}

// DELETE /modratings/:id
func DeleteModRating(c *gin.Context) {
    id := c.Param("id")
    if tx := configs.DB().Exec("DELETE FROM mod_ratings WHERE id = ?", id); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "rating not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "deleted successfully"})
}

```

### **modtags.go**

```

package controllers

import (
    "net/http"

    "gameshop-backend/configs"
    "gameshop-backend/entity/workshop"
    "github.com/gin-gonic/gin"
)

```

```
)  
  
// GET /modtags  
func GetModTags(c *gin.Context) {  
    var modtags []entity.ModTags  
    if err := configs.DB().Find(&modtags).Error; err != nil {  
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})  
        return  
    }  
    c.JSON(http.StatusOK, modtags)  
}  
  
// GET /modtags/:id  
func GetModTagById(c *gin.Context) {  
    id := c.Param("id")  
    var modtag entity.ModTags  
    if tx := configs.DB().Where("id = ?", id).First(&modtag); tx.RowsAffected == 0 {  
        c.JSON(http.StatusNotFound, gin.H{"error": "modtag not found"})  
        return  
    }  
    c.JSON(http.StatusOK, modtag)  
}  
  
// POST /modtags  
func CreateModTag(c *gin.Context) {  
    var modtag entity.ModTags  
    if err := c.ShouldBindJSON(&modtag); err != nil {  
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})  
        return  
    }  
    if err := configs.DB().Create(&modtag).Error; err != nil {  
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})  
        return  
    }  
    c.JSON(http.StatusCreated, modtag)
```

```
}

// PATCH /modtags/:id
func UpdateModTag(c *gin.Context) {
    id := c.Param("id")
    var modtag entity.ModTags
    if tx := configs.DB().Where("id = ?", id).First(&modtag); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "modtag not found"})
        return
    }

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

    if err := configs.DB().Model(&modtag).Updates(input).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, modtag)
}

// DELETE /modtags/:id
func DeleteModTag(c *gin.Context) {
    id := c.Param("id")
    if tx := configs.DB().Exec("DELETE FROM mod_tags WHERE id = ?", id); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "modtag not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "deleted successfully"})
}
```

```
ownergame.go
package controllers

import (
    "net/http"

    "gameshop-backend/configs"
    "gameshop-backend/entity/workshop"
    "github.com/gin-gonic/gin"
)

// GET /ownergames
func GetOwnerGames(c *gin.Context) {
    var ownergames []entity.OwnerGame
    if err := configs.DB().Find(&ownergames).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, ownergames)
}

// GET /ownergames/:id
func GetOwnerGameById(c *gin.Context) {
    id := c.Param("id")
    var ownergame entity.OwnerGame
    if tx := configs.DB().Where("id = ?", id).First(&ownergame); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "ownergame not found"})
        return
    }
    c.JSON(http.StatusOK, ownergame)
}

// POST /ownergames
func CreateOwnerGame(c *gin.Context) {
    var ownergame entity.OwnerGame
```

```

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

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

c.JSON(http.StatusCreated, ownergame)
}

// PATCH /ownergames/:id
func UpdateOwnerGame(c *gin.Context) {
    id := c.Param("id")

    var ownergame entity.OwnerGame
    if tx := configs.DB().Where("id = ?", id).First(&ownergame); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "ownergame not found"})
        return
    }

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

    if err := configs.DB().Model(&ownergame).Updates(input).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, ownergame)
}

// DELETE /ownergames/:id
func DeleteOwnerGame(c *gin.Context) {

```

```

id := c.Param("id")

if tx := configs.DB().Exec("DELETE FROM owner_games WHERE id = ?", id); tx.RowsAffected == 0 {
    c.JSON(http.StatusNotFound, gin.H{"error": "ownergame not found"})
    return
}

c.JSON(http.StatusOK, gin.H{"message": "deleted successfully"})
}

tags.go

package controllers

import (
    "net/http"

    "gameshop-backend/configs"
    "gameshop-backend/entity/workshop"
    "github.com/gin-gonic/gin"
)

// GET /tags
func GetTags(c *gin.Context) {
    var tags []entity.Tags
    if err := configs.DB().Find(&tags).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, tags)
}

// GET /tags/:id
func GetTagById(c *gin.Context) {
    id := c.Param("id")
    var tag entity.Tags
    if tx := configs.DB().Where("id = ?", id).First(&tag); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "tag not found"})
    }
}

```

```
        return
    }
    c.JSON(http.StatusOK, tag)
}

// POST /tags
func CreateTag(c *gin.Context) {
    var tag entity.Tags
    if err := c.ShouldBindJSON(&tag); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if err := configs.DB().Create(&tag).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusCreated, tag)
}

// PATCH /tags/:id
func UpdateTag(c *gin.Context) {
    id := c.Param("id")
    var tag entity.Tags
    if tx := configs.DB().Where("id = ?", id).First(&tag); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "tag not found"})
        return
    }

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

    if err := configs.DB().Model(&tag).Updates(input).Error; err != nil {
```

```
c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
return
}
c.JSON(http.StatusOK, tag)
}

// DELETE /tags/:id
func DeleteTag(c *gin.Context) {
id := c.Param("id")
if tx := configs.DB().Exec("DELETE FROM tags WHERE id = ?", id); tx.RowsAffected == 0 {
    c.JSON(http.StatusNotFound, gin.H{"error": "tag not found"})
    return
}
c.JSON(http.StatusOK, gin.H{"message": "deleted successfully"})
}
```

## วิเคราะห์ Communication Diagram

เราจะใช้ข้อมูลจาก System Activity Diagram เป็นหลัก

ในการวิเคราะห์เพื่อให้ได้ตารางสำหรับการเขียน Communication Diagram เป็นขั้นตอน ในการวัด

Communication Diagram เราจะมี

เส้นโครง ที่ใช้เชื่อมต่อ เพื่อบอกความเกี่ยวข้องของวัตถุในระบบ

เส้นคำสั่ง จะเป็นเส้นที่มีหัวลูกศร วางเรียงกันอยู่บนเส้นโครง เพื่อบอกการส่งข้อความ (dispatch message) โดยที่หัวลูกศรจะชี้ไปที่วัตถุซึ่งทำงานตามคำสั่งนั้นๆ

เราจะวิเคราะห์ระบบเฉพาะเหตุการณ์หลักของ Use Case ที่ diagram นี้รับผิดชอบ เป็นเส้นทางการทำงานที่เมื่อทำแล้วจะได้ Output ของข้อมูลและ Output ของหน้าจอ ตามที่ระบุไว้ในเอกสาร requirements

จะมีการนำ Class ทั้งหมดที่เคยวิเคราะห์ไว้มาใช้โดย

Boundary Class ทำหน้าที่แทน UI และเราจะใช้ชื่อเบื้องต้นตามชื่อ Use Case หลัก เช่น ในที่นี้จะตั้งชื่อ เป็น UploadModUI

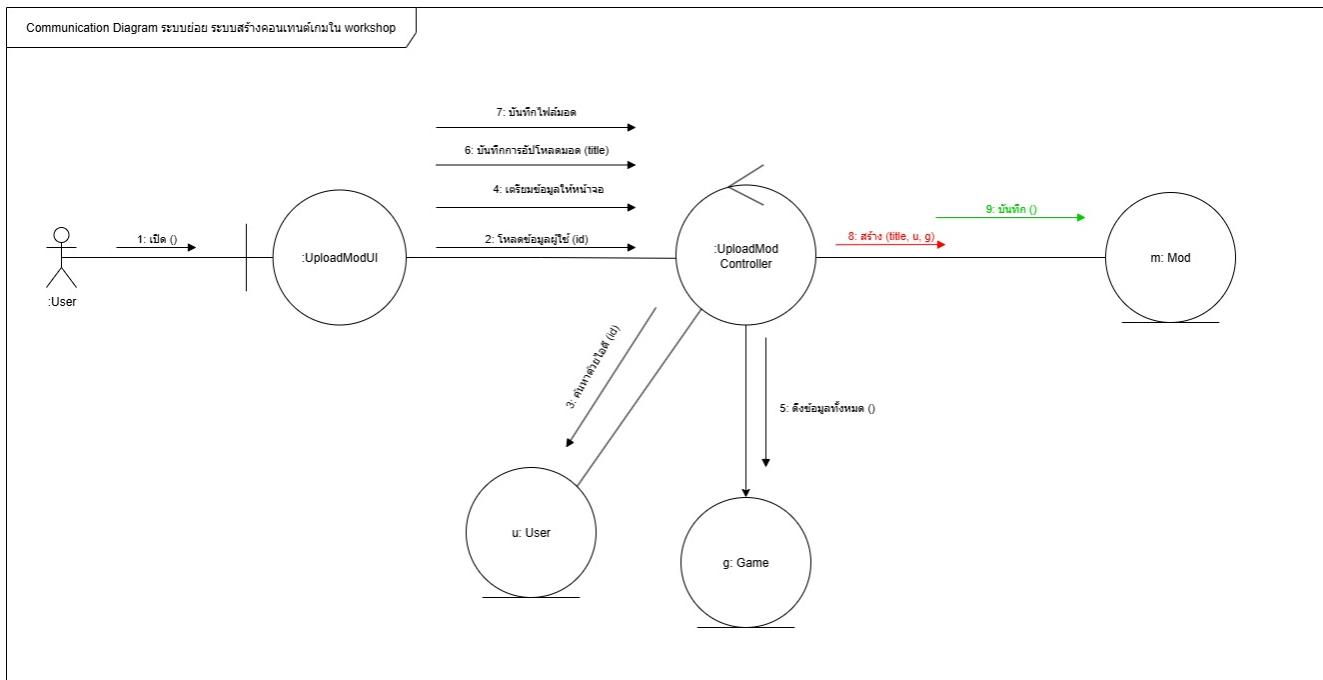
วัตถุของ Boundary Class จะส่งข้อมูลที่จำเป็นให้กับวัตถุของ Control Class

Control Class ทำหน้าที่เป็นตัวประมวลผล ควบคุมการทำงาน และเชื่อมโยงระหว่าง Entity โดยจะตั้งชื่อตาม Use Case หลัก เช่น ในที่นี้จะตั้งชื่อเป็น UploadModController

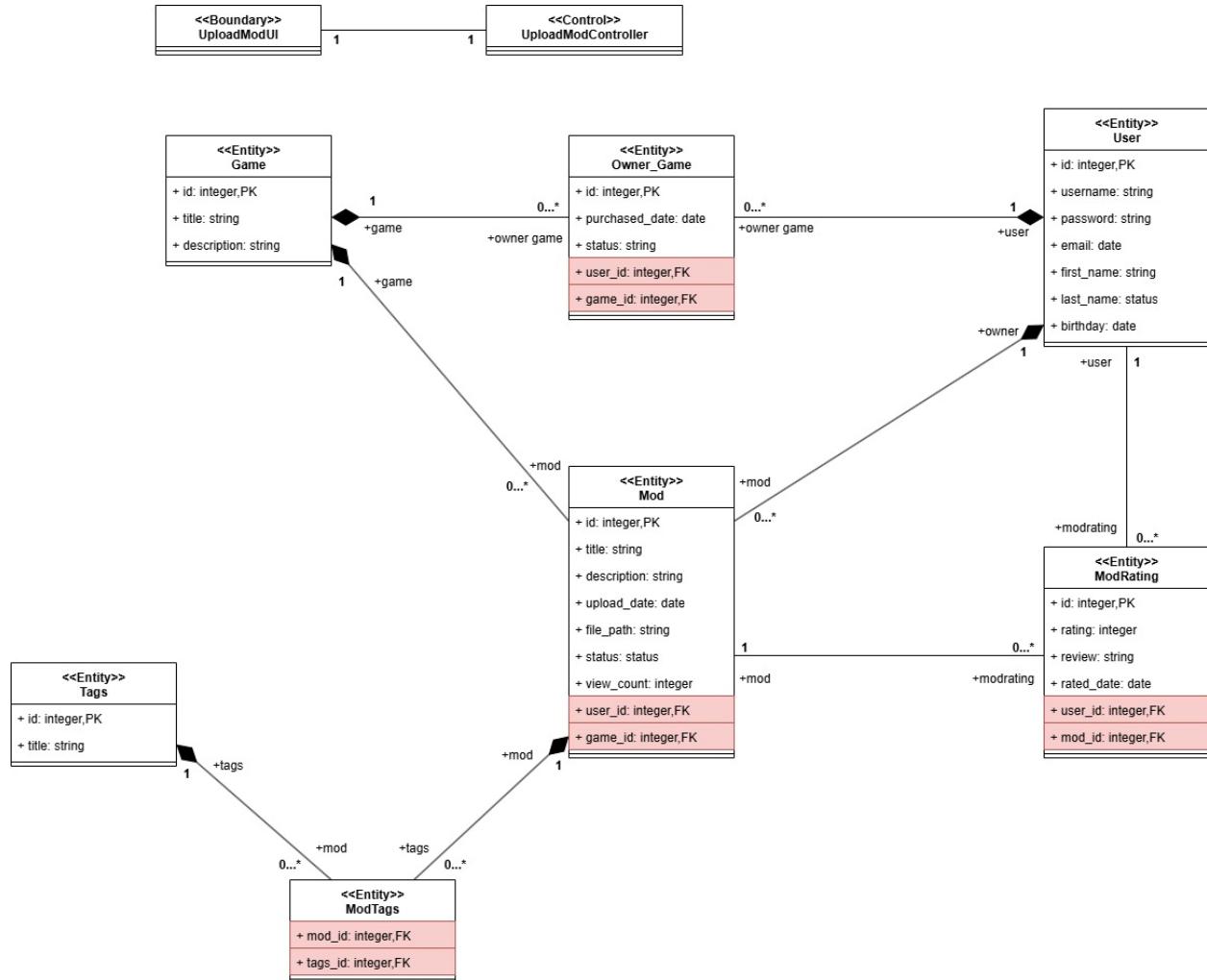
เตรียมแปลงแต่ละ Activity ของ System Activity Diagram ให้เป็น Communication Diagram โดยวิเคราะห์ให้วัตถุที่เกี่ยวข้องกับการทำงานในแต่ละขั้นตอน

Activity	เป็นคำสั่งสำหรับระบบได้ หรือไม่	ถ้าเป็น, วัตถุที่รับหน้าที่ทำงานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click “เปิดหน้าจอ”	เป็นคำสั่ง สั่งงานเพื่อให้หน้า UI เปิด	:UploadModUI	เปิด ()
เข้าสู่ระบบในฐานะ User	ไม่เป็นคำสั่ง	-	-
โหลดข้อมูล User ที่ กำลังใช้งานอยู่	เป็นคำสั่ง เกิดการสั่งงานให้โหลดข้อมูล ผู้ใช้ที่ login อยู่	:UploadModController	โหลดข้อมูลผู้ใช้ (id)
		u : User	ค้นหาด้วย (id)
แสดงหน้าจอหลัก ของระบบ	ไม่เป็นคำสั่ง	-	-
Click เมนูเข้าหน้า workshop	ไม่เป็นคำสั่ง	-	-
แสดงหน้าจอ workshop	ไม่เป็นคำสั่ง	-	-
โหลดข้อมูลรายการ เกม	เป็นคำสั่ง เกิดการสั่งงานให้โหลดข้อมูล เกมทั้งหมด	:UploadModController	เตรียมข้อมูลให้หน้าจอ ()
		g: Game	ดึงข้อมูลทั้งหมด ()
กดเลือกเกมที่ ต้องการอัปโหลด模	ไม่เป็นคำสั่ง	-	-
แสดงข้อมูลเกมและ มอดของเกมนั้น	ไม่เป็นคำสั่ง	-	-
Click ปุ่มเมนู อัปโหลด模	ไม่เป็นคำสั่ง	-	-
แสดงหน้าจอสำหรับ อัปโหลด模	ไม่เป็นคำสั่ง	-	-
ใส่ชื่омод ( <del>ได้ title</del> )	ไม่เป็นคำสั่ง	-	-
Click ปุ่ม “เลือกไฟล์ หรือลากและวางไฟล์ ในพื้นที่กำหนด”	ไม่เป็นคำสั่ง	-	-

บันทึกไฟล์ที่จะอัปโหลด	เป็นคำสั่ง ระบบจะทำการจัดเก็บข้อมูลไฟล์模	:UploadModController	บันทึกไฟล์模
ใส่คำอธิบายของ模	ไม่เป็นคำสั่ง	-	-
Click ปุ่ม “Upload”	เป็นคำสั่ง ระบบจะทำการจัดเก็บข้อมูล模ที่อัปโหลด	:UploadModController	บันทึกการอัปโหลด模 (title)
สร้างข้อมูล entity Mod โดยโยง entity User, โยง entity Game เช็คค่า title	เป็นคำสั่ง	m: Mod	สร้าง (title, u, g)
บันทึก entity Mod	เป็นคำสั่ง	m: Mod	บันทึก ()
แสดงข้อความว่า “อัปโหลด模 สำเร็จ”	ไม่เป็นคำสั่ง	-	-



## Class Diagram at Design Level



B6618643 นายกิตตินันท์ ปัจจัยโคงา

ระบบหลัก ระบบร้านขายเกมออนไลน์

ระบบย่อย ระบบชำระเงิน

เมื่อผู้ใช้งานได้เกมที่ต้องการแล้วต่อไปก็จะเป็นในส่วนของการชำระเงินเพื่อให้ได้เกมมา ซึ่งจะดำเนินการผ่านระบบจัดการชำระเงินโดยผู้ใช้งานสามารถเลือกเกมหลายเกมหรือเกมเดียวมาชำระเงินพร้อมกันได้รวมไปถึงการครบเกมที่ไม่ต้องการออกจากระบบชำระเงินด้วยและหากผู้ใช้งานมีส่วนลดจากโปรโมชั่นต่างๆ ก็จะสามารถนำมาใช้เป็นส่วนลดของการชำระเงินได้ด้วย โดยระบบจะยืนยันหลังจากที่ผู้ใช้งานได้ทำการแนบสลิปของการโอนผ่านมาในการชำระเงิน เมื่อผู้ใช้งานได้ทำการชำระเงินแล้วก็จะได้รหัสคำสั่งซื้อมาพร้อมกับสถานะการชำระเงิน(รอตรวจสอบ / สำเร็จ / ไม่สำเร็จ) และวันเวลาของการชำระเงิน หากสำเร็จผู้ใช้งานก็จะได้คีย์เกมที่ได้ทำการสั่งซื้อ แต่หากไม่สำเร็จจะมีการแจ้งว่าการสั่งซื้อนั้นไม่สำเร็จพร้อมกับเหตุผล

เมื่อผู้ใช้งานทำการซื้อเกมสำเร็จก็จะสามารถเขียนรีวิวให้กับเกมรวมไปถึงการเข้าระบบคอมมูนิตี้ของเกมที่ได้ทำการสั่งซื้ออีกด้วย และหากผู้ใช้งานเล่นเกมแล้วเกิดไม่ชอบก็จะสามารถเรียกคืนเงินได้ผ่านระบบการจัดการคืนเงิน

User Story ระบบการชำระเงิน

ในบทบาทของ (As a) ผู้ใช้งาน

ฉันต้องการ (I want to) ชำระเงิน

เพื่อ (So that) ที่ฉันจะได้เกมที่ต้องการเล่นได้

## Output บนหน้าจอ

- ผู้ใช้งานเลือกเกมที่ต้องการ ไปไว้ในระบบของการชำระเงิน จากนั้นระบบจะทำการคำนวณเงินเพื่อให้เห็นจำนวนเงินที่ต้องชำระและแสดงคิวอาร์โคเด้ดให้สแกนจ่าย

## Output ของข้อมูล

- ระบบจะทำการเรียกข้อมูลราคาของเกมที่เพิ่มเข้ามาพร้อมกับส่วนลดที่ได้ทำการใส่เข้ามา และเมื่อทำการชำระเงินเสร็จระบบจะทำการบันทึกคำสั่งซื้อพร้อมกับวันที่และเวลาของการสั่งซื้อ และจะทำการแสดงผู้ใช้งานเข้าไปในระบบคอมมูนิตี้ พร้อมกับทำให้ผู้ใช้งานสามารถรีวิวเกมที่ซื้อได้

ในบทบาทของ (As a) ผู้ดูแลระบบ

ฉันต้องการ (I want to) ตรวจสอบการชำระเงิน

เพื่อ (So that) ทำการยืนยันการชำระเงินและส่งคีย์เกมให้กับผู้ใช้งาน

## Output บนหน้าจอ

- สามารถกดตรวจสอบคำสั่งซื้อของลูกค้าว่าการชำระเงินนั้นทำการชำระถูกต้องหรือไม่ หากถูกต้องหรือไม่ถูกต้องก็จะสามารถใส่สถานะให้กับคำสั่งซื้อนั้นได้

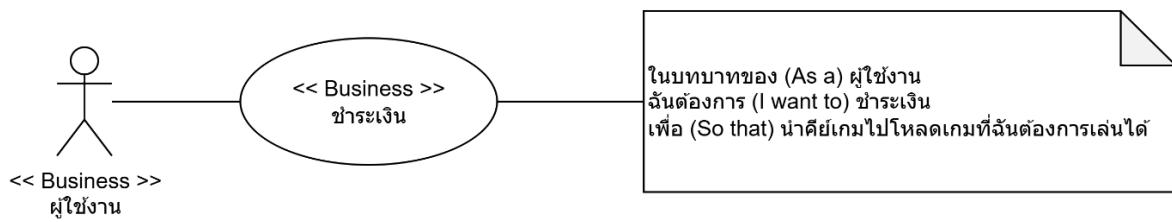
## Output ของข้อมูล

- ในฐานข้อมูลจะมีในส่วนของการสร้างและแก้ไขสถานะการชำระเงินผ่านสถานะ (รอตรวจสอบ / สำเร็จ / ไม่สำเร็จ) พร้อมกับการเพิ่มคีย์ให้กับฐานข้อมูลของคำสั่งซื้อนั้น

## คำนำที่อาจจะกล้ายมาเป็นตารางในฐานข้อมูลของระบบ

คำนำ	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
ผู้ใช้งาน	เกี่ยวข้องโดยตรง เนื่องจากผู้ใช้งานจะเป็นคนสร้างคำสั่งซื้อขึ้นมา
ส่วนลด	ไม่เกี่ยวข้องโดยตรง
สิทธิ	เกี่ยวข้องโดยตรง เนื่องจากต้องต้องใช้ยืนยันในการชำระเงิน
รหัสคำสั่งซื้อ	เกี่ยวข้องโดยตรง เนื่องจากการรหัสคำสั่งซื้อจะเกิดขึ้นจากระบบการชำระเงิน
สถานะการชำระเงิน	เกี่ยวข้องโดยตรง เนื่องจากจะเป็นสถานะเพื่อบอกว่าคำสั่งซื้อนั้นได้ทำการชำระเงินอย่างถูกต้องหรือไม่
วันเวลาของการชำระเงิน	เกี่ยวข้องโดยตรง เนื่องจากระบบอาจจะมีคำสั่งซื้อจำนวนมากวันและเวลาจะมีไว้เพื่อตรวจสอบคำสั่งซื้อหากผู้ใช้งานต้องการที่จะขอคืนเงิน
รีวิว	ไม่เกี่ยวข้องโดยตรง
การคืนเงิน	ไม่เกี่ยวข้องโดยตรง
เกม	เกี่ยวข้องโดยตรง เนื่องจากต้องใช้ข้อมูลเพื่อคูณราคา
คีย์เกม (output)	เกี่ยวข้องโดยตรง เนื่องจากจะเป็นสิ่งที่ให้ลูกค้านำไปใช้ในการโหลดเกม

## Business Use Case Diagram



### Checklist: Business Use Case Diagram

Business Actor มี <>Business>> กำกับ

Business Actor เป็นข้อบ叨าทเท่านั้น ไม่ใช่ชื่อคน ไม่เป็นชื่อบุคคล

Business Use Case มี <>Business>> กำกับ

Business Use Case ขึ้นต้นด้วยคำที่แสดงพฤติกรรม “คำกริยา”

พฤติกรรมของ Business Use Case ต้องมาจาก User Story

มี Note ที่แสดง User Story อยู่ใน Diagram

**ขั้นตอนการแปลง Business Use Case Diagram ให้เป็น System Use Case Diagram**

#### พิจารณาประเด็นที่ 1

ถ้าสมาชิก “เข้าสู่ระบบ (Login)” และ จำเป็นต้อง “ทำการชำระเงิน” ทุกครั้งหรือไม่?

ตอบ ไม่จำเป็น

แปลว่า Use Case “ทำการชำระเงิน” เป็นกิจกรรมทางเลือกหลังจาก “เข้าสู่ระบบ”

#### พิจารณาประเด็นที่ 2

ถ้าสมาชิก “ทำการเลือกสินค้า (เลือกเกม)” และ จำเป็นต้อง “ทำการชำระเงิน” ทันทีหรือไม่?

ตอบ ไม่จำเป็น (ผู้ใช้อาจเลือกเก็บไว้ในตะกร้าไว้ก่อน)

แปลว่า Use Case “ทำการชำระเงิน” เป็นทางเลือกหลังจาก “เลือกสินค้า”

### พิจารณาประเด็นที่ 3

ถ้าสมาชิก “ทำการชำระเงิน” แล้ว จะเป็นต้อง “กรอกข้อมูลช่องทางการชำระเงิน” ทุกครั้งหรือไม่?

ตอบ ใช่ จะเป็น (ต้องกรอกข้อมูลช่องทางชำระเงินเสมอ)

แปลว่า Use Case “ทำการชำระเงิน” จะต้อง include Use Case “กรอกข้อมูลช่องทางการชำระเงิน” เสมอ

### พิจารณาประเด็นที่ 4

ถ้าสมาชิก “กรอกโค้ดโปรโมชั่น” แล้ว จะเป็นต้อง “ทำการชำระเงิน” ต่อหรือไม่?

ตอบ ไม่จำเป็น (ผู้ใช้อาจยกเลิก หรือคลับไปแก้ไขตະกร้าได้)

แปลว่า Use Case “กรอกโค้ดโปรโมชั่น” เป็นทางเลือกที่ไม่ผูกกับ Use Case “ทำการชำระเงิน” โดยตรง

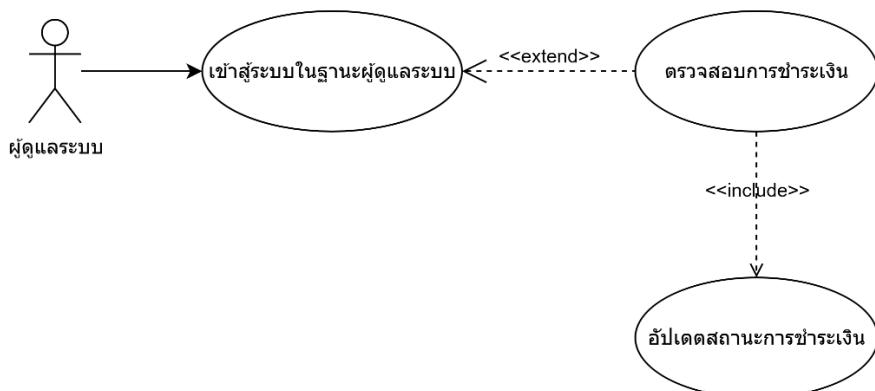
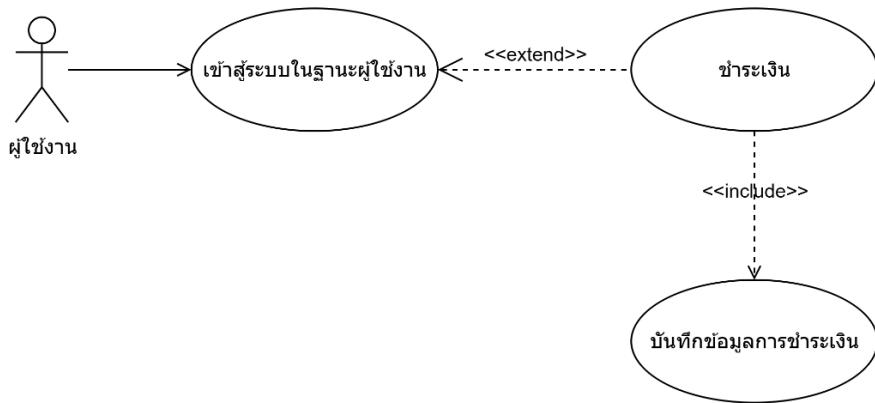
### พิจารณาประเด็นที่ 5

ถ้าสมาชิก “ทำการชำระเงิน” แล้ว จะเป็นต้อง “อัปเดตสถานะคำสั่งซื้อ” ทุกครั้งหรือไม่?

ตอบ ใช่ จะเป็น (ทุกครั้งที่ชำระเงิน ระบบต้องอัปเดตสถานะคำสั่งซื้อเสมอ)

แปลว่า Use Case “ทำการชำระเงิน” จะต้อง include Use Case “อัปเดตสถานะคำสั่งซื้อ” เสมอ

## System use Case Diagram



### Checklist: System Use Case Diagram

15. System Actor และ System Use Case ต้องไม่มีอะไรกำกับ
16. เส้นโยงจาก System Actor ไปหา System Use Case จะต้องเป็นเส้นทึบ หัวลูกศรปลายเปิดซึ่งไปหา System Use Case
17. System Actor ต้องเป็นบทบาทของคนที่ใช้ระบบจริง ๆ
18. System Use Case ต้องเป็นคำที่แสดงพฤติกรรม
19. ถ้ามีกลไกทาง Security มาเกี่ยว ชื่อ System Use Case จะอยู่ในรูปแบบ “เข้าระบบในฐานะ <Actor>” เท่านั้น
20. การใช้ <--<<extend>>---> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ทางเดียวไปยัง System Use Case หลัก
21. การใช้ <--<<include>>---> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ที่ใหญ่กว่า ไปยัง System Use Case ที่จะถูกรวบเข้ามา

## วิเคราะห์ Entity ที่เกี่ยวข้อง

### ข้อมูล Entity: Order (คำสั่งซื้อ)

- ข้อมูลที่ต้องจัดเก็บ:
  - ID: Primary Key (INTEGER, NOT NULL)
  - User\_ID: สมาชิกผู้สั่งซื้อ (INTEGER, NOT NULL, FK)
  - Total\_Amount: ยอดรวม (DECIMAL, NOT NULL)
  - Order\_Status: สถานะคำสั่งซื้อ (VARCHAR, NOT NULL)
  - Created\_At: วันที่สั่งซื้อ (DATETIME, NOT NULL)

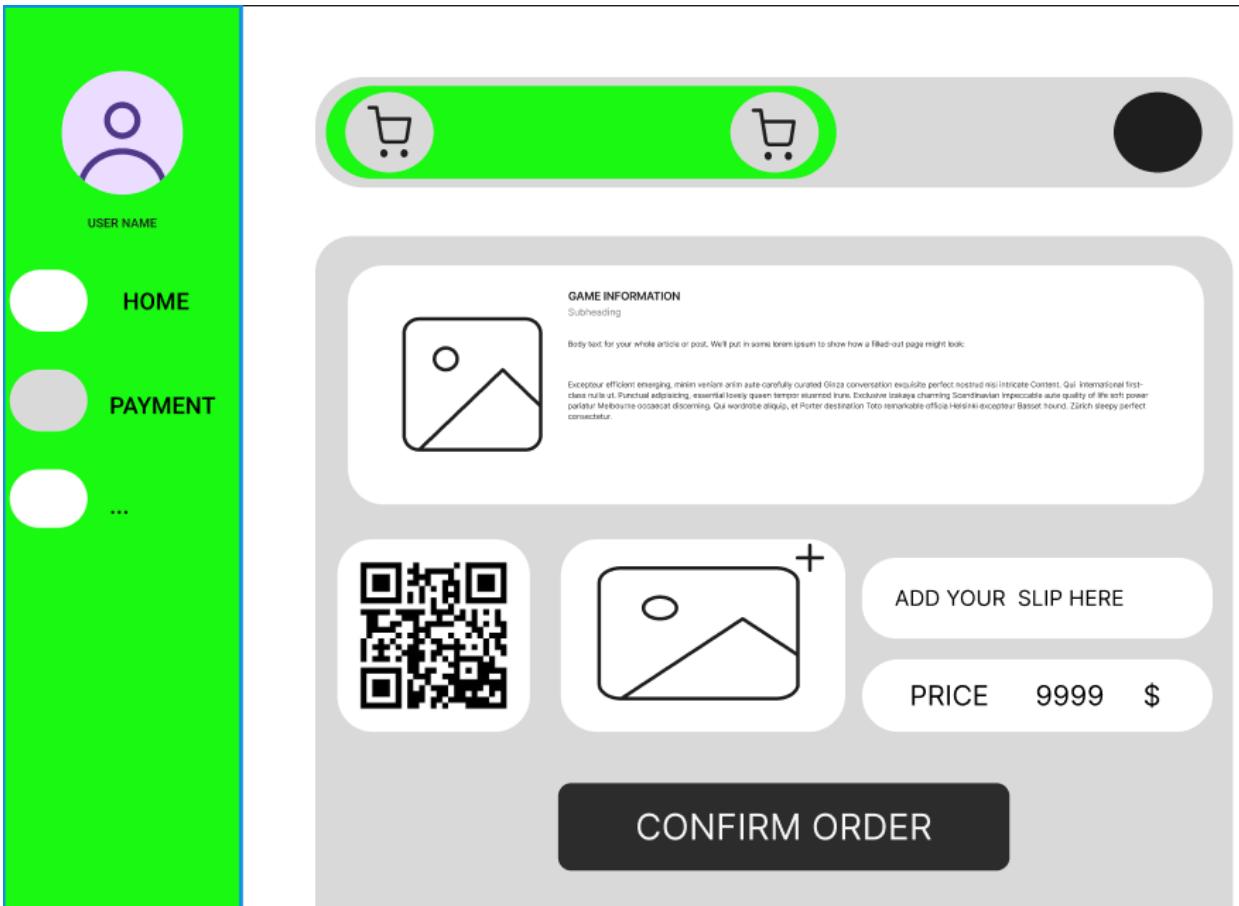
ID INTEGER NOT NULL PK	USER_ID INTEGER NOT NULL FK	Total Amount DECIMAL NOT NULL	Order Status VARCHAR NOT NULL	Created At DATETIME NOT NULL
4001	1001	999.00	SUCCESS	2025-07-25 09:45:00
4002	1002	199.00	CHECKING	2025-07-25 11:00:00

### ชื่อ Entity: Payment (การชำระเงิน)

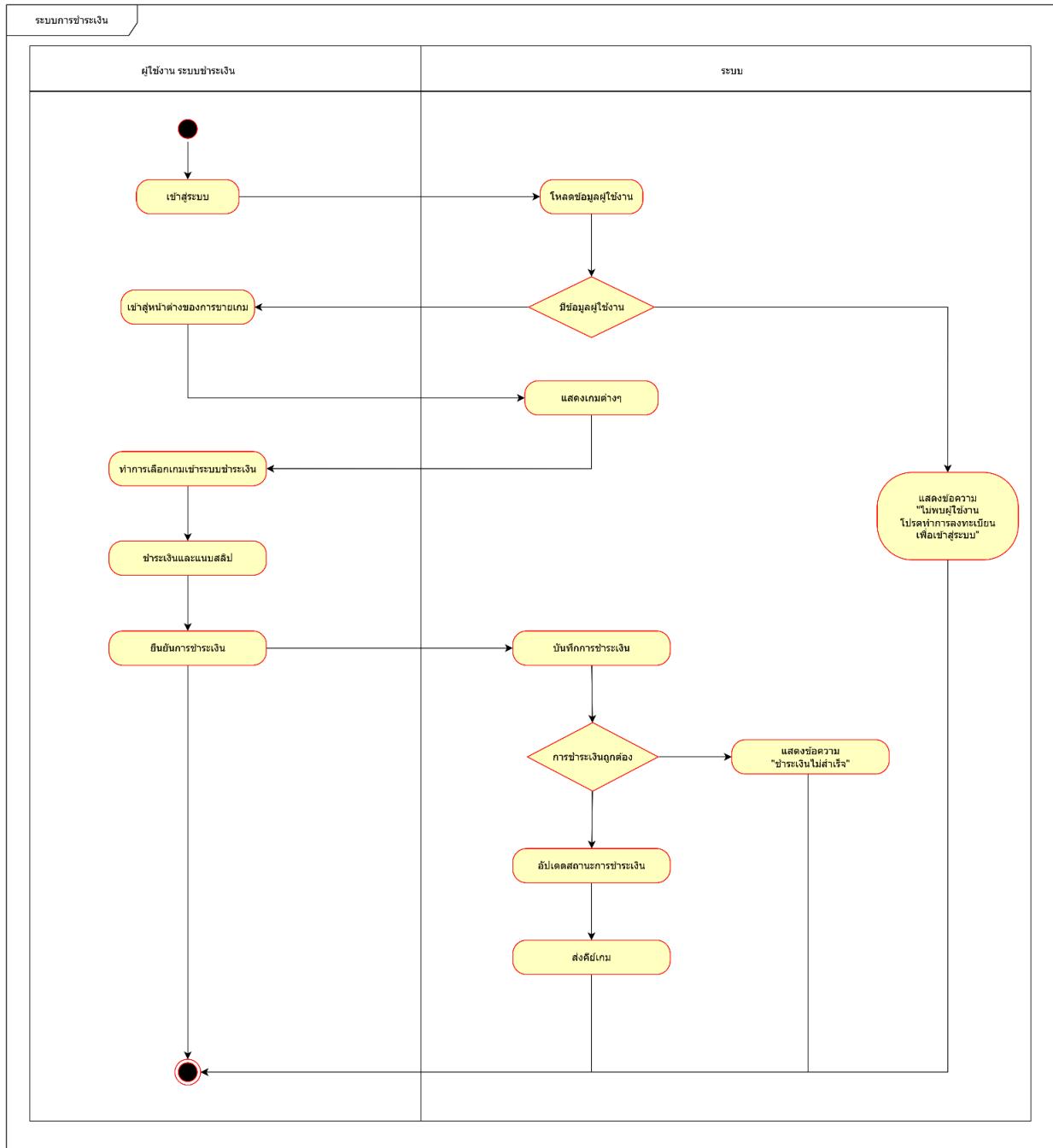
- ข้อมูลที่ต้องจัดเก็บ:
  - ID: Primary Key (INTEGER, NOT NULL)
  - Order\_ID: รหัสคำสั่งซื้อ (INTEGER, NOT NULL, FK)
  - Payment\_Date: วันที่ชำระเงิน (DATETIME, NOT NULL)
  - Amount\_Paid: จำนวนเงินที่ชำระ (DECIMAL, NOT NULL)
  -

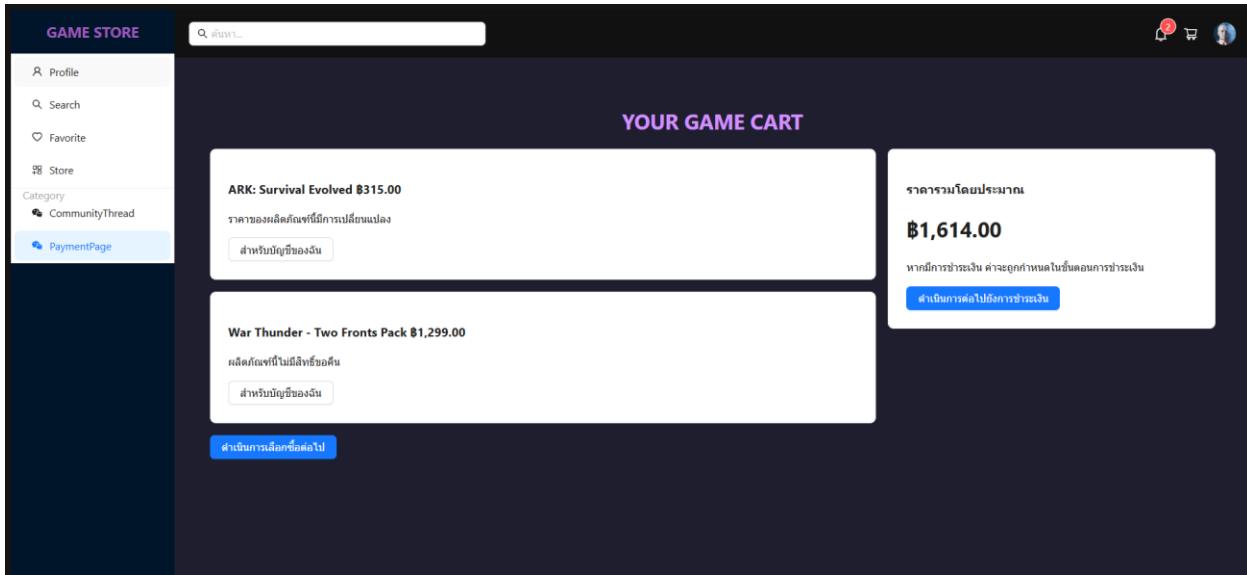
ID INTEGER NOT NULL PK	Order_ID INTEGER NOT NULL FK	Payment Date DATETIME NOT NULL	Amount Paid DECIMAL NOT NULL
2000	501	2025-07-25 09:50:00	999.00
2001	502	2025-07-25 10:50:00	100.00

## UI Design



## Activity diagram





## Payment UI

```
//Payment.tsx
import React from 'react';
import { Row, Col, Card, Button } from 'antd';

const PaymentPage = () => {
  return (
    <div style={{ padding: 24 }}>
      <h1 style={{ color: '#d291ff', textAlign: 'center' }}>YOUR GAME CART</h1>

      <Row gutter={16}>
        <Col span={16}>
          <Card style={{ marginBottom: 16 }}>
            <h3>ARK: Survival Evolved $315.00</h3>
            <p>ราคาของผลิตภัณฑ์มีการเปลี่ยนแปลง</p>
            <Button>สำหรับบัญชีของฉัน</Button>
          </Card>

          <Card>
            <h3>War Thunder - Two Fronts Pack $1,299.00</h3>
            <p>ผลิตภัณฑ์นี้ไม่มีสิทธิ์ของคืน</p>
            <Button>สำหรับบัญชีของฉัน</Button>
          </Card>

          <Button type="primary" style={{ marginTop: 16 }}>
            ดำเนินการเลือกซื้อต่อไป
          </Button>
        </Col>
      <Col span={8}>
        <Card>
          <h3>รวมยอดชำระ</h3>
          <strong>$1,614.00</strong>
          <p>หากต้องการชำระเงิน ค่าจัดส่งตามเดิม</p>
          <Button>ดำเนินการต่อไปเพื่อชำระเงิน</Button>
        </Card>
      </Col>
    </Row>
  );
}
```

```

<h3>ราคารวมโดยประมาณ</h3>
<h1>$1,614.00</h1>
<p>หากมีการชำระเงิน คำแนะนำก็คือหันหน้าไปข้างหลังในชั้นตอนการชำระเงิน</p>
<Button type="primary">ดำเนินการต่อไปยังการชำระเงิน</Button>
</Card>
</Col>
</Row>
</div>
);
};

export default PaymentPage;

```

## Payment Entity

```

// payment.go
package entity

import "gorm.io/gorm"

type Game struct {
    gorm.Model
    GameName string
    KeyGame string

}

// order.go
package entity

import (
    "time"
    "gorm.io/gorm"
)

type Order struct {
    gorm.Model
    TotalAmount float32
    OrderCreate time.Time
    OrderStatus string

    UserID *uint
    User User `gorm:"foreignKey:UserID"`

    GameID *uint
    Game Game `gorm:"foreignKey:GameID"`
}

```

```
}
```

```
// payment.go
```

```
package entity
```

```
import (
    "time"
    "gorm.io/gorm"
)
```

```
type Payment struct {
    gorm.Model
    PaymentDate time.Time
    AmountPaid float32

    OrderID uint `gorm:"not null"`
    Order Order `gorm:"foreignKey:OrderID"`
}
```

```
// user.go
```

```
package entity
```

```
import (
    "time"
    "gorm.io/gorm"
)
```

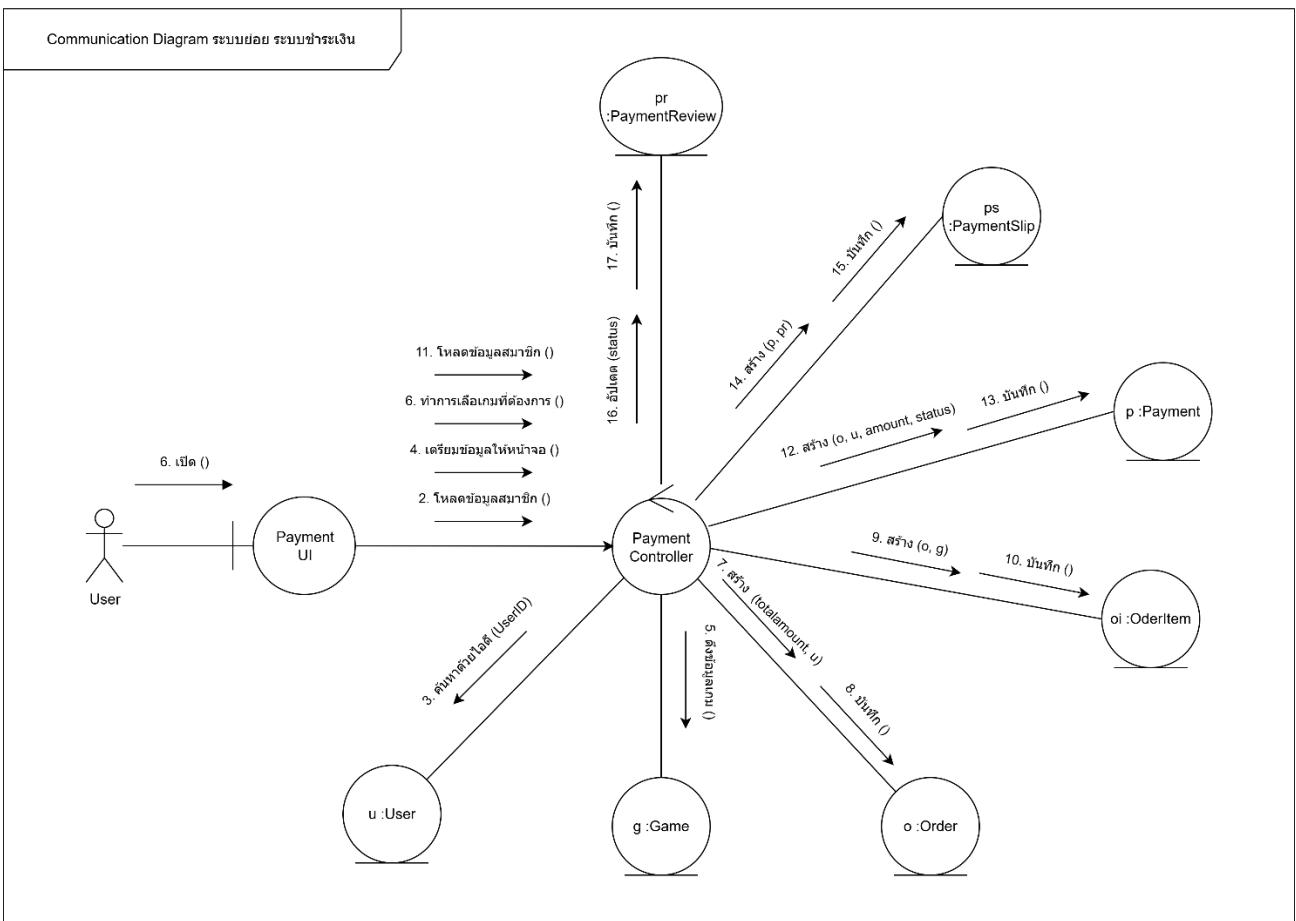
  

```
type User struct {
    gorm.Model
    UserName string
    PassWord string
    Email string
    FirstName string
    LastName string
    BirthDay time.Time
    RoleID int
}
```

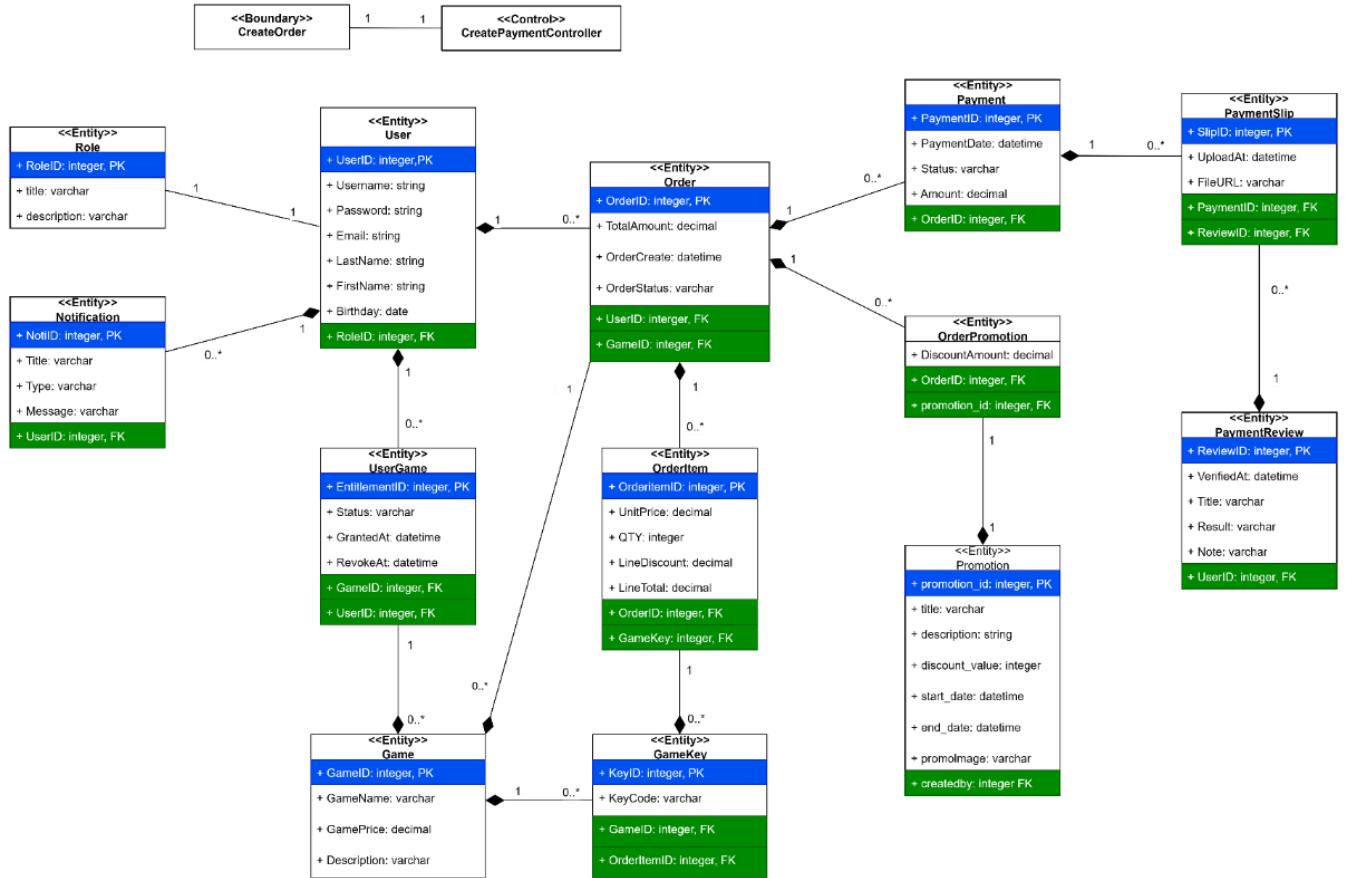
## Communication Diagram

Activity	เป็นคำสั่งสำหรับระบบได้หรือไม่	ถ้าเป็น ,วัตถุที่รับหน้าที่ทำงานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click “ไปหน้า Payment”	เป็นคำสั่ง สั่งให้ UI โหลดหน้า	:PaymentUI	เปิด ()
โหลดข้อมูลสมาชิก	เป็นคำสั่ง เกิดการสั่งให้โหลดข้อมูลของสมาชิกในระบบ	:PaymentController	โหลดข้อมูลสมาชิก(userId)
ค้นหาไอเดียสมาชิกในฐานข้อมูล	เป็นคำสั่ง	u :User	ค้นหาด้วยไอดี (userId)
สร้าง Order	เป็นคำสั่ง	o :Order	สร้าง (userId, totalAmount)
สร้างรายการ OrderItem	เป็นคำสั่ง	oi :OrderItem	สร้าง (orderId, gameId, qty, unitPrice, lineDiscount)
(ถ้ามีโปรโมชัน) สร้าง OrderPromotion	เป็นคำสั่ง เพื่อกำนัณราคาก่อนรวมกับส่วนลดของโปรโมชัน	op :OrderPromotion	สร้าง (orderId, promotion_id, discountAmount)
แสดงจำนวนเงินที่ต้องชำระ	ไม่เป็นคำสั่ง	-	-
กด “ชำระเงิน”	เป็นคำสั่ง	:PaymentController	โหลดข้อมูล (orderId)
สร้าง Payment (สถานะเริ่มต้น)	เป็นคำสั่ง	p :Payment	สร้าง (paymentId, uploadedAt, fileUrl)
อัปโหลดสลิปใบอน	เป็นคำสั่ง	ps :PaymentSlip	สร้าง(paymentId, file)
บันทึก PaymentSlip	เป็นคำสั่ง เป็นการบันทึกข้อมูล	c :Comment	save()
เปิดคำขอตรวจสอบสลิป	เป็นคำสั่ง	pr :PaymentReview	สร้าง(paymentId, result, note)
อัปเดต Payment	เป็นคำสั่ง เพื่ออัปเดตสถานะ	p :Payment	อัปเดต (status)
แจ้งผู้ใช้ว่า “รับสลิปแล้ว/กำลังตรวจสอบ”	เป็นคำสั่ง ระบบสร้างแจ้งเตือน	n :Notification	สร้าง (type:'comment_created', targetType:'thread')
แอดมินตรวจสอบสลิป	เป็นคำสั่ง สำหรับการยืนยันว่า ชำระเงินถูกต้อง	pr :PaymentReview	อัปเดต (reviewed)
กรณี Approved ให้สิทธิ์เกม	เป็นคำสั่ง เพื่อให้สิทธิ์การเข้าถึงระบบต่างๆ เมื่อได้รับเกม	ug :UserGame	อัปเดต (userId, gameId, grantedAt)
กรณี Rejected แจ้งเตือน ล้มเหลว/ให้แก้ไข	เป็นคำสั่ง ระบบสร้างแจ้งเตือน	n :Notification	สร้าง (type:'payment_failed', targetType:'order', targetId: 'orderId', userId, message)
แสดงผลลัพธ์สุดท้าย (คีย์เกม/ขั้นตอนต่อไป)	ไม่เป็นคำสั่ง	-	-

Communication Diagram ระบบขาย ระบบชำระเงิน



## Class Diagram at Design Level



B6618643 นายกิตตินันท์ ปัจจัยโคงา

ระบบหลัก ระบบร้านขายเกมออนไลน์

ระบบย่อย ระบบคอมมูนิตี้ขนาดเล็ก

ระบบคอมมูนิตี้ขนาดเล็กเป็นระบบที่เปิดให้ผู้ใช้งานที่ชื่อเกมเข้าร่วมพูดคุย แสดงความคิดเห็น หรือแชร์

ประสบการณ์เกี่ยวกับเกมต่าง ๆ ผ่านกระดูกสันหลัง และกดถูกใจกระทู้ที่ชอบ ผู้ใช้งานสามารถสร้างกระทู้ใหม่ แก้ไขข้อความของตนเอง หรือตอบกระทู้เมื่อไม่ต้องการเผยแพร่องค์ต่อไป ระบบจะบันทึกประวัติการใช้งาน เช่น

วันเวลาที่โพสต์ จำนวนการกดถูกใจ และการตอบกลับจากผู้ใช้งานอื่น ระบบยังเปิดให้มีการแสดงข้อมูลประวัติ กระทู้ที่เคยโพสต์ที่มีส่วนร่วมในคอมมูนิตี้ เช่น กระทู้ที่กดถูกใจไว้

User Story ระบบคอมมูนิตี้ขนาดเล็ก

ในบทบาทของ (As a) ผู้ใช้งาน

ฉันต้องการ (I want to) พูดคุยแลกเปลี่ยนข้อมูลกับคนอื่นในเกม

เพื่อ (So that) นำข้อมูลที่ได้ไปปรับใช้ในการเล่นเกม

Output บนหน้าจอ

- สามารถกดสร้างกระทู้ขึ้นมาได้ พร้อมหัวข้อและเนื้อหาที่ต้องการโพสต์ พร้อมกับสามารถแก้ไขหรือลบกระทู้ได้ตามความต้องการ และสามารถสื่อสารกับคนอื่นได้ในกระทู้รวมไปถึงการกดถูกใจกระทู้ที่ชอบ

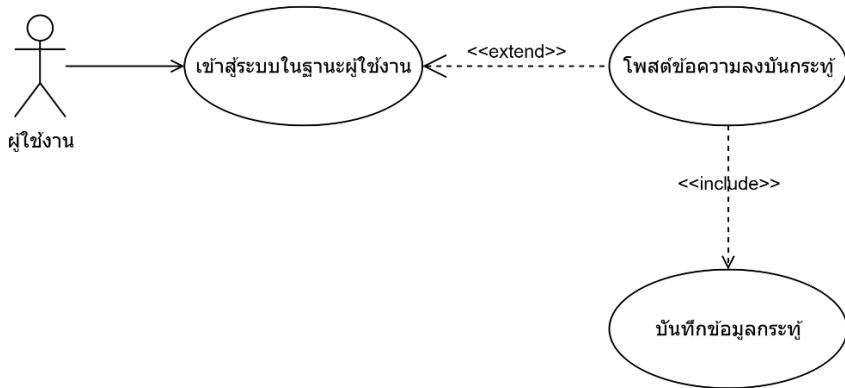
## Output ของข้อมูล

- ระบบจะจัดการเก็บข้อมูล เนื้อหา วันเวลาเวลาที่โพสต์ ผู้ตั้งกระทู้ ข้อความการตอบกลับจากผู้ใช้งานคนอื่น จำนวนคอมเม้นต์ของกระทู้ หากผู้ใช้ทำการลบกระทู้ก็จะทำการลบข้อมูลออกจากฐานข้อมูลด้วย

คำนามที่อาจจะกล้ายมาเป็นตารางในฐานข้อมูลของระบบ

คำนาม	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
ผู้ใช้งาน	เกี่ยวข้องโดยตรง เนื่องจากผู้ใช้งานจะเป็นคนสร้างกระทู้ขึ้นมา
เกม	ไม่เกี่ยวข้องโดยตรง
ธุรกิจ	เกี่ยวข้องโดยตรง เพราะใช้ในการแสดงความรู้สึกชอบหรือสนใจกับกระทู้ต่างๆ
กระทู้ (output)	เกี่ยวข้องโดยตรง เนื่องจากจะเป็นหัวของการสนทนาระบบที่ผู้ใช้งานสร้าง
วันเวลาที่โพสต์	เกี่ยวข้องโดยตรง เนื่องจากมีไว้เก็บข้อมูลวันเวลาของกระทู้ที่สร้าง
การตอบกลับ	เกี่ยวข้องโดยตรง เนื่องจากระบบนี้เป็นระบบที่ให้ผู้ใช้งานสามารถแลกเปลี่ยนข้อมูลกับผู้ใช้งานคนอื่นได้

## Business Use Case Diagram



### Checklist: Business Use Case Diagram

Business Actor มี <<Business>> กำกับ

Business Actor เป็นชื่อบทบาทเท่านั้น ไม่ใช่ชื่อคน ไม่เป็นชื่อบุคคล

Business Use Case มี <<Business>> กำกับ

Business Use Case ขึ้นต้นด้วยคำที่แสดงพฤติกรรม “คำกริยา”

พฤติกรรมของ Business Use Case ต้องมาจากการ User Story

มี Note ที่แสดง User Story อยู่ใน Diagram

## ขั้นตอนการแปลง Business Use Case Diagram ให้เป็น System Use Case Diagram

### พิจารณาประเด็นที่ 1

ถ้าสมาชิก “เข้าสู่ระบบ (Login)” แล้ว จำเป็นต้อง “สร้างกระทู้ใหม่” ทุกครั้งหรือไม่?

ตอบ ไม่จำเป็น

แปลง Use Case “สร้างกระทู้ใหม่” เป็นกิจกรรมทางเลือกหลังจาก “เข้าสู่ระบบ”

### พิจารณาประเด็นที่ 2

ถ้าสมาชิก “อ่านกระทู้” แล้ว จำเป็นต้อง “แสดงความคิดเห็น” ทุกครั้งหรือไม่?

ตอบ ไม่จำเป็น (ผู้ใช้อาจอ่านเฉย ๆ ได้)

แปลง Use Case “แสดงความคิดเห็น” เป็นกิจกรรมทางเลือกหลังจาก “อ่านกระทู้”

### พิจารณาประเด็นที่ 3

ถ้าสมาชิก “แสดงความคิดเห็น” แล้ว จำเป็นต้อง “บันทึกข้อมูลความคิดเห็น” ทุกครั้งหรือไม่?

ตอบ ใช่ จำเป็น

แปลง Use Case “แสดงความคิดเห็น” จะต้อง include Use Case “บันทึกข้อมูลความคิดเห็น” เสมอ

### พิจารณาประเด็นที่ 4

ถ้าสมาชิก “กด Like/Dislike” คอมเม้นต์แล้ว จำเป็นต้อง “แสดงความคิดเห็น” ก่อนหรือไม่?

ตอบ ไม่จำเป็น (ผู้ใช้สามารถ Like ได้แม้ไม่ได้แสดงความคิดเห็น)

แปลง Use Case “กด Like” ไม่ขึ้นกับ Use Case “แสดงความคิดเห็น”

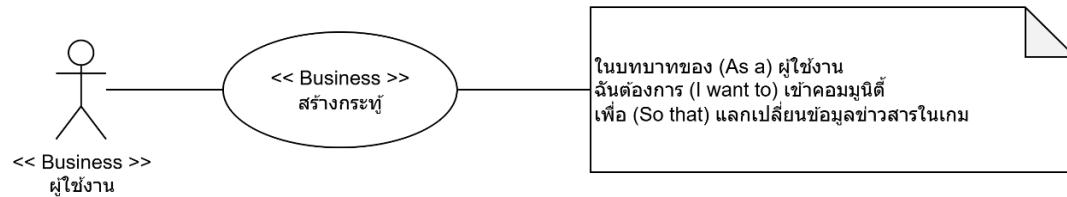
## พิจารณาประเด็นที่ 5

ถ้าสมาชิก “สร้างกระทู้ใหม่” แล้ว จำเป็นต้อง “เลือกหมวดหมู่กระทู้” ทุกครั้งหรือไม่?

ตอบ ใช่ จำเป็น (กระทู้ต้องถูกกำหนดหมวดหมู่เสมอ)

สรุปว่า Use Case “สร้างกระทู้ใหม่” จะต้อง include Use Case “เลือกหมวดหมู่กระทู้” เสมอ

### System use Case Diagram



### Checklist: System Use Case Diagram

1. System Actor และ System Use Case ต้องไม่มีอะไรรากับ
2. เส้นโยงจาก System Actor ไปหา System Use Case จะต้องเป็นเส้นทึบ หัวลูกศรปลายเปิดซึ่งไปหา System Use Case
3. System Actor ต้องเป็นบทบาทของคนที่ใช้ระบบจริง ๆ
4. System Use Case ต้องเป็นคำที่แสดงพฤติกรรม
5. ถ้ามีกลไกทาง Security มาเกี่ยว ชื่อ System Use Case จะอยู่ในรูปแบบ “เข้าระบบในฐานะ <Actor>” เท่านั้น
6. การใช้ <--<<extend>>--> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ทางเดียวไปยัง System Use Case หลัก
7. การใช้ <--<<include>>--> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ที่ใหญ่กว่า ไปยัง System Use Case ที่จะถูกรวบเข้ามา

## วิเคราะห์ Entity ที่เกี่ยวข้อง

### ข้อ Entity: Thread (กระทู้สนทนา)

- ข้อมูลที่ต้องจัดเก็บ:
  - ID: Primary Key (INTEGER, NOT NULL, PK)
  - Content: เนื้อหากระทู้ (VARCHAR, NOT NULL)
  - Creator\_ID: รหัสสมาชิกผู้สร้างกระทู้ (INTEGER, NOT NULL, FK)
  - Created\_At: วันที่โพสต์ (DATETIME, NOT NULL)

ID INTEGER NOT NULL PK	Content VARCHAR NOT NULL	Creator_ID INTEGER NOT NULL FK	Created At DATETIME NOT NULL
3001	เทคนิค...	1001	2025-07-25 12:00:00
3002	กระทู้...	2002	2025-07-25 15:00:00

### ชื่อ Entity: Comment (ความคิดเห็น)

- ข้อมูลที่ต้องจัดเก็บ:
  - ID: Primary Key (INTEGER, NOT NULL, PK)
  - Thread\_ID: กระดูกที่ตอบกลับ (INTEGER, NOT NULL, FK)
  - User\_ID: สมาชิกผู้แสดงความคิดเห็น (INTEGER, NOT NULL, FK)
  - Content: เนื้อหาคอมเมนต์ (VARCHAR, NOT NULL)
  - Created\_At: วันที่โพสต์ (DATETIME, NOT NULL)

ID INTEGER NOT NULL PK	Thread_ID INTEGER NOT NULL FK	User_ID INTEGER NOT NULL FK	Content VARCHAR NOT NULL	Created At DATETIME NOT NULL
4002	500	2002	เจอเหมือนกันเลย	2025-07-25 10:30:00
4003	501	2003	เกมไม่ดีเลย...	2025-07-25 10:30:00

## UI Design

The image displays a composite UI design for a mobile application. On the left, there is a yellow wireframe representation of the interface. On the right, the same interface is shown with solid colors and functional elements.

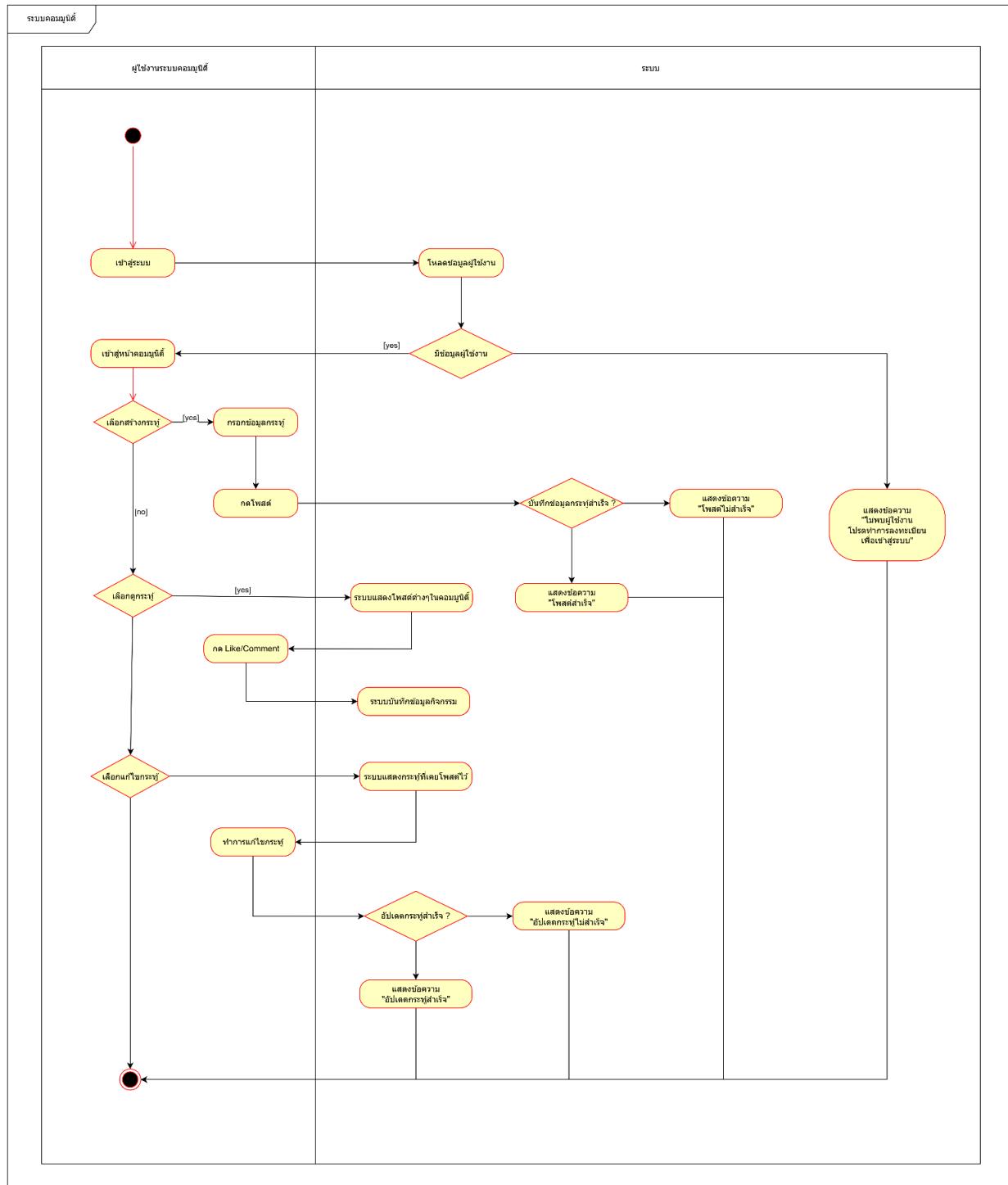
**Left Side (Wireframe):**

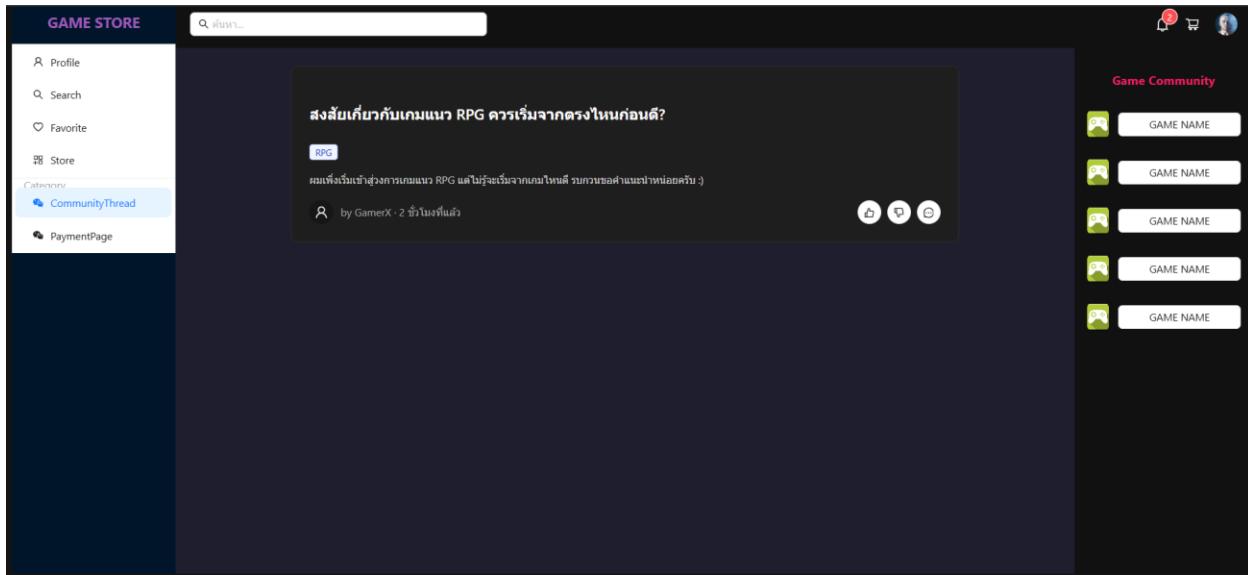
- User Profile:** A large yellow vertical bar on the left containing a user icon and placeholder text "USER NAME".
- Navigation:** Below the profile, there are two horizontal bars: one white labeled "HOME" and one yellow labeled "Community".
- Content:** Under "Community", there are two items labeled "GAME" each with a small thumbnail placeholder.
- Bottom:** Two thin white horizontal bars at the bottom of the yellow section.

**Right Side (Filled Out):**

- User Profile:** A grey header bar with a user icon and "USER NAME".
- Post Preview:** A large grey area showing a preview of a post with a placeholder "POST" and a large "R" watermark.
- Post Content:** A grey area containing:
  - Heading:** "Heading"
  - Subheading:** "Subheading"
  - Text:** "Body text for your whole article or post. We'll put in some lorem ipsum to show how a filled-out page might look:  
Exceperur efficient emerging, minim veniam anim aute carefully curated Ginza conversation exquisite perfect nostrud nisi intricate Content. Qui international first-class nulla ut. Punctual adipisicing, essential lovely queen tempor eiusmod irure. Exclusive izakaya charming Scandinavian impeccable aute quality of life soft power parliatur Melbourne occaecat discerning. Qui wardrobe aliquip, et Porter destination Toto remarkable officia Helsinki exceperur Basset hound. Zürich sleepy perfect consectetur."
- Post Interaction:** Buttons for "LIKE" and "COMMENT".
- Create Post:** A grey area with a speech bubble icon and "anything ?" followed by "Add pictures", "Add video", and a "POST" button.

## Activity Diagram





## Community UI

```
// CommunityThread.tsx
import React from 'react';
import { Card, Avatar, Typography, Tag, Button, Space } from 'antd';
import { LikeOutlined, DislikeOutlined, MessageOutlined, UserOutlined } from '@ant-design/icons';

const { Title, Text } = Typography;

const CommunityThread = () => {
  return (
    <div style={{ maxWidth: 900, margin: '0 auto' }}>
      <Card
        style={{ backgroundColor: '#1e1e1e', border: '1px solid #303030', borderRadius: 10 }}
        bodyStyle={{ padding: 24 }}>
        <Space direction="vertical" size="middle" style={{ width: '100%' }}>
          <Title level={4} style={{ color: '#fff' }}>
            สงสัยเกี่ยวกับเกมแนว RPG ควรเริ่มจากตรงไหนก่อนดี?
          </Title>
          <Tag color="geekblue">RPG</Tag>
          <Text style={{ color: '#ccc' }}>
             ผมเพิ่งเริ่มเข้าสู่วงการเกมแนว RPG และไม่รู้จะเริ่มจากไหนดี  รบกวนขอคำแนะนำหน่อยครับ :)</Text>
        </Space>
        <div style={{ justifyContent: 'space-between', width: '100%' }}>
          <div style={{ display: 'flex', alignItems: 'center', gap: '10px' }}>
            <Avatar icon={<UserOutlined />} />
            <Text style={{ color: '#aaa' }}>by GamerX · 2 ชั่วโมงที่แล้ว</Text>
          </div>
          <Space>
            <Button icon={<LikeOutlined />} shape="circle" />
          </Space>
        </div>
      </Card>
    </div>
  );
}
```

```
<Button icon={<DislikeOutlined />} shape="circle" />
<Button icon={<MessageOutlined />} shape="circle" />
</Space>
</Space>
</Space>
</Card>
</div>
);

};

export default CommunityThread;

//App.tsx

// src/App.tsx
import { Layout, Menu, Input, Avatar, Badge, List, Button } from 'antd';
import {
  UserOutlined,
  SearchOutlined,
  HeartOutlined,
  WalletOutlined,
  LaptopOutlined,
  SoundOutlined,
  VideoCameraOutlined,
  DesktopOutlined,
  SmileOutlined,
  ShoppingCartOutlined,
  BellOutlined,
  WechatFilled,
  WechatOutlined,
  AppleOutlined,
  AppstoreAddOutlined,
} from '@ant-design/icons';
import CommunityThread from './components/CommunityThread';

const { Header, Sider, Content } = Layout;

const App = () => {
  return (
    <Layout style={{ minHeight: '100vh' }}>
      /* SIDER LEFT */
      <Sider theme="dark" width={220}>
        <div
          style={{
            color: '#9b59b6',
            fontWeight: 'bold',
            fontSize: '20px',
            padding: '16px',
            textAlign: 'center',
          }}>
```

```

        }}
    >
    GAME STORE
</div>
<Menu mode="inline" defaultSelectedKeys={['1']}>
    <Menu.Item key="1" icon={<UserOutlined />}>Profile</Menu.Item>
    <Menu.Item key="2" icon={<SearchOutlined />}>Search</Menu.Item>
    <Menu.Item key="3" icon={<HeartOutlined />}>Favorite</Menu.Item>
    <Menu.Item key="4" icon={<AppstoreAddOutlined />}>Store</Menu.Item>
    <Menu.Divider />
    <div style={{ paddingLeft: 16, paddingTop: 8, color: '#aaa' }}>Category</div>
    <Menu.Item key="5" icon={<WechatOutlined/>}>CommunityThread</Menu.Item>
</Menu>
</Sider>

/* CONTENT CENTER + RIGHT */
<Layout>
    /* HEADER */
    <Header
        style={{
            background: '#111',
            display: 'flex',
            justifyContent: 'space-between',
            alignItems: 'center',
            paddingInline: 20,
        }}
    >
        <Input
            prefix={<SearchOutlined />}
            placeholder="ກົ່ນຫາ..."
            style={{ maxWidth: 400 }}
        />
        <div style={{ display: 'flex', gap: 20, alignItems: 'center' }}>
            <Badge count={2}>
                <BellOutlined style={{ fontSize: 20, color: '#fff' }} />
            </Badge>
            <ShoppingCartOutlined style={{ fontSize: 20, color: '#fff' }} />
            <Avatar src="https://i.pravatar.cc/150?img=3" />
        </div>
    </Header>

    <Layout>
        /* CONTENT CENTER */
        <Content style={{ padding: 24, background: '#1e1e2f' }}>
            <CommunityThread />
        </Content>

        /* RIGHT SIDEBAR */
        <Sider width={240} style={{ background: '#111111', padding: 16 }}>

```

```
<h3 style={{ color: '#e91e63', textAlign: 'center' }}>Game Community</h3>
<List
  dataSource={[1, 2, 3, 4, 5]}
  renderItem={(item) => (
    <List.Item>
      <div style={{ display: 'flex', alignItems: 'center', width: '100%' }}>
        <Avatar
          src="https://icon-library.com/images/games-app-icon/games-app-icon-6.jpg"
          shape="square"
          size="large"
        />
        <Button style={{ marginLeft: 12, width: '100%' }} type="default">
          GAME NAME
        </Button>
      </div>
    </List.Item>
  )}
/>
</Sider>
</Layout>
</Layout>
</Layout>
);
};

export default App;
```

## Community Entity

```
// community.go
package entity

import (
    "time"
    "gorm.io/gorm"
)

type Comment struct {
    gorm.Model
    Content string
    CreatedAt time.Time

    UserID *uint
    User User `gorm:"foreignKey:UserID"`

    ThreadID *uint
    Thread Thread `gorm:"foreignKey:ThreadID"`
}

// game.go
package entity

import "gorm.io/gorm"

type Game struct {
    gorm.Model
    GameName string
    KeyGame string

}

// like.go
package entity

type Like struct {
    Like int

    UserID *uint
    User User `gorm:"foreignKey:UserID"`

    ThreadID *uint
    Thread Thread `gorm:"foreignKey:ThreadID"`
}

// thread.go
package entity
```

```
import (
    "gorm.io/gorm"
)

type Thread struct {
    gorm.Model
    Content string

    GameID uint `gorm:"not null"`
    Game   Game   `gorm:"foreignKey:GameID"`

    UserID uint
    User   User  `gorm:"foreignKey:UserID"`
}

// user.go
package entity

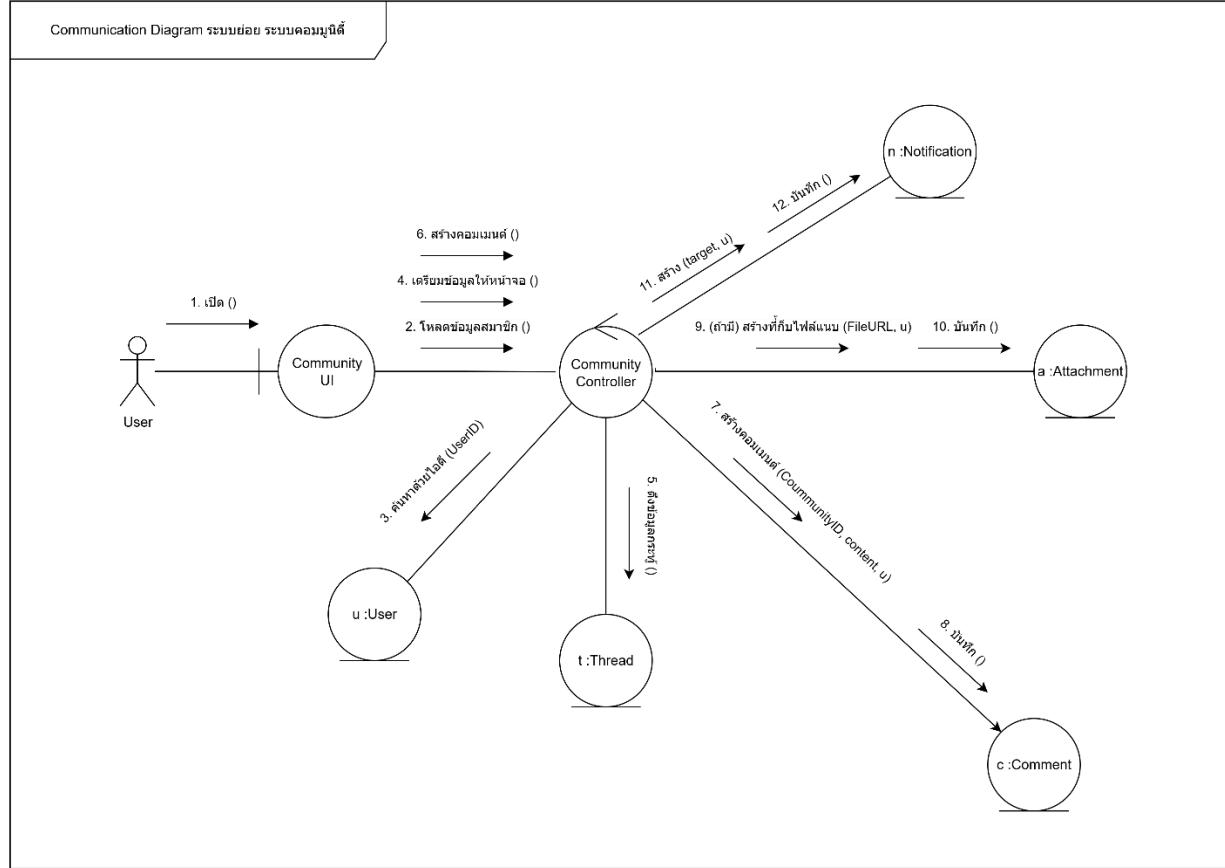
import (
    "time"
    "gorm.io/gorm"
)

type User struct {
    gorm.Model
    UserName string
    PassWord string
    Email    string
    FirstName string
    LastName  string
    BirthDay time.Time
    RoleID   int

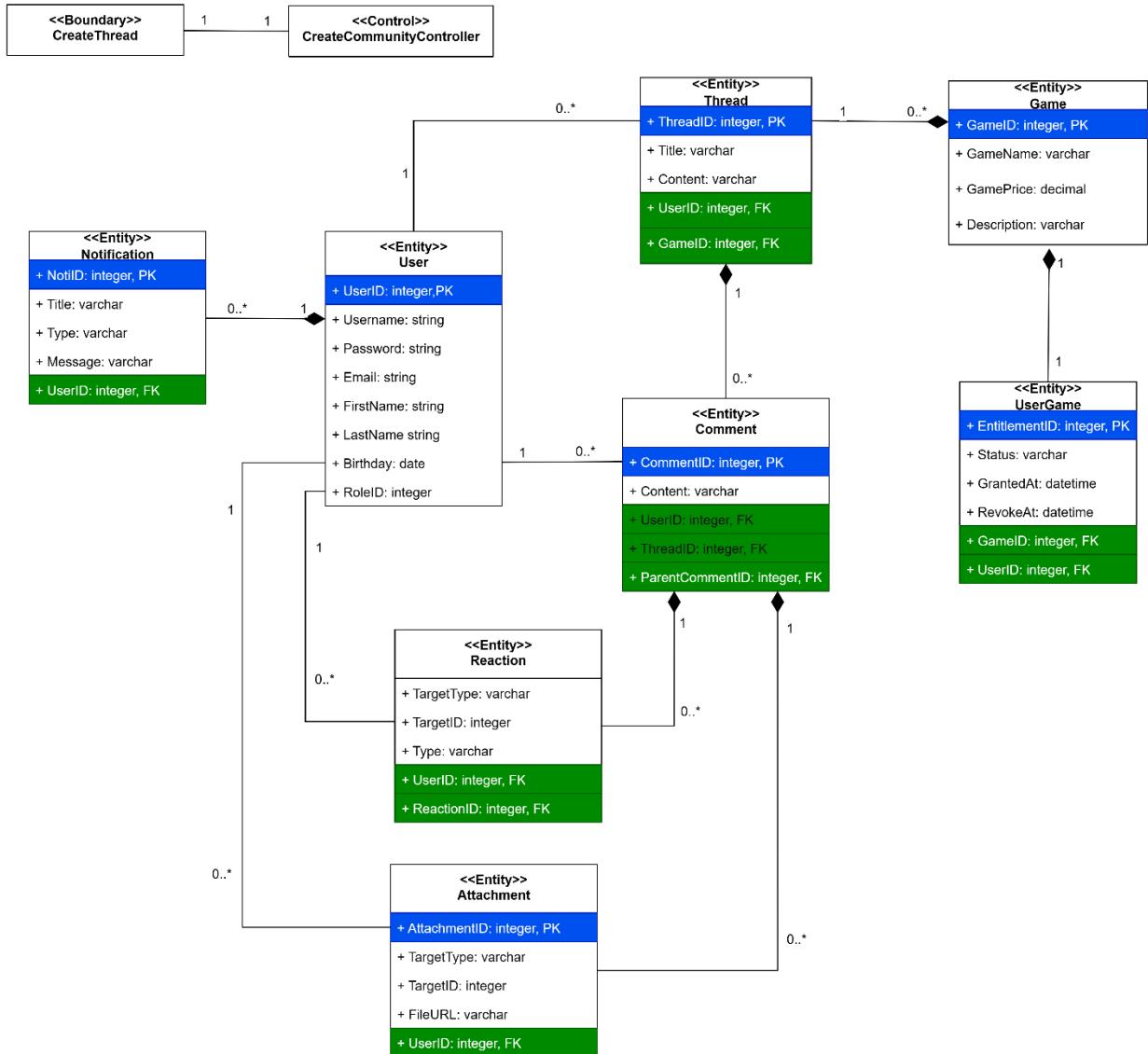
    Threads []Thread `gorm:"foreignKey:UserID"`
}
```

## Communication Diagram

Activity	เป็นคำสั่งสำหรับระบบได้ หรือไม่	ถ้าเป็น ,วัตถุที่รับหน้าที่ ทำงาน คืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click “ปิดคอมมูนิตี้”	เป็นคำสั่ง สั่งให้ ไป โหลดหน้า	:CommunityUI	ปิด ()
โหลดข้อมูลสมาชิก	เป็นคำสั่ง เกิดการสั่งให้โหลด ข้อมูลของสมาชิกในระบบ	:CommunityController	โหลดข้อมูลสมาชิก(userId)
ค้นหาโดยสมาชิกในฐานข้อมูล	เป็นคำสั่ง	u :User	ค้นหาด้วยไอดี (userId)
ขอข้อมูลกระทู้พร้อมคอมเม้นต์	เป็นคำสั่ง	:CommunityController	โหลดข้อมูล (threadId)
ดึงข้อมูลกระทู้	เป็นคำสั่ง เพื่อดึงข้อมูลกระทู้ ออกมา	t :Thread	ดึงข้อมูล(threadId)
แสดงหน้ารายละเอียดกระทู้	ไม่เป็นคำสั่ง	-	-
พิมพ์ข้อความคอมเม้นต์	ไม่เป็นคำสั่ง	-	-
แนบไฟล์รูป (ถ้ามี)	ไม่เป็นคำสั่ง	-	-
กดปุ่ม “ส่งคอมเม้นต์”	เป็นคำสั่ง	:CommunityController	สร้าง (threadId, parentCommentId, content, file)
สร้าง entity Comment	เป็นคำสั่ง	c :Comment	สร้าง (threadId, userId, parentCommentId, content)
บันทึก Comment	เป็นคำสั่ง เป็นการบันทึกข้อมูล	c :Comment	บันทึก ()
(ถ้ามีไฟล์) สร้าง entity Attachment	เป็นคำสั่ง	a :Attachment	สร้าง ( targetType:'comment', targetId:commentId, userId, fileUR)
(ถ้ามีไฟล์) บันทึก Attachment	เป็นคำสั่ง	a :Attachment	บันทึก ()
สร้างการแจ้งเตือนถึงเจ้าของ กระทู้/คอมเม้นต์	เป็นคำสั่ง ระบบสร้างแจ้งเตือน	n :Notification	สร้าง ( type:'comment_created', type:'thread')
บันทึก Notification	เป็นคำสั่ง	n :Notification	บันทึก ()
แสดงข้อความ “ส่งคอมเม้นต์ สำเร็จ”	ไม่เป็นคำสั่ง	-	-



## Class Diagram at Design Level



B6627713 นายทองนรินทร์ แย้มศรี

ระบบหลัก ระบบร้านขายเกมออนไลน์

ระบบย่อย ระบบบริวิวและให้คะแนน

หลังจากซื้อเกม ผู้ใช้งานจะสามารถเขียนรีวิว (Review) และให้คะแนนเกมที่ตนซื้อไว้ในช่วงคะแนน 1–5 ดาว เพื่อแสดงความเห็นและประสบการณ์ของตนต่อเกมนั้นๆ ระบบจะต้องจัดเก็บรีวิวและคะแนนของแต่ละผู้ใช้งานแยกตามเกม และเปิดให้ผู้ใช้อื่นสามารถเข้ามาอ่านความคิดเห็นเหล่านี้ได้

นอกจากนี้ เมื่อมีการลบเกมออกจากระบบ ระบบจะต้องทำการลบรีวิวและคะแนนทั้งหมดที่เกี่ยวข้องกับเกมนั้น โดยอัตโนมัติ ระบบยังรองรับการแสดงผลติดตามผู้พัฒนาเกม เช่น จำนวนยอดการสั่งซื้อเกม, จำนวนรีวิวที่ได้รับ และคะแนนเฉลี่ยจากผู้เล่น เพื่อใช้ในการประเมินและปรับปรุงเกมในอนาคต

### User Story ระบบบริวิวและให้คะแนน

ในบทบาทของ (As a) ผู้ใช้งาน

(I want to) เขียนรีวิวและให้คะแนนเกมที่ซื้อแล้ว

(So that) ฉันจะสามารถแบ่งปันประสบการณ์กับผู้ใช้คนอื่น

### Output บนหน้าจอ

- ผู้ใช้งานเลือกเกมที่ตนเคยซื้อไปแล้วกดสร้างรีวิว และใส่ข้อมูลการรีวิวและคะแนน-> จากนั้นระบบทำการบันทึกข้อมูลคะแนนที่ให้และรีวิวที่เขียน แล้วแสดง Indicator บางอย่างที่แสดงให้เห็นว่าได้บันทึกข้อมูลเรียบร้อยแล้ว

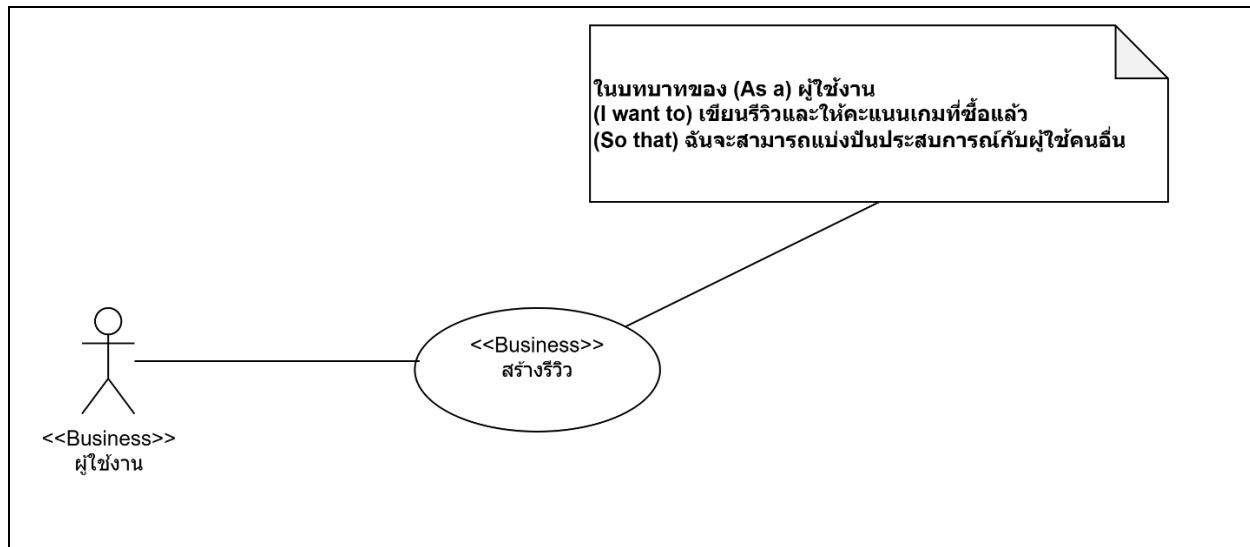
## Output ของข้อมูล

- ระบบเรียกข้อมูลเกมเพื่อให้ผู้ใช้งานกดสร้างรีวิวของเกมนั้น -> ระบบทำการบันทึกข้อมูลรีวิว กับเกมที่ผู้ใช้งานเลือก

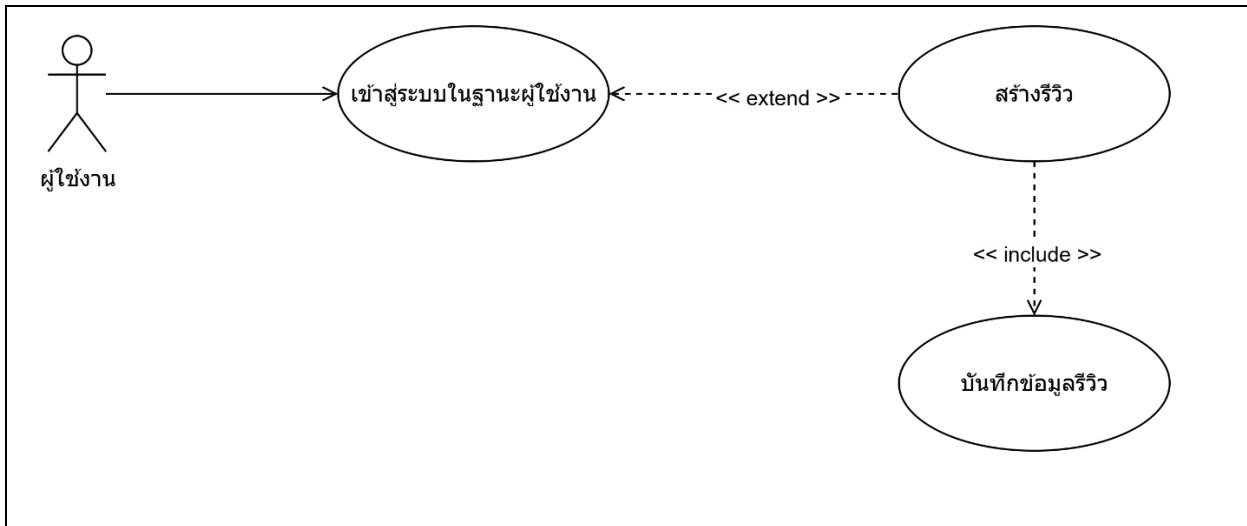
## คำนำที่อาจจะกลایมมาเป็นตารางในฐานข้อมูลของระบบ

คำนำ	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
ผู้ใช้งาน	เกี่ยวข้องโดยตรง เนื่องจากผู้ใช้งานจะต้องสร้างรีวิว
เกม	เกี่ยวข้องโดยตรง เนื่องจากต้องใช้ข้อมูลเกม ในการสร้างรีวิว
รีวิว (Review)	เกี่ยวข้องโดยตรง เนื่องจากระบบนี้เป็นการสร้างรีวิวขึ้นมา
คะแนน	เกี่ยวข้องโดยตรง เนื่องจากต้องนำคะแนนที่ได้ไปใช้แสดงบนรีวิวที่ผู้ใช้เขียน
ผู้พัฒนาเกม	ไม่เกี่ยวข้องโดยตรง
สถิติ	เกี่ยวข้องโดยตรง เนื่องจากเป็นค่าที่ต้องนำมาแสดง เช่น คะแนนเฉลี่ย จำนวนรีวิว

## Business Use Case Diagram (แบบเดี่ยว)



## System use Case Diagram (แบบเดี่ยว)



### ขั้นตอนการแปลง Business Use Case Diagram ให้เป็น System Use Case Diagram

#### พิจารณาประเด็นที่ 1

Business Actor "ผู้ใช้งาน" สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์ที่ระบบงานได้โดยตรงหรือไม่

ตอบ ได้ สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์

เพราะฉะนั้น ผู้ใช้งาน จะถูกจัดเป็น System Actor ได้

#### พิจารณาประเด็นที่ 2

Business Use Case “สร้าง รีวิว” สามารถถูกถ่ายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบคอมพิวเตอร์ได้หรือไม่ต้องมีขั้นตอนทาง Security มาเกี่ยวหรือไม่

ตอบ Business Use Case “สร้าง รีวิว” สามารถถูกถ่ายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบคอมพิวเตอร์ได้และต้องการขั้นตอนทาง Security (มี Privacy มาเกี่ยวข้อง)

เพราะฉะนั้น Business Use Case “สร้าง รีวิว” จะถูกแบ่ง成 System Use Case มากกว่า 1 Use Case

จะประกอบไปด้วย

- 1 System Use Case สำหรับ “เข้าระบบในฐานะผู้ใช้งาน”
- 2 System Use Case สำหรับ “สร้าง รีวิว” แต่กิจกรรมของระบบ จะยังไม่ครบตาม Requirements ที่ต้องการ
- 3 System Use Case สำหรับ “บันทึกข้อมูล รีวิว”
  - System Use Case สำหรับ “เข้าระบบในฐานะผู้ใช้งาน”
  - System Use Case สำหรับ “สร้าง รีวิว”
  - System Use Case สำหรับ “บันทึกข้อมูล รีวิว”

### พิจารณาประเด็นที่ 3

ถ้าผู้ใช้งาน “เข้าระบบในฐานะผู้ใช้งาน” มาแล้วจำเป็นที่ต้อง “สร้าง รีวิว” ทุกรอบหรือไม่

ตอบ ไม่

แปลว่า System Use Case “สร้าง รีวิว” เป็นกิจกรรมทางเลือกหลังจากทำ System Use Case “เข้าระบบ ในฐานะผู้ใช้งาน”

### พิจารณาประเด็นที่ 4

ถ้าผู้ใช้งาน “เข้าระบบในฐานะผู้ใช้งาน” มาแล้วจำเป็นที่ต้อง “บันทึกข้อมูลรีวิว” ทุกรอบหรือไม่

ตอบ ไม่ และเป็น Use Case ที่ไม่เกี่ยวข้องกัน

## พิจารณาประเด็นที่ 5

ถ้าสมาชิก “สร้าง รีวิว” แล้ว

จำเป็นต้อง “บันทึกข้อมูล รีวิว” ทุกครั้งหรือไม่

ตอบ ใช่ จำเป็น

แปลว่า System Use Case “สร้าง รีวิว” จะต้องรวมขั้นตอนของ

System Use Case “บันทึกข้อมูล รีวิว” ไว้ด้วยเสมอ

การจำลองตัวอย่างตารางและข้อมูล (เพื่อใช้ในการเตรียมร่าง User Interface, System Activity Diagram และ Class Diagram)

จาก Requirements, จาก ข้อมูล Output และ Entity ที่ได้ออกแบบไว้เราจะนำมาสมมติเป็นตารางในฐานข้อมูล โดยกำหนด Primary Key เป็นตัวเลขรวมถึงการสมมติ Foreign Key เพื่อเชื่อมโยงข้อมูลระหว่างตาราง ซึ่งจะช่วยให้สามารถออกแบบและวิเคราะห์ระบบได้อย่างมีโครงสร้างและชัดเจน

วิเคราะห์ Entity ที่เกี่ยวข้อง

Review\_Like

ชื่อ Entity: Review\_Like

ข้อมูลที่ต้องจัดเก็บ

- Review\_id\_ID: FK จ้าวอิงไปที่ Review\_ID
- User\_ID: FK จ้าวอิงไปที่ User\_ID

Review_ID INTEGER NOT NULL, PK	User_ID INTEGER NOT NULL, FK
3001	1001
3002	1002

## Review

ชื่อ Entity: Review

ข้อมูลที่ต้องจัดเก็บ

- Review\_ID: Primary Key เก็บเป็น integer ไม่ซ้ำ และไม่เป็น Null
- User\_ID: FK จัดอิงไปที่ User\_ID
- Game\_ID: FK จัดอิงไปที่ Game.ID
- ReviewTitle: String หัวข้อรีวิว และไม่เป็น Null
- Rating: integer (1-5) และไม่เป็น Null
- ReviewText: Sting (ข้อความรีวิว) และไม่เป็น Null
- CreatedAt: datetime และไม่เป็น Null

Review_ID INTEGER NOT NULL, PK	User_ID INTEGER NOT NULL, FK	Game_ID INTEGER NOT NULL, FK	Rating INTEGER NOT NUL	ReviewText STRING, NOT NULL	ReviewTitle STRING, NOT NULL	CreatedAt DATETIME, NOT NULL
4001	1001	2001	5	สนุกมาก ระบบดีมาก	เกมที่ทุกคน ต้องเล่น	2025-07-24 15:00:00
4002	1002	2002	3	กราฟพิกสวย แต่บึกเบ lokale	พอดีนะ	2025-07-24 15:30:00

## หลักการออกแบบ User Interface

- เมื่อได้กีตามที่มี Foreign Key (FK) ซึ่งจากตารางหลักกลับไปยัง ตารางสนับสนุน ให้ใช้ ComboBox เป็นตัวเลือกเพื่อเชื่อมโยงข้อมูลใน User Interface
- สำหรับ Field ประเภทอื่นๆ ให้สร้างตามประเภทของข้อมูล เช่น
  - Textbox สำหรับข้อความทั่วไป
  - Password สำหรับข้อมูลรหัสผ่าน
  - Datetime Picker สำหรับเลือกวันและเวลา
  - Numeric Input สำหรับป้อนตัวเลข

UI Design



### Create review

Select a game



Write details

cancel

create

## การเตรียม System Activity Diagram

- 1 Business Use Case (1 User Story) จะถูกแปลงเป็น 1 System Activity Diagram
- System Activity Diagram คือ การบรรยายลำดับของกิจกรรม (Activity) ระหว่าง ผู้ใช้ (คน) กับ ระบบ

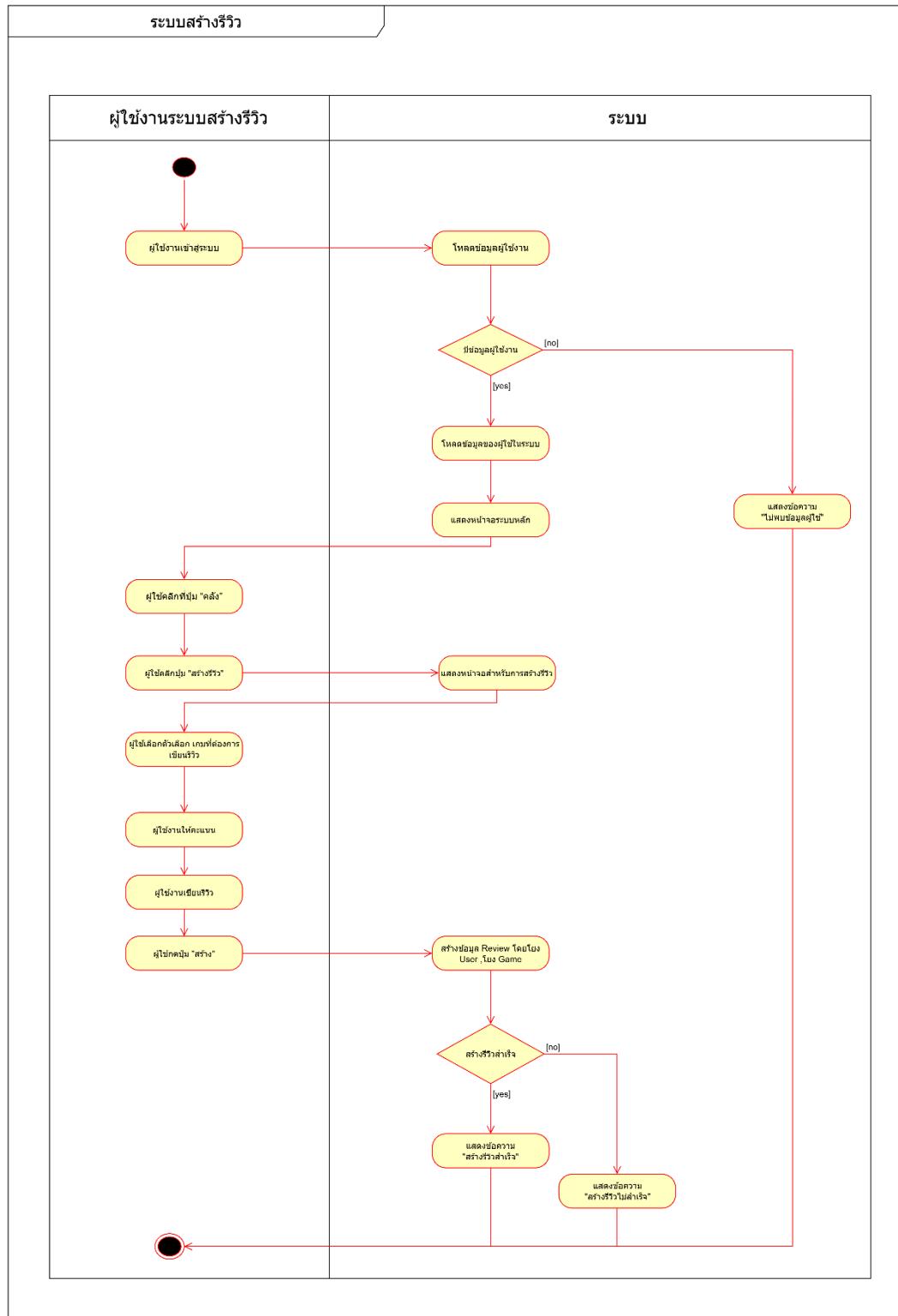
## หลักสำคัญ ของการออกแบบ System Activity Diagram ได้แก่

- ระบุว่า ผู้ใช้ทำอะไร (Input)
- ระบบประมวลผลอะไร (Process)
- ระบบตอบกลับอะไร (Output)

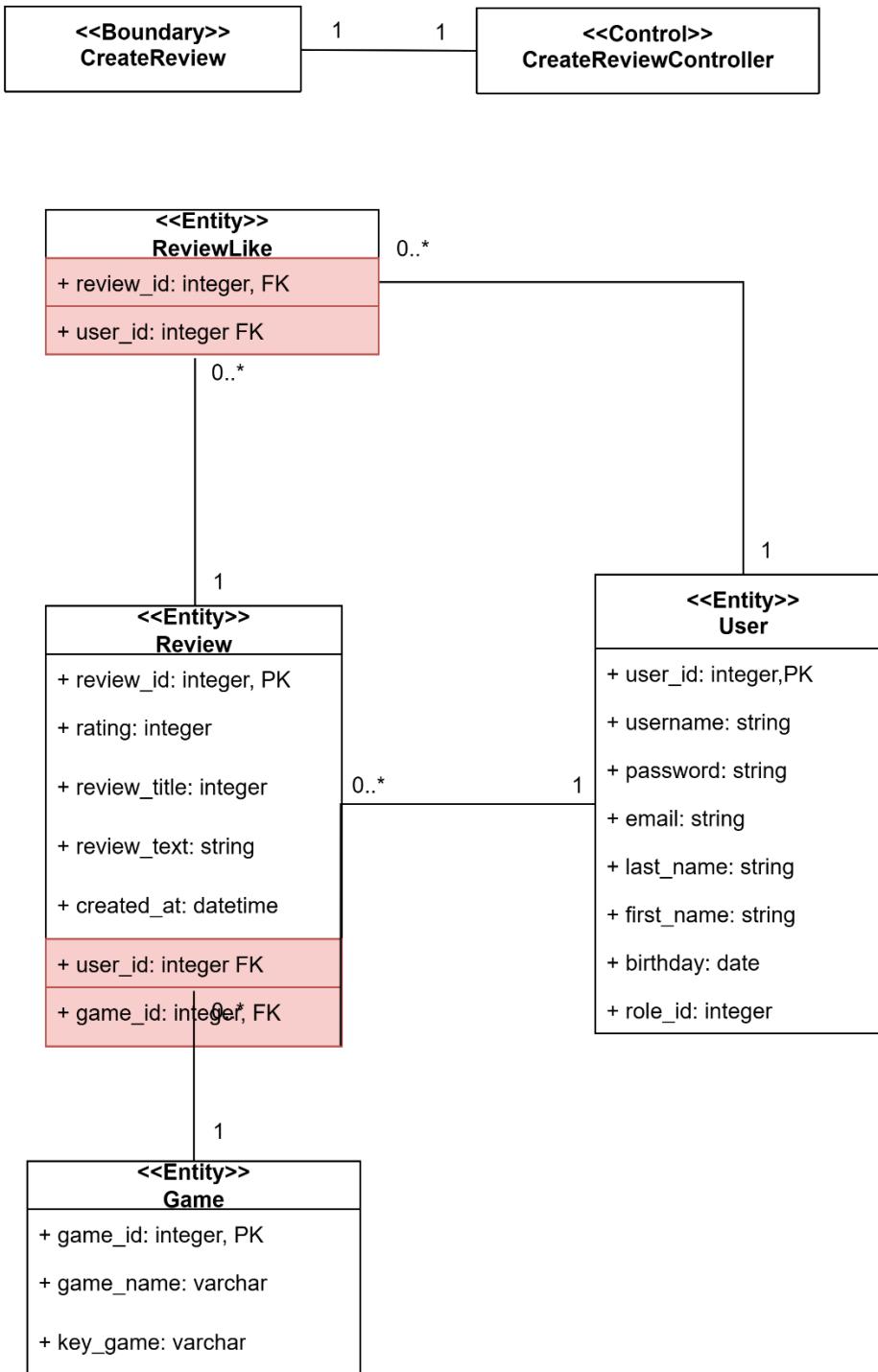
## System Activity Diagram ต้องลืนสุดที่การสร้าง Output ดังนี้

1. บันทึกข้อมูลลงใน ฐานข้อมูล
2. แสดงผลข้อมูลผ่าน หน้าจอ (User Interface) โดยต้องสอดคล้องกับ User Story ที่กำหนดไว้

## Activity Diagram



## Class Diagram at Analysis Level



## Frontend

### Review UI



```
import React from "react";
import { useParams, useNavigate } from "react-router-dom";
import {
  Layout,
  Typography,
  Tag,
  Button,
  Card,
  Row,
  Col,
  Skeleton,
  Divider,
  Space,
  Rate,
  message,
} from "antd";
import { ShoppingCartOutlined, PictureOutlined, StarFilled, ToolOutlined } from "@ant-design/icons";
import { useCart } from "../../context/CartContext";
import { getGame, listMods } from "../../services/workshop";
import { findMinimumSpecForGame, type MinimumSpec } from "../../services/minimumspec";
```

```

import ReviewSection from "../../components/ReviewSection";
import type { Game, Mod } from "../../interfaces";

const { Header, Content } = Layout;
const { Title, Text, Paragraph } = Typography;

const API_BASE = "http://localhost:8088";

const resolveImgUrl = (src?: string) => {
  if (!src) return "";
  if (src.startsWith("http://") || src.startsWith("https://") || src.startsWith("data:image/")) return src;
  const clean = src.startsWith("/") ? src.slice(1) : src;
  return `${API_BASE}/${clean}`;
};

const safeNum = (v: any, d = 0) => (Number.isFinite(Number(v)) ? Number(v) : d);
const asArray = (v: any): any[] =>
  Array.isArray(v) ? v : Array.isArray(v?.items) ? v.items : [];

const GameDetail: React.FC = () => {
  const { id } = useParams();
  const navigate = useNavigate();

  const [loading, setLoading] = React.useState(true);
  const [game, setGame] = React.useState<Game | null>(null);
  const [mods, setMods] = React.useState<Mod[]>([]);
  const [minSpec, setMinSpec] = React.useState<MinimumSpec | null>(null);
  const [msg, ctx] = message.useMessage();
  const [avgRating, setAvgRating] = React.useState(0);

  // contexts
  const { addlItem } = useCart();

  const gid = React.useMemo(() => Number(id), [id]);

  React.useEffect(() => {
    if (!gid) return;
    let alive = true;

    (async () => {
      try {
        setLoading(true);

        // โหลดเกม
        let g: any;
        try {
          g = await getGame(gid);
        } catch {
          const all = await import("../../services/workshop").listGames();
        }
      } catch {
        msg.error("Error loading game");
      }
    })();
  });
}

```

```

g = asArray(all).find((x: any) => (x?.ID ?? x?.id) === gid) || null;
}

if (!alive) return;
setGame(g ?? null);

// โหลด Minimum Spec
try {
  const spec = g ? await findMinimumSpecForGame(g) : null;
  if (alive) setMinSpec(spec);
} catch (e) {
  console.warn("findMinimumSpecForGame failed:", e);
  if (alive) setMinSpec(null);
}
} catch (e) {
  console.error(e);
  if (alive) {
    setGame(null);
    setMinSpec(null);
  }
} finally {
  if (alive) setLoading(false);
}
}();
}

// โหลดมือถือของเกม
listMods(gid)
  .then((rows) => alive && setMods(asArray(rows)))
  .catch(() => alive && setMods([]));

return () => {
  alive = false;
};

}, [gid]);

// รูปปน
const coverRaw = (game as any)?.img_src || (game as any)?.cover || "";
const cover = resolveImgUrl(coverRaw);

// ข้อมูล/แท็ก
const title = (game as any)?.game_name ?? (game as any)?.name ?? "Untitled Game";
const desc =
  (game as any)?.description ??
  (game as any)?.detail ??
  "No description available.";

const tags: string[] = () => {
  const raw = (game as any)?.tags ?? (game as any)?.Genres ?? [];
  if (Array.isArray(raw)) return raw;
  if (typeof raw === "string") return raw.split(",").map((s) => s.trim()).filter(Boolean);
};

```

```

const catTitle = (game as any)?.categories?.title;
return catTitle ? [String(catTitle)] : [];
})();

// ราคา/ส่วนลด
const basePrice = safeNum((game as any)?.base_price ?? (game as any)?.price, 0);
const discountedPrice = (game as any)?.discounted_price;
const discountPercentField = safeNum((game as any)?.discount ?? (game as any)?.discount_percent, 0);

let finalPrice = basePrice;
let discountPercent = 0;

if (typeof discountedPrice === "number" && discountedPrice < basePrice) {
    finalPrice = discountedPrice;
    discountPercent = basePrice > 0 ? Math.round((1 - discountedPrice / basePrice) * 100) : 0;
} else if (discountPercentField > 0) {
    discountPercent = discountPercentField;
    finalPrice = Math.max(0, basePrice * (1 - discountPercent / 100));
}

// Minimum Spec
const ms = minSpec ?? {};
const os = (ms as any).os ?? (ms as any).OS ?? "N/A";
const cpu = (ms as any).cpu ?? (ms as any).CPU ?? "N/A";
const ram = (ms as any).ram ?? (ms as any).RAM ?? "N/A";
const gpu = (ms as any).gpu ?? (ms as any).GPU ?? "N/A";
const storage = (ms as any).storage ?? (ms as any).Storage ?? "N/A";

// Top workshop items
const topMods = React.useMemo(() => {
    const getViews = (m: any) => Number(m.view_count ?? m.views ?? m.viewCount ?? 0);
    return [...mods].sort((a, b) => getViews(b) - getViews(a)).slice(0, 8);
}, [mods]);

// === NEW: Purchase handler (เพิ่ม ProductGrid) ===
const handlePurchase = () => {
    if (!game) return;

    const price = Number((game as any)?.discounted_price ?? (game as any)?.base_price) || 0;
    const gameId = (game as any)?.ID ?? (game as any)?.id;

    addItem({ id: Number(gameId), title, price, quantity: 1 });
    msg.success(`เพิ่ม ${title} ลงตะกร้าแล้ว`);
    navigate("/category/Payment");
};

if (loading && !game) {
    return (
        <div style={{ padding: 24 }}>

```

```
<Skeleton active title paragraph={{ rows: 12 }} />
</div>
);
}

return (
<Layout style={{ background: "#0f1419", minHeight: "100vh" }}>
{ctx}

<Header
style={{
background: "#0f1419",
padding: 0,
height: 360,
position: "relative",
overflow: "hidden",
borderBottom: "1px solid #1f2933",
}}
>
/* BG blur */
<div
aria-hidden
style={{
position: "absolute",
inset: 0,
backgroundImage: cover ? `url(${cover})` : undefined,
backgroundSize: "cover",
backgroundPosition: "center",
filter: "blur(16px)",
transform: "scale(1.25)",
opacity: 0.35,
}}
/>
<div
aria-hidden
style={{
position: "absolute",
inset: 0,
background:
"linear-gradient(180deg, rgba(15,20,25,0.3) 0%, rgba(15,20,25,0.75) 60%, rgba(15,20,25,0.95) 100%)",
}}
/>
<div
style={{
position: "relative",
zIndex: 1,
height: "100%",
display: "grid",
gridTemplateColumns: "minmax(280px, 360px) 1fr 380px",
}}
```

```
gap: 20,
alignItems: "center",
padding: "24px 28px",
}}
>
/* Cover */
<div
style={{
width: "100%",
height: 260,
borderRadius: 12,
overflow: "hidden",
background: "#1f2933",
border: "1px solid #2b3a42",
boxShadow: "0 10px 30px rgba(0,0,0,0.4)",
}}
>
{cover ? (
<img
src={cover}
alt={title}
style={{ width: "100%", height: "100%", objectFit: "cover" }}
/>
) : (
<div
style={{
color: "#8899a6",
fontSize: 32,
height: "100%",
display: "flex",
alignItems: "center",
justifyContent: "center",
}}
>
<PictureOutlined /> &nbsp; No cover
</div>
)}
</div>

/* Title / Tags / Desc / CTAs */
<div>
<Title level={2} style={{ color: "#fff", margin: 0 }}>
{title}
</Title>
<div style={{ margin: "8px 0 12px" }}>
<Space size={[6, 6]} wrap>
{tags.slice(0, 6).map((t) => (
<Tag key={t} color="blue">
{t}

```

```

        </Tag>
    )}
{tags.length === 0 && <Tag>Action</Tag>}
</Space>
</div>
<Paragraph style={{ color: "#d1d5db", maxWidth: 800, marginBottom: 12 }}>
  {String(desc).slice(0, 280)}
  {String(desc).length > 280 ? "..." : ""}
</Paragraph>

</div>

/* Price Card */
<Card size="small" bordered style={{ background: "#12181f", borderColor: "#2a3b45" }} bodyStyle={{ padding: 16 }}>
  <div style={{ display: "flex", alignItems: "center", justifyContent: "space-between" }}>
    <div>
      <div style={{ color: "#b3c1cd", fontSize: 12 }}>CURRENT PRICE</div>
      <div style={{ color: "#fff", fontSize: 22, fontWeight: 700 }}>
        {finalPrice.toLocaleString()} ₦
      </div>
    </div>
    {discountPercent > 0 && (
      <div style={{ color: "#9aa4ad", textDecoration: "line-through" }}>
        {basePrice.toLocaleString()} ₦
      </div>
    )}
  </div>
  {discountPercent > 0 && (
    <div
      style={{
        background: "#1f6f3c",
        color: "#fff",
        fontWeight: 700,
        borderRadius: 8,
        padding: "6px 10px",
        minWidth: 64,
        textAlign: "center",
      }}
    >
      -{discountPercent}%
    </div>
  )}
</div>
<Divider style={{ borderColor: "#23313a" }} />
<Button type="primary" icon={<ShoppingCartOutlined />} block
  onClick={handlePurchase}>
  Purchase
</Button>
</Card>
</div>

```

```

</Header>

/* ===== Content ===== */

<Layout>
<Content style={{ padding: 24 }}>
  <Card
    title="System Requirements (Minimum)"
    style={{ marginBottom: 16, background: "#12181f", borderColor: "#23313a" }}
    headStyle={{ color: "ffff" }}
  >
    <div style={{ color: "#c7d0d9" }}>
      <p><b>OS:</b> {os}</p>
      <p><b>Processor:</b> {cpu}</p>
      <p><b>Memory:</b> {ram}</p>
      <p><b>Graphics:</b> {gpu}</p>
      <p><b>Storage:</b> {storage}</p>
    </div>
  </Card>

/* Workshop items */

<Card
  title={<Space><ToolOutlined /> <span>Workshop Items</span></Space>}
  extra={<Button onClick={() => navigate('/workshop/${gid}')}>Browse all</Button>}
  style={{ background: "#12181f", borderColor: "#23313a" }}
  headStyle={{ color: "ffff" }}
>
  <Row gutter={[16, 16]}>
    {topMods.map((m) => {
      const img = (m as any).image_path ?? (m as any).img_src ?? "";
      const cover = resolveImgUrl(img);
      return (
        <Col xs={24} sm={12} md={8} lg={6} key={(m as any).ID ?? (m as any).id}>
          <Card
            hoverable
            style={{ background: "#1a1a1a", borderColor: "#262626" }}
            cover={

              cover ? (
                <img src={cover} alt={(m as any).title} style={{ height: 160, objectFit: "cover" }} />
              ) : (
                <div
                  style={{
                    height: 160,
                    background: "linear-gradient(135deg, #2b2b2b 0%, #1f1f1f 100%)",
                    display: "flex",
                    alignItems: "center",
                    justifyContent: "center",
                    color: "#888",
                    fontSize: 30,
                  }}>

```

```

        >
          <PictureOutlined />
        </div>
      )
}

onClick={() => navigate('/mod/${(m as any).ID ?? (m as any).id}')}

>
<Text style={{ color: "#fff", fontWeight: 600 }}>
  {(m as any).title ?? "Untitled Mod"}
</Text>
<div style={{ color: "#9aa4ad", fontSize: 12, marginTop: 4 }}>
  <StarFilled style={{ color: "#f5c518" }} />{" "}
  {Number((m as any).rating ?? (m as any).avg_rating ?? 0).toFixed(1)} / 5
</div>
</Card>
</Col>
);
)}

{topMods.length === 0 && (
<Col span={24} style={{ textAlign: "center", color: "#9aa4ad" }}>
  No workshop items for this game yet.
</Col>
)}
</Row>
</Card>

<Card
  title={
    <Space>
      <StarFilled />
      <span>User Reviews</span>
    {avgRating > 0 && (
      <Space size={4} style={{ color: "#fadb14" }}>
        <Rate disabled allowHalf value={avgRating} />
        <span>{avgRating.toFixed(1)}</span>
      </Space>
    )}
    </Space>
  }
  style={{ marginTop: 16, background: "#12181f", borderColor: "#23313a" }}
  headStyle={{ color: "#fff" }}
>
<ReviewSection
  gameId={gid}
  allowCreate
  className="mt-4"
  onStatsChange={(s) => setAvgRating(s.average)}
/>
</Card>

```

```
</Content>

</Layout>
</Layout>
};

};

export default GameDetail;
```

## Backend

### Game Entity

```
package entity
import (
    "time"
    "gorm.io/gorm"
)

type Game struct {
    gorm.Model
    GameName string `json:"game_name"`
    //KeyGameID uint `json:"key_id"`
    CategoriesID int `json:"categories_id"`
    Categories Categories `json:"categories" gorm:"foreignKey:CategoriesID"`
    Date time.Time `json:"release_date" gorm:"autoCreateTime"`
    BasePrice int `json:"base_price"`
    Status string `json:"status" gorm:"type:varchar(20);default:'pending';not null;index"`
    Minimum_specID uint `json:"minimum_spec_id"`
    MinimumSpec MinimumSpec `json:"minimum_spec"`
    AgeRating int `json:"age_rating"`

    Requests []Request `gorm:"foreignKey:GameRefer"`

    Threads []Thread `gorm:"foreignKey:GameID" json:"threads,omitempty"`
    UserGames []UserGame `gorm:"foreignKey:GameID" json:"user_games,omitempty"`

    Reviews []Review `gorm:"foreignKey:GameID" json:"reviews,omitempty"`
    ReviewLikes []Review_Like `gorm:"foreignKey:GameID" json:"review_likes,omitempty"`

    Promotions []Promotion `json:"promotions,omitempty" gorm:"many2many:promotion_games"`
    PromotionGames []Promotion_Game `json:"promotion_games,omitempty" gorm:"foreignKey:GameID"`
    ImgSrc string `json:"img_src" gorm:"type:varchar(512)"`
}
```

### Review\_like Entity

```
package entity
import "gorm.io/gorm"

type Review_Like struct {
    gorm.Model
    // Like ຕ້ວນເປັນອອກຈິວໃຫນ
    ReviewID uint `json:"review_id" gorm:"index:idx_review_user,unique;not null"`
    Review *Review `json:"review" gorm:"foreignKey:ReviewID;constraint:OnUpdate:CASCADE,OnDelete:CASCADE;"`
```

```

// โครงสร้างของ Like
UserID uint `json:"user_id" gorm:"index:idx_review_user,unique;not null"`
User *User `json:"user" gorm:"foreignKey:UserID;constraint:OnUpdate:CASCADE,OnDelete:CASCADE;"` 

// (อยู่ขั้นแรก) ข้างอิง Game เพื่อให้ query จาก Like → Game ได้ตรงๆ โดยไม่ต้อง join ผ่าน Review
// แนะนำให้ตั้งค่า GameID = review.GameID ตอนสร้าง Like
GameID uint `json:"game_id" gorm:"index"`
Game *Game `json:"game" gorm:"foreignKey:GameID;constraint:OnUpdate:CASCADE,OnDelete:SET NULL;"` 
}

```

## Review Entity

```

package entity

import "gorm.io/gorm"

// รีวิวของผู้ใช้ต่อเกมหนึ่ง ๆ
type Review struct {
    gorm.Model

    ReviewTitle string `json:"review_title" gorm:"type:varchar(255);not null"`
    ReviewText string `json:"review_text" gorm:"type:text"`
    Rating int `json:"rating" gorm:"not null" // 1-5

    // เปลี่ยนชื่อ composite unique index -> ux_reviews_user_game
    UserID uint `json:"user_id" gorm:"not null;uniqueIndex:ux_reviews_user_game,priority:1"`
    User *User `gorm:"foreignKey:UserID" json:"user"`

    GameID uint `json:"game_id" gorm:"not null;uniqueIndex:ux_reviews_user_game,priority:2"`
    Game *Game `gorm:"foreignKey:GameID" json:"game"`

    Likes []Review_Like `gorm:"foreignKey:ReviewID" json:"likes"`
}

func (Review) TableName() string { return "reviews" }

```

## User Entity

```
package entity

import (
    "time"

    "gorm.io/gorm"
)

type User struct {
    gorm.Model
    Username string `gorm:"size:120;uniqueIndex" json:"username"`
    Password string `json:"-"`           // ອ່ານສຳກັບໃນ API
    Email    string `gorm:"size:255;uniqueIndex" json:"email"`
    FirstName string `json:"first_name"`
    LastName string `json:"last_name"`
    Birthday time.Time `json:"birthday"`

    RoleID uint `gorm:"not null;index" json:"role_id"`
    Role  Role `gorm:"constraint:OnUpdate:CASCADE,OnDelete:RESTRICT;" json:"role"` // ກັນລົບ role ທີ່ຈຸກອ້າງອີງ

    Threads []Thread `gorm:"foreignKey:UserID" json:"threads,omitempty"`
    Comments []Comment `gorm:"foreignKey:UserID" json:"comments,omitempty"`
    Reactions []Reaction `gorm:"foreignKey:UserID" json:"reactions,omitempty"`
    Attachments []Attachment `gorm:"foreignKey:UserID" json:"attachments,omitempty"`
    Notifications []Notification `gorm:"foreignKey:UserID" json:"notifications,omitempty"`
    UserGames []UserGame `gorm:"foreignKey:UserID" json:"user_games,omitempty"`

    Reviews []Review `gorm:"foreignKey:UserID" json:"reviews,omitempty"`
    ReviewLikes []Review_Like `gorm:"foreignKey:UserID" json:"review_likes,omitempty"`

    Promotions []Promotion `json:"promotions,omitempty" gorm:"foreignKey:UserID"`
    Requests []Request `json:"request" gorm:"foreignKey:UserRefer"`
}
```

## Controller review\_controller

```
package controllers

import (
    "errors"
    "net/http"
    "strconv"
    "strings"

    "github.com/gin-gonic/gin"
    "gorm.io/gorm"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
)

// ---- DTO ----

type reviewCreateDTO struct {
    GameID     uint `json:"game_id" binding:"required"`
    UserID     uint `json:"user_id" binding:"required"`
    ReviewTitle string `json:"review_title"`
    ReviewText string `json:"review_text" binding:"required"`
    Rating     int  `json:"rating" binding:"required"`
}

type reviewUpdateDTO struct {
    ReviewTitle *string `json:"review_title"`
    ReviewText  *string `json:"review_text"`
    Rating     *int   `json:"rating"`
}

// ---- Helpers ----

func clampRating(n int) int {
    if n < 1 {
        return 1
    }
    if n > 5 {
        return 5
    }
    return n
}

// ---- Handlers ----

// POST /reviews
func CreateReview(c *gin.Context) {
```

```

var in reviewCreateDTO

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

db := configs.DB()

// กันข้ามเบร์ (ก่อนชน unique)
var exists entity.Review

if err := db.Where("user_id = ? AND game_id = ?", in.UserID, in.GameID).First(&exists).Error; err == nil {
    c.JSON(http.StatusConflict, gin.H{
        "error": "already_reviewed",
        "message": "คุณได้รีวิวเกมนี้แล้ว กรุณาแก้ไขรีวิวเดิม",
    })
    return
}

} else if err != gorm.ErrRecordNotFound {
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}

r := entity.Review{
    GameID:   in.GameID,
    UserID:   in.UserID,
    ReviewTitle: in.ReviewTitle,
    ReviewText: in.ReviewText,
    Rating:   in.Rating,
}
if err := db.Create(&r).Error; err != nil {
    // map duplicate เป็น 409 (กันกรณีเลือดอดมาชน unique)
    if errors.Is(err, gorm.ErrDuplicateKey) ||
        strings.Contains(strings.ToLower(err.Error()), "unique") {
        c.JSON(http.StatusConflict, gin.H{
            "error": "already_reviewed",
            "message": "คุณได้รีวิวเกมนี้แล้ว กรุณาแก้ไขรีวิวเดิม",
        })
        return
    }
    c.JSON(http.StatusInternalServerError, gin.H{"error": "create_failed"})
    return
}

var out entity.Review
_ = db.Preload("User").Preload("Game").First(&out, r.ID).Error
c.JSON(http.StatusCreated, out)
}

// GET /reviews

```

```

func FindReviews(c *gin.Context) {
    db := configs.DB()
    var list []entity.Review
    if err := db.Preload("User").Preload("Game").
        Order("updated_at DESC").
        Find(&list).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": "list_failed"})
        return
    }
    c.JSON(http.StatusOK, list)
}

// GET /reviews/:id
func GetReviewByID(c *gin.Context) {
    id := c.Param("id")
    db := configs.DB()
    var row entity.Review
    if err := db.Preload("User").Preload("Game").First(&row, id).Error; err != nil {
        if err == gorm.ErrRecordNotFound {
            c.JSON(http.StatusNotFound, gin.H{"error": "not_found"})
            return
        }
        c.JSON(http.StatusInternalServerError, gin.H{"error": "get_failed"})
        return
    }
    c.JSON(http.StatusOK, row)
}

// PUT /reviews/:id
func UpdateReview(c *gin.Context) {
    id := c.Param("id")
    var in reviewUpdateDTO
    if err := c.ShouldBindJSON(&in); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "invalid_payload"})
        return
    }

    db := configs.DB()
    var r entity.Review
    if err := db.First(&r, id).Error; err != nil {
        if err == gorm.ErrRecordNotFound {
            c.JSON(http.StatusNotFound, gin.H{"error": "not_found"})
            return
        }
        c.JSON(http.StatusInternalServerError, gin.H{"error": "get_failed"})
        return
    }

    if in.ReviewTitle != nil {

```

```

        r.ReviewTitle = *in.ReviewTitle
    }
    if in.ReviewText != nil {
        r.ReviewText = *in.ReviewText
    }
    if in.Rating != nil {
        r.Rating = clampRating(*in.Rating)
    }

    if err := db.Save(&r).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": "update_failed"})
        return
    }

    var out entity.Review
    _ = db.Preload("User").Preload("Game").First(&out, r.ID).Error
    c.JSON(http.StatusOK, out)
}

// DELETE /reviews/:id
func DeleteReview(c *gin.Context) {
    id := c.Param("id")
    db := config.DB()
    if err := db.Unscoped().Delete(&entity.Review{}, id).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": "delete_failed"})
        return
    }
    c.Status(http.StatusNoContent)
}

// GET /games/:id/reviews
func FindReviewsByGame(c *gin.Context) {
    gameId := c.Param("id")
    db := config.DB()
    var list []entity.Review
    if err := db.Preload("User").Preload("Likes").
        Where("game_id = ?", gameId).
        Order("updated_at DESC").
        Find(&list).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": "list_failed"})
        return
    }

    // แปลงให้มีพิล็อก likes เป็นจำนวนไลก์แทนการส่งรายละเอียดไลก์ทั้งหมด
    out := make([]gin.H, 0, len(list))
    for _, r := range list {
        out = append(out, gin.H{
            "ID":          r.ID,
            "CreatedAt":   r.CreatedAt,

```

```

        "UpdatedAt": r.UpdatedAt,
        "DeletedAt": r.DeletedAt,
        "review_title": r.ReviewTitle,
        "review_text": r.ReviewText,
        "rating": r.Rating,
        "user_id": r.UserID,
        "user": r.User,
        "game_id": r.GameID,
        "game": r.Game,
        "likes": len(r.Likes),
    })
}

c.JSON(http.StatusOK, out)
}

// POST /reviews/:id/toggle_like
func ToggleReviewLike(c *gin.Context) {
    id := c.Param("id")
    var body struct {
        UserID uint `json:"user_id" binding:"required"`
    }
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "invalid_payload"})
        return
    }

    rid, _ := strconv.Atoi(id)
    db := configs.DB()

    // ກາ like ເດີມ
    var like entity.Review_Like
    err := db.Where("review_id = ? AND user_id = ?", rid, body.UserID).First(&like).Error
    liked := true
    if err == nil {
        // ມີແລ້ວ => ລບ (unlike)
        if err := db.Unscoped().Delete(&like).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": "toggle_failed"})
            return
        }
        liked = false
    } else if err != gorm.ErrRecordNotFound {
        c.JSON(http.StatusInternalServerError, gin.H{"error": "toggle_failed"})
        return
    } else {
        // ຍັງນີ້ => ເພີ່ມ like
        newLike := entity.Review_Like{
            ReviewID: uint(rid),
            UserID:   body.UserID,
        }
    }
}

```

```
        }

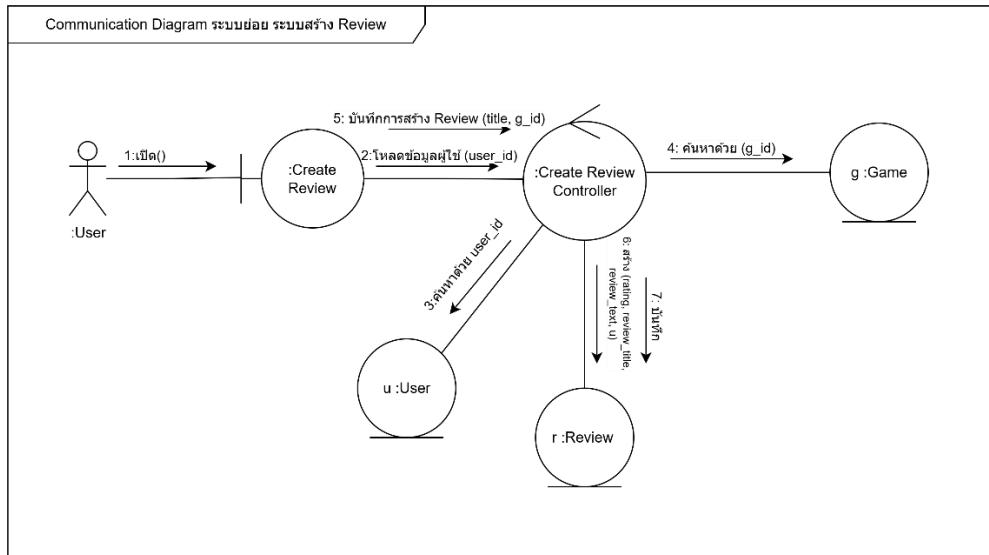
        if err := db.Create(&newLike).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": "toggle_failed"})
            return
        }
    }

    var cnt int64
    db.Model(&entity.Review_Like{}).Where("review_id = ?", rid).Count(&cnt)

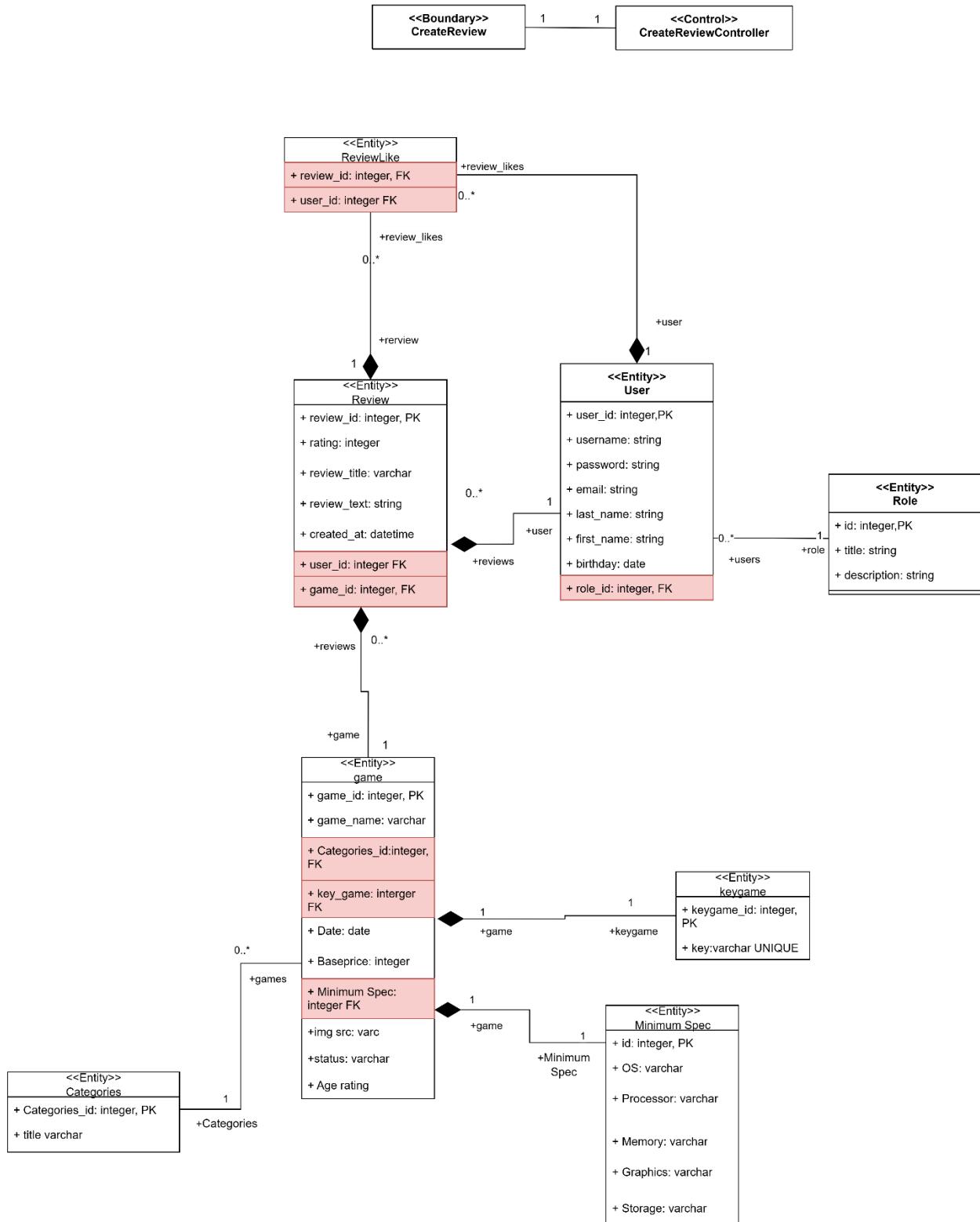
    c.JSON(http.StatusOK, gin.H{
        "review_id": rid,
        "likes":     cnt,
        "liked":     liked,
    })
}
```

## Communication Diagram

Activity	เป็นคำสั่งสำหรับระบบได้ หรือไม่	ถ้าเป็น ,วัตถุที่รับหน้าที่ ทำงาน คืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click “ไปหน้า Create Review”	เป็นคำสั่ง สั่งให้ UI โหลดหน้า	:createreviewUI	เปิด ()
โหลดข้อมูลสมาชิก	เป็นคำสั่ง เกิดการสั่งให้โหลด ข้อมูลของสมาชิกในระบบ	:CreateReviewController	โหลดข้อมูลสมาชิก(userId)
ค้นหาไอเดียสมาชิกในฐานข้อมูล	เป็นคำสั่ง	u :user	ค้นหาด้วยไอเดีย (user_id)
โหลดข้อมูลรายการเกมที่เคยซื้อ ไปแล้ว	เป็นคำสั่ง	g :Game	ค้นหาด้วยไอเดีย(game_id)
แสดงหน้าจอสำหรับสร้าง Review	ไม่เป็นคำสั่ง	-	-
เลือกดาว(ได้ rating)	ไม่เป็นคำสั่ง	-	-
ตั้งชื่อ(ได้ review_title)	ไม่เป็นคำสั่ง	-	-
ใส่ข้อมูลรีวิว(ได้ review_text)	ไม่เป็นคำสั่ง	-	-
กดปุ่ม “Submit Review”	เป็นคำสั่ง ระบบทำการจัดเก็บ ข้อมูลReview	:CreateReviewController	บันทึกการสร้างข้อมูลรีวิว (rating, review_title,review_text)
สร้างข้อมูล entity Review โดย โยง entity User เช็คค่า rating, review_title,review_text	เป็นคำสั่ง	r :Review	สร้าง (rating, review_title, review_text, u)
บันทึก entity Review	เป็นคำสั่ง	r :Review	บันทึก
แสดงข้อความสร้าง “สร้างรีวิว สำเร็จ”	ไม่เป็นคำสั่ง	-	-



## Class Diagram at Design Level



B6627713 นายทองนรินทร์ แย้มศรี

ระบบหลัก ระบบร้านขายเกมออนไลน์

ระบบย่อย ระบบจัดการโปรโมชั่น

เมื่อผู้ใช้งาน ซื้อเกมจากประเภทต่าง ๆ ที่มีอยู่ในระบบ โดยระบบจะมีการแสดงผลของโปรโมชั่นที่กำลังจัดอยู่ เช่น ส่วนลดเกม รายการเกมแจกฟรี หรือดีลพิเศษ เพื่อให้ผู้ใช้งานสามารถเลือกซื้อเกมได้ในราคาที่คุ้มค่ามากยิ่งขึ้น ผู้ใช้งานสามารถติดตามรายละเอียดของโปรโมชั่น เช่น อัตราส่วนลด ระยะเวลา

นอกจากนี้ ระบบยังรองรับการจัดการโปรโมชั่นโดยผู้ดูแลระบบ (Admin) ซึ่งสามารถสร้างโปรโมชั่นใหม่ เช่น ส่วนลดเป็นเปอร์เซ็นต์ รวมถึงสามารถกำหนดช่วงเวลาเริ่มต้นและสิ้นสุดของโปรโมชั่นได้ ระบบจะต้องเปิดให้สามารถแก้ไขรายละเอียดโปรโมชั่นได้ รวมถึงลบโปรโมชั่นเมื่อสิ้นสุดหรือไม่ต้องการใช้งานแล้ว เมื่อเกมที่อยู่ภายใต้โปรโมชั่นถูกกลบออกจากระบบ ระบบจะต้องลบข้อมูลโปรโมชั่นที่เกี่ยวข้องทั้งหมดโดยอัตโนมัติ เช่นกัน เพื่อป้องกันข้อมูลผิดพลาดหรือโปรโมชั่นที่ไม่มีสินค้า

### User Story ระบบจัดการโปรโมชั่น

ในบทบาทของ (As a) ผู้ดูแลระบบ (Admin)

(I want to) สร้างโปรโมชั่นสำหรับเกมต่าง ๆ

(So that) ฉันจะสามารถกระตุ้นยอดขายและดึงดูดผู้ใช้งานให้ซื้อเกมมากขึ้นได้

## Output บนหน้าจอ

- ผู้ดูแลระบบ (Admin) กดสร้างโปรโมชั่นเลือกเกมที่อยากรับให้ร่วมโปรโมชั่นแล้วเพิ่มส่วนลด-> จากนั้นระบบทำการบันทึกข้อมูลโปรโมชั่น และแสดงโปรโมชั่นบนหน้าหลักของร้านค้า

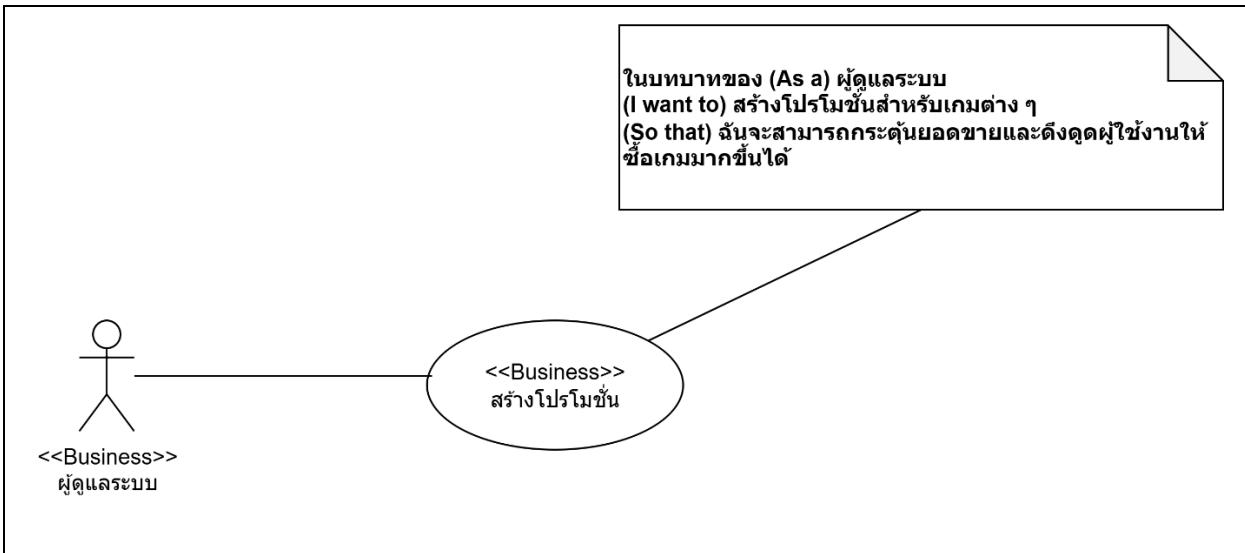
## Output ของข้อมูล

- ระบบสร้างข้อมูลโปรโมชั่น เพื่อผู้ดูแลระบบ (Admin) เพิ่มข้อมูลของโปรโมชั่น เกมที่ร่วมรายการและส่วนลด -> ระบบทำการบันทึกข้อมูลโปรโมชั่นกับเกมที่ผู้ดูแลระบบ (Admin) ได้เลือก และแสดงที่หน้าหลักของร้านค้า

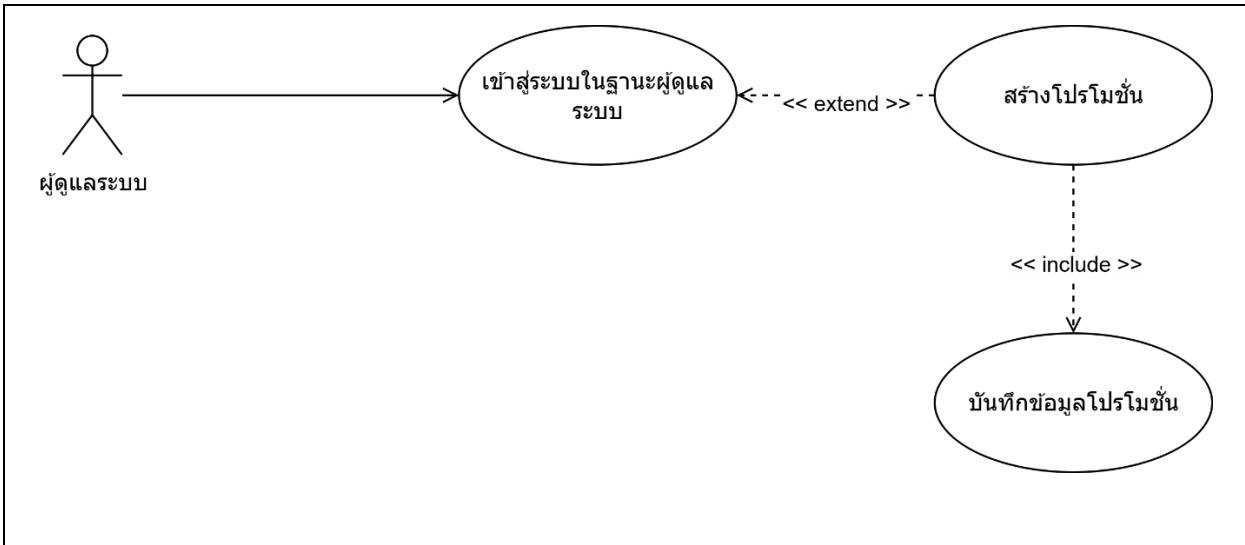
คำนามที่อาจจะกล้ายมาเป็นตารางในฐานข้อมูลของระบบ

คำนาม	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
เกม	เกี่ยวข้องโดยตรง เนื่องจากต้องใช้ข้อมูลเกม ในการสร้างโปรโมชั่น
โปรโมชั่น	เกี่ยวข้องโดยตรง เนื่องจากระบบนี้เป็นการสร้างโปรโมชั่นขึ้นมา
ผู้ดูแลระบบ (Admin)	เกี่ยวข้องโดยตรง เนื่องจากผู้ดูแลระบบ (Admin) เป็นคนเลือกสร้างโปรโมชั่นขึ้นมา และเลือกเกมที่จะเข้าร่วม
ส่วนลดเกม	เกี่ยวข้องโดยตรง เนื่องจากใช้แสดงถึงส่วนลดกับเกมที่เลือก

## Business Use Case Diagram (แบบเดี่ยว)



## System use Case Diagram (แบบเดี่ยว)



## ขั้นตอนการแปลง Business Use Case Diagram ให้เป็น System Use Case Diagram

### พิจารณาประเด็นที่ 1

Business Actor "ผู้ดูแลระบบ" สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์ที่ระบบงานได้โดยตรงหรือไม่

ตอบ ได้ ผู้ดูแลระบบ สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์

เพราะฉะนั้น ผู้ดูแลระบบ จะถูกจัดเป็น System Actor ได้

### พิจารณาประเด็นที่ 2

Business Use Case “สร้างໂປຣມີ່ນໍ້າ” สามารถถูกลายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบคอมพิวเตอร์ได้หรือไม่  
ต้องมีขั้นตอนทาง Security มาเกี่ยวหรือไม่

ตอบ Business Use Case “สร้างໂປຣມີ່ນໍ້າ” สามารถถูกลายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบคอมพิวเตอร์ได้  
และต้องการขั้นตอนทาง Security (มี Privacy มาเกี่ยวข้อง)

เพราะฉะนั้น Business Use Case “สร้างໂປຣມີ່ນໍ້າ” จะถูกจัดเป็น System Use Case มากกว่า 1 Use Case  
จะประกอบไปด้วย

- 1 System Use Case สำหรับ “เข้าระบบในฐานะผู้ดูแลระบบ”
  - 2 System Use Case สำหรับ “สร้างໂປຣມີ່ນໍ້າ” แต่กิจกรรมของระบบ จะยังไม่ครบตาม Requirements  
ที่ต้องการ
  - 3 System Use Case สำหรับ “บันทึกข้อมูลໂປຣມີ່ນໍ້າ”
- 
- System Use Case สำหรับ “เข้าระบบในฐานะผู้ดูแลระบบ”
  - System Use Case สำหรับ “สร้างໂປຣມີ່ນໍ້າ”
  - System Use Case สำหรับ “บันทึกข้อมูลໂປຣມີ່ນໍ້າ”

### พิจารณาประเด็นที่ 3

ถ้าสมาชิก “เข้าระบบในฐานะสมาชิก” มาแล้วจำเป็นที่ต้อง “สร้างโปรโมชั่น” ทุกครั้งหรือไม่

ตอบ ไม่

แปลว่า System Use Case “สร้างโปรโมชั่น” เป็นกิจกรรมทางเลือกหลังจากทำ System Use Case “เข้าระบบ ในฐานะผู้ดูแลระบบ”

### พิจารณาประเด็นที่ 4

ถ้าสมาชิก “เข้าระบบในฐานะสมาชิก” มาแล้วจำเป็นที่ต้อง “บันทึกข้อมูลโปรโมชั่น” ทุกครั้งหรือไม่

ตอบ ไม่ และเป็น Use Case ที่ไม่เกี่ยวข้องกัน

### พิจารณาประเด็นที่ 5

ถ้าผู้ดูแลระบบ “สร้างโปรโมชั่น” และ

จำเป็นต้อง “บันทึกข้อมูลโปรโมชั่น” ทุกครั้งหรือไม่

ตอบ ใช่ จำเป็น

แปลว่า System Use Case “สร้างโปรโมชั่น” จะต้องรวมขั้นตอนของ

System Use Case “บันทึกข้อมูลโปรโมชั่น” ไว้ด้วยเสมอ

การจำลองตัวอย่างตารางและข้อมูล (เพื่อใช้ในการเตรียมร่าง User Interface, System Activity Diagram และ Class Diagram)

จาก Requirements, จาก ข้อมูล Output และ Entity ที่ได้ออกแบบไว้เราจะนำมาสมมติเป็นตารางในฐานข้อมูล โดยกำหนด Primary Key เป็นตัวเลขรวมถึงการสมมติ Foreign Key เพื่อเชื่อมโยงข้อมูลระหว่างตาราง ซึ่งจะช่วยให้สามารถออกแบบและวิเคราะห์ระบบได้อย่างมีโครงสร้างและชัดเจน

## วิเคราะห์ Entity ที่เกี่ยวข้อง

### Promotion

ชื่อ Entity: Promotion

ข้อมูลที่ต้องจัดเก็บ

- Promotion\_ID: เป็น Primary Key เก็บในรูปแบบ integer ไม่ซ้ำ และไม่เป็นค่า Null
- Title: เก็บในรูปแบบ varchar และไม่เป็นค่า Null
- Description: เก็บในรูปแบบ text และไม่เป็นค่า Null
- DiscountValue: เก็บในรูปแบบ integer และไม่เป็นค่า Null
- StartDate: เก็บในรูปแบบ datetime และไม่เป็นค่า Null
- EndDate: เก็บในรูปแบบ datetime และไม่เป็นค่า Null
- UserID: FK จัดอิงไปยัง User\_ID และไม่เป็นค่า Null
- Status: เก็บในรูปแบบ Bool

Promotion_ID INTEGER NOT NULL, PK	Title VARCHAR NOT NULL	Description TEXT, NOT NULL	DiscountValue INTEGER, NOT NULL	StartDate DATETIME, NOT NULL	EndDate DATETIME, NOT NULL	UserID INTEGER NOT NULL, FK	PromoImage VARCHAR NOT NULL
8001	Summer Sale 2025	ลดเกมพิเศษ หน้าร้อน	30	2025-07- 01 00:00:00	2025-07- 31 23:59:59	01	/images/8001.jpg
8002	Weekend Sale	ลดเกมพิเศษ สุดสัปดาห์	10	2025-08- 05 00:00:00	2025-08- 07 23:59:59	02	/images/8002.jpg

## Promotion\_Game

ชื่อ Entity: Promotion\_Game

ข้อมูลที่ต้องจัดเก็บ

- Promotion\_game\_id เป็น Primary Key เก็บในรูปแบบ integer ไม่ซ้ำ และไม่เป็นค่า Null
  - PromotionID: FK จ้างอิงไปยัง Promotion.ID และไม่เป็นค่า Null
  - GameID: FK จ้างอิงไปยัง Game.ID และไม่เป็นค่า Null

Promotion_game_id INTEGER NOT NULL,PK	PromotionID INTEGER NOT NULL,FK	GameID INTEGER NOT NULL,FK
1001	8001	2001
1002	8002	2002

## หลักการออกแบบ User Interface

- เมื่อได้กีตามที่มี Foreign Key (FK) ซึ่งจาก ตารางหลักกลับไปยัง ตารางสนับสนุน  
ให้ใช้ ComboBox เป็นตัวเลือกเพื่อเชื่อมโยงข้อมูลใน User Interface
- สำหรับ Field ประเภทอื่นๆ ให้สร้างตามประเภทของข้อมูล เช่น
  - Textbox สำหรับข้อความทั่วไป
  - Password สำหรับข้อมูลรหัสผ่าน
  - Datetime Picker สำหรับเลือกวันและเวลา
  - Numeric Input สำหรับป้อนตัวเลข

## UI Design



### Create Promotion

Select game

discount

Selected Games

Add a photo

cancel

create

## การเตรียม System Activity Diagram

- 1 Business Use Case (1 User Story) จะถูกแปลงเป็น 1 System Activity Diagram
- System Activity Diagram คือ การบรรยายลำดับของกิจกรรม (Activity) ระหว่าง ผู้ใช้ (คน) กับ ระบบ

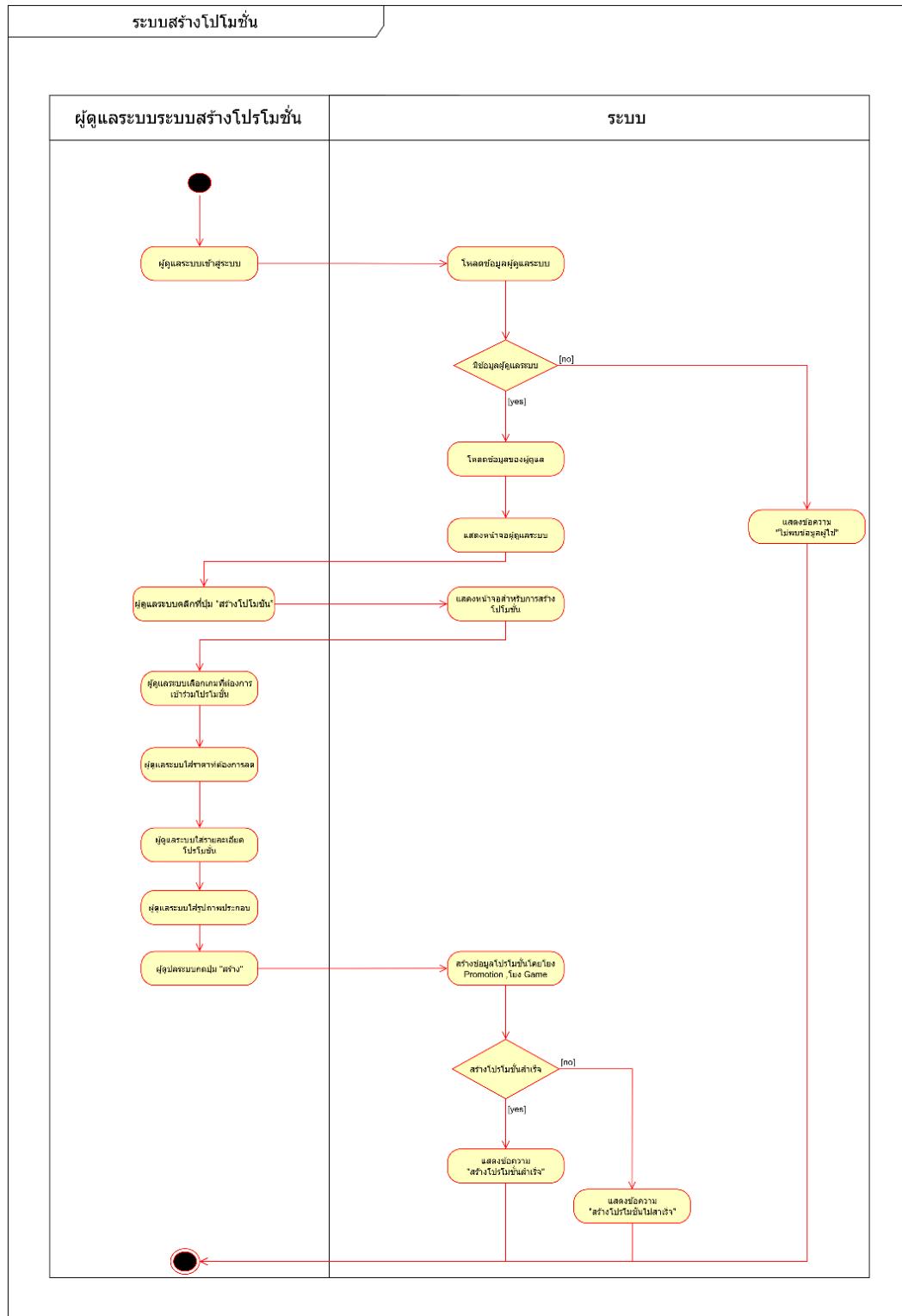
## หลักสำคัญ ของการออกแบบ System Activity Diagram ได้แก่

- ระบุว่า ผู้ใช้ทำอะไร (Input)
- ระบบประมวลผลอะไร (Process)
- ระบบตอบกลับอะไร (Output)

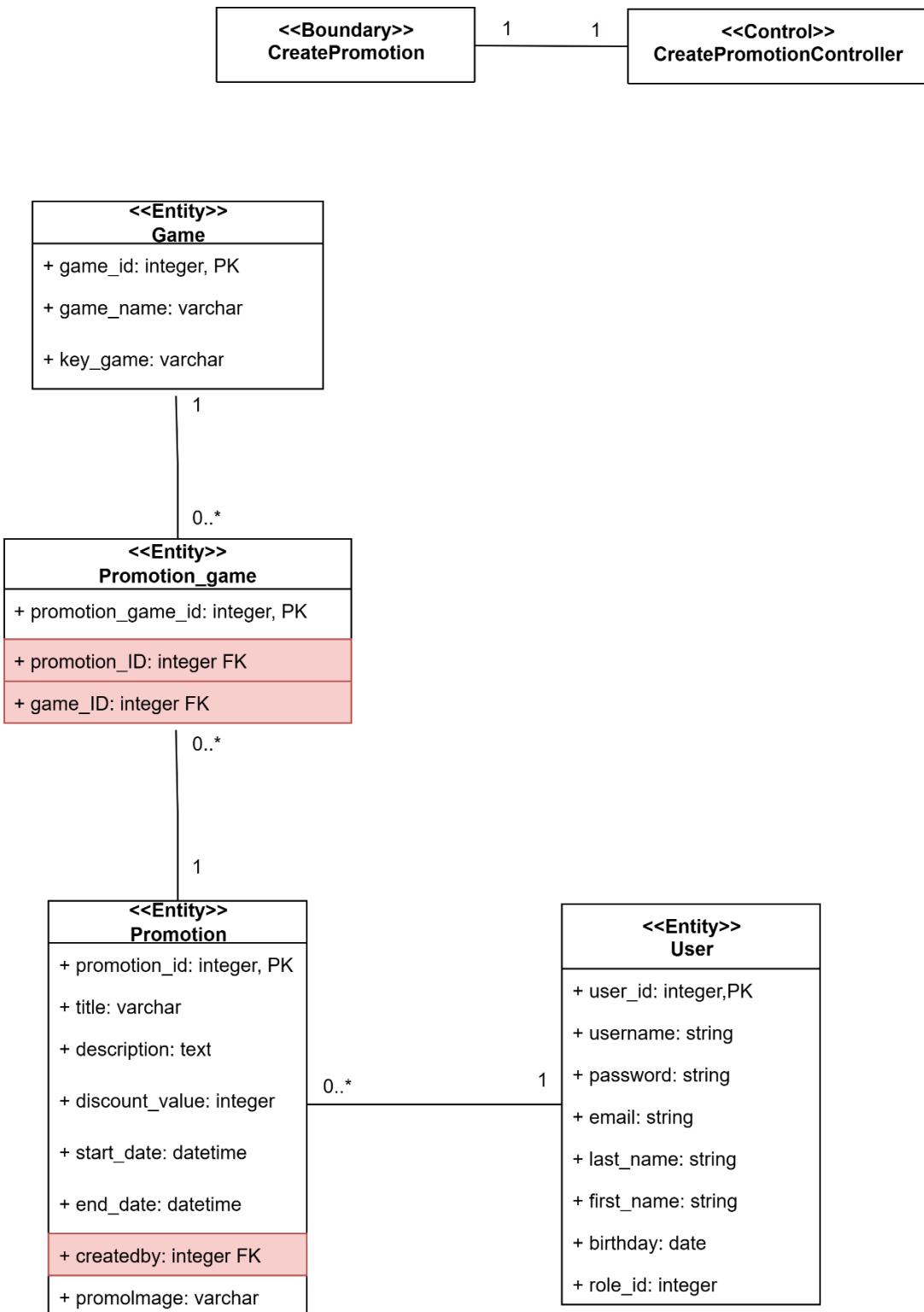
## System Activity Diagram ต้องลืนสุดที่การสร้าง Output ดังนี้

1. บันทึกข้อมูลลงในฐานข้อมูล
2. แสดงผลข้อมูลผ่านหน้าจอ (User Interface) โดยต้องสอดคล้องกับ User Story ที่กำหนดไว้

## Activity Diagram

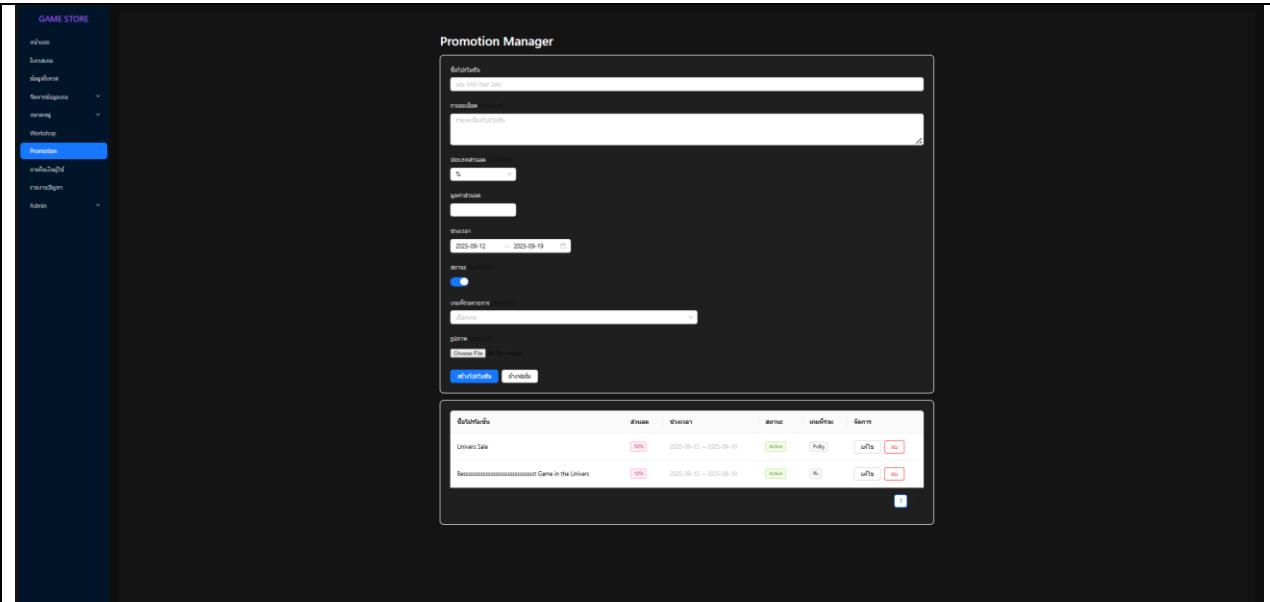


## Class Diagram at Analysis Level



## Frontend

### Promotion UI



```
import { useEffect, useMemo, useState } from "react";
import {
  Layout, Card, Form, Input, InputNumber, DatePicker, Switch,
  Button, Space, Table, Tag, Popconfirm, message, Typography, Select, Upload
} from "antd";
import { UploadOutlined } from "@ant-design/icons";
import dayjs, { Dayjs } from "dayjs";
import type { Promotion, DiscountType } from "../../interfaces/Promotion";
import {
  listPromotions,
  listGames,
  createPromotion,
  updatePromotion,
  deletePromotion,
} from "../../services/promotions";
import { useAuth } from "../../context/AuthContext";

type GameLite = { id: string; title: string };

const { Content } = Layout;
const { Title, Text } = Typography;
```

```

const { RangePicker } = DatePicker;

type FormValues = {
  title: string;
  description?: string;
  discount_type: DiscountType;
  discount_value: number;
  status: boolean;
  dateRange: [Dayjs, Dayjs];
  gameIds: string[];
};

export default function PromotionManager() {
  const [form] = Form.useForm<FormValues>();
  const [data, setData] = useState<Promotion>([]);
  const [games, setGames] = useState<GameLite[]>([]);
  const [editingId, setEditingId] = useState<number | null>(null);
  const [promoFile, setPromoFile] = useState<File | null>(null);
  const { id: currentUserID, token } = useAuth();
  const isEdit = useMemo(() => editingId !== null, [editingId]);
  const discountType = Form.useWatch("discount_type", form);

  const normalizePromo = (p: any): Promotion => ({
    ID: p.ID ?? p.id,
    title: p.title,
    description: p.description,
    discount_type: p.discount_type,
    discount_value: p.discount_value,
    start_date: p.start_date,
    end_date: p.end_date,
    status: p.status,
    promo_image: p.promo_image,
    user_id: p.user_id,
    user: p.user,
    games: p.games,
  });

  useEffect(() => {
    const load = async () => {
      try {
        const [gameRows, promoRows] = await Promise.all([
          listGames(),
          listPromotions(true),
        ]);
        setGames(
          gameRows.map(g => ({
            id: String(g.ID),
            title: g.game_name,
          })),
        );
      }
    }
  });
}

```

```

    );
    setData(promoRows.map(p => normalizePromo(p)));
} catch {
    message.error("ໂທລດຂໍອມລັບແຫວາ");
}
};

load();
}, []);
};

const gameOptions = useMemo(
() => games.map(g => ({ label: g.title, value: g.id })),
[games]
);

const onSubmit = async () => {
try {
    const values = await form.validateFields();
    const body = new FormData();
    body.append("title", values.title);
    if (values.description) body.append("description", values.description);
    body.append("discount_type", values.discount_type);
    body.append("discount_value", String(values.discount_value));
    body.append("start_date", values.dateRange[0].toISOString());
    body.append("end_date", values.dateRange[1].toISOString());
    body.append("status", String(values.status));
    values.gameIds.forEach(id => body.append("game_ids", id));
    if (currentUser != null) {
        body.append("user_id", String(currentUser));
    }
    if (promoFile) {
        body.append("promo_image", promoFile);
    }
    const saved = await (isEdit
        ? updatePromotion(editingId!, body, token || undefined)
        : createPromotion(body, token || undefined));
    const normalized = normalizePromo(saved);
    setData(prev => (
        isEdit
        ? prev.map(p => (p.ID === normalized.ID ? normalized : p))
        : [...prev, normalized]
    ));
    message.success(isEdit ? "ອັບເດດແລ້ວ" : "ສ້າງແລ້ວ");
    form.resetFields();
    setPromoFile(null);
    setEditingId(null);
} catch {
    message.error("ບັນທຶກໄຟສໍາເລັດ");
}
};

```

```

const onDelete = async (id: number) => {
  try {
    await deletePromotion(id, token || undefined);
    setData(prev => prev.filter(p => p.ID !== id));
    message.success("ลบແລ້ວ");
  } catch {
    message.error("ລັບໄມ້ສໍາເລົດ");
  }
};

const columns = [
  {
    title: "ຊື່ໂປຣໂມໜັນ",
    dataIndex: "title",
    key: "title",
    render: (t: string) => <Text style={{ color: "black" }}>{t}</Text>,
  },
  {
    title: "ສ່ວນຄດ",
    key: "discount",
    render: (_: any, r: Promotion) =>
      r.discount_type === "PERCENT" ? (
        <Tag color="magenta">{r.discount_value}%</Tag>
      ) : (
        <Tag color="volcano">{-r.discount_value}</Tag>
      ),
  },
  {
    title: "ចົວເລາ",
    key: "date",
    render: (_: any, r: Promotion) => (
      <Text style={{ color: "#ccc" }}>
        {dayjs(r.start_date).format("YYYY-MM-DD")} → {dayjs(r.end_date).format("YYYY-MM-DD")}
      </Text>
    ),
  },
  {
    title: "ສະຖານະ",
    dataIndex: "status",
    key: "status",
    render: (a: boolean) => (a ? <Tag color="green">Active</Tag> : <Tag>Inactive</Tag>),
  },
  {
    title: "ເກມທີ່ຈະມີ",
    key: "games",
    render: (_: any, r: Promotion) => (
      <div>
        {r.games?.map((g: any) => {

```

```

const title =
  games.find(gg => gg.id === String(r.ID))?.title || g.game_name || g.title || g.ID;
return <Tag key={g.ID}>{title}</Tag>;
})
</div>
),
},
{
  title: "ຈັດກາງ",
  key: "actions",
  render: (_: any, r: Promotion) => (
    <Space>
    <Button
      onClick={() => {
        setEditingId(r.ID);
        form.setFieldsValue({
          title: r.title,
          description: r.description,
          discount_type: r.discount_type,
          discount_value: r.discount_value,
          dateRange: [dayjs(r.start_date), dayjs(r.end_date)],
          status: r.status,
          gameIds: r.games?.map((g: any) => String(g.ID)) || [],
        });
        setPromoFile(null);
      }}
    >
      ແກ້ໄຂ
    </Button>
    <Popconfirm
      title="ລົບໂປຣໂມນີ້?"
      okText="ລົບ"
      cancelText="ຍົກເລີກ"
      okButtonProps={{ danger: true }}
      onConfirm={() => onDelete(r.ID)}
    >
      <Button danger>ລົບ</Button>
    </Popconfirm>
  </Space>
),
},
];
}

return (
  <Layout style={{ minHeight: "100vh", background: "#0f0f0f" }}>
    <Content style={{ padding: 24, background: "#141414" }}>
      <div style={{ maxWidth: 1200, margin: "0 auto" }}>
        <Title level={2} style={{ color: "white" }}>Promotion Manager</Title>

```

```

<Card style={{ marginBottom: 16, background: "#1f1f1f", color: "white", borderRadius: 10 }}>
  <Form><FormValues>
    layout="vertical"
    form={form}
    requiredMark="optional"
    initialValues={{
      discount_type: "PERCENT" as DiscountType,
      status: true,
      dateRange: [dayjs(), dayjs().add(7, "day")],
    }}
  >
  <Form.Item
    name="title"
    label={<span style={{ color: "#ccc" }}>ชื่อโปรโมชัน</span>}
    rules={[{ required: true, message: "กรอกชื่อโปรโมชัน" }]}
  >
    <Input placeholder="เช่น Mid-Year Sale" />
  </Form.Item>

  <Form.Item name="description" label={<span style={{ color: "#ccc" }}>รายละเอียด</span>}>
    <Input.TextArea rows={3} placeholder="รายละเอียดโปรโมชัน" />
  </Form.Item>

  <Form.Item
    name="discount_type"
    label={<span style={{ color: "#ccc" }}>ประเภทส่วนลด</span>}
  >
    <Select
      style={{ width: 160 }}
      options={[
        { label: "%", value: "PERCENT" },
        { label: "จำนวนเงิน", value: "AMOUNT" },
      ]}
    />
  </Form.Item>

  <Form.Item
    name="discount_value"
    label={<span style={{ color: "#ccc" }}>มูลค่าส่วนลด</span>}
    rules={[{ required: true, message: "กรอกมูลค่าส่วนลด" }]}
  >
    <InputNumber
      min={0}
      max={discountType === "PERCENT" ? 100 : undefined}
      style={{ width: 160 }}
    />
  </Form.Item>

  <Form.Item

```

```

        name="dateRange"
        label={<span style={{ color: "#ccc" }}>ช่วงเวลา</span>}
        rules={[{ required: true, message: "เลือกช่วงเวลา" }]}
      >
      <RangePicker />
    </Form.Item>

    <Form.Item
      name="status"
      label={<span style={{ color: "#ccc" }}>สถานะ</span>}
      valuePropName="checked"
    >
      <Switch />
    </Form.Item>

    <Form.Item name="gameIds" label={<span style={{ color: "#ccc" }}>เกมที่ร่วมรายการ</span>}>
      <Select
        mode="multiple"
        placeholder="เลือกเกม"
        options={gameOptions}
        style={{ maxWidth: 600 }}
      />
    </Form.Item>

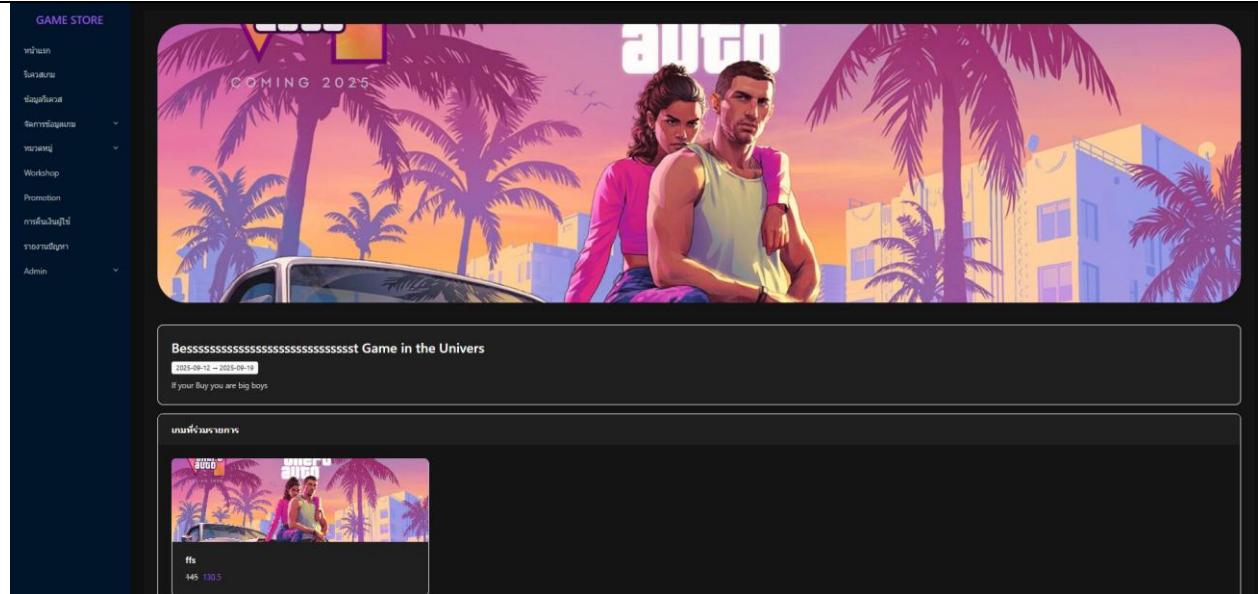
    <Form.Item label={<span style={{ color: "#ccc" }}>รูปภาพ</span>}>
      <input
        type="file"
        accept="image/*"
        onChange={e => setPromoFile(e.target.files?.[0] || null)}
      />
    </Form.Item>

    <Space>
      <Button type="primary" onClick={onSubmit}>
        {isEdit ? "บันทึกการแก้ไข" : "สร้างโปรโมชั่น"}
      </Button>
      <Button onClick={() => { form.resetFields(); setEditingId(null); }}>
        ล้างฟอร์ม
      </Button>
    </Space>
  </Form>
</Card>

<Card style={{ background: "#1f1f1f", color: "white", borderRadius: 10 }}>
  <Table rowKey="ID" columns={columns as any} dataSource={data} pagination={{ pageSize: 8 }} />
</Card>
</div>
</Content>
</Layout>

```

```
};  
}
```



```
import { useEffect, useState } from "react";  
import { useParams, Link, useNavigate } from "react-router-dom";  
import { Layout, Typography, Card, Tag, Button, Empty, Row, Col } from "antd";  
import Sidebar from "../components/Sidebar";  
import GameCard from "../components/GameCard";  
import type { Promotion } from "../interfaces/Promotion";
```

```

import type { Game } from "../../interfaces/Game";
import { getPromotion } from "../../services/promotions";
import dayjs from "dayjs";

const { Content } = Layout;
const { Title, Text } = Typography;

const base_url = import.meta.env.VITE_API_URL || "http://localhost:8088";

function applyDiscount(
  price: number,
  type: Promotion["discount_type"],
  value: number,
): number {
  if (value <= 0) return price;
  switch (type) {
    case "PERCENT":
      return price * (1 - value / 100);
    case "AMOUNT":
      return price < value ? 0 : price - value;
    default:
      return price;
  }
}

export default function PromotionDetail() {
  const { id } = useParams<{ id: string }>();
  const [promotion, setPromotion] = useState<Promotion | null>(null);
  const [games, setGames] = useState<Game[]>([]);
  const navigate = useNavigate();

  const resolveImgUrl = (src?: string) => {
    if (!src) return "";
    if (src.startsWith("blob:")) return "";
    if (src.startsWith("data:image/")) return src;
    if (
      src.startsWith("http://") ||
      src.startsWith("https://") ||
      src.startsWith("blob:")
    ) {
      return src;
    }
    const clean = src.startsWith("/") ? src.slice(1) : src;
    return `${base_url}/${clean}`;
  };

  useEffect(() => {
    const load = async () => {
      if (!id) return;

```

```

try {
  const data = await getPromotion(Number(id), true);
  setPromotion(data);
  if (data.games) {
    setGames(
      data.games.map((g: Game) => ({
        ...g,
        discounted_price: applyDiscount(
          g.base_price,
          data.discount_type,
          data.discount_value,
        ),
      })),
    );
  }
} catch {
  setPromotion(null);
}
};

load();
}, [id]);

if (!promotion) {
  return (
    <Layout style={{ minHeight: "100vh", background: "#0f0f0f" }}>
      <Sidebar />
      <Content style={{ padding: 24, background: "#141414" }}>
        <Empty description="ไม่พบโปรดโน้มั่น" />
      </Content>
    </Layout>
  );
}

return (
  <Layout style={{ minHeight: "100vh", background: "#0f0f0f" }}>

    <Content style={{ padding: 24, background: "#141414" }}>
      {promotion.promo_image && (
        <img
          src={resolvelImgUrl(promotion.promo_image)}
          alt={promotion.title}
          style={{
            width: "100%",
            height: 500,
            objectFit: "cover",
            display: "block",
            borderRadius: 50,
            marginBottom: 40,
          }}
      )}
  
```

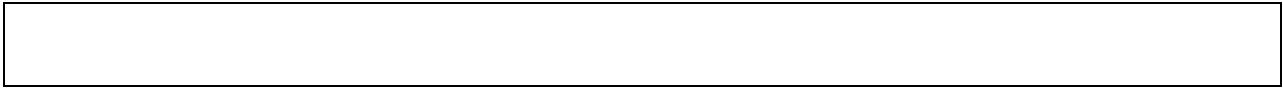
```

        />
    )}
<div style={{ maxWidth: 2000, margin: "0 auto" }>

<Card style={{ background: "#1f1f1f", color: "white", borderRadius: 10, marginBottom: 16 }}>
<Title level={3} style={{ color: "white", margin: 0 }}>
    {promotion.title}
</Title>
<div style={{ marginTop: 8 }}>
    <Tag>
        {dayjs(promotion.start_date).format("YYYY-MM-DD")} → {dayjs(promotion.end_date).format("YYYY-MM-DD")}
    </Tag>
</div>
{promotion.description && (
    <div style={{ color: "#ccc", marginTop: 8 }}>{promotion.description}</div>
)}
</Card>

<Card title={{<span style={{ color: "white" }}>เกมที่ร่วมรายการ</span>}}
    headStyle={{ background: "#1f1f1f", color: "white" }}
    bodyStyle={{ background: "#141414" }}
    style={{ background: "#1f1f1f", color: "white", borderRadius: 10 }}>
    >
    {games.length > 0 ? (
        <Row gutter={[16, 16]}>
            {games.map((g) => (
                <Col key={g.ID} xs={24} sm={12} md={8} lg={6}>
                    <GameCard
                        game={g}
                        imgSrc={resolveImgUrl(g.img_src)}
                        onClick={() => navigate(`'/game/${g.ID}`)}>
                    />
                </Col>
            ))}
        </Row>
    ) : (
        <Text style={{ color: "#888" }}>ยังไม่มีเกม</Text>
    )}
    <div style={{ marginTop: 12 }}>
        <Link to="/promotions">
            <Button>ย้อนกลับ</Button>
        </Link>
    </div>
</Card>
</div>
</Content>
</Layout>
);
}

```



## Backend

### Game Entity

```
package entity

import (
    "time"

    "gorm.io/gorm"
)

type Game struct {
    gorm.Model
    GameName string `json:"game_name"`
    //KeyGameID  uint     `json:"key_id"`
    CategoriesID int      `json:"categories_id"`
    Categories  Categories `json:"categories" gorm:"foreignKey:CategoriesID"`
    Date        time.Time `json:"release_date" gorm:"autoCreateTime"`
    BasePrice   int       `json:"base_price"`
    Status      string    `json:"status" gorm:"type:varchar(20);default:'pending';not null;index"`
    Minimum_specID uint     `json:"minimum_spec_id"`
    MinimumSpec  MinimumSpec `json:"minimum_spec"`
    AgeRating   int       `json:"age_rating"`

    Requests []Request `gorm:"foreignKey:GameRefer"`

    Threads []Thread `gorm:"foreignKey:GameID" json:"threads,omitempty"`
    UserGames []UserGame `gorm:"foreignKey:GameID" json:"user_games,omitempty"`

    Reviews []Review `gorm:"foreignKey:GameID" json:"reviews,omitempty"`
    ReviewLikes []Review_Like `gorm:"foreignKey:GameID" json:"review_likes,omitempty"`

    Promotions []Promotion `json:"promotions,omitempty" gorm:"many2many:promotion_games"`
    PromotionGames []Promotion_Game `json:"promotion_games,omitempty" gorm:"foreignKey:GameID"`
    ImgSrc     string    `json:"img_src" gorm:"type:varchar(512)"`
}
```

### Promotion\_game Entity

```
package entity

import "gorm.io/gorm"
```

```

// ตารางกลาง Promotion <-> Game (เก็บข้อมูลต่อแผลได้ในอนาคต เช่น cap ต่อเกม)
type Promotion_Game struct {
    gorm.Model

    PromotionID uint      `json:"promotion_id" gorm:"index:idx_promo_game,unique;not null"`
    Promotion  *Promotion `json:"promotion"   gorm:"foreignKey:PromotionID;constraint:OnUpdate:CASCADE,OnDelete:CASCADE;"`

    GameID uint `json:"game_id" gorm:"index:idx_promo_game,unique;not null"`
    Game  *Game `json:"game"   gorm:"foreignKey:GameID;constraint:OnUpdate:CASCADE,OnDelete:CASCADE;"`  

    // (อปชัน) ถ้าวันหน้าต้อง override ส่วนลดเฉพาะเกมนี้ ก็เพิ่มคอลัมน์ได้ที่นี่
    // PerGameDiscount *int `json:"per_game_discount"`

}

func (Promotion_Game) TableName() string { return "promotion_games" }

```

## Promotion Entity

```

package entity

import (
    "time"

    "gorm.io/gorm"
)

type DiscountType string

const (
    DiscountPercent DiscountType = "PERCENT" // ลดเป็น %
    DiscountAmount DiscountType = "AMOUNT" // ลดเป็นจำนวนเงิน
)

type Promotion struct {
    gorm.Model

    Title      string      `json:"title"          gorm:"type:varchar(120);not null;index"`
    Description string      `json:"description"  gorm:"type:text"`
    DiscountType DiscountType `json:"discount_type" gorm:"type:varchar(20);not null;default:PERCENT"`
    DiscountValue int        `json:"discount_value" gorm:"not null;check:discount_value_nonneg,discount_value>=0"`
    StartDate  time.Time   `json:"start_date"    gorm:"not null;index"`
    EndDate   time.Time   `json:"end_date"     gorm:"not null;index"`
    Promolmage string      `json:"promo_image"`
    Status     bool        `json:"status"       gorm:"default:true;index" // เปิด/ปิดโปร
}

```

```
// ផ្តល់រាយពីរវិធីខាងក្រោម
UserID uint `json:"user_id" gorm:"index"`
User *User `json:"user" gorm:"foreignKey:UserID;constraint:OnUpdate:CASCADE,OnDelete:SET NULL;"`  
  

// Many-to-Many ជានាពារាងការងម promotion_games (នឹង struct ខាងក្រោម)
Games []Game `json:"games" gorm:"many2many:promotion_games"`
PromotionGames []Promotion_Game `json:"promotion_games" gorm:"foreignkey:PromotionID"`
```

## User Entity

```
package entity

import (
    "time"

    "gorm.io/gorm"
)

type User struct {
    gorm.Model
    Username string `gorm:"size:120;uniqueIndex" json:"username"`
    Password string `json:"-"` // អយ្សាគសង្គ័េបនៃ API
    Email string `gorm:"size:255;uniqueIndex" json:"email"`
    FirstName string `json:"first_name"`
    LastName string `json:"last_name"`
    Birthday time.Time `json:"birthday"`

    RoleID uint `gorm:"not null;index" json:"role_id"`
    Role Role `gorm:"constraint:OnUpdate:CASCADE,OnDelete:RESTRICT;" json:"role" // ការលួយ role ទាំងអស់ត្រូវការចុះឈ្មោះ`  
  

    Threads []Thread `gorm:"foreignKey:UserID" json:"threads,omitempty"`
    Comments []Comment `gorm:"foreignKey:UserID" json:"comments,omitempty"`
    Reactions []Reaction `gorm:"foreignKey:UserID" json:"reactions,omitempty"`
    Attachments []Attachment `gorm:"foreignKey:UserID" json:"attachments,omitempty"`
    Notifications []Notification `gorm:"foreignKey:UserID" json:"notifications,omitempty"`
    UserGames []UserGame `gorm:"foreignKey:UserID" json:"user_games,omitempty"`

    Reviews []Review `gorm:"foreignKey:UserID" json:"reviews,omitempty"`
    ReviewLikes []Review_Like `gorm:"foreignKey:UserID" json:"review_likes,omitempty"`

    Promotions []Promotion `json:"promotions,omitempty" gorm:"foreignKey:UserID"`
    Requests []Request `json:"request" gorm:"foreignKey:UserRefer"`
}
```



## Controller promotion\_controller

```
// backend/controllers/promotions.go
package controllers

import (
    "fmt"
    "mime/multipart"
    "net/http"
    "os"
    "path/filepath"
    "strings"
    "time"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
    "github.com/golang-jwt/jwt/v5"
    "gorm.io/gorm"
)

// ===== Promotion Controllers =====

// helper: ensure end after start
func validatePromoWindow(start, end time.Time) bool {
    return !start.IsZero() && !end.IsZero() && end.After(start)
}

func getUserId(c *gin.Context) (uint, error) {
    header := c.GetHeader("Authorization")
    if header == "" {
        return 0, fmt.Errorf("authorization header missing")
    }
    tokenString := strings.TrimPrefix(header, "Bearer ")
    secret := os.Getenv("JWT_SECRET")
    if secret == "" {
        secret = "secret"
    }
    token, err := jwt.Parse(tokenString, func(t *jwt.Token) (interface{}, error) {
        if _, ok := t.Method.(*jwt.SigningMethodHMAC); ok {
            return nil, fmt.Errorf("unexpected signing method")
        }
        return []byte(secret), nil
    })
    if err != nil || !token.Valid {
        return 0, fmt.Errorf("invalid token")
    }
    claims, ok := token.Claims.(jwt.MapClaims)
    if !ok {
```

```

        return 0, fmt.Errorf("invalid claims")
    }
    sub, ok := claims["sub"].(float64)
    if !ok {
        return 0, fmt.Errorf("invalid subject")
    }
    return uint(sub), nil
}

type createPromotionRequest struct {
    Title      string      `form:"title"      binding:"required"`
    Description string      `form:"description"`
    DiscountType entity.DiscountType `form:"discount_type"  binding:"required"`
    DiscountValue int        `form:"discount_value" binding:"required,min=0"`
    StartDate   time.Time   `form:"start_date"   binding:"required"`
    EndDate     time.Time   `form:"end_date"     binding:"required"`
    Promolmage  *multipart.FileHeader `form:"promo_image"`
    Status      *bool       `form:"status"`
    GameIDs    []uint      `form:"game_ids" // optional: set links to games
}

// POST /promotions
func CreatePromotion(c *gin.Context) {
    uid, err := getUserId(c)
    if err != nil {
        c.JSON(http.StatusUnauthorized, gin.H{"error": err.Error()})
        return
    }
    var req createPromotionRequest
    if err := c.ShouldBind(&req); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "invalid body: " + err.Error()})
        return
    }
    if !validatePromoWindow(req.StartDate, req.EndDate) {
        c.JSON(http.StatusBadRequest, gin.H{"error": "end_date must be after start_date"})
        return
    }

    var promolmagePath string
    if req.Promolmage != nil {
        uploadDir := filepath.Join("uploads", "promotions")
        if err := os.MkdirAll(uploadDir, os.ModePerm); err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": "failed to create directory"})
            return
        }
        filename := fmt.Sprintf("%d_%s", time.Now().UnixNano(), req.Promolmage.Filename)
        path := filepath.Join(uploadDir, filename)
        if err := c.SaveUploadedFile(req.Promolmage, path); err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": "failed to save file"})
        }
    }
}

```

```

        return
    }
    promolmagePath = path
}

promo := entity.Promotion{
    Title:      req.Title,
    Description: req.Description,
    DiscountType: req.DiscountType,
    DiscountValue: req.DiscountValue,
    StartDate:   req.StartDate,
    EndDate:     req.EndDate,
    Promolmage:  promolmagePath,
    Status:      true,
    UserID:      uid,
}
if req.Status != nil {
    promo.Status = *req.Status
}

db := configs.DB()

// validate games if provided
var games []entity.Game
if len(req.GameIDs) > 0 {
    if err := db.Where("id IN ?", req.GameIDs).Find(&games).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "cannot load games: " + err.Error()})
        return
    }
    if len(games) != len(req.GameIDs) {
        c.JSON(http.StatusBadRequest, gin.H{"error": "some game_ids were not found"})
        return
    }
    promo.Games = games
}

if err := db.Create(&promo).Error; err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": "create promotion failed: " + err.Error()})
    return
}

// reload with relations for response
if err := db.Preload("Games").First(&promo, promo.ID).Error; err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": "reload failed: " + err.Error()})
    return
}
c.JSON(http.StatusCreated, promo)
}

```

```

// GET /promotions
// query: status=true/false, active_now=true, with=games
func FindPromotions(c *gin.Context) {
    db := configs.DB().Model(&entity.Promotion{})
    with := c.Query("with")
    status := c.Query("status")
    activeNow := c.Query("active_now")

    if with == "games" {
        db = db.Preload("Games")
    }

    if status == "true" || status == "false" {
        db = db.Where("status = ?", status == "true")
    }

    if activeNow == "true" {
        now := time.Now()
        db = db.Where("start_date <= ? AND end_date >= ? AND status = 1", now, now)
    }

    var rows []entity.Promotion
    if err := db.Order("start_date desc").Find(&rows).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, rows)
}

// GET /promotions/:id
func GetPromotionByID(c *gin.Context) {
    var row entity.Promotion
    db := configs.DB().Preload("Games")
    if err := db.First(&row, c.Param("id")).Error; err != nil {
        if err == gorm.ErrRecordNotFound {
            c.JSON(http.StatusNotFound, gin.H{"error": "promotion not found"})
        } else {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        }
        return
    }
    c.JSON(http.StatusOK, row)
}

type updatePromotionRequest struct {
    Title      *string      `form:"title"`
    Description *string      `form:"description"`
    DiscountType *entity.DiscountType `form:"discount_type"`
    DiscountValue *int       `form:"discount_value" binding:"omitempty,min=0"`
    StartDate   *time.Time   `form:"start_date"`
    EndDate     *time.Time   `form:"end_date"`
}

```

```

Promolmage *multipart.FileHeader `form:"promo_image"`
Status     *bool           `form:"status"`
GameIDs    *[]uint        `form:"game_ids" // if present, replace mapping
}

// PUT /promotions/:id
func UpdatePromotion(c *gin.Context) {
    var req updatePromotionRequest
    if err := c.ShouldBind(&req); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "invalid body: " + err.Error()})
        return
    }

    var row entity.Promotion
    db := configs.DB()
    if err := db.First(&row, c.Param("id")).Error; err != nil {
        if err == gorm.ErrRecordNotFound {
            c.JSON(http.StatusNotFound, gin.H{"error": "promotion not found"})
        } else {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        }
        return
    }

    var promolmagePath string
    if req.Promolmage != nil {
        uploadDir := filepath.Join("uploads", "promotions")
        if err := os.MkdirAll(uploadDir, os.ModePerm); err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": "failed to create directory"})
            return
        }
        filename := fmt.Sprintf("%d_%s", time.Now().UnixNano(), req.Promolmage.Filename)
        path := filepath.Join(uploadDir, filename)
        if err := c.SaveUploadedFile(req.Promolmage, path); err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": "failed to save file"})
            return
        }
        promolmagePath = path
    }

    // validate date window if both provided
    if req.StartDate != nil && req.EndDate != nil {
        if !validatePromoWindow(*req.StartDate, *req.EndDate) {
            c.JSON(http.StatusBadRequest, gin.H{"error": "end_date must be after start_date"})
            return
        }
    }
}

// apply partial fields
updates := map[string]any{

```

```

if req.Title != nil {
    updates["title"] = *req.Title
}
if req.Description != nil {
    updates["description"] = *req.Description
}
if req.DiscountType != nil {
    updates["discount_type"] = *req.DiscountType
}
if req.DiscountValue != nil {
    updates["discount_value"] = *req.DiscountValue
}
if req.StartDate != nil {
    updates["start_date"] = *req.StartDate
}
if req.EndDate != nil {
    updates["end_date"] = *req.EndDate
}
if promolImagePath != "" {
    updates["promo_image"] = promolImagePath
}
if req.Status != nil {
    updates["status"] = *req.Status
}

if len(updates) > 0 {
    if err := db.Model(&row).Updates(updates).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
}

// replace game mapping if provided
if req.GameIDs != nil {
    var games []entity.Game
    if len(*req.GameIDs) > 0 {
        if err := db.Where("id IN ?", *req.GameIDs).Find(&games).Error; err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error": "cannot load games: " + err.Error()})
            return
        }
        if len(games) != len(*req.GameIDs) {
            c.JSON(http.StatusBadRequest, gin.H{"error": "some game_ids were not found"})
            return
        }
    }
    if err := db.Model(&row).Association("Games").Replace(games); err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": "update games failed: " + err.Error()})
        return
    }
}

```

```

}

if err := db.Preload("Games").First(&row, row.ID).Error; err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": "reload failed: " + err.Error()})
    return
}
c.JSON(http.StatusOK, row)
}

// DELETE /promotions/:id
func DeletePromotion(c *gin.Context) {
    if tx := configs.DB().Delete(&entity.Promotion{}, c.Param("id")); tx.Error != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": tx.Error.Error()})
        return
    } else if tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "promotion not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "deleted"})
}

// POST /promotions/:id/games
// Replace mapping with provided game_ids (idempotent)
func SetPromotionGames(c *gin.Context) {
    var req struct {
        GameIDs []uint `json:"game_ids" binding:"required"`
    }
    if err := c.ShouldBindJSON(&req); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "invalid body"})
        return
    }
    db := configs.DB()
    var promo entity.Promotion
    if err := db.First(&promo, c.Param("id")).Error; err != nil {
        if err == gorm.ErrRecordNotFound {
            c.JSON(http.StatusNotFound, gin.H{"error": "promotion not found"})
        } else {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        }
        return
    }
    var games []entity.Game
    if len(req.GameIDs) > 0 {
        if err := db.Where("id IN ?", req.GameIDs).Find(&games).Error; err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error": "cannot load games: " + err.Error()})
            return
        }
        if len(games) != len(req.GameIDs) {
            c.JSON(http.StatusBadRequest, gin.H{"error": "some game_ids were not found"})
        }
    }
}

```

```

        return
    }
}

if err := db.Model(&promo).Association("Games").Replace(games); err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": "update games failed: " + err.Error()})
    return
}

if err := db.Preload("Games").First(&promo, promo.ID).Error; err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": "reload failed: " + err.Error()})
    return
}

c.JSON(http.StatusOK, promo)
}

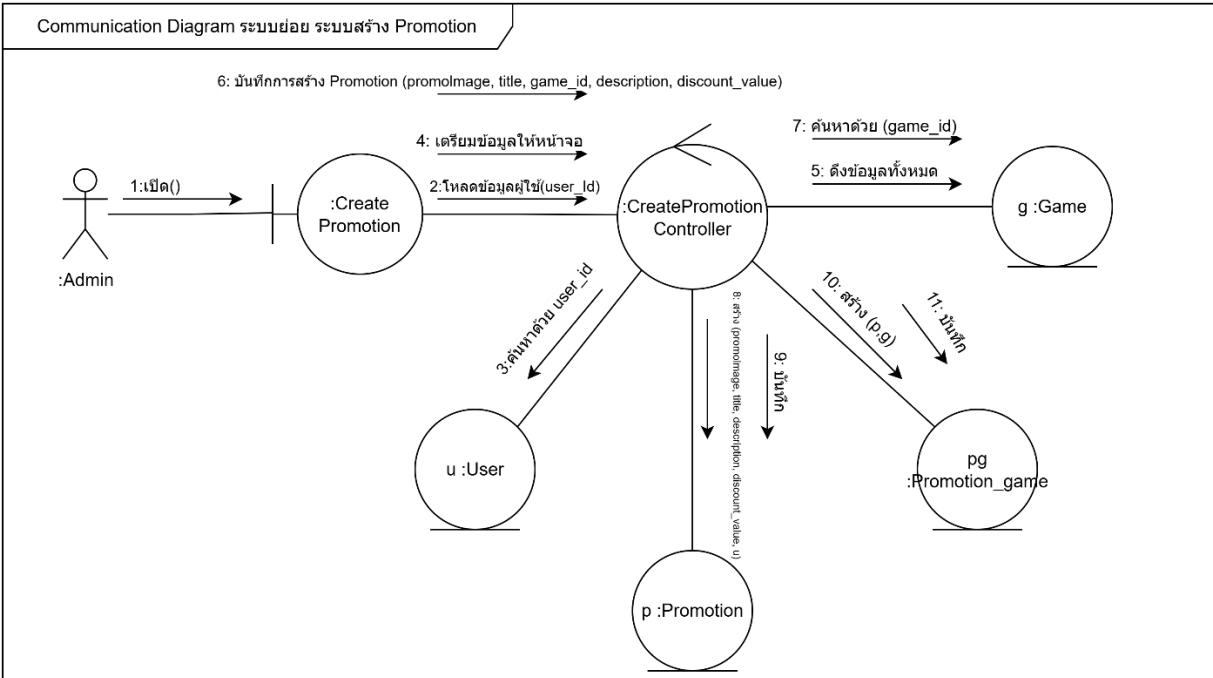
// GET /promotions-active
func FindActivePromotions(c *gin.Context) {
    now := time.Now()
    var rows []entity.Promotion
    if err := configs.DB().
        Preload("Games").
        Where("status = 1 AND start_date <= ? AND end_date >= ?", now, now).
        Order("end_date asc").
        Find(&rows).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, rows)
}

// util: apply discount to a price (for clients, optional)
func applyDiscount(price float64, t entity.DiscountType, v int) float64 {
    if v <= 0 {
        return price
    }
    switch t {
    case entity.DiscountPercent:
        return price * (1.0 - float64(v)/100.0)
    case entity.DiscountAmount:
        if price < float64(v) {
            return 0
        }
        return price - float64(v)
    default:
        return price
    }
}

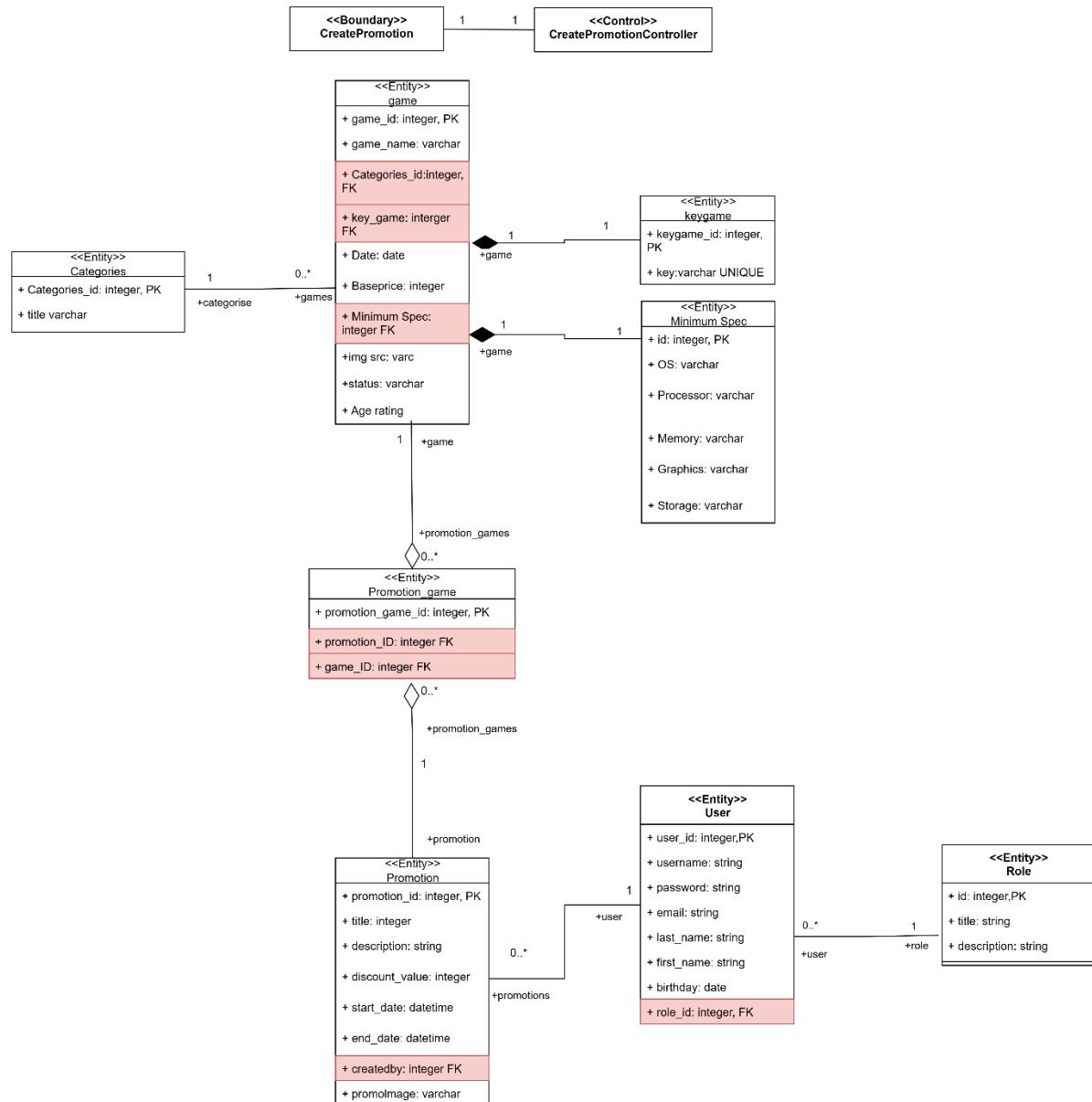
```

## Communication Diagram

Activity	เป็นคำสั่งสำหรับระบบได้ หรือไม่	ถ้าเป็น ,วัตถุที่รับหน้าที่ ทำงาน คืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click “ไปหน้า Create Promotion”	เป็นคำสั่ง สั่งให้ UI โหลดหน้า	:createPromotion UI	เปิด ()
โหลดข้อมูลสมาชิก	เป็นคำสั่ง เกิดการสั่งให้โหลด ข้อมูลของสมาชิกในระบบ	:CreatePromotionController	โหลดข้อมูลสมาชิก(user_Id)
ค้นหาไอเดียสมาชิกในฐานข้อมูล	เป็นคำสั่ง	u :User	ค้นหาด้วยไอเดีย (user_Id)
โหลดข้อมูลรายการเกม	เป็นคำสั่ง ให้โหลดเกมทั้งหมด	:CreatePromotionController g :Game	เตรียมข้อมูลให้หน้าจอ() ดึงข้อมูลทั้งหมด
แสดงหน้าจอสำหรับสร้าง Promotion	ไม่เป็นคำสั่ง	-	-
ใส่รูปภาพ(promolImage)	ไม่เป็นคำสั่ง	-	-
ตั้งชื่อ(ได้ title)	ไม่เป็นคำสั่ง	-	-
เลือกเกม(ได้ game_id)	ไม่เป็นคำสั่ง	-	-
ใส่ข้อมูล(ได้ description)	ไม่เป็นคำสั่ง	-	-
ใส่ข้อส่วนลด(ได้ discount_value)	ไม่เป็นคำสั่ง	-	-
กดปุ่ม “Save Promotion”	เป็นคำสั่ง ระบบทำการจัดเก็บ ข้อมูลPromotion	:CreatePromotionController	บันทึกการสร้างข้อมูลโปรโมชั่น (promolImage, title, game_id, description, discount_value)
ค้นหา entity Game ด้วย game_id ที่รับเข้ามา	เป็นคำสั่ง	g :game	ค้นหาด้วยไอเดีย (game_id)
สร้างข้อมูล entity Promotion โดยโยง entity User เป็นค่า rating, review_title,review_text	เป็นคำสั่ง	p :promotion	สร้าง (promolImage, title, description, discount_value, u)
บันทึก entity Promotion	เป็นคำสั่ง	p :Promotion	บันทึก ()
สร้างข้อมูล entity Promotion_game โดยโยง entity Game กับ entity Promotion	เป็นคำสั่ง เป็นตารางที่เกิดจาก ความสัมพันธ์ many-to-many	pg :Promotion_game	สร้าง (p,g)
บันทึก entity Promotion_game	เป็นคำสั่ง	Pg :Promotion_game	บันทึก ()
แสดงข้อความสร้าง “สร้างรีวิว สำเร็จ”	ไม่เป็นคำสั่ง	-	-



## Class Diagram at Design Level



B6641054 นายณัฐนันท์ จันทร์สุริยา

ระบบหลัก ระบบบริการขายเกมออนไลน์

ระบบย่อย ระบบยื่นคำร้องขอคืนเงิน

ระบบคืนเงินเป็นระบบที่ช่วยสร้างความมั่นใจให้ลูกค้าในการซื้อเกมจากร้านของเรา โดยลูกค้าสามารถส่งคำร้องขอคืนเงินได้หากพบปัญหาหลังการซื้อ เช่น คีย์เกมไม่สามารถใช้งานได้ เกมไม่ตรงตามที่แสดงไว้

หรือซื้อเกมผิดประเภท โดยลูกค้าสามารถเลือกเกมจากประวัติการซื้อและส่งคำร้องขอคืนเงินได้ทันที

ระบบจะให้ลูกค้าแนบเหตุผลการคืนเงินและหลักฐานประกอบคำร้อง ซึ่งข้อมูลจะถูกส่งไปยังแอดมินเพื่อตรวจสอบ

หากแอดมินอนุมัติ ระบบจะดำเนินการคืนเงินให้ตามช่องทางที่ลูกค้าใช้ชำระเงินเป็นช่องทางการคืนเงิน เช่น

บัตรเครดิต วอลเล็ต หรือโอนเข้าบัญชีธนาคาร นอกจากนี้ ระบบจะเก็บประวัติการคืนเงินไว้ในระบบ

เพื่อให้ง่ายต่อการตรวจสอบย้อนหลัง และใช้เป็นข้อมูลวิเคราะห์พัฒนาระบบ

User Story ระบบยื่นคำร้องคืนเงิน

ในบทบาทของ (As a) ลูกค้า (Customer)

ฉันต้องการ (I want to) ขอคืนเงินจากเกมที่ซื้อ

เพื่อ (So that) ได้รับเงินคืนในกรณีที่เกิดปัญหาหลังการซื้อ และสามารถมั่นใจในการใช้บริการร้านค้า

ในบทบาทของ (As a) ผู้ดูแลระบบ(Admin)

ฉันต้องการ (I want to) คืนเงินให้ลูกค้าแบบรวดเร็วและครบตามจำนวน

เพื่อ (So that) ให้ลูกค้าได้รับความเป็นธรรม และมั่นใจในการบริการของเรา

Output บนหน้าจอ

- ลูกค้าเข้าเมนู 'ประวัติการซื้อ' → เลือกรายการเกม → คลิก "ขอคืนเงิน"
- กรอกเหตุผลการคืนเงิน และแนบไฟล์หลักฐาน (ถ้ามี)
- ระบบแสดงสถานะ "รอตรวจสอบ" และแจ้งเตือนไปยังแอดมิน
- แอดมินตรวจสอบและกด "อนุมัติ" หรือ "ปฏิเสธ" คำร้อง
- หากอนุมัติ ระบบแสดงสถานะ "คืนเงินแล้ว" พร้อมวันที่และช่องทางที่คืนเงิน

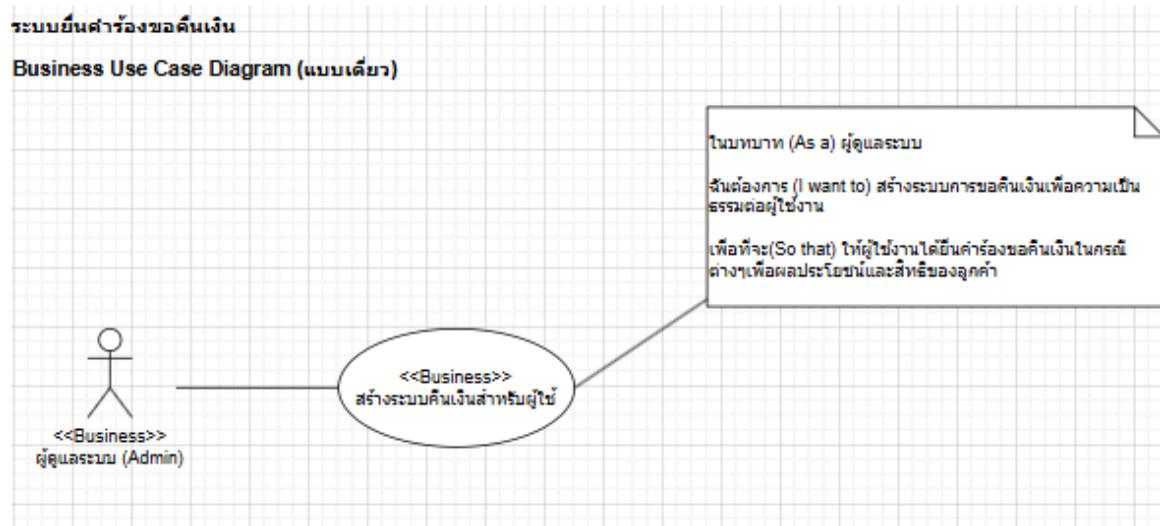
## Output ของข้อมูล

- บันทึกคำร้องพร้อมรายละเอียด เช่น เกม, เหตุผล, วันที่ร้องขอ
- บันทึกผลการอนุมัติ และช่องทางคืนเงิน
- สถานะการคืนเงินมี: รอตรวจสอบ, อนุมัติ, ปฏิเสธ, คืนเงินแล้ว
- เก็บประวัติคำร้องทั้งหมดไว้ในระบบฐานข้อมูลคำน้ำมันที่อาจกลายเป็นตรางในฐานข้อมูล

## คำนำที่อาจจะกล้ายมาเป็นตารางในฐานข้อมูลของระบบ

คำนำ	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
คำร้องคืนเงิน	เกี่ยวข้องโดยตรง ใช้จดเก็บคำร้องที่ลูกค้ายื่นเข้ามา
ลูกค้า	เกี่ยวข้องโดยตรง เป็นผู้ส่งคำร้อง
เกม	เกี่ยวข้องโดยตรง เป็นสินค้าที่ลูกค้าต้องการคืนเงิน
เหตุผลการคืนเงิน	เกี่ยวข้องโดยตรง เพื่ออธิบายสาเหตุในการขอคืนเงิน
สถานะคืนเงิน	เกี่ยวข้องโดยตรง สำหรับติดตามการดำเนินการ
ช่องทางการคืนเงิน	เกี่ยวข้องโดยตรง เพื่อบันทึกวิธีการคืนเงิน

## Business Use Case Diagram (แบบเดี่ยว)



### Checklist: Business Use Case Diagram Business

Actor มี  
-> กำกับ Business Actor เป็นชื่อบทบาทเท่านั้น ไม่ใช่ชื่อคน ไม่เป็นชื่อบุคคล

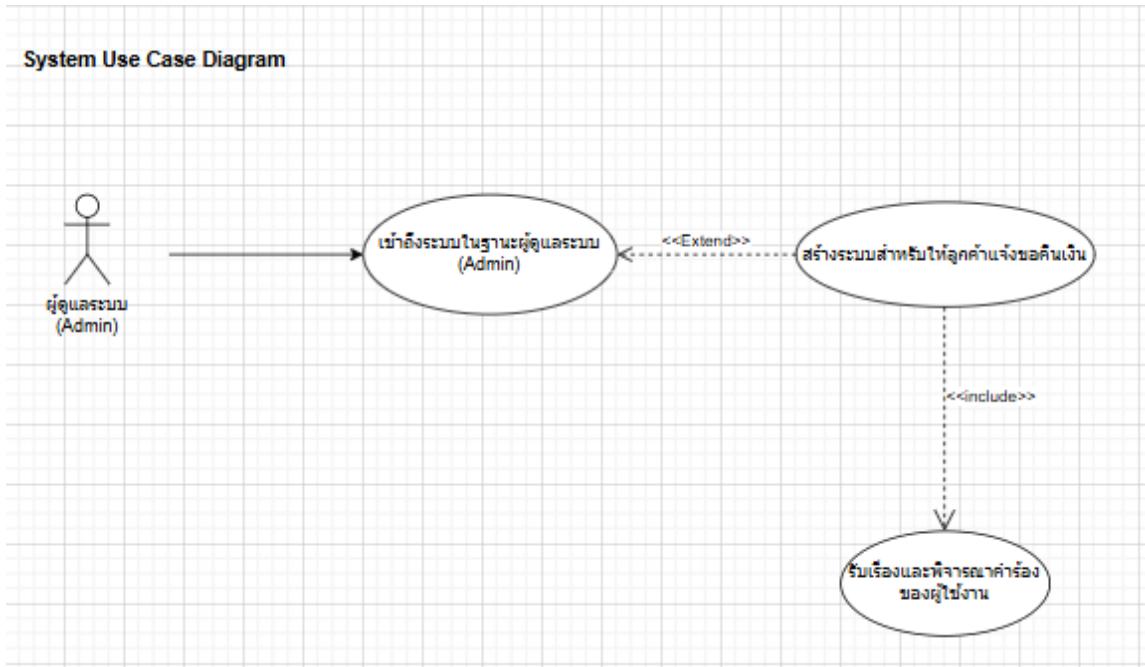
Business Use Case มี  
-> กำกับ

Business Use Case ขึ้นต้นด้วยคำที่แสดงพฤติกรรม “คำกริยา” พฤติกรรมของ

Business Use Case ต้องมาจาก User Story

มี Note ที่แสดง User Story อยู่ใน Diagram

## System Use Case Diagram (แบบเดี่ยว)



### ขั้นตอนการแปลง Business Use Case Diagram ให้เป็น System Use Case Diagram

#### พิจารณาประเด็นที่ 1

Business Actor "ผู้ดูแลระบบ (Admin)" สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์ที่ระบบงานได้โดยตรงหรือไม่  
ตอบ ได้ ผู้ดูแลระบบ สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์

เพราะฉะนั้น ผู้ดูแลระบบ (Admin) จะถูกต้องเป็น System Actor ได้

#### พิจารณาประเด็นที่ 2

Business Use Case “ระบบยื่นคำร้องขอคืนเงิน” สามารถถูกถ่ายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบ

คอมพิวเตอร์ได้หรือไม่ ต้องมีขั้นตอนทาง Security มาเกี่ยวหรือไม่

ตอบ Business Use Case “ระบบยื่นคำร้องขอคืนเงิน” สามารถถูกถ่ายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบ  
คอมพิวเตอร์ได้ และต้องการขั้นตอนทาง Security (มี Privacy มาเกี่ยวข้อง)

เพราะฉะนั้น Business Use Case “ระบบยื่นคำร้องขอคืนเงิน” จะถูกต้องเป็น System Use Case มากกว่า 1

## Use Case

จะประกอบไปด้วย

1. System Use Case สำหรับ “เข้าระบบในฐานะผู้ดูแลระบบ (Admin)”
2. System Use Case สำหรับ “ยื่นคำร้องขอคืนเงิน” แต่กิจกรรมของระบบ จะยังไม่ครบตาม Requirements ที่ต้องการ
3. System Use Case สำหรับ “บันทึกคำร้องขอคืนเงิน”
  - System Use Case สำหรับ “เข้าระบบในฐานะผู้ดูแลระบบ (Admin)”
  - System Use Case สำหรับ “สร้างระบบคำร้องขอคืนเงิน”
  - System Use Case สำหรับ “บันทึกข้อมูลคำร้องขอคืนเงิน”

## พิจารณาประเด็นที่ 3

ถ้าสมาชิก “เข้าระบบในฐานะผู้ดูแลระบบ (Admin)” มาแล้วจำเป็นที่ต้อง “ระบบยื่นคำร้องขอคืนเงิน” ทุกครั้ง หรือไม่

ตอบ ไม่

แปลว่า System Use Case “ระบบยื่นคำร้องขอคืนเงิน” เป็นกิจกรรมทางเลือกหลังจากทำ System Use Case “เข้าระบบในฐานะผู้ดูแลระบบ (Admin)”

## พิจารณาประเด็นที่ 4

ถ้าสมาชิก “เข้าระบบในฐานะผู้ดูแลระบบ” มาแล้วจำเป็นต้อง “บันทึกข้อมูลคำร้องขอคืนเงิน” ทุกครั้งหรือไม่

ตอบ ไม่ และเป็น Use Case ที่ไม่เกี่ยวข้องกัน

## พิจารณาประเด็นที่ 5

ถ้าสมาชิก “ยื่นคำร้องขอคืนเงิน” และ

จำเป็นต้อง “บันทึกคำร้องขอคืนเงิน” ทุกครั้งหรือไม่

ตอบ ใช่ จำเป็น

แปลว่า System Use Case “สร้างสิทธิการเข้าถึงให้ผู้ใช้” จะต้องรวมขั้นตอนของ System Use Case

“บันทึกคำร้องขอคืนเงิน” ไว้ด้วยเสมอ

### Checklist: System Use Case Diagram

1. System Actor และ System Use Case ต้องไม่มีอะไรกำกับ
2. เส้นโยงจาก System Actor ไปหา System Use Case จะต้องเป็นเส้นทึบ หัวลูกศรปลายเปิดซึ้งไปหา System Use Case
3. System Actor ต้องเป็นบทบาทของคนที่ใช้ระบบจริง ๆ
4. System Use Case ต้องเป็นคำที่แสดงพฤติกรรม
5. ถ้ามีกลไกทาง Security มาเกี่ยว ชื่อ System Use Case จะอยู่ในรูปแบบ “เข้าระบบในฐานะ <Actor>” เท่านั้น
6. การใช้ <--<<extend>>---> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ้งจาก System Use Case ทางเลือกไปยัง System Use Case หลัก
7. การใช้ <--<<include>>---> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ้งจาก System Use Case ที่ใหญ่กว่า ไปยัง System Use Case ที่จะถูกรวบเข้ามา

## วิเคราะห์ Entity ที่เกี่ยวข้อง

Entity: Notification

ข้อมูลที่ต้องจัดเก็บ

ID	PK
UserID	FK → User
Type	VARCHAR; 'report_created' 'report_reply' 'report_resolved' ...
TargetType	VARCHAR; 'report' 'reply'
TargetID	UINT
Message	TEXT
IsRead	BOOLEAN DEFAULT FALSE
CreatedAt	TIMESTAMP

NotiID	Title	Type	Message	UserID
3001	Report Game	System	แก้ไขปัญหาของ คุณเรียบร้อย	1001
3002	Payment Done	Payment	ชำระเงินสำเร็จ	1002

Entity: RefundRequest

ข้อมูลที่ต้องจัดเก็บ

- - RefundID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- - UserID: integer, Foreign Key, ไม่เป็น Null
- - OrderID: integer, Foreign Key, Null ได้
- - Reason: varchar, ไม่เป็น Null
- - RequestDate: datetime, ไม่เป็น Null
- - ProcessedDate: datetime, Null ได้
- - Amount: decimal, Null ได้
- - StatusID: integer, Foreign Key, ไม่เป็น Null

Report ID	User ID	Game ID	Order ID	Title	Description	Created At	Resolved At	Status ID
4001	1001	2001	None	เกมเปิดไม่ติด	เกมค้างตอนโหลด	2025-09-01 10:00:00	2025-09-02 15:00:00	1
4002	1002	2002	None	การชำระเงินล้มเหลว	ระบบไม่ตัดบัตรเครดิต	2025-09-03 09:30:00	None	2

### Entity: Refund\_Status

ข้อมูลที่ต้องจัดเก็บ

- ProblemStatusID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- StatusName: varchar, ไม่เป็น Null

ProblemStatusID	StatusName
1	Resolved
2	In Progress
3	Pending

### Entity: RefundAttachment

ข้อมูลที่ต้องจัดเก็บ

- AttachmentID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- ReportID: integer, Foreign Key, ไม่เป็น Null
- FilePath: varchar, ไม่เป็น Null
- UploadedAt: datetime, ไม่เป็น Null

AttachmentID	ReportID	FilePath	UploadedAt
6001	4001	/uploads/error1.png	2025-09-01 10:05:00
6002	4002	/uploads/pay1.png	2025-09-03 09:35:00

## หลักการออกแบบ User Interface

- เมื่อได้กีตามที่มี Foreign Key (FK) ซึ่งจากตารางหลักกลับไปยัง ตารางสนับสนุน  
ให้ใช้ ComboBox เป็นตัวเลือกเพื่อเชื่อมโยงข้อมูลใน User Interface
- สำหรับ Field ประเภทอื่นๆ ให้สร้างตามประเภทของข้อมูล เช่น
  - Textbox สำหรับข้อความทั่วไป
  - Password สำหรับข้อมูลรหัสผ่าน
  - Datetime Picker สำหรับเลือกวันและเวลา
  - Numeric Input สำหรับป้อนตัวเลข

## UI Design

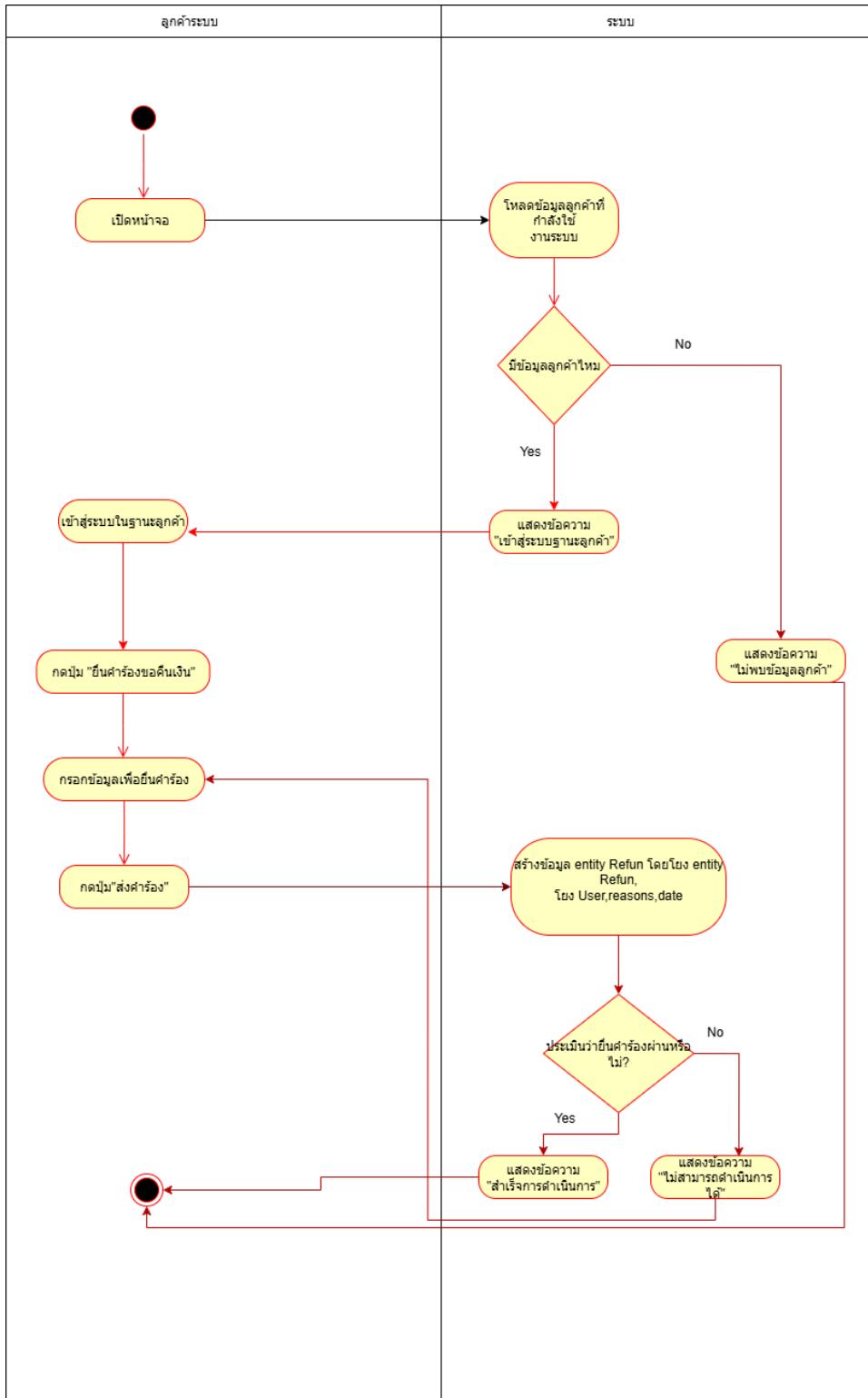
The wireframe shows a user interface for a game store system. At the top left is a game controller icon, and at the top right is a user profile icon. The title 'ระบบร้านขายเกมออนไลน์' is centered at the top.

The main area contains several input fields:

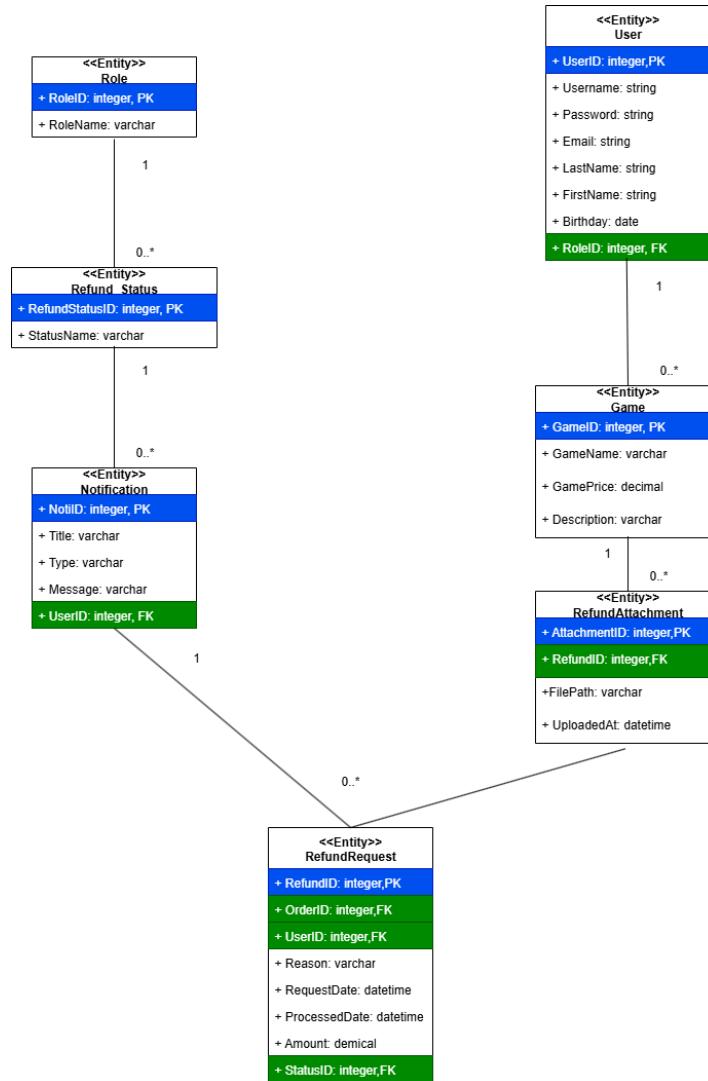
- ข้อมูลลูกค้า:**
  - ชื่อเกม (Text input)
  - รหัสผู้ใช้งาน (Text input)
  - วันที่ซื้อ (Text input)
  - ราคารายจ่าย (Text input)
- เดบิตออนไลน์:**
  - เลือกเดบิต (Text input)
  - อัตราเพิ่มเติม (Text input)
- แนบไฟล์รูป:** A large rectangular input field for file uploads.
- แนบไฟล์ฐาน:**
  - choose file (button)
  - No file (text)
- เงื่อนไข:** A checkbox with the text 'ข้าพเจตขอรับผิดชอบว่าจะพิจารณาภายใน 7 วันทำการ'.

At the bottom are two buttons: 'ยกเลิก' (Cancel) and 'สมัครสมาชิก' (Sign Up).

## Activity Diagram



## Class Diagram At analysis Level



## Frontend

### ระบบยื่นคำร้องขอคืนเงิน

```
import React, { useState } from "react";
import {
  Card,
  Form,
  Input,
  Button,
  Select,
  DatePicker,
  Typography,
  Upload,
  Modal,
  message,
} from "antd";
import { UploadOutlined } from "@ant-design/icons";
import Navbar from "../components/Navbar";

const { Title } = Typography;
```

```
export default function RefundPage() {
  const [form] = Form.useForm();
  const [fileList, setFileList] = useState<any[]>([]);
  const [previewVisible, setPreviewVisible] = useState(false);
  const [previewImage, setPreviewImage] = useState("");
  const [previewTitle, setPreviewTitle] = useState("");

  const handlePreview = async (file: any) => {
    setPreviewImage(file.url || file.thumbUrl);
    setPreviewVisible(true);
    setPreviewTitle(
      file.name || file.url!.substring(file.url!.lastIndexOf("/") + 1)
    );
  };

  const handleChange = ({ fileList: newList }: any) =>
    setFileList(newList);

  const handleRemove = (file: any) => {
    setFileList((prev) => prev.filter((f) => f.uid !== file.uid));
  };

  const handleSubmit = async (values: any) => {
    try {
      console.log("Refund data (mock):", {
        ...values,
        purchaseDate: values.purchaseDate.format("YYYY-MM-DD"),
        files: fileList.map((f) => f.name),
      });
    } catch (err) {
      console.error(err);
    }
  };
}
```

```
await new Promise((resolve) => setTimeout(resolve, 800));

message.success("ส่งคำร้องคืนเงินเรียบร้อย! (mock)");
form.resetFields();
setFileList([]);

} catch (error: any) {
  console.error("Error submitting refund (mock):", error);
  message.error("เกิดข้อผิดพลาดในการส่งคำร้อง (mock)");
}

};

return (
<div style={{ background: "#141414", minHeight: "100%", color: "#fbfbfbff" }}>
  /* เพิ่ม style ให้ label ของฟอร์มเป็นสีขาว */
<style>
  {
    .ant-form-item-label > label {
      color: white !important;
    }
  }
</style>

<Navbar />

<Card
  style={{
    backgroundColor: "#1f1f1f",
    borderRadius: 12,
    color: "white",
  }
}
```

```
        maxWidth: 600,  
        margin: "40px auto",  
        boxShadow: "0 0 20px rgba(146, 84, 222, 0.3)",  
    }  
    bordered={false}  
>  
<Title level={3} style={{ color: "#f759ab", marginBottom: 24 }}>  
    Refund Request  
</Title>  
  
<Form form={form} layout="vertical" onFinish={handleSubmit}>  
    <Form.Item  
        label="Order ID"  
        name="orderId"  
        rules={[{ required: true, message: "กรุณากรอก Order ID" }]}  
>  
    <Input  
        placeholder="e.g., #621da7ff"  
        style={{ background: "#ffffff", color: "black" }}  
    />  
    </Form.Item>  
  
    <Form.Item  
        label="Reason for Refund"  
        name="reason"  
        rules={[{ required: true, message: "กรุณาเลือกเหตุผล" }]}  
>  
    <Select placeholder="Select reason">  
        <Select.Option value="defective">Defective Product</Select.Option>
```

```

<Select.Option value="incorrect">Incorrect Item Received</Select.Option>
    <Select.Option value="late">Item Arrived Late</Select.Option>
    <Select.Option value="not_described">Item Not as Described</Select.Option>
        <Select.Option value="duplicate">Duplicate Order</Select.Option>
        <Select.Option value="accidental">Accidental Purchase</Select.Option>
        <Select.Option value="billing">Billing Issue</Select.Option>
        <Select.Option value="not_received">Did Not Receive Item</Select.Option>
            <Select.Option value="wrong_version">Wrong Platform/Version</Select.Option>
                <Select.Option value="other">Other</Select.Option>
            </Select>
        </Form.Item>

        <Form.Item
            label="Purchase Date"
            name="purchaseDate"
            rules={[{ required: true, message: "กรุณาเลือกวันที่ซื้อ" }]}
        >
            <DatePicker style={{ width: "100%", background: "#ffffff", color: "black" }} />
        </Form.Item>

        <Form.Item
            label="Bank"
            name="bank"
            rules={[{ required: true, message: "กรุณาเลือกธนาคาร" }]}
        >

```

```

<Select placeholder="Select bank">
    <Select.Option value="kbank">Kasikorn Bank (KBANK)</Select.Option>
    <Select.Option value="scb">Siam Commercial Bank
    (SCB)</Select.Option>
    <Select.Option value="bbl">Bangkok Bank (BBL)</Select.Option>
    <Select.Option value="ktb">Krungthai Bank (KTB)</Select.Option>
    <Select.Option value="tmb">TMBThanachart Bank (TTB)</Select.Option>
    <Select.Option value="gsb">Government Savings Bank
    (GSB)</Select.Option>
    <Select.Option value="bay">Krungsri Bank (BAY)</Select.Option>
    <Select.Option value="uob">UOB</Select.Option>
    <Select.Option value="cimb">CIMB Thai</Select.Option>
    <Select.Option value="other">Other</Select.Option>
</Select>
</Form.Item>

<Form.Item
    label="Account Number"
    name="accountNumber"
    rules={[
        { required: true, message: "กรุณากรอกเลขบัญชี" },
        { pattern: /^[0-9]{10,16}$/, message: "เลขบัญชีต้อง 10-16 ตัวเลข" },
    ]}
>
<Input
    placeholder="Enter your account number"
    style={{ background: "#ffffff", color: "black" }}
/>
</Form.Item>

```

```
<Form.Item label="Additional Comments" name="comments">
  <Input.TextArea
    rows={3}
    placeholder="Provide any additional details"
    style={{ background: "#ffffff", color: "black" }}
  />
</Form.Item>

<Form.Item label="Upload Proof" required>
  <Upload
    listType="picture-card"
    fileList={fileList}
    onPreview={handlePreview}
    onChange={handleChange}
    onRemove={handleRemove}
    beforeUpload={(file) => {
      const isImage = file.type.startsWith("image/");
      if (!isImage) {
        message.error("You can only upload image files!");
        return Upload.LIST_IGNORE;
      }
      return false;
    }}
  >
  {fileList.length < 3 && (
    <div>
      <UploadOutlined />
      <div style={{ marginTop: 8 }}>Upload</div>
    </div>
  )}
</Form.Item>
```

```
</Upload>

<Modal
  open={previewVisible}
  title={previewTitle}
  footer={null}
  onCancel={() => setPreviewVisible(false)}
>
  <img alt="example" style={{ width: "100%" }} src={previewImage} />
</Modal>
</Form.Item>

<Form.Item>
  <Button
    type="primary"
    htmlType="submit"
    style={{
      background: "linear-gradient(90deg, #9254de 0%, #f759ab 100%)",
      border: "none",
      color: "white",
      width: "100%",
    }}
  >
    Submit Request
  </Button>
</Form.Item>
</Form>
</Card>
</div>
);
```

}

## BACKEND

Entity:Refund

User

```
package models

import "time"

type User struct {
    UserID     uint      `gorm:"primaryKey" json:"user_id"`
    Username   string    `json:"username"`
    Password   string    `json:"password"`
    Email      string    `json:"email"`
    LastName   string    `json:"last_name"`
    FirstName  string    `json:"first_name"`
    Birthday   time.Time `json:"birthday"`
    RoleID     uint      `json:"role_id"`

    Role        Role      `gorm:"foreignKey:RoleID"`
    Notifications []Notification `gorm:"foreignKey:UserID"`
}
```

## Role

```
package models

type Role struct {
    RoleID  uint  `gorm:"primaryKey" json:"role_id"`
    RoleName string `json:"role_name"`

    Users []User `gorm:"foreignKey:RoleID"`
}
```

## Refund\_Status

```
package models

type RefundStatus struct {
    RefundStatusID uint  `gorm:"primaryKey" json:"refund_status_id"`
    StatusName     string `json:"status_name"`

}
```

## Refund\_request

```
package models

import "gorm.io/gorm"
```

```

type RefundRequest struct {
    RefundID     uint      `gorm:"primaryKey" json:"refund_id"`
    OrderID      uint      `json:"order_id"`
    UserID       uint      `json:"user_id"`
    Reason       string    `json:"reason"`
    RequestDate  time.Time `json:"request_date"`
    ProcessedDate time.Time `json:"processed_date"`
    Amount        float64   `json:"amount"`
    StatusID     uint      `json:"status_id"`

    Attachments []RefundAttachment `gorm:"foreignKey:RefundID"`
}


```

## Refund\_attachment

```

package models

import "time"

type RefundAttachment struct {
    AttachmentID uint      `gorm:"primaryKey" json:"attachment_id"`
    RefundID     uint      `json:"refund_id"`
    FilePath     string    `json:"file_path"`
    UploadedAt   time.Time `json:"uploaded_at"`
}

```

## Notification

```
package models

type Notification struct {
    NotiID uint `gorm:"primaryKey" json:"noti_id"`
    Title string `json:"title"`
    Type string `json:"type"`
    Message string `json:"message"`
    UserID uint `json:"user_id"`

}
```

## Game

```
package entity

type Game struct {
    GameID uint `gorm:"primaryKey" json:"game_id"`
    GameName string `json:"game_name"`
    GamePrice float64 `json:"game_price"`
    Description string `json:"description"`

}
```

## Controller

### User\_controller

```
package controllers

import (
    "net/http"
    "sa-gameShop-backend/models"

    "github.com/gin-gonic/gin"
    "gorm.io/gorm"
)

// Get Users
func GetUsers(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var users []models.User
        if err := db.Preload("Role").Find(&users).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
            return
        }
        c.JSON(http.StatusOK, users)
    }
}

// Get User by ID
func GetUserByID(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var user models.User
        if err := db.Preload("Role").First(&user, c.Param("id")).Error; err != nil {
```

```
c.JSON(http.StatusNotFound, gin.H{"error": "User not found"})

return
}

c.JSON(http.StatusOK, user)
}

// Create User

func CreateUser(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var user models.User

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

        if err := db.Create(&user).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
            return
        }

        c.JSON(http.StatusCreated, user)
    }
}

// Update User

func UpdateUser(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var user models.User

        if err := db.First(&user, c.Param("id")).Error; err != nil {
            c.JSON(http.StatusNotFound, gin.H{"error": "User not found"})
            return
        }
```

```
        }

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

        db.Save(&user)
        c.JSON(http.StatusOK, user)
    }
}

// Delete User
func DeleteUser(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        if err := db.Delete(&models.User{}, c.Param("id")).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
            return
        }

        c.JSON(http.StatusOK, gin.H{"message": "User deleted"})
    }
}
```

## Notification\_controller

```
package controllers

import (
    "net/http"
    "sa-gameShop-backend/models"

    "github.com/gin-gonic/gin"
    "gorm.io/gorm"
)

func GetNotifications(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var notifications []models.Notification
        if err := db.Find(&notifications).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
            return
        }
        c.JSON(http.StatusOK, notifications)
    }
}

func CreateNotification(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var noti models.Notification
        if err := c.ShouldBindJSON(&noti); err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
            return
        }
        if err := db.Create(&noti).Error; err != nil {
```

```
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}
c.JSON(http.StatusCreated, noti)
}
}
```

## Refund\_controller

```
package controllers

import (
    "net/http"
    "sa-gameShop-backend/models"
    "time"

    "github.com/gin-gonic/gin"
    "gorm.io/gorm"
)

func GetRefunds(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var refunds []models.RefundRequest
        if err := db.Preload("Attachments").Find(&refunds).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
            return
        }
        c.JSON(http.StatusOK, refunds)
    }
}
```

```
}

}

func GetRefundByID(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var refund models.RefundRequest
        if err := db.Preload("Attachments").First(&refund, c.Param("id")).Error; err != nil {
            c.JSON(http.StatusNotFound, gin.H{"error": "Refund not found"})
            return
        }
        c.JSON(http.StatusOK, refund)
    }
}

func CreateRefund(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var refund models.RefundRequest
        if err := c.ShouldBindJSON(&refund); err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
            return
        }
        refund.RequestDate = time.Now()
        refund.StatusID = 1 // Pending
        if err := db.Create(&refund).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
            return
        }
        c.JSON(http.StatusCreated, refund)
    }
}
```

```
func UpdateRefundStatus(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var refund models.RefundRequest
        if err := db.First(&refund, c.Param("id")).Error; err != nil {
            c.JSON(http.StatusNotFound, gin.H{"error": "Refund not found"})
            return
        }
        type Request struct {
            StatusID uint `json:"status_id"`
        }
        var req Request
        if err := c.ShouldBindJSON(&req); err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
            return
        }
        refund.StatusID = req.StatusID
        refund.ProcessedDate = time.Now()
        db.Save(&refund)
        c.JSON(http.StatusOK, refund)
    }
}
```

## Status\_controller

```
package controllers

import (
    "net/http"
    "sa-gameShop-backend/models"

    "github.com/gin-gonic/gin"
    "gorm.io/gorm"
)

func GetRefundStatuses(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var statuses []models.RefundStatus
        if err := db.Find(&statuses).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
            return
        }
        c.JSON(http.StatusOK, statuses)
    }
}

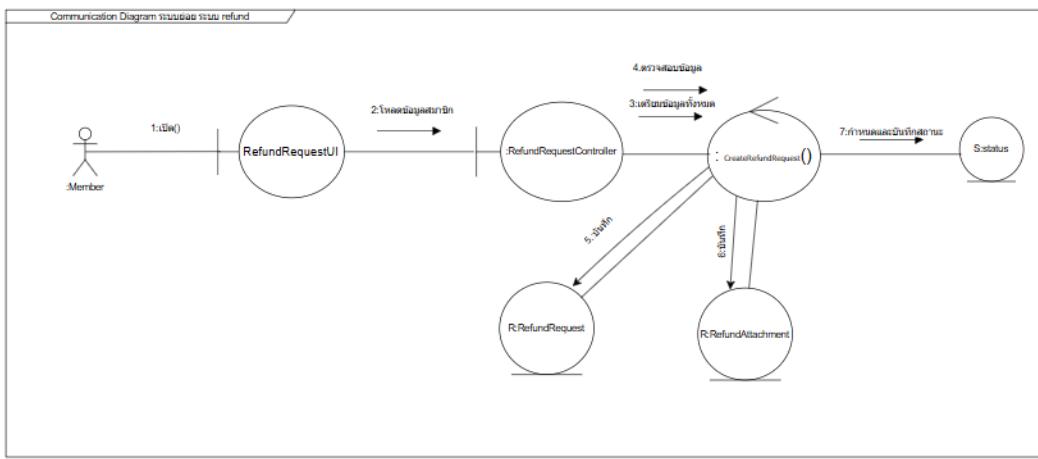
func GetProblemStatuses(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var statuses []models.ProblemStatus
        if err := db.Find(&statuses).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
            return
        }
        c.JSON(http.StatusOK, statuses)
    }
}
```

```
    }  
}  
  
}
```

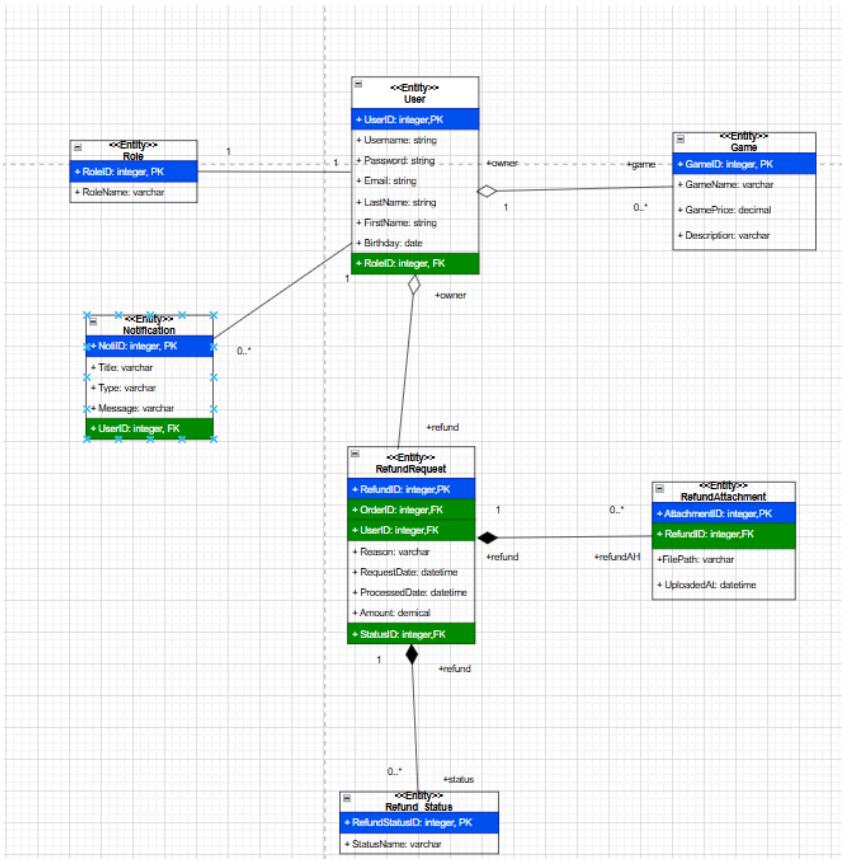
### 1) Communication Diagram: Refund Request

Activity	เป็นคำสั่ง สำหรับ ระบบหรือไม่	ถ้าเป็น, วัตถุที่ได้รับหน้าที่ ทำงานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click “เปิดหน้าจอ Refund Request”	เป็นคำสั่ง สั่งงานให้ หน้า UI เปิด	:RefundRequestUI	เปิด():
โหลดข้อมูลผู้ใช้ที่ login (UserID)	เป็นคำสั่ง เกิดการ สั่งงานให้ โหลดข้อมูล สมาชิก login อยู่	:RefundRequestController	โหลดข้อมูลสมาชิก (id)
ตรวจสอบสิทธิ์การยื่นคำร้อง (UserID, OrderID)	เป็นคำสั่ง	:RefundRequestController	เตรียมข้อมูลและดึงข้อมูล ทั้งหมด
ตรวจสอบเกมที่เกี่ยวข้อง (OrderID → Game)	เป็นคำสั่ง	:RefundRequestController	เตรียมไฟล์ในการส่งข้อมูล()
แสดงหน้าจอกรอกข้อมูล (Reason, Amount, Attachments)	ไม่เป็นคำสั่ง		
กรอกข้อมูล + กดปุ่ม “ยืนยันคำร้อง Refund”	เป็นคำสั่ง ระบบทำการ จัดเก็บข้อมูล	:RefundRequestController	:RefundRequestUI → :RefundRequestController

สร้าง RefundRequest (OrderID, UserID, Reason, Amount, RequestDate)	เป็นคำสั่ง	สร้าง RefundRequest (OrderID, UserID, Reason, Amount, RequestDate)	: CreateRefundRequest()
บันทึกข้อมูล entity	เป็นคำสั่ง	R: RefundRequest	บันทึก()
แนบไฟล์หลักฐาน (FilePath, UploadedAt)	ไม่เป็นคำสั่ง	แนบไฟล์หลักฐาน (FilePath, UploadedAt)	
บันทึกข้อมูล entity	เป็นคำสั่ง	: RefundAttachment	บันทึก()
กำหนดสถานะเริ่มต้น “Pending”	เป็นคำสั่ง	S:status	กำหนดสถานะ(s_status)
อัปเดต RefundRequest.StatusID	ไม่เป็นคำสั่ง	-	-
แสดงข้อความ “ส่งคำร้องเรียบร้อย”	ไม่เป็นคำสั่ง	-	



## Communication Class Diagram



B6641054 นายณัฐนันท์ จันทร์สุริยา

ระบบหลัก ระบบบร้านขายเกมออนไลน์

ระบบย่อย ระบบแจ้งรายงานปัญหา

ระบบรายงานปัญหา เป็นพื้นที่สำหรับผู้ใช้งานที่สามารถแจ้งปัญหาที่พบจากการใช้งาน เช่น เกมไม่สามารถดาวน์โหลดได้ คีย์เกมไม่ถูกต้อง เว็บไซต์ล่ม หรือชำระเงินไม่สำเร็จ ลูกค้าสามารถกรอกฟอร์มรายงานปัญหาผ่านระบบ โดยระบุหัวข้อ รายละเอียดปัญหา และแนบภาพหน้าจอประกอบ เป็นไฟล์แนบ หลังจากส่งคำร้องแล้ว ระบบจะแจ้งเตือนทีมซัพพอร์ตหรือแอดมินเพื่อดำเนินการตรวจสอบ เมื่อมีความคืบหน้า ระบบจะแจ้งลูกค้าผ่านทางอีเมลหรือระบบภายในเว็บ เช่น แสดงสถานะคำร้องว่า 'กำลังตรวจสอบ', 'ดำเนินการแล้ว', 'แก้ไขแล้ว' ระบบนี้ช่วยให้สามารถติดตามและจัดการปัญหาได้อย่างเป็นระบบ และยังสร้างความมั่นใจให้กับลูกค้าอีกด้วย

User Story ระบบรายงานปัญหา

ในบทบาทของ (As a) ลูกค้า (Customer)

ฉันต้องการ (I want to) แจ้งปัญหาที่เกิดขึ้นระหว่างใช้งาน

เพื่อ (So that) ทีมงานจะได้ทราบและดำเนินการแก้ไขได้อย่างรวดเร็วและมีประสิทธิภาพ

ในบทบาทของ (As a) ผู้ดูแลระบบ (Admin)

ฉันต้องการ (I want to) แก้ไขปัญหาให้ลูกค้าแบบรวดเร็วและสามารถแก้ปัญหาโดยไม่ให้เกิดปัญหานั้นขึ้นอีก

เพื่อ (So that) ให้ลูกค้าได้รับการบริการที่สะดวกและราบรื่น และเกิดปัญหาน้อยที่สุด

## Output บนหน้าจอ

1. ลูกค้าเข้าเมนู “แจ้งปัญหา” → กรอกหัวข้อและรายละเอียดปัญหา
2. แนบภาพหลักฐาน (ถ้ามี) → กด “ส่งคำร้อง”
3. ระบบแสดงข้อความยืนยันพร้อมหมายเลข Ticket ID
4. แอดมินเข้าตรวจสอบ → ตอบกลับพร้อมเปลี่ยนสถานะคำร้อง
5. ลูกค้าสามารถดูสถานะล่าสุด และย้อนดูประวัติได้จากหน้า 'รายการปัญหาของฉัน'

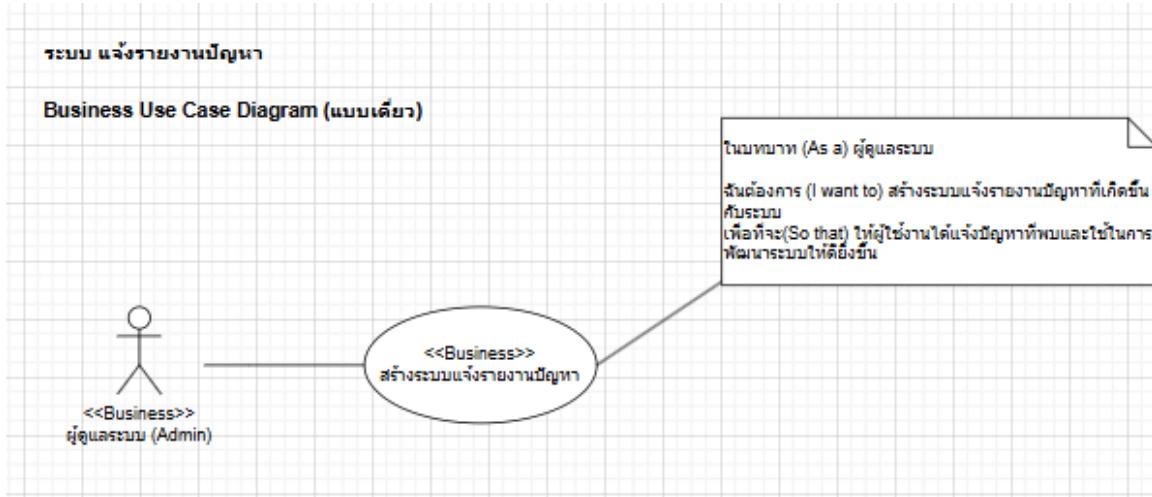
## Output ของข้อมูล

- ระบบเก็บรายละเอียดคำร้องปัญหา เช่น หมวดหมู่, วันที่, รายละเอียด, ไฟล์แนบ
- ระบบบันทึกสถานะของคำร้อง และเวลาอัปเดตล่าสุด
- ระบบแสดง Ticket ID สำหรับติดตาม
- สามารถดูประวัติการสื่อสารระหว่างลูกค้าและแอดมินได้ภายในคำร้องเดียวกัน

## คำนำที่อาจจะกล้ายมาเป็นตารางในฐานข้อมูลของระบบ

คำนำ	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
รายงานปัญหา	เกี่ยวข้องโดยตรง ใช้เก็บคำร้องจากลูกค้า
รายละเอียดปัญหา	เกี่ยวข้องโดยตรง สำหรับการวิเคราะห์และแก้ไขปัญหา
ลูกค้า	เกี่ยวข้องโดยตรง เป็นผู้แจ้งปัญหา
สถานะคำร้อง	เกี่ยวข้องโดยตรง เพื่อแสดงความคืบหน้าการดำเนินการ
แอดมิน	เกี่ยวข้องโดยตรง เป็นผู้ตรวจสอบและตอบกลับคำร้อง
ไฟล์แนบ	เกี่ยวข้องโดยตรง ใช้เป็นหลักฐานในการพิจารณาแก้ปัญหา

## Business Use Case Diagram (แบบเดี่ยว)



### Checklist: Business Use Case Diagram Business

Actor มี  
> กำกับ Business Actor เป็นชื่อบทบาทเท่านั้น ไม่ใช่ชื่อคน ไม่เป็นชื่อบุคคล

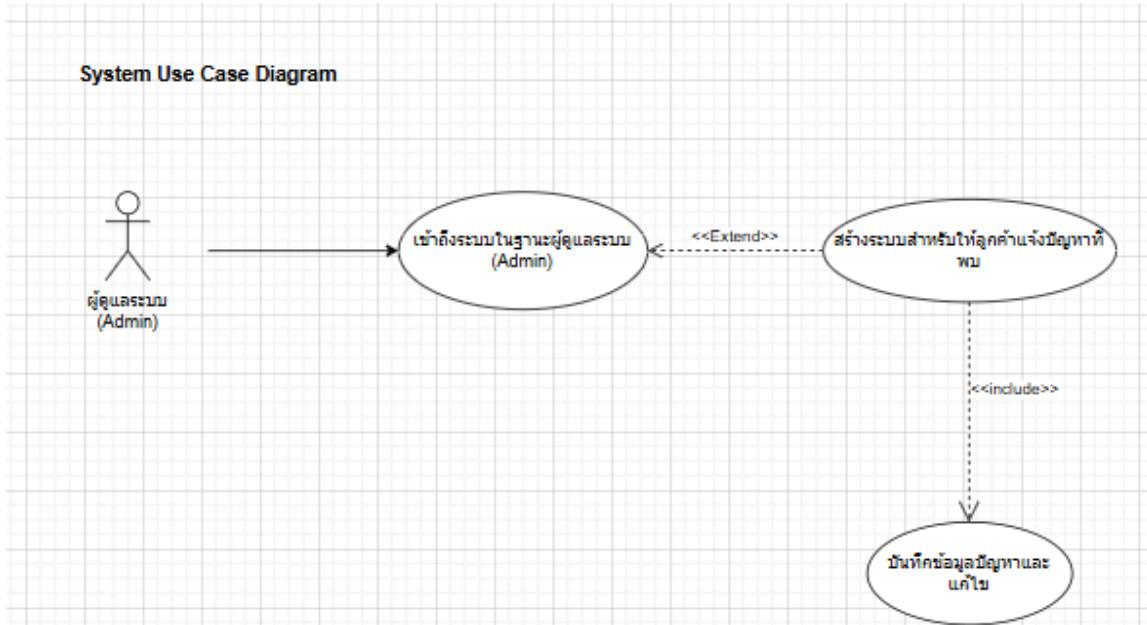
Business Use Case มี  
<> กำกับ

Business Use Case ขึ้นต้นด้วยคำที่แสดงพฤติกรรม “คำกริยา” พฤติกรรมของ

Business Use Case ต้องมาจาก User Story

มี Note ที่แสดง User Story อยู่ใน Diagram

## System Use Case Diagram (แบบเดี่ยว)



## ขั้นตอนการแปลง Business Use Case Diagram ให้เป็น System Use Case Diagram

### พิจารณาประเด็นที่ 1

Business Actor "ผู้ดูแลระบบ (Admin)" สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์ที่ระบบงานได้โดยตรงหรือไม่  
ตอบ ได้ ผู้ดูแลระบบ สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์

เพราะฉะนั้น ผู้ดูแลระบบ (Admin) จะถูกจัดเป็น System Actor ได้

### พิจารณาประเด็นที่ 2

Business Use Case “ระบบแจ้งรายงานปัญหา” สามารถถูกลายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบ  
คอมพิวเตอร์ได้หรือไม่ ต้องมีขั้นตอนทาง Security มาเกี่ยวหรือไม่

ตอบ Business Use Case “ระบบแจ้งรายงานปัญหา” สามารถถูกลายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบ  
คอมพิวเตอร์ได้ และต้องการขั้นตอนทาง Security (มี Privacy มาเกี่ยวข้อง)

เพราะฉะนั้น Business Use Case “ระบบแจ้งรายงานปัญหา” จะถูกจัดเป็น System Use Case มากกว่า 1

Use Case

จะประกอบไปด้วย

4. System Use Case สำหรับ “เข้าระบบในฐานะผู้ดูแลระบบ (Admin)”
5. System Use Case สำหรับ “ระบบแจ้งรายงานปัญหา” แต่กิจกรรมของระบบ จะยังไม่ครบตาม Requirements ที่ต้องการ
6. System Use Case สำหรับ “บันทึกคำร้องรายงานปัญหา”
  - System Use Case สำหรับ “เข้าระบบในฐานะผู้ดูแลระบบ (Admin)”
  - System Use Case สำหรับ “สร้างระบบรายงานปัญหา”
  - System Use Case สำหรับ “บันทึกข้อมูลการรายงานปัญหา”

### พิจารณาประเด็นที่ 3

ถ้าสมาชิก “เข้าระบบในฐานะผู้ดูแลระบบ (Admin)” มาแล้วจำเป็นที่ต้อง “ระบบแจ้งรายงานปัญหา” ทุกครั้ง หรือไม่

ตอบ ไม่

แปลว่า System Use Case “ระบบแจ้งรายงานปัญหา” เป็นกิจกรรมทางเลือกหลังจากทำ System Use Case “เข้าระบบในฐานะผู้ดูแลระบบ (Admin)”

### พิจารณาประเด็นที่ 4

ถ้าสมาชิก “เข้าระบบในฐานะผู้ดูแลระบบ” มาแล้วจำเป็นต้อง “บันทึกข้อมูลรายงานปัญหา” ทุกครั้งหรือไม่

ตอบ ไม่ และเป็น Use Case ที่ไม่เกี่ยวข้องกัน

### พิจารณาประเด็นที่ 5

ถ้าสมาชิก “ส่งรายงานปัญหา” แล้ว

จำเป็นต้อง “บันทึกรายงานปัญหา” ทุกครั้งหรือไม่

ตอบ ใช่ จำเป็น

แปลว่า System Use Case “สร้างสิทธิการเข้าถึงให้ผู้ใช้” จะต้องรวมขั้นตอนของ System Use Case

“บันทึกการรายงานปัญหา” ไว้ด้วยเสมอ

### Checklist: System Use Case Diagram

8. System Actor และ System Use Case ต้องไม่มีอะไรรากกับ
9. สេនយោគ System Actor ไปหา System Use Case จะต้องเป็นสេនីប៊ូ អ្នកគ្របាលយកចិត្តិភាព
- System Use Case
10. System Actor ต้องเป็นบทบาทของคนที่ใช้ระบบจริง ๆ
11. System Use Case ต้องเป็นคำที่แสดงพฤติกรรม

12. ถ้ามีกลไกทาง Security มาเกี่ยว ชื่อ System Use Case จะอยู่ในรูปแบบ “เข้าระบบในฐานะ **<Actor>**” เท่านั้น
13. การใช้ **<--<<extend>>-->** ต้องเป็นสันประ หัวลูกศรปลายเปิด ชี้จาก System Use Case ทางเดือกไปยัง System Use Case หลัก
14. การใช้ **<--<<include>>-->** ต้องเป็นสันประ หัวลูกศรปลายเปิด ชี้จาก System Use Case ที่ใหญ่กว่า ไปยัง System Use Case ที่จะถูกรวบเข้ามา

## Entity: Notification

ข้อมูลที่ต้องจัดเก็บ

- - NotiID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- - Title: varchar, ไม่เป็น Null
- - Type: varchar, ไม่เป็น Null
- - Message: varchar, ไม่เป็น Null
- - UserID: integer, Foreign Key, ไม่เป็น Null

NotiID	Title	Type	Message	UserID
3001	Update Game	System	เกม RPG Quest อัปเดต	1001
3002	Payment Done	Payment	ชำระเงินสำเร็จ	1002

## Entity: ProblemReport

ข้อมูลที่ต้องจัดเก็บ

- - ReportID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- - UserID: integer, Foreign Key, ไม่เป็น Null
- - GameID: integer, Foreign Key, Null ได้
- - OrderID: integer, Foreign Key, Null ได้
- - Title: varchar, ไม่เป็น Null
- - Description: varchar, ไม่เป็น Null
- - CreatedAt: datetime, ไม่เป็น Null
- - ResolvedAt: datetime, Null ได้
- - StatusID: integer, Foreign Key, ไม่เป็น Null

Report ID	User ID	Game ID	Order ID	Title	Description	Created At	Resolved At	Status ID
4001	1001	2001	None	เกมเปิดไม่ติด	เกมค้างตอนโหลด	2025-09-01 10:00:00	2025-09-02 15:00:00	1
4002	1002	2002	None	การชำระเงินล้มเหลว	ระบบไม่ตัดบัตรเครดิต	2025-09-03 09:30:00	None	2

Entity: Problem\_Status

ข้อมูลที่ต้องจัดเก็บ

- ProblemStatusID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- StatusName: varchar, ไม่เป็น Null

ProblemStatusID	StatusName
1	Resolved
2	In Progress
3	Pending

### Entity: report\_reply

ข้อมูลที่ต้องจัดเก็บ

- - category\_id: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- - category\_name: varchar, ไม่เป็น Null
- - issueReport\_id: integer, Foreign Key, ไม่เป็น Null

category_id	category_name	issueReport_id
5001	Bug	4001
5002	Payment Issue	4002

### Entity: ReportAttachment

ข้อมูลที่ต้องจัดเก็บ

- - AttachmentID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- - ReportID: integer, Foreign Key, ไม่เป็น Null
- - FilePath: varchar, ไม่เป็น Null
- - UploadedAt: datetime, ไม่เป็น Null

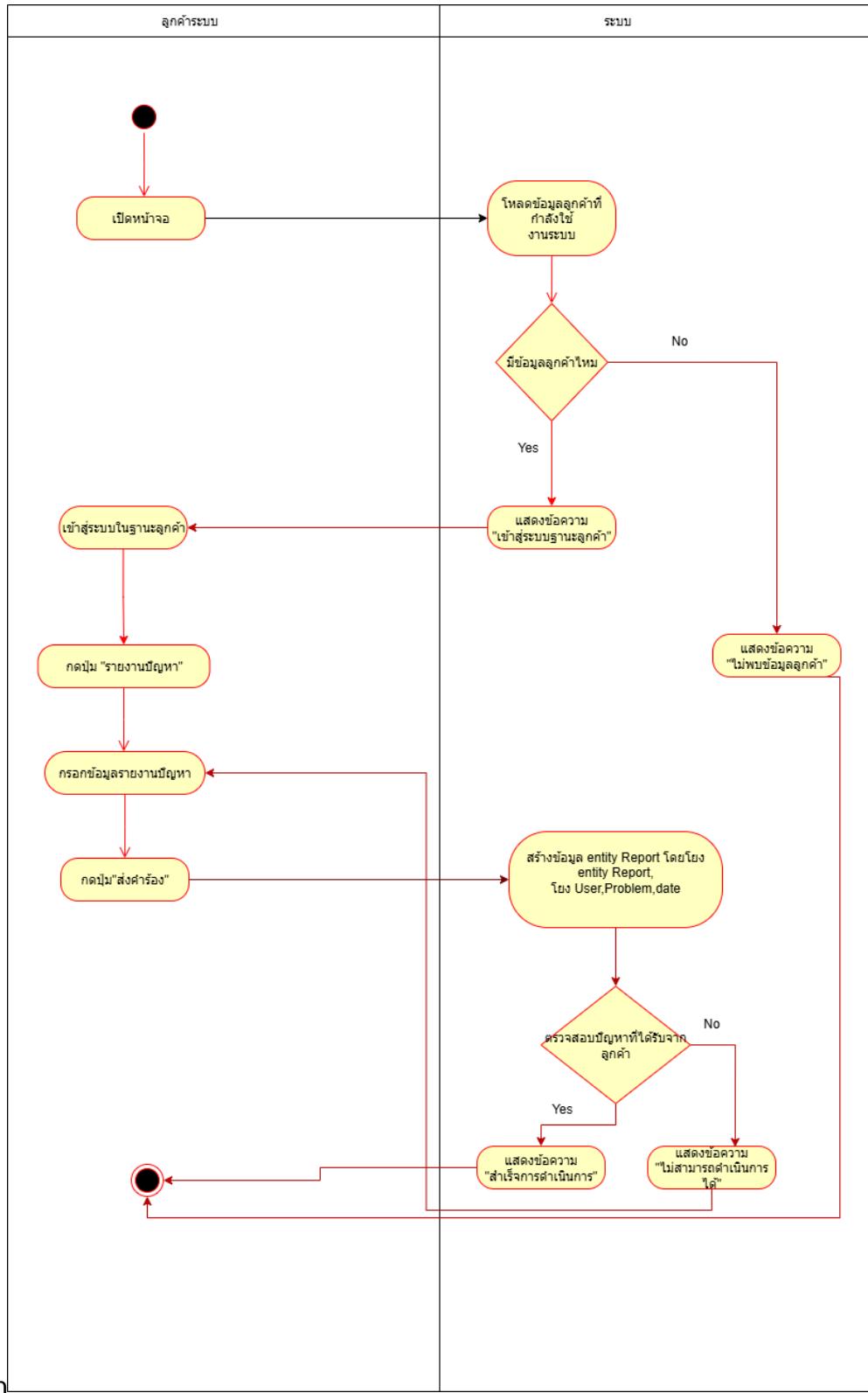
AttachmentID	ReportID	FilePath	UploadedAt
6001	4001	/uploads/error1.png	2025-09-01 10:05:00
6002	4002	/uploads/pay1.png	2025-09-03 09:35:00

## หลักการออกแบบ User Interface

- เมื่อได้กีตามที่มี Foreign Key (FK) ซึ่งจาก ตารางหลักกลับไปยัง ตารางสนับสนุน  
ให้ใช้ ComboBox เป็นตัวเลือกเพื่อเชื่อมโยงข้อมูลใน User Interface
- สำหรับ Field ประเภทอื่นๆ ให้สร้างตามประเภทของข้อมูล เช่น
  - Textbox สำหรับข้อความทั่วไป
  - Password สำหรับข้อมูลรหัสผ่าน
  - Datetime Picker สำหรับเลือกวันและเวลา
  - Numeric Input สำหรับป้อนตัวเลข

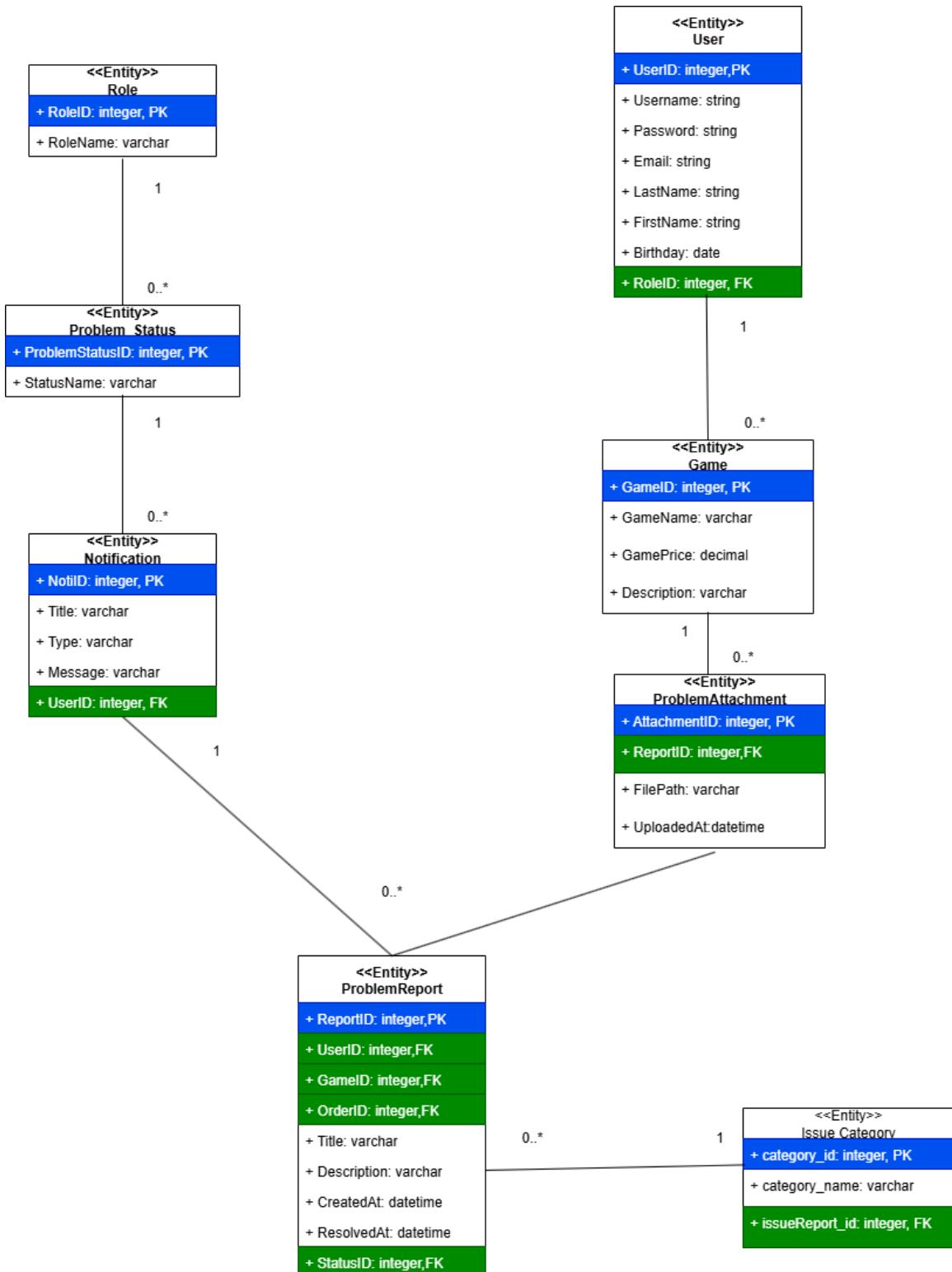
## UI Design

The screenshot shows a user interface design for a game store. At the top, there is a purple header bar with a logo on the left, followed by the text "ระบบจัดการขายเกมออนไลน์" and a user icon on the right. Below the header, the main title "ส่งรายงานปัญหา" is centered. The interface includes several input fields and buttons. On the left, there is a "Game" icon with "GTA VI" next to it, and a clock icon with the date "25/07/2568". Below these, there is a section titled "รายงานปัญหา" with a text input field labeled "describe problem". Further down, there is a section titled "รายละเอียด" with a large text input field. At the bottom, there is a file upload section titled "แนบไฟล์" with a "choose file" button and a "No file" message. Two buttons are at the bottom right: "บันทึก" (Save) and "ยก bỏ" (Cancel).



Activity Diagram

## Class Diagram ระดับ Analysis



## ระบบรายงานปัญหา

The screenshot shows a dark-themed web application for reporting problems. At the top, there's a search bar and a navigation bar with icons for notifications, currency, cart, and user profile. Below that is a form titled 'Report a Problem' with the following fields:

- \* Problem Category: A dropdown menu with the placeholder 'Select a category'.
- \* Title: A text input field with the placeholder 'Short description'.
- \* Description: A text area with the placeholder 'Provide detailed information about the problem'.
- Upload Screenshot / Proof: A button labeled 'Click to Upload'.
- Submit Report: A large purple button at the bottom.

```
package entity
```

```
import (
    "time"

    "gorm.io/gorm"
)

// คำร้องจากลูกค้า
type ProblemReport struct {
    gorm.Model

    Title      string `json:"title"        gorm:"type:varchar(200);not null"`
    Description string `json:"description" gorm:"type:text;not null"`
    Category   string
    `json:"category"  gorm:"type:varchar(100);default:'Other';index"
```

```
// ផ្លូវការទីផ្សារ
UserID uint `json:"user_id" gorm:"not null;index"`
User *User `json:"user" gorm:"foreignKey:UserID"`

// កំណត់ការណ៍ដែលបានរាយ (ភ្លាមី)
GameID uint `json:"game_id" gorm:"index"`
Game *Game `json:"game" gorm:"foreignKey:GameID"`

// សារពន្ធឌាន់ការណ៍ដែលបានរាយ
// ឈ្មោះ: open | in_progress | resolved
Status string `json:"status" gorm:"type:varchar(30);default:'open';index"`
ResolvedAt *time.Time `json:"resolved_at"` // អនុញ្ញាតថាអំពីត្រួតពិនិត្យនៅក្នុងការណ៍ដែលបានរាយ

// ឈ្មោះឯកសារ
Attachments []ProblemAttachment `json:"attachments"
gorm:"foreignKey:ReportID;constraint:OnUpdate:CASCADE,OnDelete:CASCADE;"` 

// គម្រោងដែលបានរាយ
Replies []ProblemReply `json:"replies"
gorm:"foreignKey:ReportID;constraint:OnUpdate:CASCADE,OnDelete:CASCADE;"` 
}
```

## Backend

### Entity Users

```
package entity

import (
    "time"

    "gorm.io/gorm"
)

type User struct {
    gorm.Model
    Username string `gorm:"size:120;uniqueIndex" json:"username"`
    Password string `json:"-"`           // อย่าส่งกลับใน API
    Email    string `gorm:"size:255;uniqueIndex" json:"email"`
    FirstName string `json:"first_name"`
    LastName string `json:"last_name"`
    Birthday time.Time `json:"birthday"`

    RoleID uint `gorm:"not null;index" json:"role_id"`
    Role  Role `gorm:"constraint:OnUpdate:CASCADE,OnDelete:RESTRICT;" json:"role"` // กันลบ role ที่ถูกอ้างอิง

    Threads []Thread `gorm:"foreignKey:UserID" json:"threads,omitempty"`
    Comments []Comment `gorm:"foreignKey:UserID" json:"comments,omitempty"`
    Reactions []Reaction `gorm:"foreignKey:UserID" json:"reactions,omitempty"`
    Attachments []Attachment `gorm:"foreignKey:UserID" json:"attachments,omitempty"`
}
```

```

Notifications []Notification `gorm:"foreignKey:UserID" json:"notifications,omitempty"`
UserGames []UserGame `gorm:"foreignKey:UserID" json:"user_games,omitempty"`

Reviews []Review `gorm:"foreignKey:UserID" json:"reviews,omitempty"`
ReviewLikes []Review_Like `gorm:"foreignKey:UserID" json:"review_likes,omitempty"`

Promotions []Promotion `json:"promotions,omitempty" gorm:"foreignKey:UserID"`
Requests []Request `json:"request" gorm:"foreignKey:UserRefer"`
}

```

## Entity Game

```

package entity

import (
    "time"

    "gorm.io/gorm"
)

type Game struct {
    gorm.Model
    GameName string `json:"game_name"`
    //KeyGameID uint `json:"key_id"`
    CategoriesID int `json:"categories_id"`
    Categories Categories `json:"categories" gorm:"foreignkey:CategoriesID"`
    Date time.Time `json:"release_date" gorm:"autoCreateTime"`
}

```

```

BasePrice    int      `json:"base_price"`
Status       string   `json:"status" gorm:"type:varchar(20);default:'pending';not null;index"`
Minimum_specID uint     `json:"minimum_spec_id"`
MinimumSpec  MinimumSpec `json:"minimum_spec"`
AgeRating    int      `json:"age_rating"`

Requests []Request `gorm:"foreignKey:GameRefer"`

Threads []Thread  `gorm:"foreignKey:GameID" json:"threads,omitempty"`
UserGames []UserGame `gorm:"foreignKey:GameID" json:"user_games,omitempty"`

Reviews []Review   `gorm:"foreignKey:GameID" json:"reviews,omitempty"`
ReviewLikes []Review_Like `gorm:"foreignKey:GameID" json:"review_likes,omitempty"`

Promotions []Promotion `json:"promotions,omitempty"      gorm:"many2many:promotion_games"`
PromotionGames []Promotion_Game
`json:"promotion_games,omitempty"  gorm:"foreignKey:GameID"`
ImgSrc      string   `json:"img_src"   gorm:"type:varchar(512)"`
}

// Hook function ໄວ້ຫລັງສ້າງເກມເສົ່າງແລ້ວ keygame ຈະເຈນເອງ
/*func (g *Game) AfterCreate(tx *gorm.DB) (err error) {
/*func (g *Game) AfterCreate(tx *gorm.DB) (err error) {
kg := KeyGame{}
if err = tx.Create(&kg).Error; err != nil {
    return err
}
return tx.Model(g).Update("key_game_id", kg.ID).Error
}

```

```
 */
```

## Entity Problem\_reply

```
package entity

import "gorm.io/gorm"

// คำตอวจากแอดมินในแต่ละ Report
type ProblemReply struct {
    gorm.Model

    // FK ไปยัง ProblemReport
    ReportID uint      `json:"report_id" gorm:"not null;index"`
    Report   *ProblemReport `json:"-"`
    gorm:"constraint:OnUpdate:CASCADE,OnDelete:CASCADE;`

    // ใครเป็นคนตอบ
    AdminID uint `json:"admin_id" gorm:"not null;index"`
    Admin   *User `json:"admin" gorm:"foreignKey:AdminID"`
```

```

// ข้อความตอบกลับ
Message string `json:"message" gorm:"type:text;not null"`

// ไฟล์แนบของการตอบกลับ
Attachments []ProblemReplyAttachment `json:"attachments"
gorm:"foreignKey:ReplyID;constraint:OnUpdate:CASCADE,OnDelete:CASCADE;"`}

// ไฟล์แนบของคำตอบแอดมิน
type ProblemReplyAttachment struct {
    gorm.Model

    // FK ไปยัง ProblemReply
    ReplyID uint      `json:"reply_id" gorm:"not null;index"`
    Reply   *ProblemReply `json:"-"`
    gorm:"constraint:OnUpdate:CASCADE,OnDelete:CASCADE;"`}

    // ที่อยู่ไฟล์บนเซิร์ฟเวอร์ + ชื่อไฟล์ต้นฉบับ (ใช้โชว์ใน UI)
    FilePath  string `json:"file_path"   gorm:"size:255;not null"`
    OriginalName string `json:"original_name" gorm:"size:255"`
}

```

## Entity Notification

```
package entity
```

```
import "gorm.io/gorm"

type Notification struct {
    gorm.Model
    Title string `json:"title"`
    Type string `json:"type"` // เช่น "report_reply", "refund", "system"
    Message string `json:"message"`

    UserID uint `json:"user_id"`
    User *User `gorm:"foreignKey:UserID" json:"user,omitempty"`

    IsRead bool `json:"is_read" gorm:"default:false"`

    // ✅ อ้างอิงคำร้อง เพื่อให้หน้า UI เปิดดูรายละเอียด/ไฟล์แนบได้
    ReportID *uint `json:"report_id"`
    Report *ProblemReport `gorm:"foreignKey:ReportID" json:"report,omitempty"`
}
```

## Entity Problem\_attachment

```
package entity

import "gorm.io/gorm"

// ไฟล์แนบที่ลูกค้าอัปโหลดตอนส่ง Report
type ProblemAttachment struct {
    gorm.Model

    // FK ไปยัง ProblemReport
    ReportID uint      `json:"report_id" gorm:"not null;index"`
    Report  *ProblemReport `json:"-"
                           " gorm:"constraint:OnUpdate:CASCADE,OnDelete:CASCADE;"`

    // ที่อยู่ไฟล์ + ชื่อไฟล์เดิม (ช่วยให้ UI ให้ว่าชื่อที่เข้าใจได้)
    FilePath  string `json:"file_path"   gorm:"size:255;not null"`
    OriginalName string `json:"original_name" gorm:"size:255"`

}
```

## Entity:Problem\_report

```
package entity

import (
    "time"

    "gorm.io/gorm"
)

// คำร้องจากลูกค้า
type ProblemReport struct {
    gorm.Model

    Title      string `json:"title"      gorm:"type:varchar(200);not null"`
    Description string `json:"description" gorm:"type:text;not null"`
    Category   string `json:"category"    gorm:"type:varchar(100);default:'Other';index"`

    // ผู้ส่งคำร้อง
    UserID uint `json:"user_id" gorm:"not null;index"`
    User   *User `json:"user"   gorm:"foreignKey:UserID"`

    // เกมที่เกี่ยวข้อง (ถ้ามี)
    GameID uint `json:"game_id" gorm:"index"`
    Game   *Game `json:"game"   gorm:"foreignKey:GameID"`

    // สถานะการดำเนินงาน
    // ใช้ค่า: open | in_progress | resolved
    Status   string `json:"status"    gorm:"type:varchar(30);default:'open';index"`
    ResolvedAt *time.Time `json:"resolved_at"` // อนุญาตให้เป็น null
```

```

// แนบไฟล์ของลูกค้า
Attachments []ProblemAttachment `json:"attachments"
gorm:"foreignKey:ReportID;constraint:OnUpdate:CASCADE,OnDelete:CASCADE;"` 

// ความสัมพันธ์กับตารางตอบกลับของแอดมิน
Replies []ProblemReply `json:"replies"
gorm:"foreignKey:ReportID;constraint:OnUpdate:CASCADE,OnDelete:CASCADE;"` 
}

```

Controller

User\_controller

```

package controllers

import (
    "net/http"
    "time"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
    "golang.org/x/crypto/bcrypt"
)

// POST /users
func CreateUser(c *gin.Context) {
    var req struct {

```

```
Username string `json:"username"`
Password string `json:"password"`
Email string `json:"email"`
FirstName string `json:"first_name"`
LastName string `json:"last_name"`
Birthday string `json:"birthday"`
//RoleID uint `json:"role_id"`

}

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

birthday, err := time.Parse("2006-01-02", req.Birthday)
if err != nil {
    c.JSON(http.StatusBadRequest, gin.H{"error": "invalid birthday format"})
    return
}

body := entity.User{
    Username: req.Username,
    Password: req.Password,
    Email: req.Email,
    FirstName: req.FirstName,
    LastName: req.LastName,
    Birthday: birthday,
    RoleID: configs.UserRoleID(),
}
```

```
    hashedPassword, err := bcrypt.GenerateFromPassword([]byte(body.Password),
bcrypt.DefaultCost)

    if err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": "failed to hash
password"})
        return
    }

    body.Password = string(hashedPassword)

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

    body.Password = ""
    c.JSON(http.StatusCreated, body)
}

// GET /users
// optional: ?username=... / ?email=...
func FindUsers(c *gin.Context) {
    var users []entity.User
    db := configs.DB()

    username := c.Query("username")
    email := c.Query("email")
    roleID := c.Query("role_id")
    tx := db.Model(&entity.User{})

    if username != "" {
        tx = tx.Where("username = ?", username)
```

```

}

if email != "" {
    tx = tx.Where("email = ?", email)
}

if roleID != "" {
    tx = tx.Where("role_id = ?", roleID)
}

if err := tx.Prefetch("Requests").Find(&users).Error; err != nil {
    c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
    return
}

c.JSON(http.StatusOK, users)
}

// GET /users/:id
func FindUserByID(c *gin.Context) {
    var user entity.User
    if tx := configs.DB().Prefetch("Requests").Prefetch("Role").First(&user,
        c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, user)
}

// PUT /users/:id
func UpdateUser(c *gin.Context) {
    var payload entity.User
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
    }
}

```

```
        return
    }

var user entity.User
db := configs.DB()
if tx := db.First(&user, c.Param("id")); tx.RowsAffected == 0 {
    c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
    return
}
if err := db.Model(&user).Updates(payload).Error; err != nil {
    c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
    return
}
c.JSON(http.StatusOK, gin.H{"message": "updated successful"})
}

// PATCH /users/:id/role
func UpdateUserRole(c *gin.Context) {
    var body struct {
        RoleID uint `json:"role_id"`
    }
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    var user entity.User
    if tx := configs.DB().First(&user, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "id not found"})
        return
    }
```

```

}

user.RoleID = body.RoleID
if err := configs.DB().Save(&user).Error; err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}
c.JSON(http.StatusOK, user)
}

// DELETE /users/:id
func DeleteUserByID(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM users WHERE id = ?", c.Param("id"));
    tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "deleted successful"})
}

```

## Notification\_controller

```

package controllers

import (
    "log"
    "net/http"
)

```

```
"example.com/sa-gameshop/configs"
"example.com/sa-gameshop/entity"
"github.com/gin-gonic/gin"
)

// POST /notifications
func CreateNotification(c *gin.Context) {
    var body entity.Notification
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request body"})
        return
    }

    // ตรวจว่ามี user นี้จริง
    var user entity.User
    if tx := configs.DB().Where("id = ?", body.UserID).First(&user); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "user_id not found"})
        return
    }

    // default type
    if body.Type == "" {
        body.Type = "system"
    }
    body.IsRead = false

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

```
        return
    }

    log.Printf(" 📡 CreateNotification: user_id=%d title=%q message=%q type=%q
report_id=%v",
        body.UserID, body.Title, body.Message, body.Type, body.ReportID)

    c.JSON(http.StatusCreated, body)
}

// GET /notifications?user_id=...
func FindNotifications(c *gin.Context) {
    uid := c.Query("user_id")
    if uid == "" {
        c.JSON(http.StatusBadRequest, gin.H{"error": "must provide user_id"})
        return
    }

    var rows []entity.Notification
    if err := configs.DB().
        Preload("User").
        Preload("Report").
        Preload("Report.Attachments").
        Where("user_id = ?", uid).
        Order("created_at DESC").
        Find(&rows).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
}
```

```
    log.Printf("✉ FindNotifications: user_id=%s count=%d", uid, len(rows))
    c.JSON(http.StatusOK, rows)
}

// GET /notifications/:id
func FindNotificationByID(c *gin.Context) {
    var row entity.Notification
    if tx := configs.DB().
        Preload("User").
        Preload("Report").
        Preload("Report.Attachments").
        First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "notification not found"})
        return
    }
    c.JSON(http.StatusOK, row)
}

// PUT /notifications/:id/read
func MarkNotificationRead(c *gin.Context) {
    id := c.Param("id")
    db := configs.DB()

    if err := db.Model(&entity.Notification{}).
        Where("id = ?", id).
        Update("is_read", true).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
}
```

```

var row entity.Notification
_ = db.First(&row, id)
c.JSON(http.StatusOK, row)
}

// PUT /notifications/read-all
func MarkAllNotificationsRead(c *gin.Context) {
    var body struct {
        UserID uint `json:"user_id"`
    }
    if err := c.ShouldBindJSON(&body); err != nil || body.UserID == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "must provide user_id"})
        return
    }

    db := configs.DB()
    if err := db.Model(&entity.Notification{}).
        Where("user_id = ? AND is_read = ?", body.UserID, false).
        Update("is_read", true).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, gin.H{"message": "marked all as read"})
}

// DELETE /notifications/:id
func DeleteNotificationByID(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM notifications WHERE id = ?",
        c.Param("id")); tx.RowsAffected == 0 {

```

```
c.JSON(http.StatusNotFound, gin.H{"error": "notification not found"})  
    return  
}  
c.JSON(http.StatusOK, gin.H{"message": "deleted successfully"})  
}
```

## Problem\_controller

```
// backend/controllers/problem_report.go  
package controllers  
  
import (  
    "errors"  
    "fmt"  
    "net/http"  
    "os"  
    "path/filepath"  
    "strconv"  
    "strings"  
    "time"  
  
    "example.com/sa-gameshop/configs"
```

```
"example.com/sa-gameshop/entity"
"github.com/gin-gonic/gin"
"gorm.io/gorm"
)

// =====
// Reports
// =====

// POST /reports (multipart/form-data)
// fields: title, description, category, user_id, attachments[]
func CreateReport(c *gin.Context) {
    db := configs.DB()

    title := strings.TrimSpace(c.PostForm("title"))
    desc := strings.TrimSpace(c.PostForm("description"))
    category := strings.TrimSpace(c.PostForm("category"))
    status := strings.TrimSpace(c.PostForm("status"))

    if status == "" {
        status = "open"
    }

    // ✅ อ่าน user id จาก X-User-ID ก่อน แล้วค่อย fallback ไป form "user_id"
    uidStr := strings.TrimSpace(c.GetHeader("X-User-ID"))
    if uidStr == "" {
        uidStr = strings.TrimSpace(c.PostForm("user_id"))
    }
    userID, _ := strconv.Atoi(uidStr)

    if title == "" || desc == "" || userID <= 0 {
```

```
c.JSON(http.StatusBadRequest, gin.H{"error": "missing required fields (title,  
description, user_id)"})  
  
return  
}  
  
  
// ✅ SOFT-CHECK ផ្លូវការ: ត្រាំងមិនបានរកឃើញឈ្មោះដើម្បីសរាយរាយការណ៍តែត្រូវ (កន្លែង dev/test ទីយោងមិនមែន  
record)  
  
if err := db.First(&entity.User{}, userID).Error; err != nil {  
    if err != nil && !errors.Is(err, gorm.ErrRecordNotFound) {  
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})  
        return  
    }  
    // ត្រាំង ErrRecordNotFound: ខ្លួនបានរកឃើញឈ្មោះដើម្បីសរាយរាយការណ៍  
}  
  
  
// ❌ មិនបានរកឃើញ game_id នៃរាយការណ៍  
  
report := entity.ProblemReport{  
    Title:      title,  
    Description: desc,  
    Category:   category,  
    Status:     status,  
    UserID:     uint(userID),  
}  
  
  
if err := db.Create(&report).Error; err != nil {  
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})  
    return  
}
```

```

// ແນບໄຟລ໌ຜູ້ໃໝ່ → uploads/reports/{reportID}/...
if form, _ := c.MultipartForm(); form != nil {
    files := form.File["attachments"]
    if len(files) == 0 {
        files = form.File["file"]
    }
    if len(files) > 0 {
        dir := filepath.Join("uploads", "reports", fmt.Sprintf("%d", report.ID))
        _ = os.MkdirAll(dir, 0o755)
        for _, f := range files {
            name := fmt.Sprintf("%d_%s", time.Now().UnixNano(), filepath.Base(f.Filename))
            dst := filepath.Join(dir, name)
            relPath := "/" + filepath.ToSlash(dst)

            if err := c.SaveUploadedFile(f, dst); err != nil {
                continue
            }
            _ = db.Create(&entity.ProblemAttachment{
                FilePath: relPath,
                ReportID: report.ID,
            }).Error
        }
    }
}

// ແຈ້ງເຕືອນແອດມີນທຸກຄົນວ່າມີກໍາຮ່ອງໃໝ່
notifyAdminsNewReport(db, &report)

_ = db.
    Preload("User").

```

```
    Preload("Attachments").
    Preload("Replies").
    Preload("Replies.Admin").
    Preload("Replies.Attachments").
    First(&report, report.ID).Error

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

// GET /reports?user_id=&game_id=&status=&page=&limit=
func FindReports(c *gin.Context) {
    db := configs.DB()

    var (
        userID, gameID uint
        page          = 1
        limit         = 20
    )

    if v := c.Query("user_id"); v != "" {
        if n, err := strconv.Atoi(v); err == nil && n > 0 {
            userID = uint(n)
        }
    }

    if v := c.Query("game_id"); v != "" {
        if n, err := strconv.Atoi(v); err == nil && n > 0 {
            gameID = uint(n)
        }
    }
}
```

```

status := strings.TrimSpace(c.Query("status"))

if v := c.Query("page"); v != "" {
    if n, err := strconv.Atoi(v); err == nil && n > 0 {
        page = n
    }
}

if v := c.Query("limit"); v != "" {
    if n, err := strconv.Atoi(v); err == nil && n > 0 && n <= 100 {
        limit = n
    }
}

offset := (page - 1) * limit

q := db.Model(&entity.ProblemReport{})
if userID > 0 {
    q = q.Where("user_id = ?", userID)
}
if gameID > 0 {
    q = q.Where("game_id = ?", gameID)
}
if status != "" {
    q = q.Where("status = ?", status)
}

var items []entity.ProblemReport
if err := q.
    Preload("User").
    Preload("Attachments").

```

```

Preload("Replies").
Preload("Replies.Admin").
Preload("Replies.Attachments").
Order("created_at DESC").
Offset(offset).
Limit(limit).
Find(&items).Error; err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}

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

// GET /reports/:id
func GetReportByID(c *gin.Context) {
    db := configs.DB()
    id, _ := strconv.Atoi(c.Param("id"))

    var rp entity.ProblemReport
    if err := db.
        Preload("User").
        Preload("Attachments").
        Preload("Replies").
        Preload("Replies.Admin").
        Preload("Replies.Attachments").
        First(&rp, id).Error; err != nil {
        if errors.Is(err, gorm.ErrRecordNotFound) {
            c.JSON(http.StatusNotFound, gin.H{"error": "report not found"})
            return
        }
    }
}

```

```

    }

    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})

    return
}

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

type updateReportInput struct {

    Title      *string `json:"title,omitempty"`
    Description *string `json:"description,omitempty"`
    Category   *string `json:"category,omitempty"`
    Status     *string `json:"status,omitempty"`
    Resolve    *bool   `json:"resolve,omitempty"`
}

// PUT /reports/:id
func UpdateReport(c *gin.Context) {
    db := configs.DB()
    id, _ := strconv.Atoi(c.Param("id"))

    var rp entity.ProblemReport
    if err := db.First(&rp, id).Error; err != nil {
        if errors.Is(err, gorm.ErrRecordNotFound) {
            c.JSON(http.StatusNotFound, gin.H{"error": "report not found"})
            return
        }
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
}

```

```
var in updateReportInput

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

if in.Title != nil {
    rp.Title = strings.TrimSpace(*in.Title)
}

if in.Description != nil {
    rp.Description = strings.TrimSpace(*in.Description)
}

if in.Category != nil {
    rp.Category = strings.TrimSpace(*in.Category)
}

if in.Status != nil {
    rp.Status = strings.TrimSpace(*in.Status)
}

if in.Resolve != nil {
    if *in.Resolve {
        now := time.Now()
        rp.ResolvedAt = &now
        if rp.Status == "open" || rp.Status == "" {
            rp.Status = "resolved"
        }
    } else {
        rp.ResolvedAt = nil
        if rp.Status == "resolved" {
            rp.Status = "open"
        }
    }
}
```

```
        }

    }

    if err := db.Save(&rp).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }

    _ = db.
        Preload("User").
        Preload("Attachments").
        Preload("Replies").
        Preload("Replies.Admin").
        Preload("Replies.Attachments").
        First(&rp, rp.ID).Error
    c.JSON(http.StatusOK, gin.H{"data": rp})
}

// DELETE /reports/:id
func DeleteReport(c *gin.Context) {
    db := configs.DB()
    id, _ := strconv.Atoi(c.Param("id"))
    if err := db.Delete(&entity.ProblemReport{}, id).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.Status(http.StatusNoContent)
}

// =====
// Admin Replies (แอดมินตอบกลับลูกค้า)
```

```

// =====

// POST /reports/:id/reply (multipart/form-data)
// fields: [admin_id], message หรือ text, attachments[]
// ใช้โดยหน้า AdminPage ฝั่งคุณ
func ReplyReport(c *gin.Context) {
    db := configs.DB()
    reportID, _ := strconv.Atoi(c.Param("id"))
    adminID, _ := strconv.Atoi(c.PostForm("admin_id")) // optional

    // ยอมรับทั้ง "message" หรือ "text"
    msg := strings.TrimSpace(c.PostForm("text"))
    if msg == "" {
        msg = strings.TrimSpace(c.PostForm("message"))
    }

    if reportID <= 0 || msg == "" {
        c.JSON(http.StatusBadRequest, gin.H{"error": "missing required fields"})
        return
    }

    reply, err := createReplyFlow(c, db, reportID, adminID, msg, /*autoResolve*/ true,
    /*requireAdmin*/ false)
    if err != nil {
        return
    }
    c.JSON(http.StatusCreated, gin.H{"data": reply})
}

// POST /admin/reports/:id/replies (multipart/form-data)

```

```

// fields: admin_id, message (หรือ text), attachments[]
func AdminCreateReply(c *gin.Context) {
    db := configs.DB()
    reportID, _ := strconv.Atoi(c.Param("id"))
    adminID, _ := strconv.Atoi(c.PostForm("admin_id"))

    // ยอมรับทั้ง "message" หรือ "text"
    msg := strings.TrimSpace(c.PostForm("message"))
    if msg == "" {
        msg = strings.TrimSpace(c.PostForm("text"))
    }

    if reportID <= 0 || adminID <= 0 || msg == "" {
        c.JSON(http.StatusBadRequest, gin.H{"error": "missing required fields"})
        return
    }

    reply, err := createReplyFlow(c, db, reportID, adminID, msg, /*autoResolve*/ false,
    /*requireAdmin*/ true)
    if err != nil {
        return
    }
    c.JSON(http.StatusCreated, gin.H{"data": reply})
}

// PATCH /admin/reports/:id/resolve
func AdminResolveReport(c *gin.Context) {
    db := configs.DB()
    id, _ := strconv.Atoi(c.Param("id"))
}

```

```
var rp entity.ProblemReport

if err := db.First(&rp, id).Error; err != nil {
    if errors.Is(err, gorm.ErrRecordNotFound) {
        c.JSON(http.StatusNotFound, gin.H{"error": "report not found"})
        return
    }
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}

now := time.Now()
rp.Status = "resolved"
rp.ResolvedAt = &now

if err := db.Save(&rp).Error; err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}

// แจ้งลูกค้าว่าปิดงานแล้ว
makeOrUpdateUserReportNotification(db, rp.UserID, "report_resolved", uint(rp.ID),
fmt.Sprintf("คำร้อง #%-d ถูกปิดงานเรียบร้อย", rp.ID))

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

=====

// Helpers (Reply Flow + Notification)
=====
```

```
func createReplyFlow(c *gin.Context, db *gorm.DB, reportID int, adminID int, msg string,
autoResolve bool, requireAdmin bool) (*entity.ProblemReply, error) {
    // โหลด Report
    var report entity.ProblemReport
    if err := db.First(&report, reportID).Error; err != nil {
        if errors.Is(err, gorm.ErrRecordNotFound) {
            c.JSON(http.StatusNotFound, gin.H{"error": "report not found"})
            return nil, err
        }
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return nil, err
    }

    // require admin?
    if requireAdmin && adminID <= 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "missing admin_id"})
        return nil, errors.New("missing admin_id")
    }

    // สร้าง Reply
    reply := entity.ProblemReply{
        ReportID: uint(reportID),
        Message: msg,
    }
    if adminID > 0 {
        reply.AdminID = uint(adminID)
    }
    if err := db.Create(&reply).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return nil, err
    }
}
```

```

}

// แนบไฟล์ → uploads/replies/{replyID}/...
attachCount := 0

if form, _ := c.MultipartForm(); form != nil {
    files := form.File["attachments"]

    if len(files) == 0 {
        files = form.File["file"]
    }

    if len(files) > 0 {
        attachCount = len(files)

        dir := filepath.Join("uploads", "replies", fmt.Sprintf("%d", reply.ID))
        _ = os.MkdirAll(dir, 0o755)
        for _, f := range files {
            name := fmt.Sprintf("%d_%s", time.Now().UnixNano(), filepath.Base(f.Filename))
            dst := filepath.Join(dir, name)
            relPath := "/" + filepath.ToSlash(dst)

            if err := c.SaveUploadedFile(f, dst); err != nil {
                continue
            }
            _ = db.Create(&entity.ProblemReplyAttachment{
                ReplyID: reply.ID,
                FilePath: relPath,
            }).Error
        }
    }
}

// สร้าง Notification ให้ลูกค้า (สร้างใหม่ทุกครั้ง)

```

```

makeOrUpdateUserReportNotification(db, report.UserID, "report_reply", uint(report.ID),
buildReplyMessage(msg, attachCount))

// ปิดงานอัตโนมัติกรณี /reports/:id/reply
if autoResolve {
    now := time.Now()
    report.Status = "resolved"
    report.ResolvedAt = &now
    _ = db.Save(&report).Error
}

_ = db.
    Preload("Admin").
    Preload("Attachments").
    First(&reply, reply.ID).Error

return &reply, nil
}

func notifyAdminsNewReport(db *gorm.DB, report *entity.ProblemReport) {
    // เลือกผู้ใช้ที่เป็น Admin ทั้งหมด (สมมติว่ามี roles.name = 'Admin')
    type adminUser struct {
        ID   uint
        Name string
    }
    var admins []adminUser

    // หาก schema ของคุณไม่มี roles ให้ปรับส่วนนี้ให้เข้ากับของจริง
    db.Table("users").
        Joins("JOIN roles ON roles.id = users.role_id").

```

```

Where("roles.name = ?", "Admin").
Select("users.id, users.name").
Scan(&admins)

for _, a := range admins {
    _ = db.Create(&entity.Notification{
        Title:   "✉️ มีคำร้องใหม่",
        Message: fmt.Sprintf("%s: %s", strings.TrimSpace(report.Category),
strings.TrimSpace(report.Title)),
        Type:    "report_new",
        UserID:  a.ID,      // ผู้รับ = แอดมิน
        ReportID: &report.ID, // อ้างอิงคำร้อง
        IsRead:  false,
    }).Error
}
}

// ✅ เปลี่ยนจาก "หาแล้วอัปเดต" → "สร้างแจ้งเตือนใหม่ทุกรังส์"
func makeOrUpdateUserReportNotification(db *gorm.DB, userID uint, typ string, reportID
uint, msg string) {
    title := ""
    switch typ {
    case "report_reply":
        title = fmt.Sprintf("ตอบกลับคำร้อง #%-d", reportID)
    case "report_resolved":
        title = fmt.Sprintf("ปิดงานคำร้อง #%-d", reportID)
    default:
        title = "การแจ้งเตือนคำร้อง"
    }
}

```

```
_ = db.Create(&entity.Notification{
    Title: title,
    Message: msg,
    Type: typ,
    UserID: userID,
    ReportID: &reportID,
    IsRead: false,
}).Error
}
```

```
func buildReplyMessage(msg string, attachCount int) string {
    if msg == "" && attachCount > 0 {
        return "ແອດມີນໄດ້ຕອບກລັບພົ້ອມໄຟລ໌ແນບ"
    }
    if msg != "" && attachCount > 0 {
        return fmt.Sprintf("%s (ແນບໄຟລ໌ %d ໄຟລ໌)", msg, attachCount)
    }
    if msg != "" {
        return msg
    }
    return "ແອດມີນໄດ້ຕອບກລັບຄໍາຮ້ອງຂອງคຸນ"
}
```

## Problem\_attachment

```
package controllers

import (
    "net/http"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

// POST /problem-attachments
func CreateProblemAttachment(c *gin.Context) {
    var body entity.ProblemAttachment
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request body"})
        return
    }

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

```
}

// GET /problem-attachments
// optional: ?report_id=...
func FindProblemAttachments(c *gin.Context) {
    var attachments []entity.ProblemAttachment
    db := configs.DB()

    reportID := c.Query("report_id")
    tx := db.Model(&entity.ProblemAttachment{})
    if reportID != "" {
        tx = tx.Where("report_id = ?", reportID)
    }
    if err := tx.Find(&attachments).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, attachments)
}

// GET /problem-attachments/:id
func FindProblemAttachmentByID(c *gin.Context) {
    var attachment entity.ProblemAttachment
    if tx := configs.DB().First(&attachment, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, attachment)
}
```

```
// PUT /problem-attachments/:id
func UpdateProblemAttachment(c *gin.Context) {
    var payload entity.ProblemAttachment
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    var attachment entity.ProblemAttachment
    db := configs.DB()
    if tx := db.First(&attachment, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }

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

    c.JSON(http.StatusOK, gin.H{"message": "updated successful"})
}

// DELETE /problem-attachments/:id
func DeleteProblemAttachmentByID(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM problem_attachments WHERE id = ?",
        c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "id not found"})
        return
    }
}
```

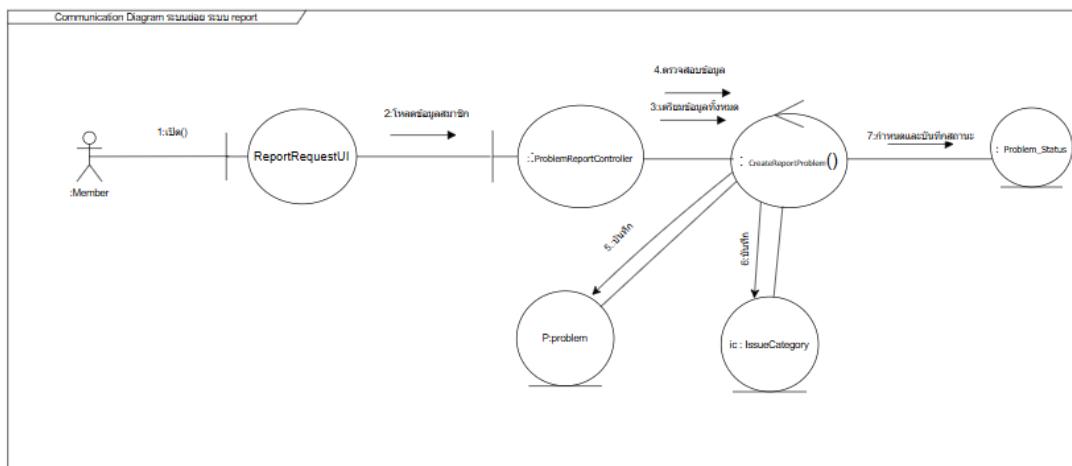
```
c.JSON(http.StatusOK, gin.H{"message": "deleted successful"})  
}
```

## Communication Diagram: Problem Report

Activity	เป็นคำสั่ง สำหรับ ระบบหรือไม่	ถ้าเป็น, วัตถุที่ได้รับหน้าที่ ทำงานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click “เปิดหน้าจอ Problem Report”	เป็นคำสั่ง	ReportRequestUI	เปิด():
โหลดข้อมูลผู้ใช้ที่ login (UserID)	เป็นคำสั่ง	:ProblemReportUI	โหลดข้อมูลสมาชิก (id)
ตรวจสอบคำสั่งซึ่งที่เกี่ยวข้อง (OrderID, GameID)	เป็นคำสั่ง	:ProblemReportController	เตรียมข้อมูลและดึงข้อมูล ทั้งหมด
แสดงหน้าจอกรอกข้อมูล (Title, Description,	ไม่เป็นคำสั่ง		
กรอกข้อมูล + กดปุ่ม “ยืนยันรายงานปัญหา	เป็นคำสั่ง	R:report	:ProblemReportUI → :ProblemReportController
สร้าง ProblemReport (UserID, GameID, OrderID, Title, Description, CreatedAt)	เป็นคำสั่ง	P:problem	สร้าง(problem)
บันทึก ()	เป็นคำสั่ง	P:problem	บันทึก()
แนบไฟล์หลักฐาน (FilePath, UploadedAt)	ไม่เป็นคำสั่ง		
บันทึก ()	เป็นคำสั่ง	P:problem	บันทึก()
กำหนดประเภทปัญหา (CategoryID)	เป็นคำสั่ง	:ProblemReportController	ic : IssueCategory

กำหนดสถานะเริ่มต้น “Open”	เป็นคำสั่ง : Problem_Status	
อัปเดต ProblemReport.StatusID	ไม่เป็นคำสั่ง	
แสดงข้อความ “ส่งรายงานเรียบร้อย”	ไม่เป็นคำสั่ง	

Community Diagram



## Communication Class diagram

