

ระบบร้านขายเกมออนไลน์

ระบบร้านเกมออนไลน์ เป็นระบบที่ให้บริการสำหรับผู้ใช้งานสามารถทำการเข้าสู่ระบบ Login เพื่อ เลือกชื่อเกม ต่าง ๆ ที่เปิดขายบนแพลตฟอร์ม โดยระบบจะจัดแสดงข้อมูลผ่าน ระบบจัดการข้อมูลเกม ที่รวมถึงชื่อเกม ประเภทเกม ค่ายผู้ผลิต ราคา คำอธิบาย และภาพตัวอย่าง เพื่ออำนวยความสะดวกให้กับ ผู้ใช้งาน ระบบมี ระบบ workshop ที่ผู้ใช้งานจะสามารถนำมودเกมที่ผู้ใช้งานสร้างสรรค์เองมาอปโหลดลงบนระบบของเราเพื่อเผยแพร่ มอดของตัวเองและระบบยังสามารถให้ผู้ใช้คนอื่นสามารถดาวน์โหลดไปใช้ได้แล้วก็ยังสามารถมาให้คะแนน มอดได้ด้วย โดยผู้ใช้งานสามารถเลือกชื่อเกมผ่าน ระบบจัดการการชำระเงิน พร้อมกับการประมวลผลส่วนลดจาก ระบบ จัดการโปรโมชั่น ซึ่งสามารถเพิ่มโค้ดส่วนลดหรือโปรโมชั่นตามเทศกาลได้เมื่อมีความผิดพลาดหรือความไม่ พึง พอดีในการซื้อ ระบบมี ระบบการจัดการคืนเงิน ที่ให้ผู้ใช้งานส่งคืนคืนเงิน พร้อมตรวจสอบเงื่อนไขตามที่ ระบบกำหนด หากผู้ใช้ต้องการเสนอให้ร้านนำเกมที่ยังไม่เข้ามาจำหน่าย สามารถใช้งานระบบ

- รีวิวเกมสำหรับการขาย ซึ่งสมาชิกสามารถพิมพ์ชื่อเกมและรายละเอียดเกมที่ต้องการ พร้อมระบบให้คะแนน ผู้ใช้คนอื่น เพื่อช่วยให้ผู้ดูแลพิจารณา เพื่อสร้างประสบการณ์ที่ดีขึ้น สมาชิกสามารถแสดงความคิดเห็นและให้ คะแนนเกมผ่าน ระบบบุรีวิวและให้คะแนน ซึ่งรองรับการให้คะแนนใน ระดับ 1–5 ดาว พร้อมข้อความรีวิวและ ระบบนี้จะเชื่อมกับหน้าเกมโดยตรง ระบบยังมีระบบคอมมูนิตี้- ขนาดเล็กที่เปิดให้ผู้ใช้งานตั้งกระทู้ถาม-ตอบ และเปลี่ยนประสบการณ์ และพูดคุยกันกับเกม พร้อม ความสามารถในการแสดงความคิดเห็น กดถูกใจ กด บันทึก หรือแจ้งโพสต์ที่ไม่เหมาะสม โดยมีผู้ดูแลระบบคอย ควบคุมเนื้อหา สำหรับความปลอดภัยและการควบคุม สิทธิการใช้งาน ระบบมี ระบบจัดการสิทธิผู้ใช้งาน ที่สามารถกำหนด ระดับสิทธิของผู้ใช้งานทั่วไป ผู้ดูแลระบบ และพนักงาน ใน การเข้าถึงฟังก์ชันต่าง ๆ ภายใต้ระบบ นอกจากนี้ยัง มี ระบบรายงานปัญหา ที่ให้สามารถ แจ้งปัญหาการใช้งาน เช่น ปัญหาในการซื้อเกม ระบบขัดข้อง ซึ่ง จะถูกส่งต่อให้แอดมินทำการตรวจสอบและ ดำเนินการต่อไป

ระบบที่เกิดขึ้นจาก ระบบฐานข่ายเกมออนไลน์

- ระบบย่อย ระบบปรีเคิลเกมสำหรับการขาย
- ระบบย่อย ระบบจัดการข้อมูลเกม
- ระบบย่อย ระบบสร้างคอนเทนต์เกมใน workshop
- ระบบย่อย ระบบจัดการสิทธิผู้ใช้งาน
- ระบบย่อย ระบบจัดการการชำระเงิน
- ระบบย่อย ระบบคอมมูนิตี้ขนาดเล็ก
- ระบบย่อย ระบบรีวิวและให้คะแนน
- ระบบย่อย ระบบจัดการโปรโมชัน
- ระบบย่อย ระบบการจัดการคืนเงิน
- ระบบย่อย ระบบรายงานปัญหา

B6611613 นายณัฐพงษ์ดันนัย แหนงกระโทก

ระบบหลัก ระบบร้านขายเกมออนไลน์

ระบบย่อย ระบบบริเวณสำหรับการขาย

ระบบซื้อ-ขายเกม เป็นระบบที่ให้ลูกค้า (Customer) สามารถเข้ามาซื้อเกมผ่านระบบของเราได้ โดยสามารถทำการลงทะเบียน เข้ามาซื้อเกมโดยลูกค้าสามารถรีบีเวสเกมที่อยากรับซื้อในแต่ละเดือนผ่านระบบบริเวณสำหรับการขาย โดยจะใช้รหัสผ่านซื้อเกม หมวดหมู่ของเกม วันที่รีบีเวส และเหตุผลที่อยากรับซื้อเกมนั้น เพื่อที่ระบบการขายจะได้เห็นความต้องการลูกค้าแล้วนำเกมที่ลูกค้าการจำแนกมาขายได้

User Story ระบบบริเวณสำหรับการขาย

ในบทบาทของ (As a) ผู้ดูแลระบบ

ฉันต้องการ (I want to) ข้อมูลจำนวนรีบีเวสเกมจากผู้ใช้

เพื่อ (So that) จะสามารถใช้ข้อมูลนั้นวิเคราะห์ว่าเกมใดควรวางขายในระบบ

ในบทบาทของ (As a) ลูกค้า

ฉันต้องการ (I want to) ส่งรีบีเวสเกมที่อยากรับซื้อในร้านจำนวนมาก

เพื่อที่ (So that) สามารถซื้อเกมที่ฉันสนใจอนาคตได้

Output บนหน้าจอ

- ลูกค้าจะสามารถรีบีเวสเกม(Create)จากที่ต้องการได้ จากนั้นระบบจะดึงข้อมูล(Read)มาแสดงเกมที่มีการรีบีเวสและจำแนกเรียบง่ายตามแต่ละเกมที่ลูกค้าเลือกมา

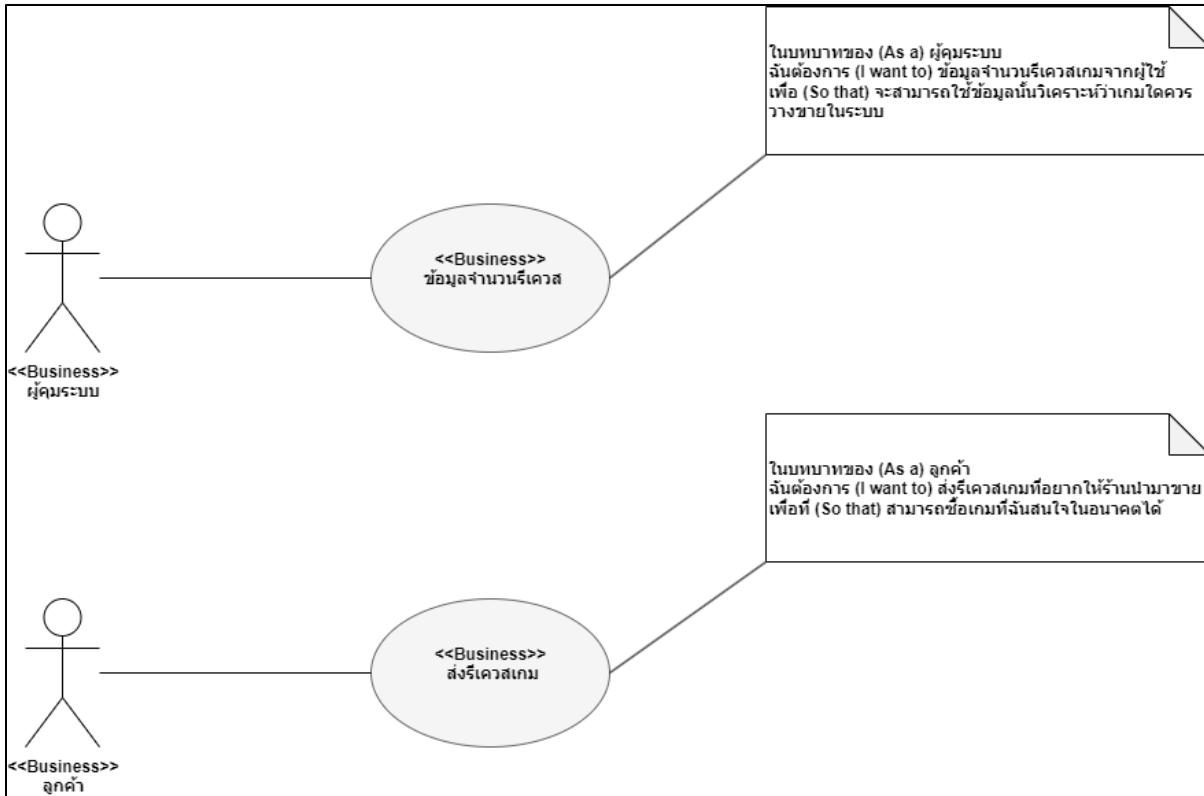
Output ของข้อมูล

- ระบบจะทำการเพิ่มข้อมูลของผู้ใช้ที่รีบีเวสเกมมา เข้าในฐานข้อมูลของเกมที่มีการรีบีเวส

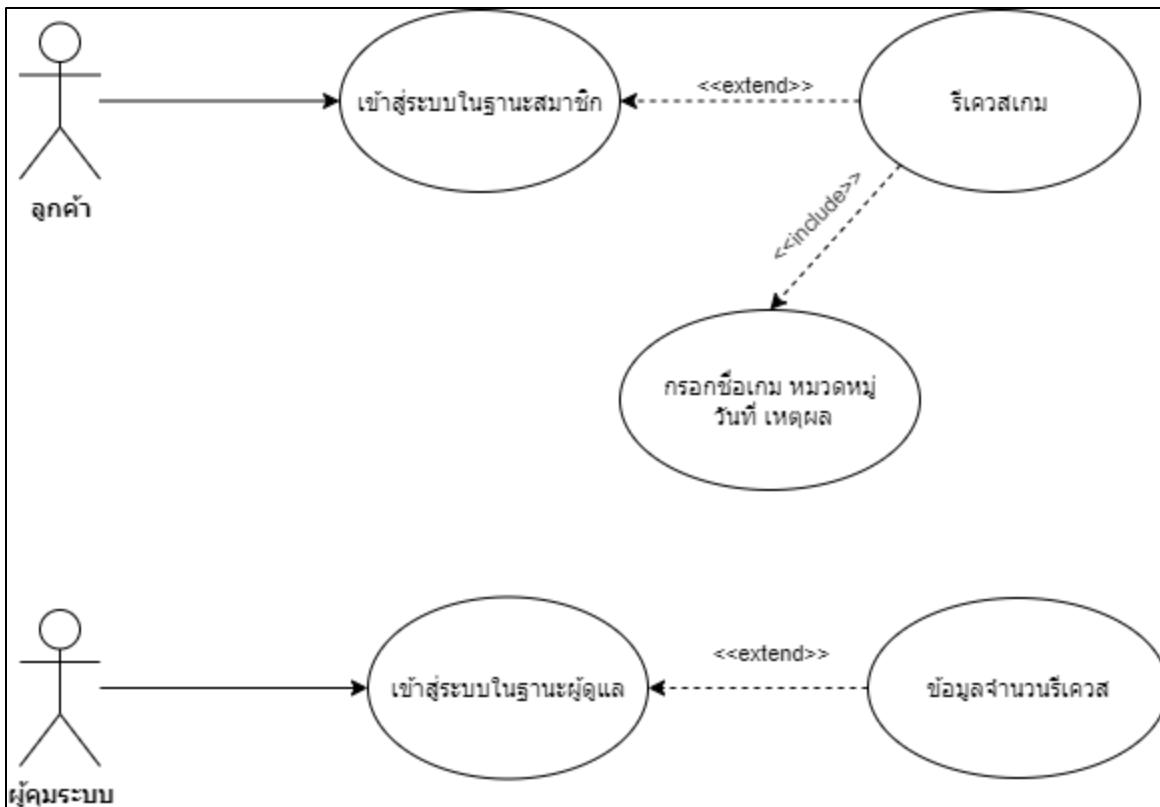
คำนำที่อาจจะถูกยกมาเป็นตารางในฐานข้อมูลของระบบ

คำนำ	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
ลูกค้า	เกี่ยวข้องโดยตรง เนื่องจากผู้ใช้งานจะเป็นคนสร้างรีเควสต์ซึ่งมีลูกค้าเป็นหนึ่งในตัวอย่างเช่นเดียวกัน
เกม	เกี่ยวข้องโดยตรง กับจำนวนลูกค้าที่รีเควสต์เกมนั้นๆ
หมวดหมู่	เกี่ยวข้องโดยตรง กับเกม
วันที่	ไม่เกี่ยวข้องโดยตรง
เหตุผล	เกี่ยวข้องโดยตรง เนื่องจากลูกค้าอาจมีเหตุผลในการรีเควสต์เกมนั้นหรือไม่ก็ได้
จำนวน	เกี่ยวข้องโดยตรง เนื่องจากในเกมเดียวก็มีจำนวนลูกค้าที่รีเควสต์เกมนั้นต่างกัน ซึ่งเป็นตัวตัดสินในการเลือกเกมมาวางขาย
รีเควซ์เกม	เกี่ยวข้องโดยตรง เนื่องจากผู้ใช้งานจะเป็นคนสร้างรีเควสต์

Business Use Case Diagram (แบบเดี่ยว)



System use Case Diagram (แบบเดี่ยว)



Checklist: อธิบาย

- การรีเควส์เกมลูกค้าจำเป็นต้องล็อกอินในฐานะสมาชิกก่อน
- ในระหว่างที่รีเควส์เกมจำเป็นต้องกรอกรายละเอียดเกมที่รีเควสไป
- ผู้ดูแลระบบจะเข้าถึงข้อมูลได้ต้องล็อกอินเข้าสู่ระบบก่อน

Checklist: System Use Case Diagram

- System Actor และ System Use Case ต้องไม่มีอะไรกำกับ
- เส้นเชื่อมจาก System Actor ไปหา System Use Case จะต้องเป็นเส้นทึบ หัวลูกศรปลายเปิดซึ่งไปหา System Use Case
- System Actor ต้องเป็นบทบาทของคนที่ใช้ระบบจริง ๆ
- System Use Case ต้องเป็นคำที่แสดงพฤติกรรม
- ถ้ามีกลไกทาง Security มาเกี่ยว ชื่อ System Use Case จะอยู่ในรูปแบบ “เข้าระบบในฐานะ <Actor>” เช่นนั้น
- การใช้ <--<<extend>>---> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ทางเลือกไปยัง System Use Case หลัก

7. การใช้ <--<<include>>--> ต้องเป็นส่วนประ ทวัลุกศรปลายเปิด ชี้จาก 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 Request 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*

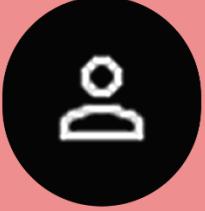
กรุณากรอกชื่อเกنم

姓氏*

กรุณากรอกนามสกุล

วันที่*

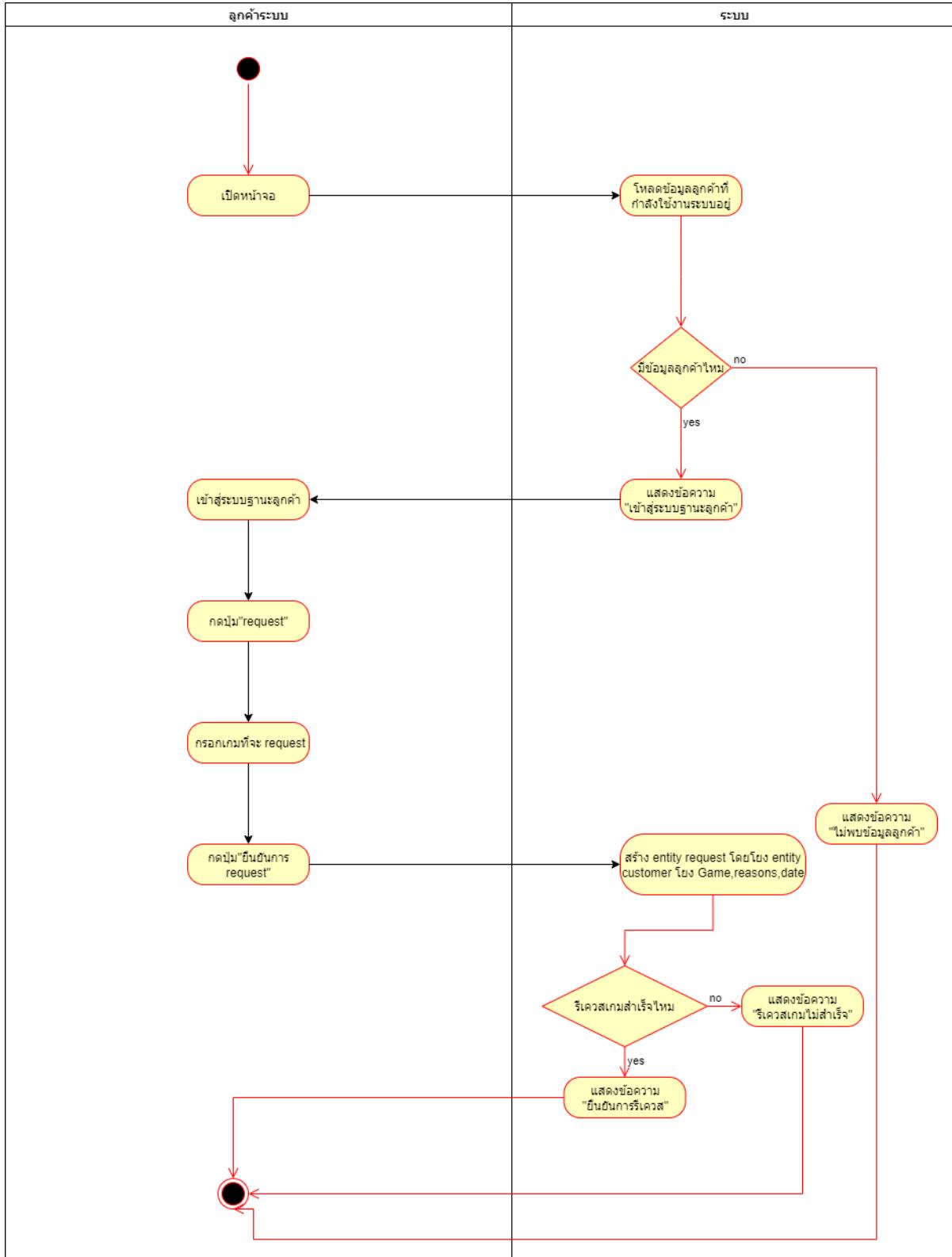
กรุณากรอกวันที่

 Nattakdanai Yangkratok



ยืนยัน

Activity Diagram



B6611613 นายณัฐก์ดันย แหยงกระโภก

ระบบหลัก ระบบร้านขายเกมออนไลน์

ระบบย่อย ระบบจัดการเกม

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

ส่วนบทบาทผู้ควบคุมจะสามารถจัดการข้อมูลเกมเพื่อแสดงรายละเอียดเกม สเปกขั้นต่ำของคอมพิวเตอร์ รายงานของเกม และรีวิวของลูกค้าผ่านระบบรีวิวและให้คะแนน ที่เก็บข้อมูล ชื่อเกมที่รีวิว คะแนน ความคิดเห็น วันที่รีวิว

User Story ระบบจัดการเกม

ในบทบาทของ (As a) แอดมิน

ฉันต้องการ (I want to) จัดการเกม

เพื่อ (So that) เพิ่มเกมเข้าสู่ระบบ

Output บนหน้าจอ

- ผู้ควบคุมจะสามารถกดแก้ไขข้อมูลเกม -> ผู้ควบคุมจะสามารถกดเพิ่ม, แก้ไข หรือ ลบ ข้อมูลเกมนั้นๆได้

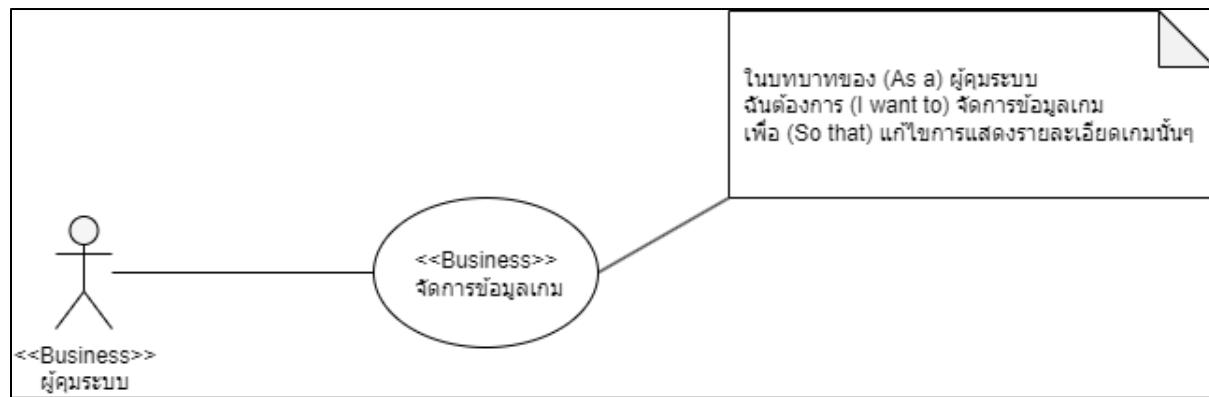
Output ของข้อมูล

- ระบบจะดึงข้อมูล(Read)รายละเอียดของเกมนั้นๆ มาเพื่อให้ผู้คุมระบบสามารถแก้ไขรายละเอียด (update)ของเกมนั้นๆ

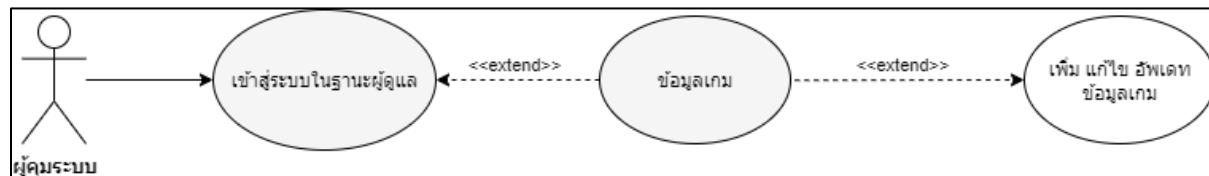
คำนำที่อาจจะกลایมมาเป็นตารางในฐานข้อมูลของระบบ

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

- ผู้คุมระบบสามารถเข้าถึงข้อมูลเกมได้ฝ่ายเดียว
- การเข้าถึงข้อมูลเกมนั้นไม่จำเป็นที่จะต้องแก้ไขข้อมูลเกมก็ได้

Checklist: System Use Case Diagram

- System Actor และ System Use Case ต้องไม่มีอะไรกำกับ
- เส้นโยงจาก System Actor ไปหา System Use Case จะต้องเป็นเส้นทึบ หัวลูกศรปลายเปิดซึ่งไปหา System Use Case
- System Actor ต้องเป็นบทบาทของคนที่ใช้ระบบจริง ๆ
- System Use Case ต้องเป็นคำที่แสดงพฤติกรรม
- ถ้ามีกลไกทาง Security มาเกี่ยว ชื่อ System Use Case จะอยู่ในรูปแบบ “เข้าระบบในฐานะ <Actor>” เท่านั้น
- การใช้ <--<<extend>>---> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ทางเลือกไปยัง System Use Case หลัก
- 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	USERNA ME	PASSWORD VARCHAR NOT NULL	EMAIL VARCHAR NOT NULL	FIRST_NA ME	LAST_NA VARCHAR NOT NULL	BIRTHD AY
admin	VERCHAR NOT NULL, Unique			VARCHAR NOT NULL		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
 - Point: เก็บในรูปแบบ integer 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	C_id	K_id	Date	M_id	Img_src	Status	Agerating
Game	VARCHAR NOT NULL	INTEGER FK	INTEGER FK	DATE NOT NULL	INTEGER FK	VARCHAR	VARCHAR	Integer

NULL, PK								
1001	Kf2	1	1	19- 12- 2000	1	fasf	pending	16
1002	Db	2	2	19- 12- 2000	2	asfasf	pending	20

คีย์เกม(keygame)

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

ID Keygame	Key VARCHAR UNIQUE
INTEGER NOT NULL, PK	
1001	asdasd
1002	adad

หมวดหมู่ (categories)

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

ID	Title
Categories	VARCHAR
INTEGER	UNIQUE
NOT	
NULL, PK	
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	OS VARCHAR	Processor VARCHAR	Memory VARCHAR	Graphics VARCHAR	Storage VARCHAR
INTEGER NOT NULL, PK					
1001	Window10	I5 10300	32 gb	Gtx 1090	12 gb

1002	Window 7	I7 11900	16 gb	Rtx 2020	200 mb
------	----------	----------	-------	----------	--------

UI Design

Edit

ชื่อเกม*

คะแนน*

- /10

รีวิว*



Nattakdanai Yangkratok

16/05/2018



Minimum Spec*

OS*

Processor*

Memory*

Graphics*

Storage*

Reward*

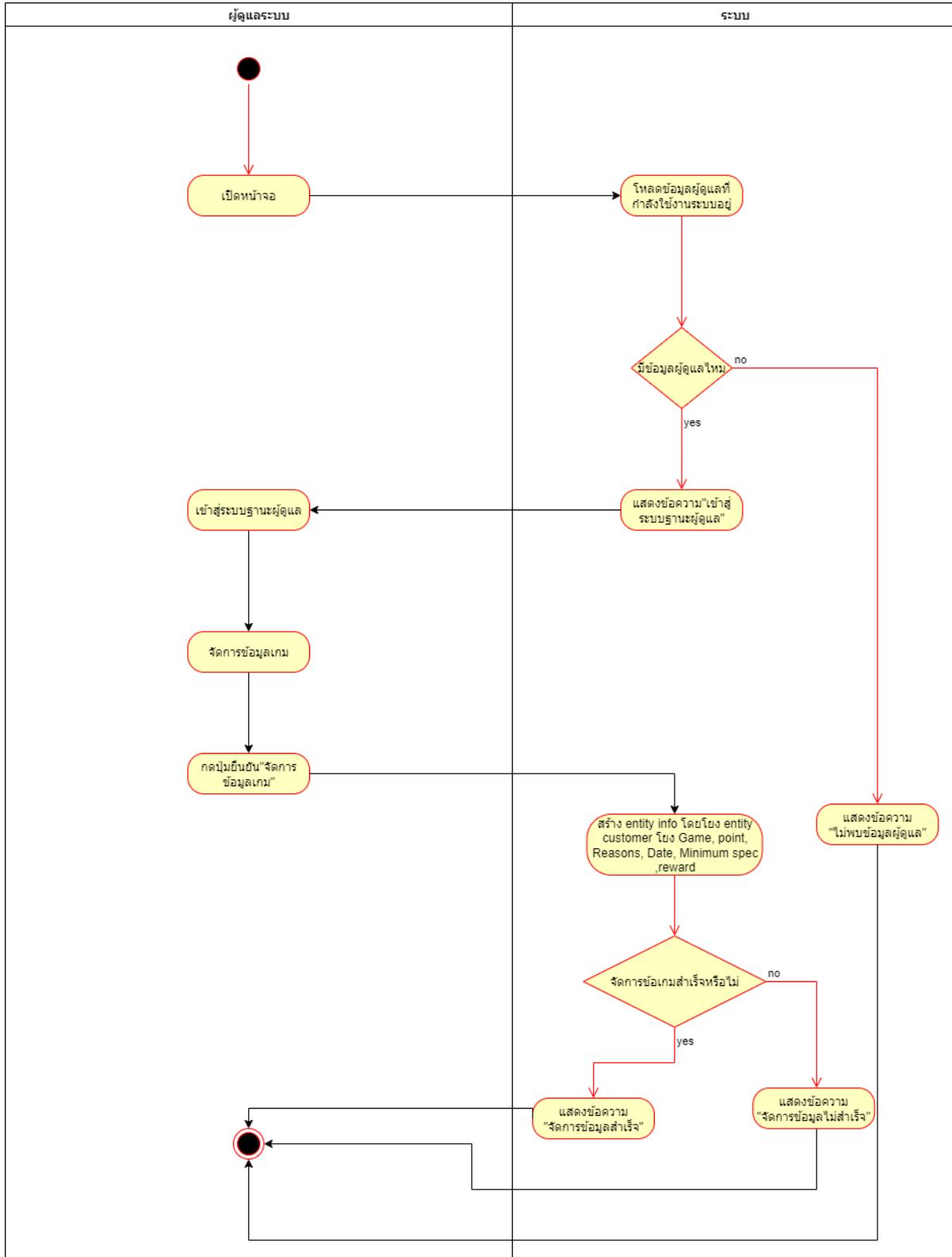


Jack Dorson

16/05/2018

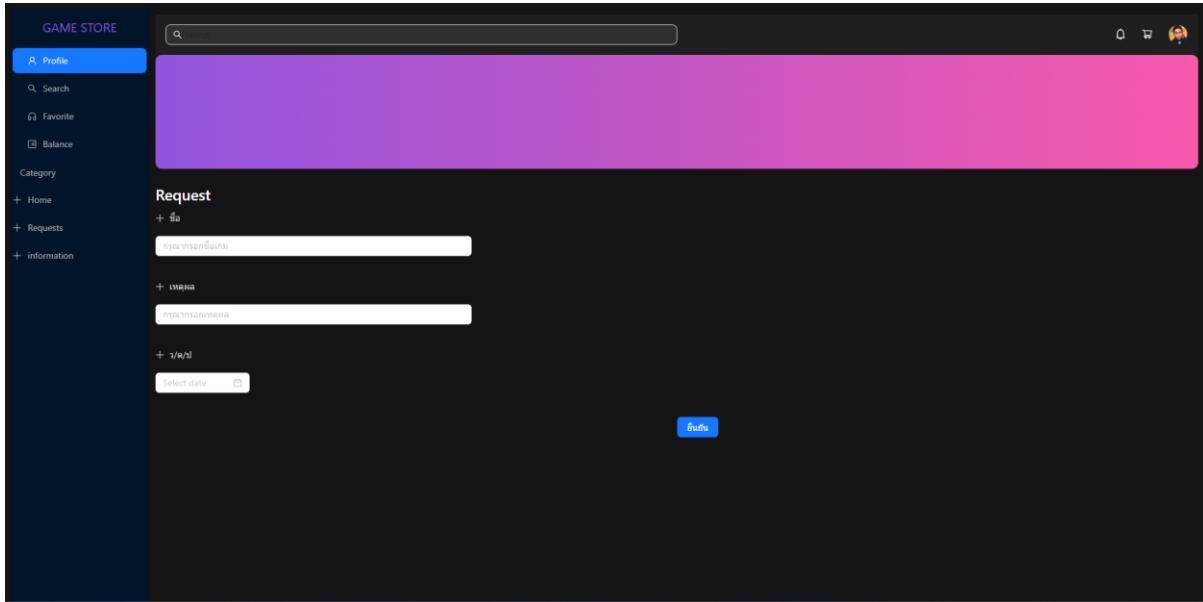
ยืนยัน

Activity Diagram



Frontend

ระบบปรีเคส



```
import Sidebar from "../components/Sidebar";
import { Layout, Space, Button } from 'antd';
import Navbar from "../components/Navbar";
import { Typography, Input, DatePicker } from "antd";
import { Col, Row } from 'antd';

import {
  PlusOutlined ,
} from '@ant-design/icons';

const { Title } = Typography;

const Request = () => {

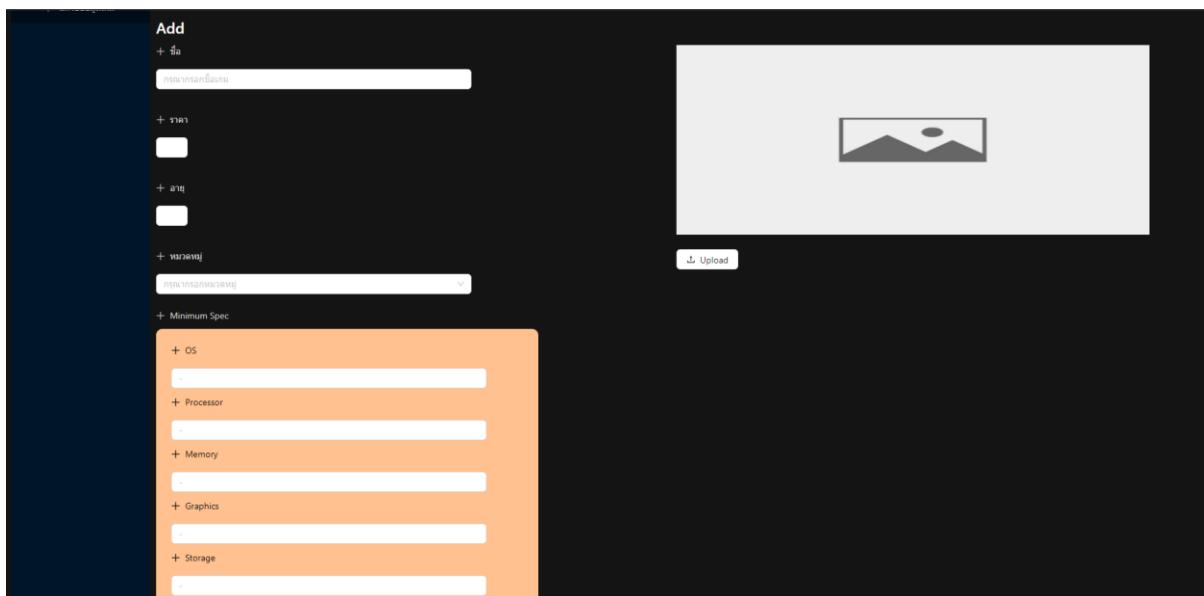
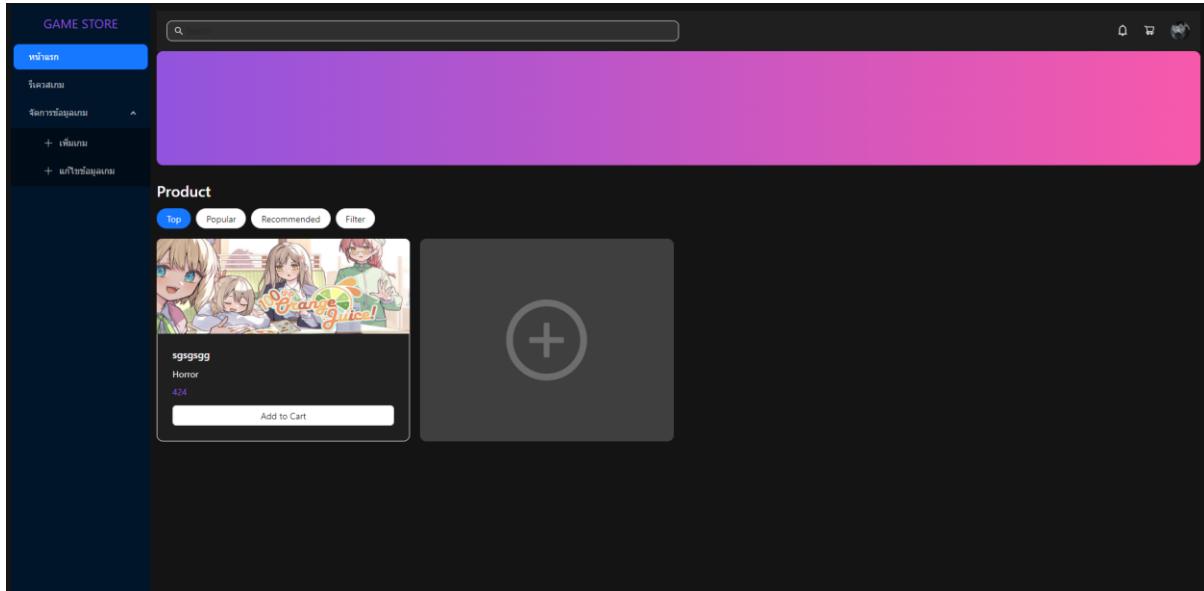
  return(
    <Layout>
      <Sidebar />
      <Layout style={{ background: '#141414', flex: 1 , minHeight: '100vh'}}>
        <div style={{ padding: '10px' }}>
          <Navbar />
          <div style={{ background: 'linear-gradient(90deg, #9254de 0%, #f759ab 100%)', height: 180, borderRadius: 10, marginBottom: 24 }}></div>
        <Title level={3} style={{ color: 'white' }}>Request</Title> /* title ต้อง import Typography*/
        <Space><PlusOutlined style={{background: '#141414', color: '#d6d6d6ff}}/><text style={{ color: '#d6d6d6ff }}>จ่อ</text></Space>
        <br />
        <br />
      </Layout>
    )
}
```

```
<Input placeholder="ក្រុណាករណីខែក" style={{ width: 500}}/>
<br />
<br />
<br />
<Space><PlusOutlined style={{background: '#141414', color: '#d6d6d6ff'}}/><text style={{ color: '#d6d6d6ff' }}>ឡាមត
</text></Space>
<br />
<br />
<Input placeholder="ក្រុណាករណីខែមុន" style={{ width: 500}}/>
<br />
<br />
<br />
<Space><PlusOutlined style={{background: '#141414', color: '#d6d6d6ff'}}/><text style={{ color: '#d6d6d6ff' }}>៣/៩/១
</text></Space>
<br />
<br />
<DatePicker />
<br />
<br />
<br />
<br />
<Row>
<Col offset={12}><Button type="primary">បង្កើត</Button></Col>
</Row>
</div>
</Layout>
</Layout>
};

};

export default Request;
```

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(""); // ถ้าคลิกไฟล์อื่นก็เคลียร์ 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: categoryld,
        });
        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("เพิ่มเกมスペック:", 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>
      </div>
      <Title level={3} style={{ color: 'white' }}>Add</Title> /* title នៃការបង្កើត */
      <Row style={{marginBottom: 24 }}>
        <Col span={12}>
          <Space><PlusOutlined style={{background: '#141414', color: '#d6d6d6ff'}}/><text style={{ color: '#d6d6d6ff' }}>កុំពួក</text></Space>
        <br />
        <br />
        <Input placeholder="ក្រុមករករបស់ខ្លួន" 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' }}>រាជការ</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' }}>អាយុ</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' }}>អ្នកចុះហត្ថលេខា</text></Space>
        <br />
        <br />
        <Select placeholder="ក្រុមករករបស់អ្នកចុះហត្ថលេខា" 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/*/*" style={{width: "100%", height: "40px", border: "1px solid #ccc", padding: "5px", margin: "10px 0", display: "block", text-align: "center", color: "#ccc", cursor: "pointer", outline: "none", border-radius: "5px", background: "#fff", position: "relative", overflow: "hidden", transition: "border-color 0.3s ease-in-out, background-color 0.3s ease-in-out, color 0.3s ease-in-out, transform 0.3s ease-in-out, opacity 0.3s ease-in-out, box-shadow 0.3s ease-in-out, border-radius 0.3s ease-in-out, background-size 0% 0% / 100% 100%}}>Upload</Upload></Upload>
      </Col>
    </Layout>
  </div>
)

```

```

        </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;
  }
  // ແປດໄສ img_src ໃຫ້ເປັນ absolute URL ເສັນອະນຸມາດ
  const resolveImgUrl = (src?: string) => {
    if (!src) return "";
  }
}

```

```

if (src.startsWith("blob:")) return "";
if (src.startsWith("data:image/")) return src;
// ถ้าเป็น blob: เดิม จะมักใช้ไม่ได้ในหน้าอื่น — ควรแก้ไข backend ให้ส่ง URL カラ
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={resovleImgUrl(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 ໄວ້ລັບສຽງເກມເສີ່ງແລ້ວ 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.Category
    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 ใหม่ลง 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
    }

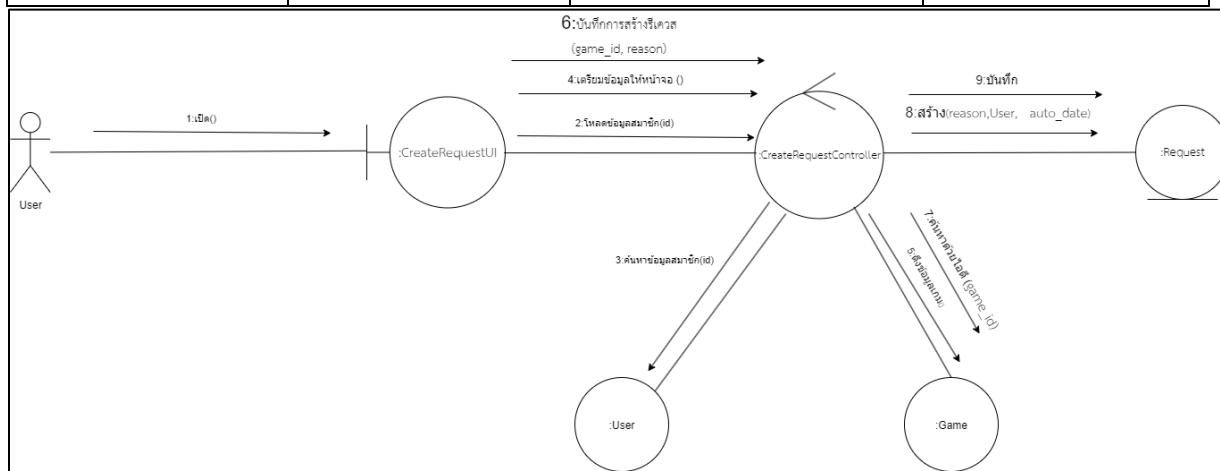
    if err := configs.DB().Create(&minimum_specs).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }

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

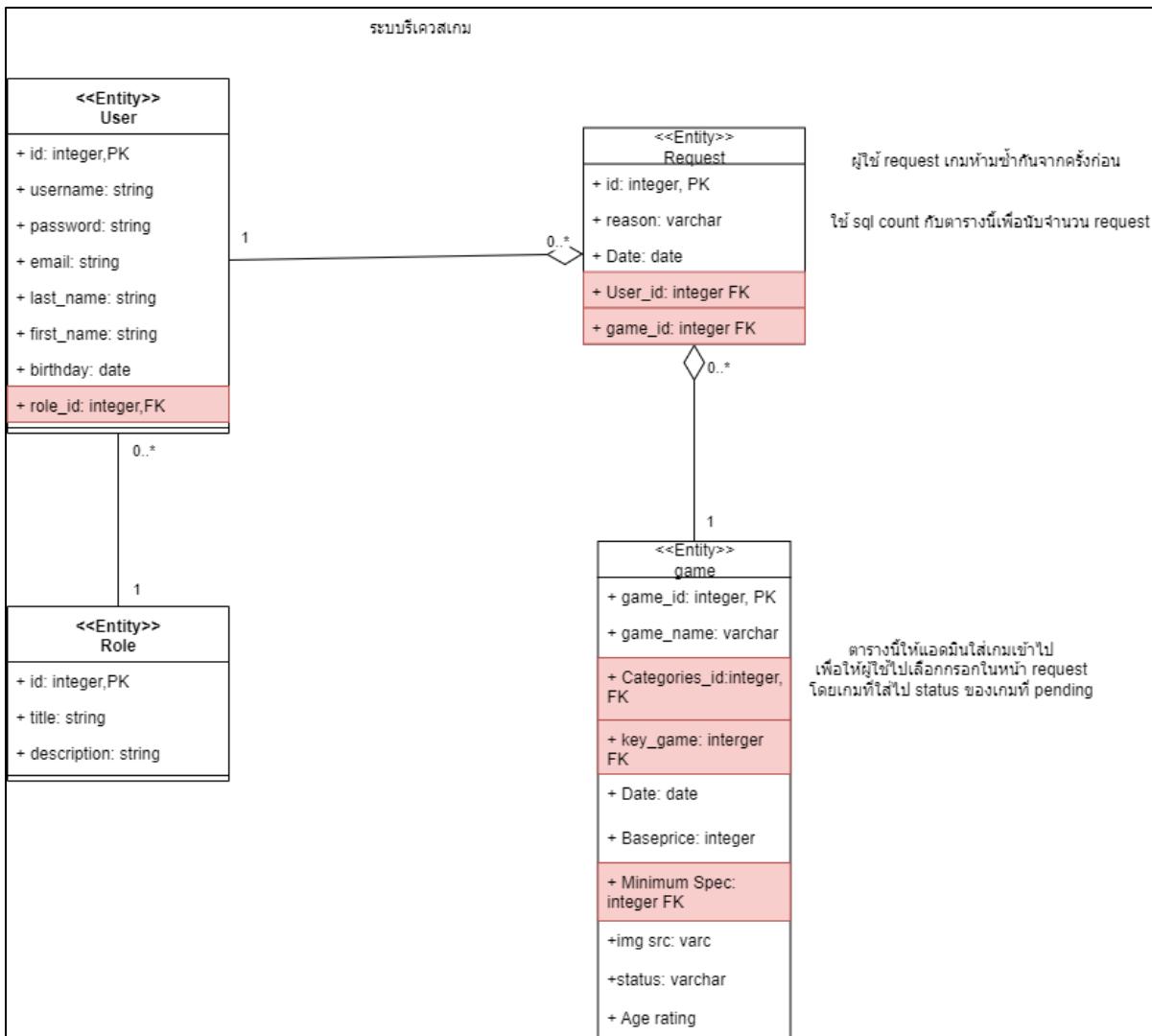
Communication diagram(ระบบบีทีเคเวสเกม)

Activity	เป็นคำสั่งสำหรับระบบได้หรือไม่	ถ้าเป็น ,วัตถุที่รับหน้าที่ทำงานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click "เปิดหน้าจอ"	เป็นคำสั่ง สั่งงาน เพื่อให้หน้า UI เปิด	:CreateRequestUI	เปิด ()
โหลดข้อมูลผู้ใช้ที่กำลังใช้งานระบบอยู่	เป็นคำสั่ง เกิดการ สั่งงานให้โหลดข้อมูลสมาชิกที่ login อยู่	:CreateRequestController	โหลดข้อมูลสมาชิก (id)
		U:User	ค้นหาด้วยไอดี (id)
โหลดข้อมูลรายการเกม	เป็นคำสั่ง เกิดการ สั่งงานให้โหลดข้อมูลเกมที่มีสถานะ pending	:CreateRequestController	เตรียมข้อมูลให้หน้าจอ ()
		:game	ดึงข้อมูลเกม()
แสดงหน้าจอสำหรับสร้าง "Request"	ไม่เป็นคำสั่ง	-	-
กดเลือกเกม (ได้game_id)	ไม่เป็นคำสั่ง	-	-
กรอกเหตุผล (ได้reason)	ไม่เป็นคำสั่ง	-	-
กดปุ่ม "Request"	เป็นคำสั่ง	:CreateRequestController	บันทึกการสร้างรีเควส (game_id, reason)

	ระบบทำการจัดเก็บข้อมูล Request		
ค้นหา entity game ด้วย game_id ของ game ที่รับเข้ามา	เป็นคำสั่ง	:game	ค้นหาด้วยอีดี (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 "สร้างรีเควสสำเร็จ"	เป็นคำสั่ง	-	-



Class Diagram at Design Level(ระบบบีรีเควสเกม)

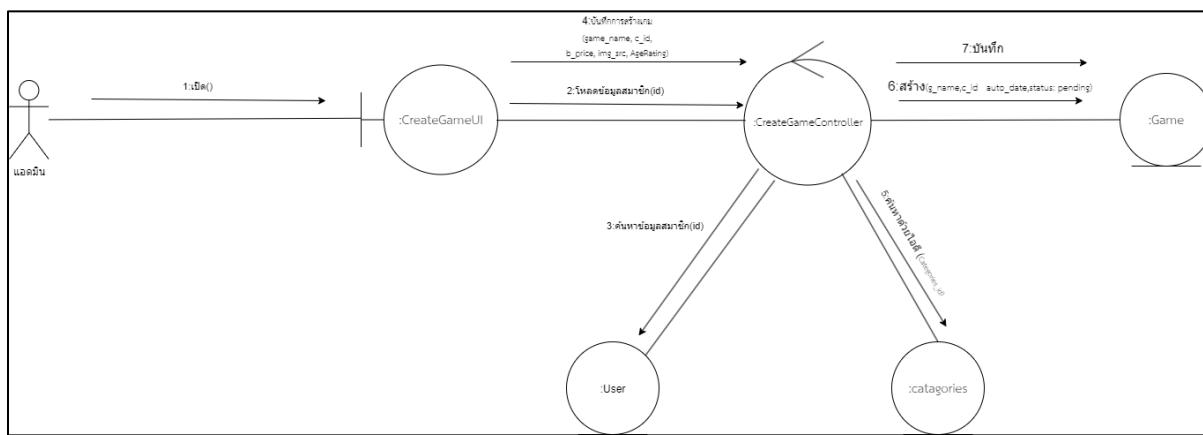


Communication diagram(ระบบจัดการเกม) (สร้างเกม)

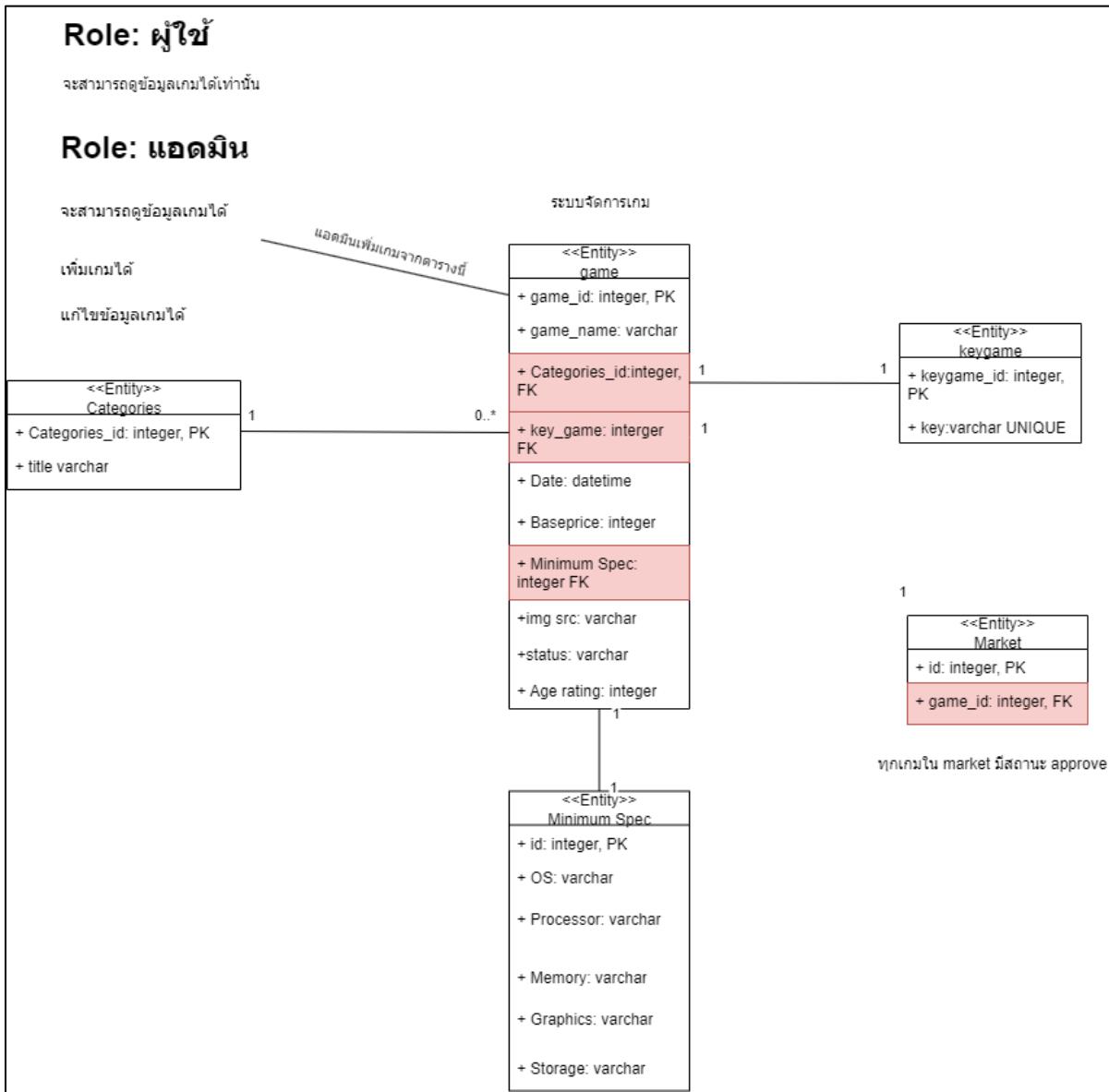
Activity	เป็นคำสั่งสำหรับระบบได้หรือไม่	ถ้าเป็น , วัตถุที่รับหน้าที่ทำงานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click "เปิดหน้าจอ"	เป็นคำสั่ง สั่งงาน เพื่อให้หน้า UI เปิด	:CreateGameUI	เปิด ()
		:CreateGameController	โหลดข้อมูลสมาชิก (id)

โหลดข้อมูลผู้ใช้ที่กำลังใช้งานระบบอยู่	เป็นคำสั่ง เกิดการสั่งงานให้โหลดข้อมูลสมาชิกที่ login อยู่	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"	เป็นคำสั่ง ระบบทำการจัดเก็บข้อมูล 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 เช็ต ค่า game_name, b_price, img_src	เป็นคำสั่ง	:Game	สร้าง(๔_name,c_id auto_date,status: pending)

, AgeRating			
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 เมื่อลูกค้าเล่นเกมแล้วทางเราก็อยากให้มารีวิวและให้คะแนนเกมโดยมีคะแนนตั้งแต่ 1-5 คะแนน เพื่อเชิญชวนคนอื่นๆ ให้มาพิจารณาดูว่าเกมนั้นเป็นอย่างไรบ้าง โดยในระบบของเรา มี **ผู้ใช้งาน** หลายประเภท เช่น ลูกค้า พนักงาน **ผู้ดูแลระบบ (Admin)** ก็เลยจำเป็นต้องมีระบบจัดการสิทธิผู้ใช้งานเพื่อแยกสิทธิการใช้งานและจัดการสิทธิการใช้งานของผู้ใช้แต่ละประเภท โดยแอดมินจะสามารถ **create, update Role** ของผู้ใช้ได้ และก็สามารถ **delete Role** ได้ โดย **Role** ต่างๆ จะถูกกำหนด **สิทธิ** ให้ว่าสามารถทำอะไรได้ 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

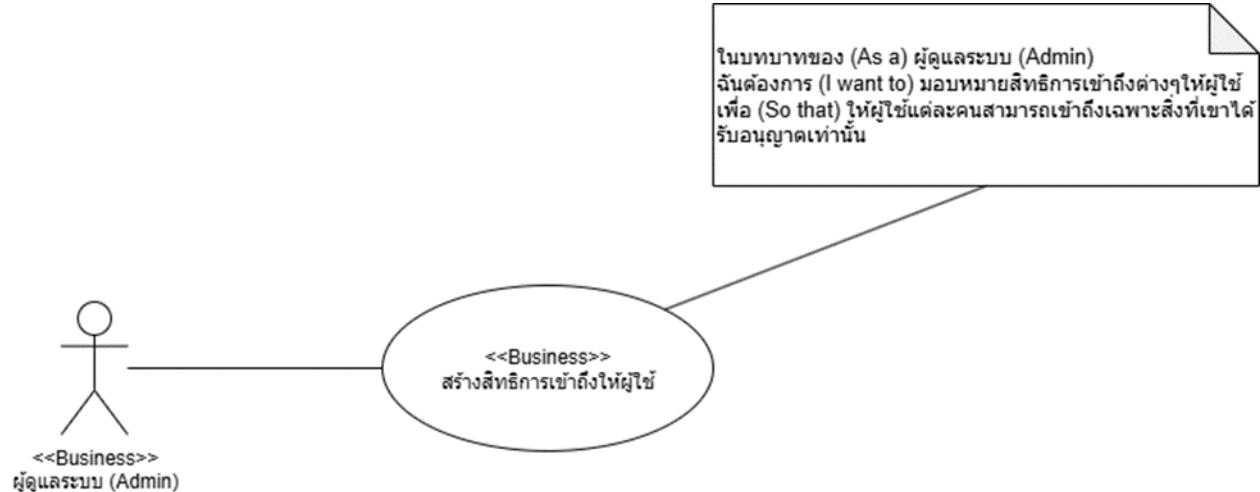
Output ของข้อมูล

- ระบบสร้าง Role ใหม่ พร้อมบันทึกสิทธิที่ Role นั้นมี -> ระบบจะเรียกข้อมูลผู้ใช้มาเพื่อให้แออดมินกำหนด Role ให้ผู้ใช้ -> ระบบจะบันทึกข้อมูลและอัพเดทสิทธิของผู้ใช้ในฐานข้อมูล

คำนามที่อาจจะกล้ายมาเป็นตารางในฐานข้อมูลของระบบ

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

Business Use Case Diagram (แบบเดี่ยว)



Checklist: Business Use Case Diagram

Business Actor มี <<Business>> กำกับ

Business Actor เป็นชื่อบทบาทเท่านั้น ไม่ใช่ชื่อคน ไม่เป็นชื่อบุคคล

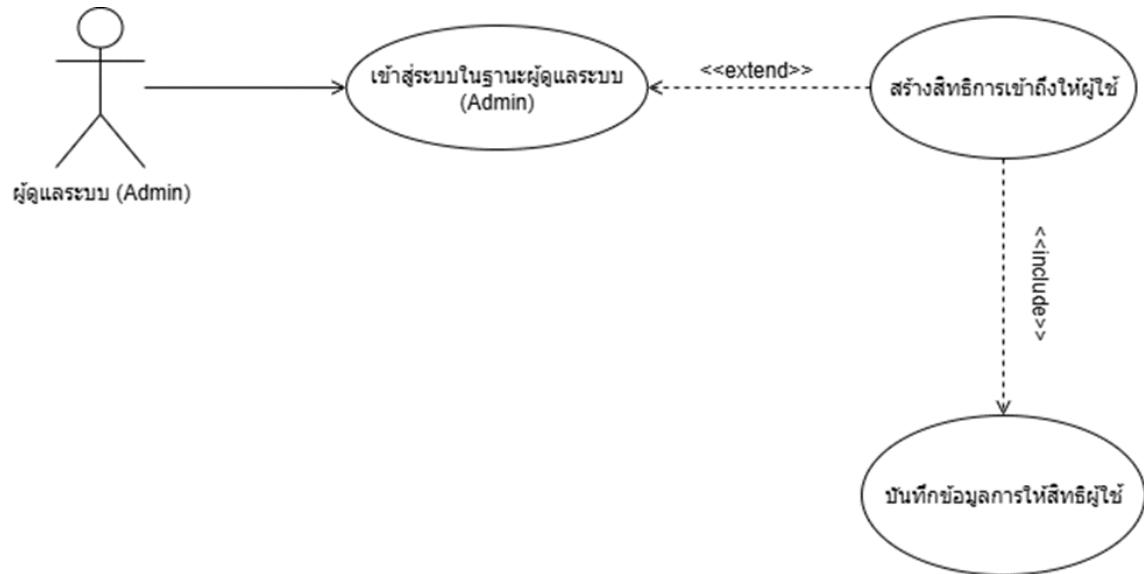
Business Use Case มี <<Business>> กำกับ

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” สามารถกลยุทธ์เป็น กิจกรรมที่เกี่ยวข้องกับระบบ

คอมพิวเตอร์ได้หรือไม่ต้องมีขั้นตอนทาง Security มาเกี่ยวหรือไม่

ตอบ Business Use Case “สร้างสิทธิการเข้าถึงให้ผู้ใช้” สามารถกลยุทธ์เป็น กิจกรรมที่เกี่ยวข้องกับระบบ

คอมพิวเตอร์ได้และต้องการขั้นตอนทาง Security (มี Privacy มาเกี่ยวข้อง)

เพราะฉะนั้น Business Use Case “สร้างสิทธิการเข้าถึงให้ผู้ใช้” จะกลยุทธ์เป็น 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” ทุกรอบ หรือไม่

ตอบ ไม่

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

พิจารณาประเด็นที่ 4

ถ้าสมาชิก “เข้าระบบในฐานะผู้ดูแลระบบ” มาแล้วจำเป็นต้อง “บันทึกข้อมูลการให้สิทธิผู้ใช้” ทุกรอบ หรือไม่
ตอบ ไม่ และเป็น Use Case ที่ไม่เกี่ยวข้องกัน

พิจารณาประเด็นที่ 5

ถ้าสมาชิก “สร้างสิทธิการเข้าถึงให้ผู้ใช้” และ
จำเป็นต้อง “บันทึกข้อมูลการให้สิทธิผู้ใช้” ทุกรอบ หรือไม่

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 จะต้องเป็นเส้นทึบ หัวลูกศรปลายเปิดชี้ไปหา 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 ชี้จาก System Use Case ทางเดียวไปยัง System Use Case หลัก
14. การใช้ <--<<include>>--> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ชี้จาก 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 INTEGE R NOT NULL, PK	USERNAM E VARCHAR NOT NULL, 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@gmai l.co m	SA	68	31-12- 2004
1002	User02	Encrypt(12345 6)	user02@gmai l.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 เช่นแก้ไขเกม หรือลบเกมออกจากร้าน
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

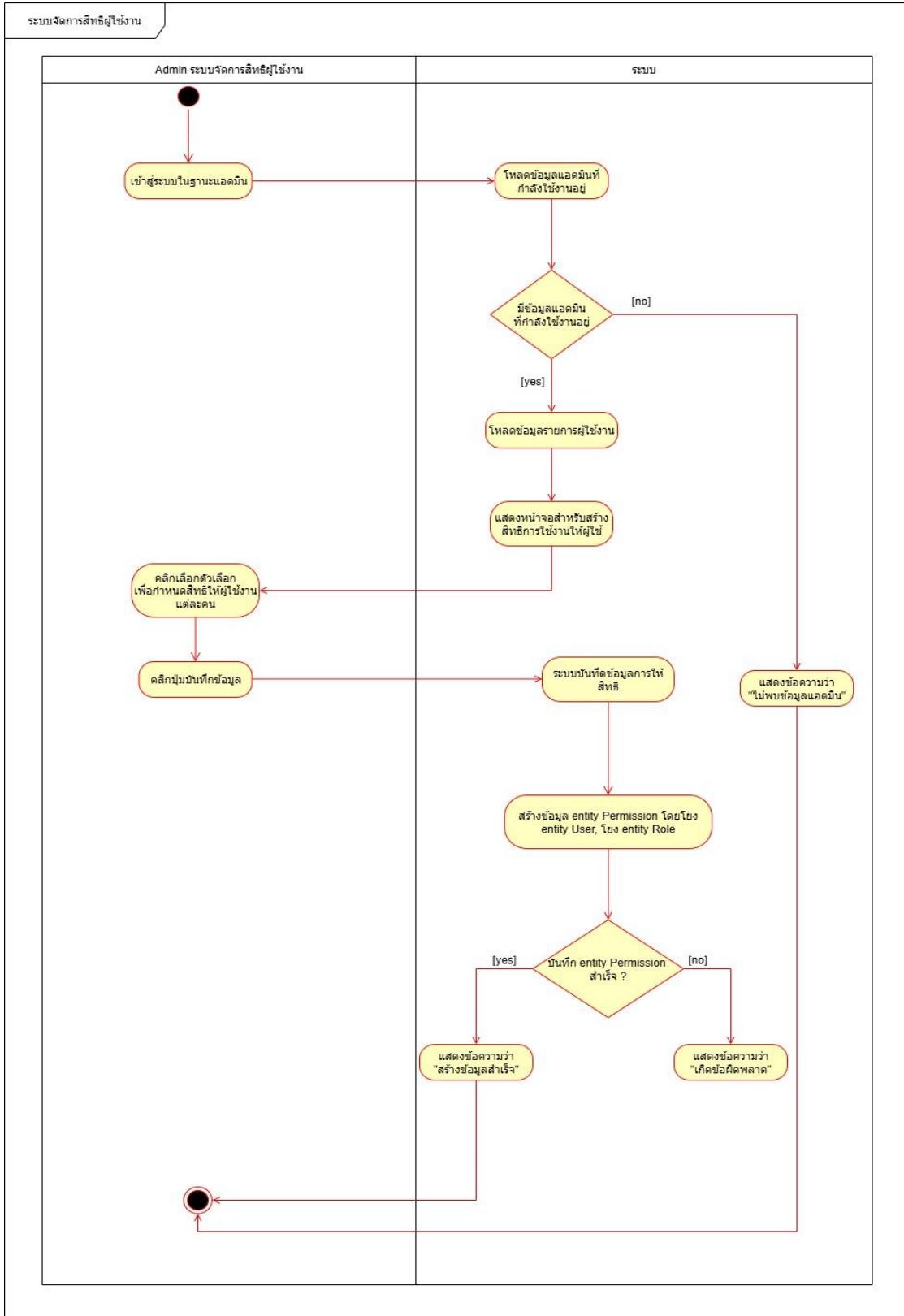
หลักการออกแบบ User Interface

- 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

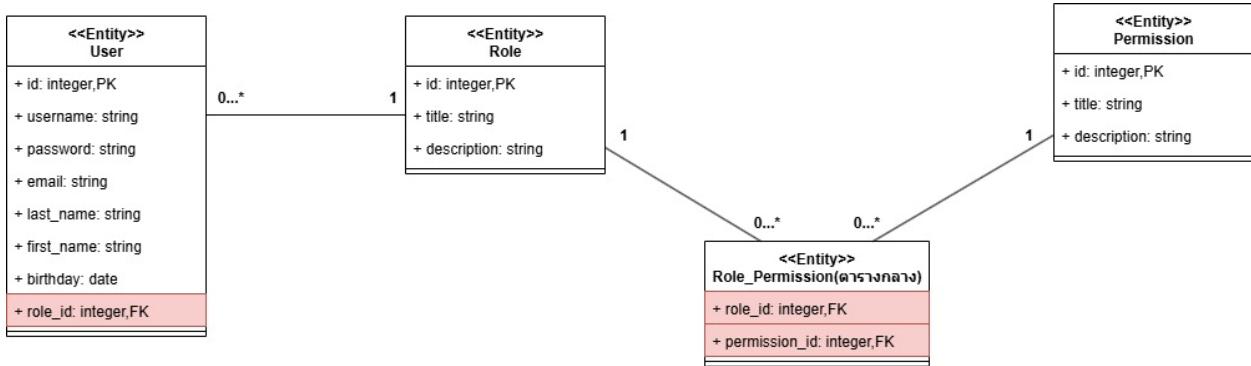
UI Design

The UI design mockup illustrates a user management application interface. On the left, a vertical yellow sidebar contains a user icon, the username "Admin01", and a blue button labeled "บันทึกข้อมูล". The main content area features a title "จัดการสิทธิการเข้าถึง" (Access Control Management) and a search bar. Below the title, there are five data rows, each consisting of a colored user icon (green, cyan, yellow, red, pink), the label "Username", and a dropdown menu labeled "Option 1". At the bottom of the main content area, there is a navigation footer with page numbers 1, 2, 3, ..., and a right arrow.

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", "จัดการสินค้าในร้าน" (Manage products in store), 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 "GAME STORE" application with the "จัดการสินค้าในร้าน" link selected in the sidebar. The main area displays the "แก้ไขบทบาท – ADMIN" (Edit role – Admin) page. It includes tabs for "การแจ้งเตือน", "การอนุญาต", and "จัดการสถานะ". The "ชื่อผู้ดูแล *" field contains "Admin", and the "ค่าอิมบล์เดนเน็ง" field contains "ผู้ดูแลและรับผิดชอบ". The "สี ลักษณะ *" section shows a color palette with several 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("ລົບບໍາຫາທີ່ເຮືອຍແລ້ວ");
};

const handleAddRole = () => {
  const newRole = {
    key: Date.now().toString(),
    role: "ຕຳແໜ່ງໃໝ່",
    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: "ບໍາຫາ",
  dataIndex: "role",
  key: "role",
  render: (text: string, record: any) => (
    <Space>
    {record.icon}
  )
};
```

```
{text}
</Space>
),
},
{
  title: "ສມາຊິກ",
  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) => {
    // 🔒 ຫ້າມລົບ role @everyone
    if (record.key === "everyone") return null;

    const items = [
```

```

{
  key: "delete",
  label: (
    <Popconfirm
      title="ຄູນແນ່ໃຈຫົວໜ້າມີທີ່ຈະລົບບົດບານີ້?"
      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" }}>
        ບົດບານ
      </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

```
import React, { useState } from "react";
import {
  Typography,
  Input,
  Tabs,
  Button,
  Row,
  Col,
  Switch,
  Modal,
```

```
List,  
Checkbox,  
} from "antd";  
  
import { ArrowLeftOutlined, MoreOutlined, PlusOutlined } from "@ant-design/icons";  
import { useParams, useNavigate } from "react-router-dom";  
  
const { Title, Text } = Typography;  
  
const RoleEdit: React.FC = () => {  
  const { id } = useParams();  
  const navigate = useNavigate();  
  
  const [roles, setRoles] = useState<  
    { id: string; name: string; color: string; description: string }[]>[  
    { id: "1", name: "Admin", color: "#1890ff", description: "ຜູ້ຜະລາດບັນຫຼາມ" },  
    { id: "2", name: "Customer", color: "#52c41a", description: "ລູກຄ້າທົ່ວໄປ" },  
  ];  
  
  const [activeRoleId, setActiveRoleId] = useState(id ?? "1");  
  const currentRole = roles.find((r) => r.id === activeRoleId) || roles[0];  
  
  const [roleName, setRoleName] = useState(currentRole.name);  
  const [roleDescription, setRoleDescription] = useState(currentRole.description);  
  const [color, setColor] = useState(currentRole.color);  
  const [activeTab, setActiveTab] = useState("display");  
  
  const [isModalVisible, setIsModalVisible] = useState(false);  
  const [selectedMembers, setSelectedMembers] = useState<number>([]);  
  const [searchText, setSearchText] = useState("");  
  const [permissionSearch, setPermissionSearch] = useState("");  
  
  const mockMembers = [  
    { id: 1, name: "User_A", tag: "user_a#1234" },  
    { id: 2, name: "User_B", tag: "user_b#5678" },
```

```

{ id: 3, name: "User_C", tag: "user_c#9101" },
{ id: 4, name: "User_D", tag: "user_d#1121" },
{ id: 5, name: "User_E", tag: "user_e#3141" },
];

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

const permissionsList = [
  { key: "view_store", label: "ดูร้านค้า", description: "เข้าชมเกมทั้งหมดในร้านค้า" },
  { key: "purchase_game", label: "ซื้อเกม", description: "สามารถซื้อเกมผ่านระบบ" },
  { key: "view_own_library", label: "ดูคลังเกมของตัวเอง", description: "เข้าดูเกมที่ซื้อแล้ว" },
  { key: "download_game", label: "ดาวน์โหลดเกม", description: "ดาวน์โหลดเกมที่ซื้อแล้ว" },
  { key: "write_review", label: "เขียนรีวิว", description: "เขียนรีวิวหรือให้คะแนนเกม" },
  { key: "comment_game", label: "คอมเม้นต์เกม", description: "แสดงความคิดเห็นบนหน้ารายละเอียดเกม" },
  { key: "edit_profile", label: "แก้ไขโปรไฟล์", description: "แก้ไขข้อมูลส่วนตัว เช่น ชื่อ หรือรูปโปรไฟล์" },
  { key: "view_order_history", label: "ดูประวัติคำสั่งซื้อ", description: "ตรวจสอบคำสั่งซื้อที่ผ่านมา" },
  { key: "manage_users", label: "จัดการผู้ใช้", description: "เพิ่ม, ลบ หรือแก้ไขบัญชีผู้ใช้ทั่วไป" },
  { key: "manage_roles", label: "จัดการบทบาท", description: "กำหนดบทบาทและสิทธิ์ของผู้ใช้และแออดมิน" },
  { key: "manage_games", label: "จัดการเกม", description: "เพิ่ม, แก้ไข, หรือลบเกมในร้าน" },
  { key: "manage_categories", label: "จัดการหมวดหมู่เกม", description: "จัดการหมวดหมู่เกมเพื่อเรียงหมวดง่าย" },
  { key: "view_sales_reports", label: "ดูรายงานการขาย", description: "ดูสถิติการขายและรายได้" },
  { key: "process_refunds", label: "คืนเงิน", description: "ดำเนินการคืนเงินให้ลูกค้า" },
  { key: "moderate_reviews", label: "ตรวจสอบรีวิว", description: "ลบหรือแก้ไขรีวิวที่ไม่เหมาะสม" },
  { key: "manage_discounts", label: "จัดการโปรโมชั่น", description: "สร้างหรือแก้ไขโปรโมชั่น/ส่วนลดต่าง ๆ" },
  { key: "view_analytics", label: "ดู Analytics", description: "ดูข้อมูลเชิงลึก เช่น การใช้งานผู้ใช้, ยอดขาย" },
];

```

```

    { key: "manage_orders", label: "จัดการคำสั่งซื้อ", description: "ตรวจสอบและปรับสถานะคำสั่งซื้อ" },
    { key: "send_notifications", label: "ส่งแจ้งเตือน", description: "ส่งข้อความหรืออีเมลแจ้งผู้ใช้" },
    { key: "configure_settings", label: "ตั้งค่าระบบ", description: "ตั้งค่าทั่วไปของร้าน เช่น วิธีชำระเงิน,
    การแสดงผล" },
];

const [permissions, setPermissions] = useState<Record<string, boolean>>(
  Object.fromEntries(permissionsList.map((p) => [p.key, false]))
);

const showModal = () => {
  setIsModalVisible(true);
  setSelectedMembers([]);
  setSearchText("");
};

const handleOk = () => {
  console.log("Adding members:", selectedMembers);
  setIsModalVisible(false);
  setSelectedMembers([]);
};

const handleCancel = () => {
  setIsModalVisible(false);
  setSelectedMembers([]);
};

const handleCheckboxChange = (memberId: number) => {
  setSelectedMembers((prev) =>
    prev.includes(memberId)
      ? prev.filter((id) => id !== memberId)
      : [...prev, memberId]
  );
};

```

```
const filteredMembers = mockMembers.filter(
  (m) =>
    m.name.toLowerCase().includes(searchText.toLowerCase()) ||
    m.tag.toLowerCase().includes(searchText.toLowerCase())
);

const updateRole = () => {
  setRoles((prev) =>
    prev.map((r) =>
      r.id === activeRoleId
        ? { ...r, name: roleName, color, description: roleDescription }
        : r
    )
  );
};

const addRole = () => {
  const newId = (roles.length + 1).toString();
  const newRole = { id: newId, name: "New Role", color: "#a0a0a0", description: "" };
  setRoles([...roles, newRole]);
  setActiveRoleId(newId);
  setRoleName(newRole.name);
  setColor(newRole.color);
  setRoleDescription(newRole.description);
};

return (
  <div style={{ background: "#141414", minHeight: "100vh", display: "flex" }}>
    {/* Role List */}
    <div
      style={{
        width: 220,
        padding: 12,
        borderRight: "1px solid #333",
        height: "100%"
      }>
```

```
        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(-1)}
    style={{ color: "white" }}
>
    ย้อนกลับ
</Button>
<Button
    type="text"
    size="small"
    icon={<PlusOutlined />}
    onClick={addRole}
    style={{ color: "white" }}
/>
</div>

<div style={{ flex: 1, overflowY: "auto" }}>
{roles.map((role) => {
    const isActive = role.id === activeRoleId;
    return (

```

```
<div
  key={role.id}
  onClick={() => {
    setActiveRoleId(role.id);
    setRoleName(role.name);
    setColor(role.color);
    setRoleDescription(role.description);
  }}
  style={{
    display: "flex",
    alignItems: "center",
    padding: "8px 12px",
    marginBottom: 4,
    borderRadius: 6,
    cursor: "pointer",
    color: "white",
    background: isActive ? "#2f3136" : "transparent",
    borderLeft: isActive ? `4px solid ${role.color}` : "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, marginRight: 8 }}>●</span>
  <span>{role.name}</span>
</div>
);
}]}
</div>
</div>
```

```
/* Right Panel */
<div
  style={{
    flex: 1,
    display: "flex",
    flexDirection: "column",
    height: "100vh",
    padding: "16px 32px",
    boxSizing: "border-box",
  }}
>
  /* Header */
  <div
    style={{
      display: "flex",
      marginBottom: 16,
      alignItems: "center",
      gap: 16,
      flexShrink: 0,
    }}
  >
    <Title level={4} style={{ color: "white", margin: 0 }}>
      ແກ້ໄຂບໍທາາຫ – {roleName.toUpperCase()}
    </Title>
    <Button
      type="primary"
      onClick={updateRole}
      style={{ marginLeft: "auto", padding: "0 16px", height: 36 }}
    >
      ບັນທຶກ
    </Button>
  </div>

  /* Tabs */
  <Tabs
```

```
activeKey={activeTab}
onChange={setActiveTab}
items={[
  { key: "display", label: <Text style={{ color: "white" }}>การแสดงผล</Text> },
  { key: "permission", label: <Text style={{ color: "white" }}>การอนุญาต</Text> },
  { key: "members", label: <Text style={{ color: "white" }}>จัดการสมาชิก</Text> },
]}
style={{ flexShrink: 0 }}
/>

/* Tab Content Scrollable */
<div
  style={{
    flex: 1,
    overflowY: "auto",
    marginTop: 16,
    paddingRight: 8,
    scrollbarWidth: "thin",
    scrollbarColor: "#555 #2f3136",
  }}
>
<style>
  `

  div::-webkit-scrollbar {
    width: 8px;
  }

  div::-webkit-scrollbar-track {
    background: #2f3136;
  }

  div::-webkit-scrollbar-thumb {
    background-color: #555;
    border-radius: 4px;
  }
`>
</style>
```

```
/* Display Tab */

{activeTab === "display" && (
  <div>
    <div style={{ marginBottom: 16 }}>
      <Text style={{ color: "white" }}>
        ชื่อตำแหน่ง <Text type="danger">*</Text>
      </Text>
      <Input
        value={roleName}
        onChange={(e) => setRoleName(e.target.value)}
        style={{
          marginTop: 8,
          background: "#2f3136",
          color: "white",
          border: "none",
        }}
      />
    </div>

    <div style={{ marginBottom: 16 }}>
      <Text style={{ color: "white" }}>คำอธิบายตำแหน่ง</Text>
      <Input.TextArea
        value={roleDescription}
        onChange={(e) => setRoleDescription(e.target.value)}
        rows={3}
        style={{
          marginTop: 8,
          background: "#2f3136",
          color: "white",
          border: "none",
        }}
      />
    </div>
  
```

```

<div>
  <Text style={{ color: "white" }}>
    สี ตำแหน่ง <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 white" : "2px solid transparent",
          }}
        />
      </Col>
    )))
  </Row>
</div>
</div>
)}

/* Permission Tab */
{activeTab === "permission" && (
  <div>
    <Input
      placeholder="ค้นหาการอนุญาต"
      value={permissionSearch}
      onChange={(e) => setPermissionSearch(e.target.value)}
      style={{
        marginBottom: 16,
        background: "#2f3136",
      }}
    </Input>
  </div>
)
}

```

```
        color: "white",
        border: "none",
    )}
/>
{permissionsList
    .filter(
        (p) =>
            p.label.toLowerCase().includes(permissionSearch.toLowerCase()) ||
            p.description.toLowerCase().includes(permissionSearch.toLowerCase())
    )
    .map((perm) => (
        <div
            key={perm.key}
            style={{
                display: "flex",
                justifyContent: "space-between",
                alignItems: "center",
                padding: "8px 0",
                borderBottom: "1px solid #333",
            }}
        >
        <div>
            <Text style={{ color: "white", fontWeight: 500 }}>{perm.label}</Text>
            <br />
            <Text style={{ color: "#aaa", fontSize: 12 }}>{perm.description}</Text>
        </div>
        <Switch
            checked={permissions[perm.key]}
            onChange={(checked) =>
                setPermissions((prev) => ({ ...prev, [perm.key]: checked }))
            }
        />
        </div>
    )))
}
</div>
```

```
}

/* Members Tab */

{activeTab === "members" && (
  <div style={{ display: "flex", flexDirection: "column" }}>
    <div
      style={{{
        display: "flex",
        justifyContent: "space-between",
        marginBottom: 16,
      }}}
    >
      <Input
        placeholder="ค้นหาสมาชิก"
        style={{ width: "70%", background: "#2f3136", color: "white" }}
        value={searchText}
        onChange={(e) => setSearchText(e.target.value)}
      />
      <Button type="primary" onClick={showModal}>
        เพิ่มสมาชิก
      </Button>
    </div>

    <div
      style={{{
        flex: 1,
        display: "flex",
        justifyContent: "center",
        alignItems: "center",
        color: "#aaa",
      }}}
    >
      <div style={{ textAlign: "center" }}>
        <div style={{ fontSize: 48, marginBottom: 16 }}>
          <svg
```

```
    xmlns="http://www.w3.org/2000/svg"
    fill="currentColor"
    viewBox="0 0 24 24"
    width="48"
    height="48"
  >
  <path d="M12 12c2.21 0 4-1.79 4-4s-1.79-4-4-4 1.79-4 4 1.79 4 4 4zm0 2c-2.67 0-8
1.34-8 4v2h16v-2c0-2.66-5.33-4-8-4z" />
</svg>
</div>
<div>
  <div>
    <div style="background-color: #f0f0f0; padding: 10px; border-radius: 5px; margin-bottom: 10px;">
      <img alt="Search icon" style="width: 24px; height: 24px; vertical-align: middle; margin-right: 10px;"/>
      <span>ຄົນຫາສາຂົກ</span>
    </div>
    <div style="background-color: #f0f0f0; padding: 10px; border-radius: 5px; margin-bottom: 10px;">
      <img alt="Search icon" style="width: 24px; height: 24px; vertical-align: middle; margin-right: 10px;"/>
      <span>ເພີ່ມສາຂົກໃຫ້ກັບທະບາຖິ່ນ</span>
    </div>
    <div style="background-color: #f0f0f0; padding: 10px; border-radius: 5px; margin-bottom: 10px;">
      <img alt="Search icon" style="width: 24px; height: 24px; vertical-align: middle; margin-right: 10px;"/>
      <span>ລົງທະບຽນ</span>
    </div>
    <div style="background-color: #f0f0f0; padding: 10px; border-radius: 5px; margin-bottom: 10px;">
      <img alt="Search icon" style="width: 24px; height: 24px; vertical-align: middle; margin-right: 10px;"/>
      <span>ລົງທະບຽນ</span>
    </div>
    <div style="background-color: #f0f0f0; padding: 10px; border-radius: 5px; margin-bottom: 10px;">
      <img alt="Search icon" style="width: 24px; height: 24px; vertical-align: middle; margin-right: 10px;"/>
      <span>ລົງທະບຽນ</span>
    </div>
  </div>
  <div style="text-align: right; margin-top: 10px;">
    <button style="border: none; background-color: transparent; font-size: 1em; margin-right: 10px;">ຍົກເລີກ</button>
    <button style="border: none; background-color: transparent; font-size: 1em; border: 1px solid #ccc; padding: 0 10px;">ອີເມວ</button>
  </div>
</div>
</div>

/* Modal */
<Modal
  title={<span style={{ color: "black" }}>ເພີ່ມສາຂົກ</span>}
  open={isModalVisible}
  onClose={handleClose}
  onOk={handleOk}
  onCancel={handleCancel}
  okText="ເພີ່ມ"
  cancelText="ຍົກເລີກ"
>
<Input.Search
  placeholder="ຄົນຫາສາຂົກ"

```

```

        onChange={(e) => setSearchText(e.target.value)}
        style={{ marginBottom: 16 }}
      />
      <div style={{ maxHeight: 300, overflowY: "auto" }}>
        <List
          dataSource={filteredMembers}
          renderItem={(member) => (
            <List.Item
              key={member.id}
              style={{
                backgroundColor: selectedMembers.includes(member.id)
                  ? "#f0f0f0"
                  : "transparent",
                borderRadius: 4,
                cursor: "pointer",
              }}
              onClick={() => handleCheckboxChange(member.id)}
            >
              <Checkbox
                checked={selectedMembers.includes(member.id)}
                onChange={() => handleCheckboxChange(member.id)}
                style={{ marginRight: 8 }}
              />
              <span>{member.name}</span>
              <span style={{ marginLeft: 8, color: "#999" }}>{member.tag}</span>
            </List.Item>
          )}
        />
      </div>
    </Modal>
  </div>
);

};

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 ในการวาด

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 ในที่นี่จะตั้งชื่อเป็น CreatePermissionUI

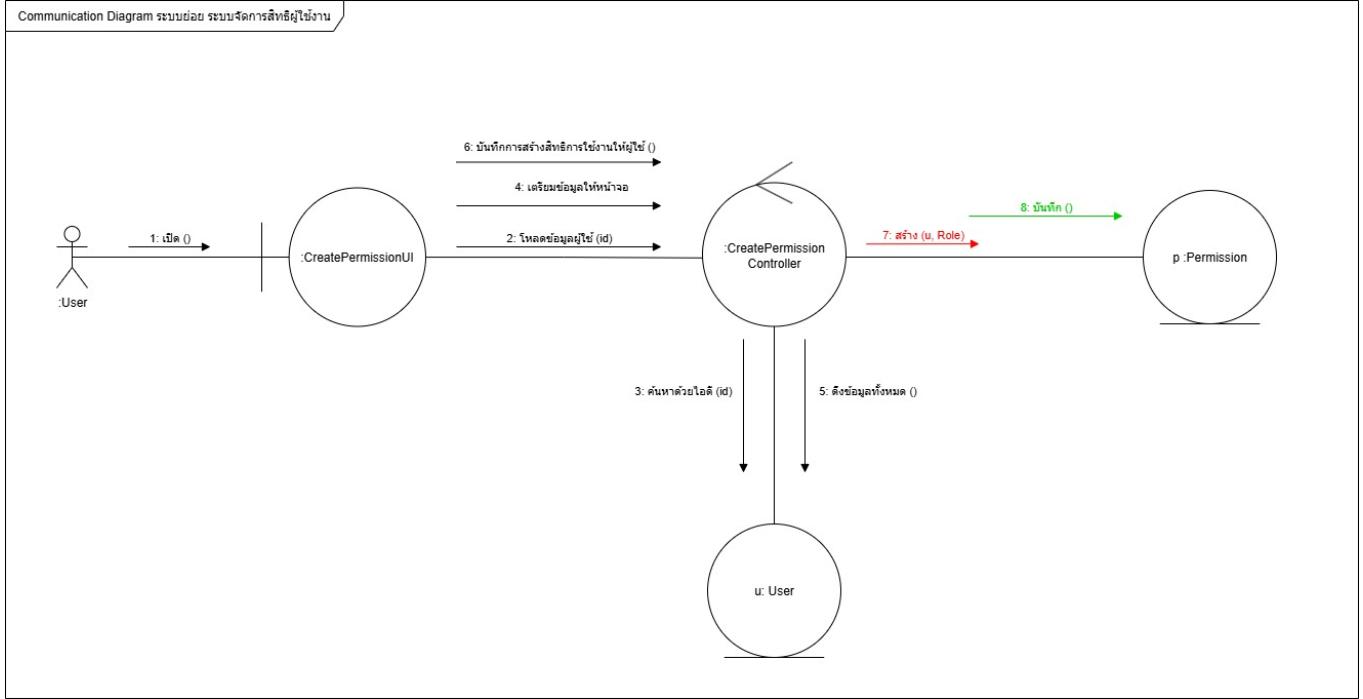
วัตถุของ Boundary Class จะส่งข้อมูลที่จำเป็นให้กับวัตถุของ Control Class

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

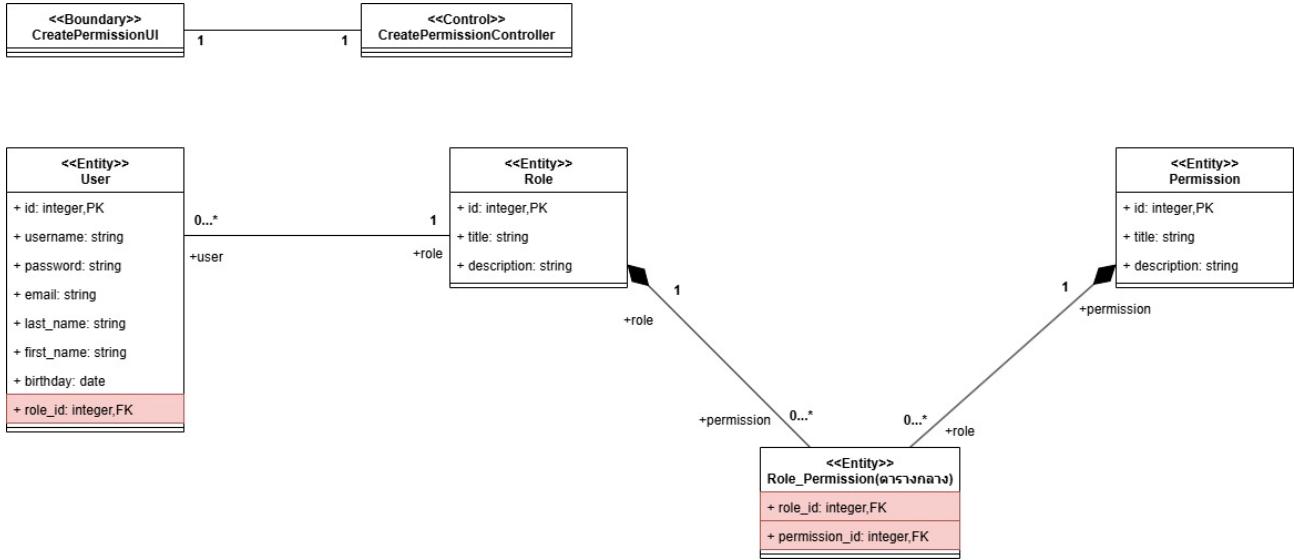
เตรียมแปลงแต่ละ Activity ของ System Activity Diagram ให้เป็น Communication Diagram โดยวิเคราะห์ให้วัตถุที่เกี่ยวข้องกับการทำงานในแต่ละขั้นตอน

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 เสิร์ฟของเกม 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 พวกเขาสามารถกลับมา 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) เผยแพร่มอดเกมที่ฉันสร้างให้ผู้ใช้งานคนอื่นๆได้เล่นได้

Output บนหน้าจอ

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

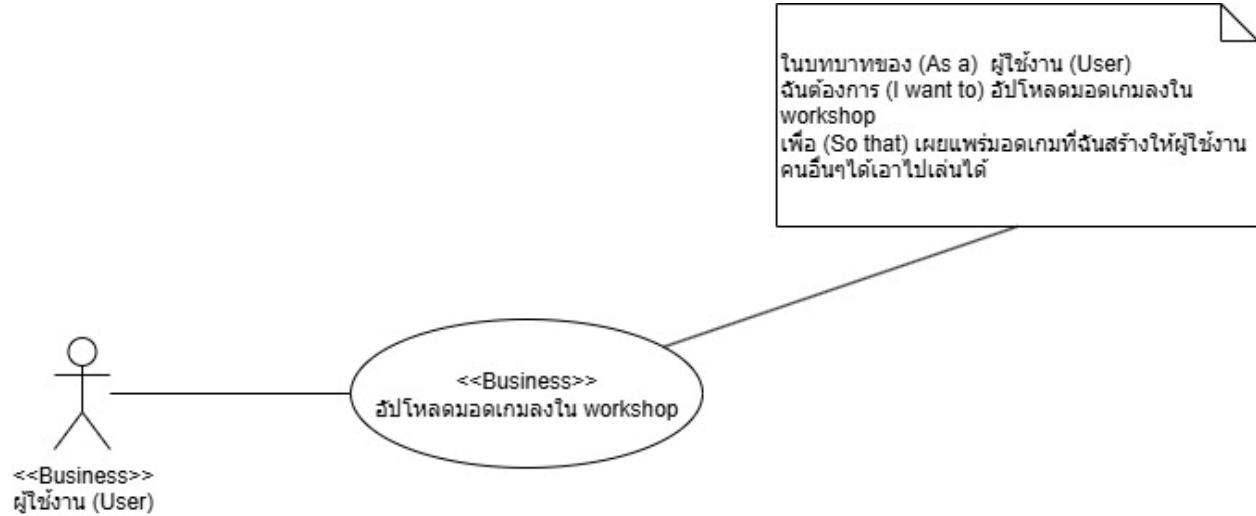
Output ของข้อมูล

- ระบบจะอัปโหลดมอดเกมลงใน workshop -> ระบบจะบันทึกข้อมูลของมอดเกมที่เราอัปโหลดลงในฐานข้อมูล

คำนำที่อาจจะกล้ายมาเป็นตารางในฐานข้อมูลของระบบ

คำนำ	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
ผู้ใช้งาน (User)	เกี่ยวข้องโดยตรง เนื่องจากเป็นสิ่งที่แอดมินต้องสร้าง, อัปเดต หรือ ลบ
เกม	เกี่ยวข้องโดยตรง เนื่องจากต้องอัปโหลดมอดูลของเกมนั้นๆ
Mod	เกี่ยวข้องโดยตรง เนื่องจากเป็นสิ่งที่จะอัปโหลดลง workshop
เกมที่ตนเองเป็นเจ้าของ (Owner_game)	เกี่ยวข้องโดยตรง เนื่องจากก่อนจะอัปโหลดมอดของเกมนั้นๆ ได้ 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

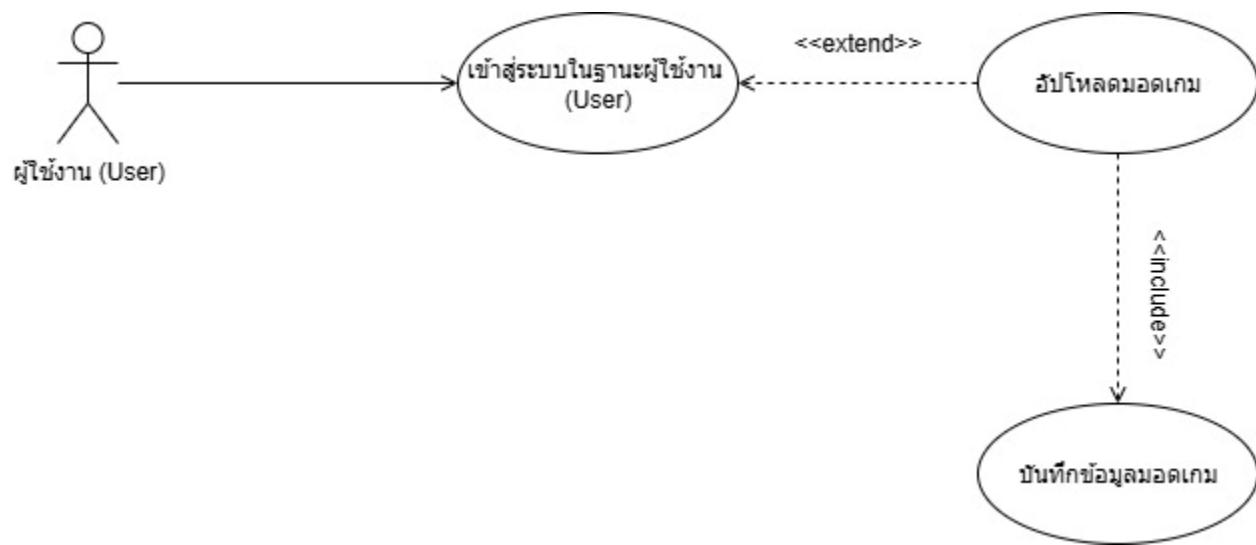
Business Use Case มี <<Business>> กำกับ

Business Use Case ขึ้นต้นด้วยคำที่แสดงพฤติกรรม “คำกริยา”

พฤติกรรมของ Business Use Case ต้องมาจากการ Story

มี Note ที่แสดง User Story อุปกรณ์ใน 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” ทุกรึงหรือไม่

ตอบ ไม่

แปลว่า System Use Case “อัปโหลดมอดเกม” เป็นกิจกรรมทางเลือกหลังจากทำ System Use Case

“เข้าระบบในฐานะผู้ใช้งาน (User)”

พิจารณาประเด็นที่ 4

ถ้าสามารถ “เข้าระบบในฐานะผู้ใช้งาน” มาแล้วจำเป็นต้อง “บันทึกข้อมูลมอดเกม” ทุกครั้งหรือไม่
ตอบ ไม่ และเป็น Use Case ที่ไม่เกี่ยวข้องกัน

พิจารณาประเด็นที่ 5

ถ้าสามารถ “อัปโหลดมอดเกม” และ^{จะ}
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>” เพ่านั้น
- 6 การใช้ <--<<extend>>--> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ทางเดียวไปยัง System Use Case หลัก

7 การใช้ <--<<include>>--> ต้องเป็นสันประ 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 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, 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.l.c om	SA	68	10-08-2000
1002	User02	Encrypt(1234 56)	demo02@gmai.l.c 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, PK	PURCHASE_DATE DATE NOT NULL	STATUS VARCHAR NOT NULL	USER_ID DATE NOT NULL, FK	GAME_ID INTEGER NOT NULL, FK
3001	10-10-2568	เป็นเจ้าของเกม	1001	2001

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
 - Game_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, PK	TITLE VARCH AR	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, FK	GAME_ ID INTEGE R NOT NULL, FK
4001	funfun	เป็นมอดเกม ที่ให้ผู้เล่นได้ เล่นค่าน ใหม่ๆ	20-10-2568	C:\User\acer\Docu ments	Publis ed	1001	2001

MOD_ID INTEGER NOT NULL, FK	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

คะແນນ

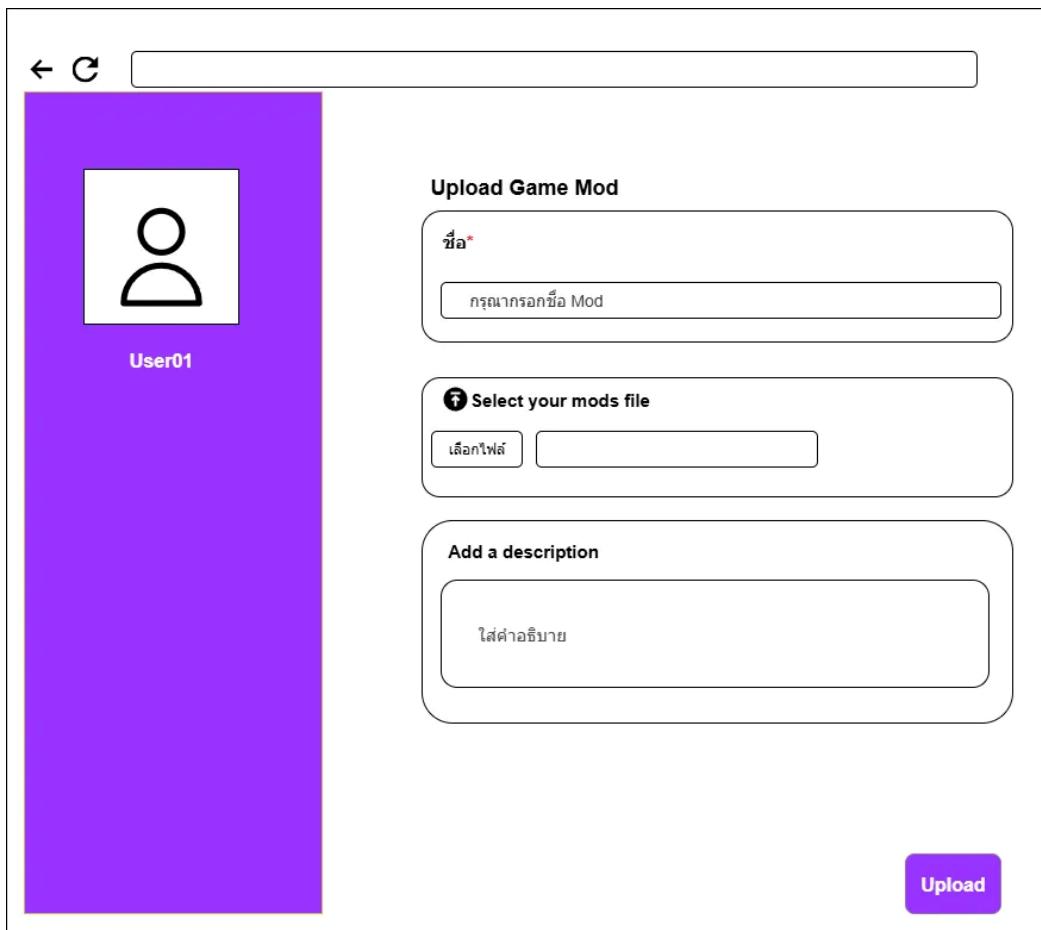
- ชื่อ Entity: ModRating
- ข้อมูลที่ต้องจัดเก็บ
 - ID: เป็น Primary Key เก็บในรูปแบบ integer ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
 - rated_date: เก็บในรูปแบบ date ไม่สามารถเป็นค่า Null ได้
 - review: เก็บในรูปแบบ varchar สามารถเป็นค่า Null ได้
 - rating: เก็บในรูปแบบ integer สามารถเป็นค่า 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) ซึ่งจากตารางหลักกลับไปยังตารางสนับสนุน ให้ใช้ ComboBox เป็นตัวเลือกเพื่อเชื่อมโยงข้อมูลใน User Interface
- สำหรับ Field ประเภทอื่นๆ ให้สร้างตามประเภทของข้อมูล เช่น
 - Textbox สำหรับข้อความทั่วไป
 - Password สำหรับข้อมูลรหัสผ่าน
 - Datetime Picker สำหรับเลือกวันและเวลา
 - Numeric Input สำหรับป้อนตัวเลข

UI Design



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

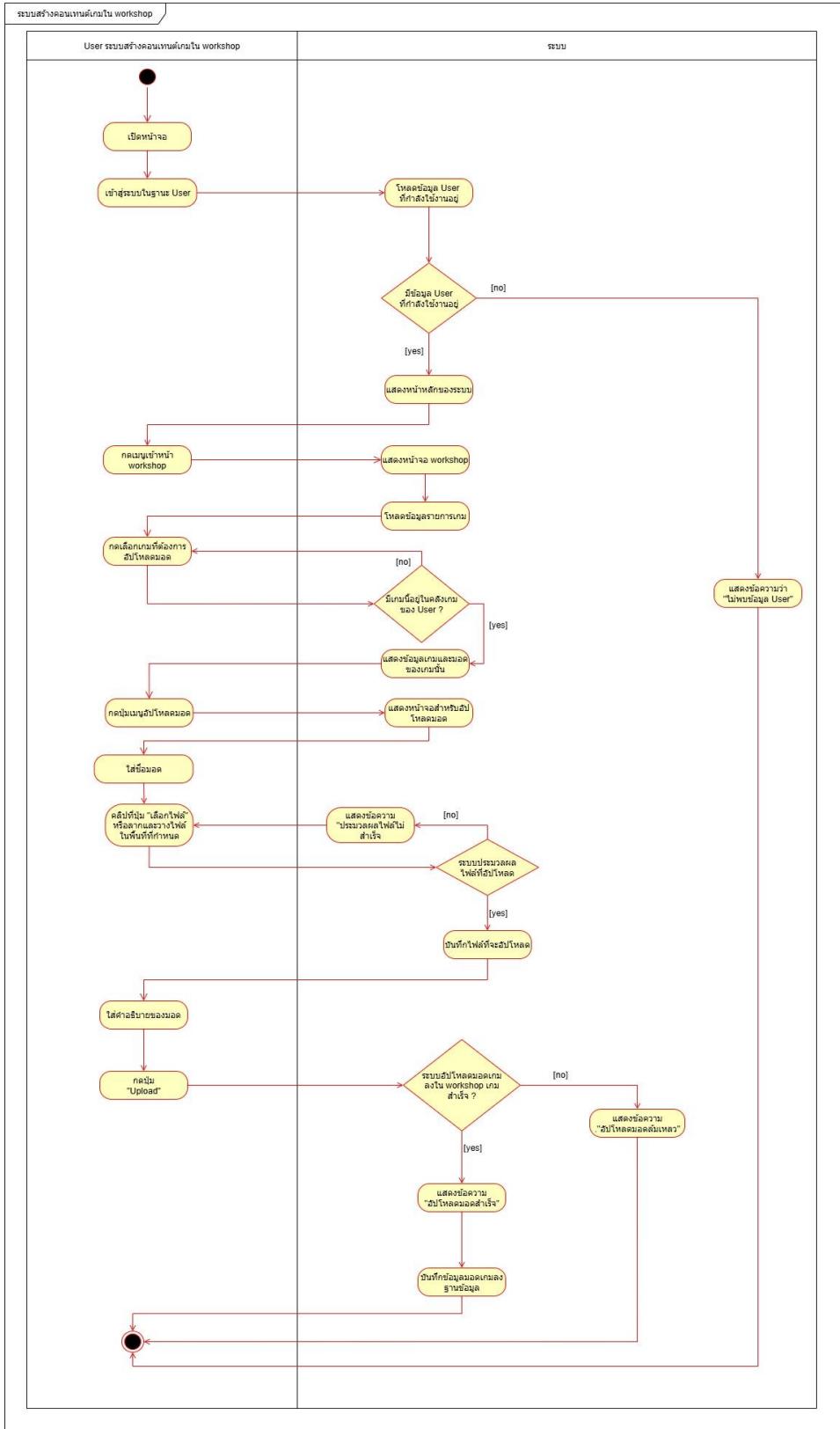
หลักสำคัญ ของการออกแบบ System Activity Diagram ได้แก่

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

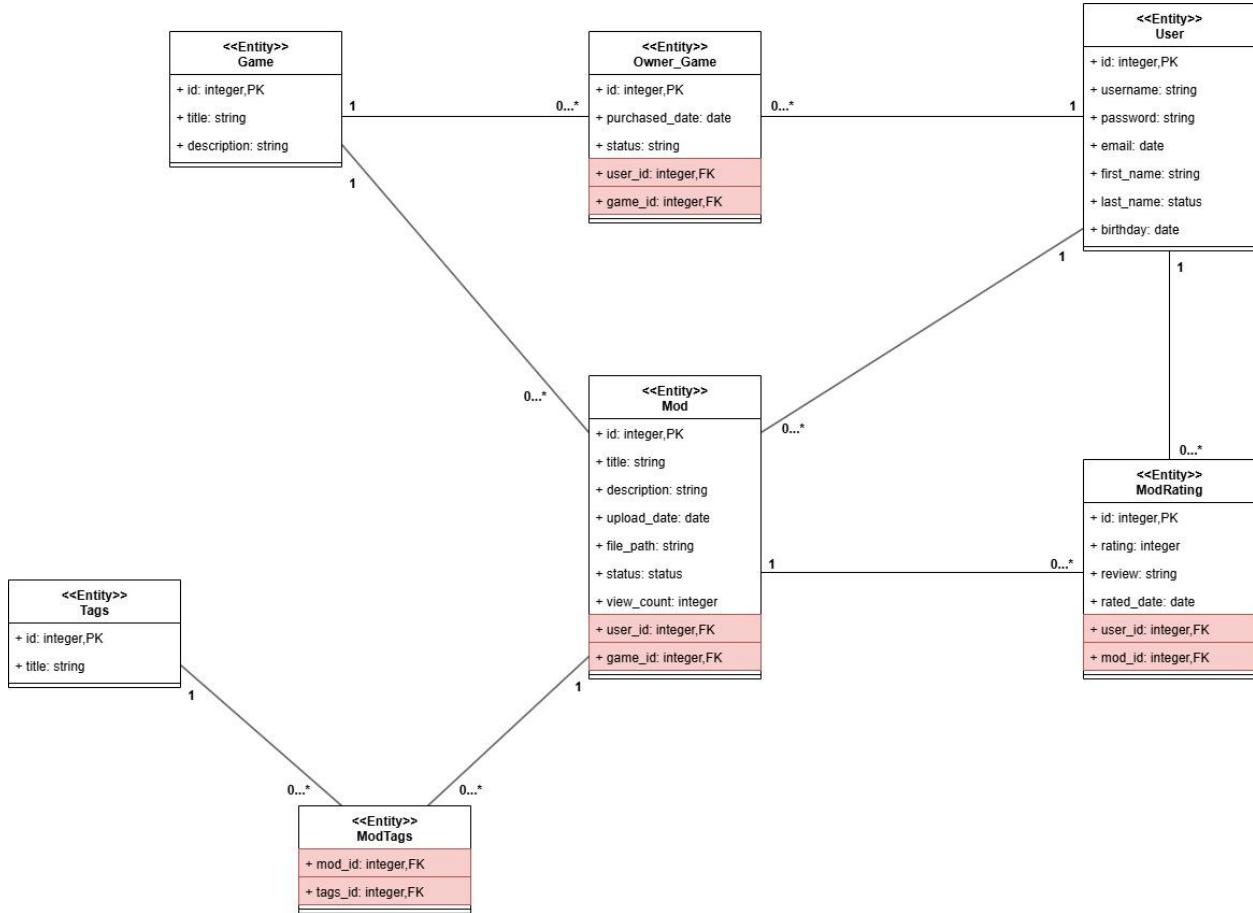
System Activity Diagram ต้องสื่นสุดที่การสร้าง Output ดังนี้

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

Activity Diagram

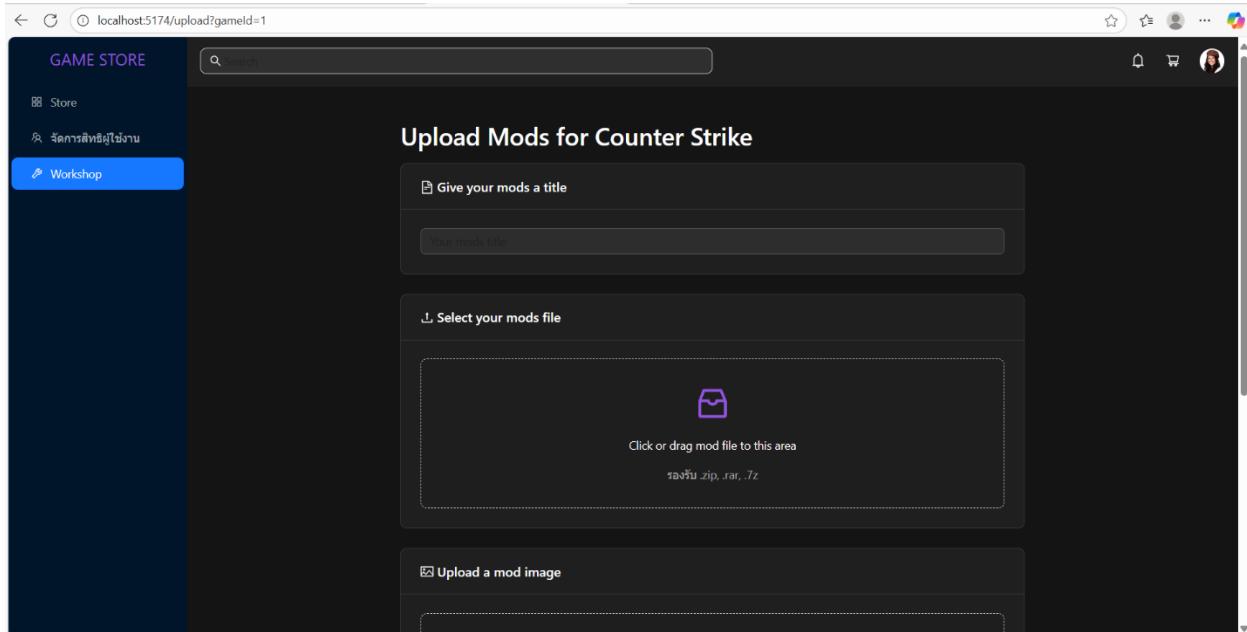


Class Diagram at Analysis Level

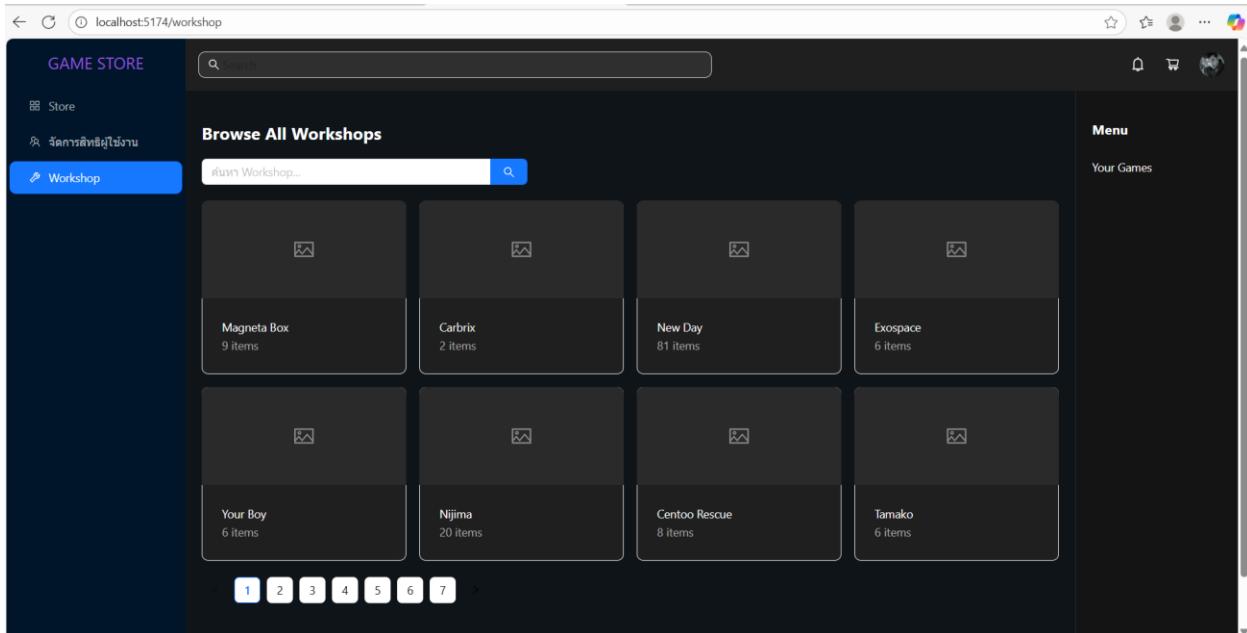


UI

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 page has a dark theme with a sidebar on the left labeled "GAME STORE" containing "Store", "จัดการสิ่งที่คุณชื่นชอบ", and "Workshop". The main content area features a large image of a game map titled "Weapons Course". Below the image, the mod title "Weapons Course" is displayed with a "Download" button. A placeholder text message "This is a placeholder description for the mod. In the future, you can load real mod details from the backend." is shown. A "Comments" section with a comment input field and "Add Comment" button follows. To the right, a pink sidebar displays "Creator: Asmisint", "Uploaded: 2025-09-05", "Unique Visitors" (0), "Current Downloads" (0), and a "Rate this Mod" section with a 4/5 rating and a "Your Rating: 4 / 5" input field.

SourceCode

WorkshopUpload Page

```
import React, { useState } from "react";
import { Typography, Card, Input, Button, Upload, message } from "antd";
import Navbar from "./components/Navbar";
import { UploadOutlined, FileTextOutlined, PictureOutlined, InboxOutlined } from "@ant-design/icons";
import { useSearchParams } from "react-router-dom";

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

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

// mock ข้อมูลเกมที่ user มี
const userGames: WorkshopItem[] = [
  { id: "1", title: "Counter Strike", items: 6 },
  { id: "2", title: "Half-Life", items: 3 },
];

const Workshop = () => {
  const [searchParams] = useSearchParams();
  const gameId = searchParams.get("gameId");
  const game = userGames.find((g) => g.id === gameId);

  // State
  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>("");  
  
// File handlers  
const handleModFile = (file: File) => {  
    setModFile(file);  
    message.success(` ${file.name} selected`);  
    return false; // prevent auto upload  
};  
  
const handleModImage = (file: File) => {  
    setModImage(file);  
    setImagePreview(URL.createObjectURL(file));  
    message.success(` ${file.name} selected`);  
    return false; // prevent auto upload  
};  
  
const handleUpload = () => {  
    if (!modTitle || !modFile) {  
        message.error("กรุณาใส่ชื่อเม็ดและเลือกไฟล์เม็ดก่อนอัปโหลด");  
        return;  
    }  
    console.log("Title:", modTitle);  
    console.log("Description:", modDescription);  
    console.log("File:", modFile);  
    console.log("Image:", modImage);  
  
    message.success("อัปโหลดเรียบร้อยแล้ว!");  
    setModTitle("");  
    setModDescription("");  
    setModFile(null);  
    setModImage(null);  
    setImagePreview("");  
};  
  
return (
```

```

<div style={{ background: "#141414", minHeight: "100vh" }}>

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

    {/* ชื่อโมด */}
    <Card
      title={
        <span style={{ color: "white" }}>
          <FileTextOutlined /> Give your mods a title
        </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>

    {/* เลือกไฟล์มืด */}
    <Card
      title={
        <span style={{ color: "white" }}>
          <UploadOutlined /> Select your mods file
        </span>
      }
      style={{ background: "#1f1f1f", marginBottom: "24px", borderRadius: 8, borderColor: "#404040" }}
      headStyle={{ color: "white", borderBottomColor: "#404040" }}
    >

```

```

<Dragger beforeUpload={handleModFile} showUploadList={false}>
  <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" }}>ຮອງຮັບ .zip, .rar, .7z</p>
</Dragger>
{modFile && <p style={{ color: "white", marginTop: 8 }}>Selected: {modFile.name}</p>}
</Card>

/* ເລືອກຮູ່ມືອດ */
<Card
  title={
    <span style={{ color: "white" }}>
      <PictureOutlined /> Upload a mod image
    </span>
  }
  style={{ background: "#1f1f1f", marginBottom: "24px", borderRadius: 8, borderColor: "#404040" }}
  headStyle={{ color: "white", borderBottomColor: "#404040" }}
>
<Dragger beforeUpload={handleModImage} showUploadList={false} accept="image/*">
  <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>

```

```
/* คำอธิบาย */
<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>

/* ปุ่มอัปโหลด */
<div style={{ textAlign: "right" }}>
  <Button
    type="primary"
    size="large"
    style={{ background: "#9254de", borderColor: "#9254de" }}
    onClick={handleUpload}
  >
    Upload
  </Button>
</div>
</div>
</div>
);
};
```

```
export default Workshop;
```

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>
    }
  </List>
</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 ไม่สามารถอัปโหลดมืดได้");
    }
  }
}

```

```
};

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={{ color: "white", text-align: "center", height: 200 }}>
          <h1>Workshop</h1>
          <p>A place to learn and share knowledge</p>
        </div>
      )}
    
```

```
/>
) : (
<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",
    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 || "No description available."}
    </Paragraph>
  </div>
</Content>
```

```
"This is a placeholder description for the mod. In the future, you can load real mod details from the backend."}
```

```
</Paragraph>
```

```
/* 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>  
    <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 />
<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": "owner game 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 ในการวาด

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 ในที่นี่จะตั้งชื่อ เป็น UploadModUI

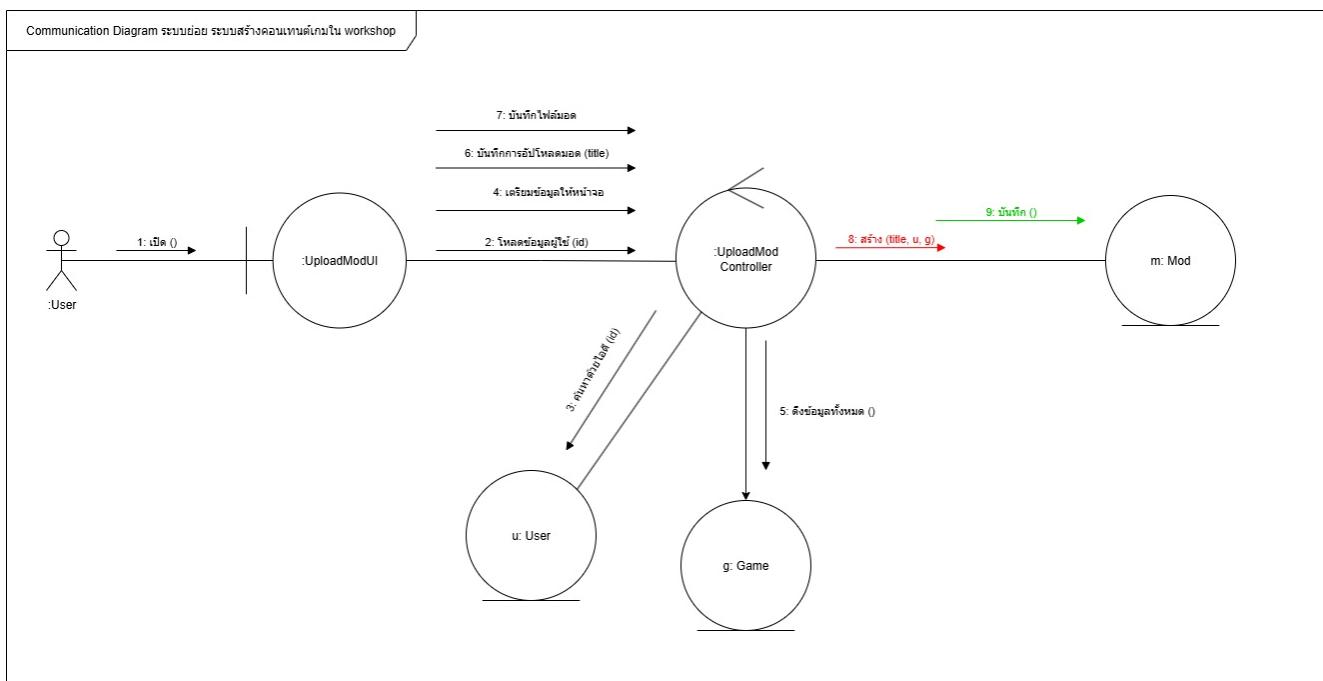
วัตถุของ Boundary Class จะส่งข้อมูลที่จำเป็นให้กับวัตถุของ Control Class

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

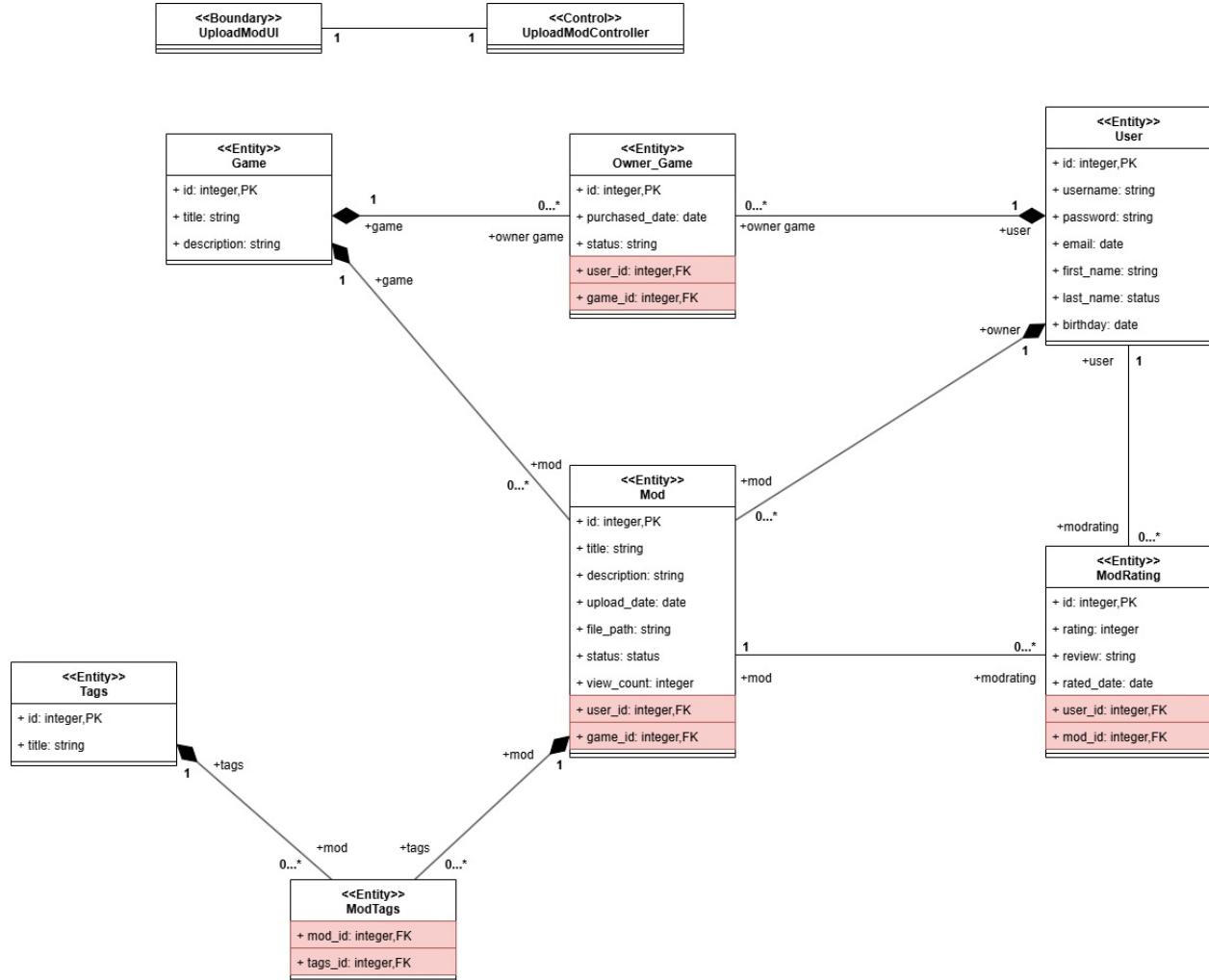
เตรียมแปลงแต่ละ Activity ของ System Activity Diagram ให้เป็น Communication Diagram โดยวิเคราะห์ให้วัตถุที่เกี่ยวข้องกับการทำงานในแต่ละขั้นตอน

Activity	เป็นคำสั่งสำหรับระบบได้ หรือไม่	ถ้าเป็น, วัตถุที่รับหน้าที่ ทำงานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click “เปิดหน้าจอ”	เป็นคำสั่ง สั่งงานเพื่อให้หน้า UI เปิด	: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 (ได้ title)	ไม่เป็นคำสั่ง	-	-
Click ปุ่ม “เลือกไฟล์ หรือลากและวางไฟล์ ในพื้นที่กำหนด”	ไม่เป็นคำสั่ง	-	-

บันทึกไฟล์ที่จะอัปโหลด	เป็นคำสั่ง ระบบจะทำการจัดเก็บข้อมูลไฟล์มود	:UploadModController	บันทึกไฟล์มود
ใส่คำอธิบายของมود	ไม่เป็นคำสั่ง	-	-
Click ปุ่ม “Upload”	เป็นคำสั่ง ระบบจะทำการจัดเก็บข้อมูลมอดที่อัปโหลด	:UploadModController	บันทึกการอัปโหลดมอด (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 “อัปโหลดมอดสำเร็จ”	ไม่เป็นคำสั่ง	-	-



Class Diagram at Design Level



B6618643 นายกิตตินันท์ ปัจจัยโคงา

ระบบหลัก ระบบร้านขายเกมออนไลน์

ระบบย่อย ระบบชำระเงิน

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

เมื่อผู้ใช้งานทำการซื้อเกมสำเร็จก็จะสามารถเขียนรีวิวให้กับเกมรวมไปถึงการเข้าระบบคอมมูนิตี้ของเกมที่ได้ทำการสั่งซื้อไว้ด้วย และหากผู้ใช้งานเล่นเกมแล้วเกิดไม่ชอบก็จะสามารถเรียกคืนเงินได้ผ่านระบบการจัดการคืนเงิน

User Story ระบบการชำระเงิน

ในบทบาทของ (As a) ผู้ใช้งาน

ฉันต้องการ (I want to) ชำระเงิน

เพื่อ (So that) ที่ฉันจะได้เกมที่ต้องการเล่นได้

Output บนหน้าจอ

- ผู้ใช้งานเลือกเกมที่ต้องการ ไปไว้ในระบบของการชำระเงิน จากนั้นระบบจะทำการคำนวณเงินเพื่อให้เห็นจำนวนเงินที่ต้องชำระและแสดงคิวอาร์โคเด้ก์ให้สแกนจ่าย

Output ของข้อมูล

- ระบบจะทำการเรียกข้อมูลราคาของเกมที่เพิ่มเข้ามาพร้อมกับส่วนลดที่ได้ทำการใส่เข้ามา และเมื่อทำการชำระเงินเสร็จระบบจะทำการบันทึกคำสั่งซื้อพร้อมกับวันที่และเวลาของการสั่งซื้อ และจะทำการเออดผู้ใช้งานเข้าไปในระบบคอมมูนิตี้ พร้อมกับทำให้ผู้ใช้งานสามารถรีวิวเกมที่ซื้อได้

ในบทบาทของ (As a) ผู้ดูแลระบบ

ฉันต้องการ (I want to) ตรวจสอบการชำระเงิน

เพื่อ (So that) ทำการยืนยันการชำระเงินและส่งคีย์เกมให้กับผู้ใช้งาน

Output บนหน้าจอ

- สามารถกดตรวจสอบคำสั่งซื้อของลูกค้าว่าการชำระเงินนั้นทำการชำระถูกต้องหรือไม่ หากถูกต้องหรือไม่ หากถูกต้องก็จะสามารถใส่สถานะให้กับคำสั่งซื้อนั้นได้

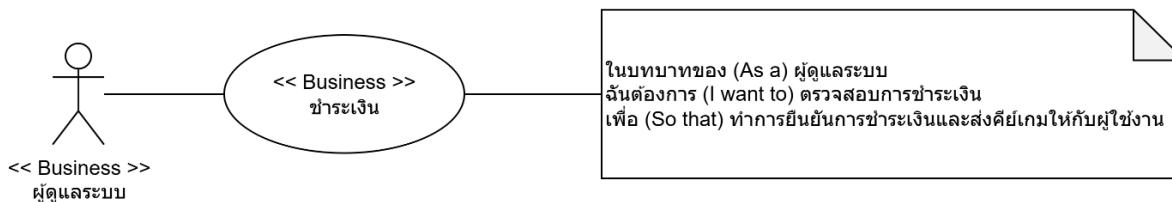
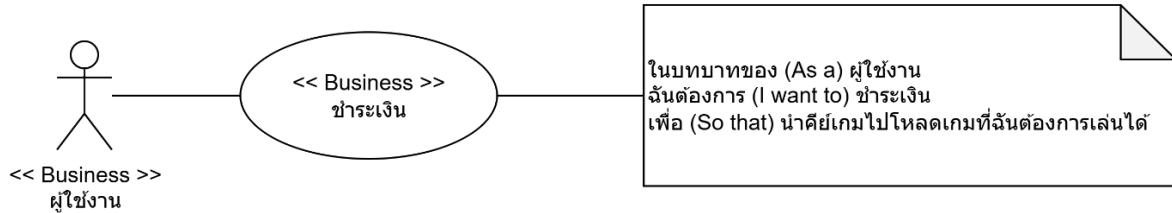
Output ของข้อมูล

- ในฐานข้อมูลจะมีในส่วนของการสร้างและแก้ไขสถานะการชำระเงินผ่านสถานะ (รอตรวจสอบ / สำเร็จ / ไม่สำเร็จ) พร้อมกับการเพิ่มคีย์ให้กับฐานข้อมูลของคำสั่งซื้อนั้น

คำนำที่อาจจะกลایมานเป็นตารางในฐานข้อมูลของระบบ

คำนำ	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
ผู้ใช้งาน	เกี่ยวข้องโดยตรง เนื่องจากผู้ใช้งานจะเป็นคนสร้างคำสั่งซื้อขึ้นมา
ส่วนลด	ไม่เกี่ยวข้องโดยตรง
สลิป	เกี่ยวข้องโดยตรง เนื่องจากต้องต้องใช้ยืนยันในการชำระเงิน
รหัสคำสั่งซื้อ	เกี่ยวข้องโดยตรง เนื่องจากการหักคำสั่งซื้อจะเกิดขึ้นจากระบบการชำระเงิน
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 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)” 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” และ จำเป็นต้อง “กรอกข้อมูลช่องทางการชำระเงิน” ทุกครั้งหรือไม่?

ตอบ ใช่ จำเป็น (ต้องกรอกข้อมูลช่องทางชำระเงินเสมอ)

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

พิจารณาประเด็นที่ 4

ถ้าสมาชิก “กรอกโค้ดโปรโมชั่น” และ จำเป็นต้อง “ทำการชำระเงิน” ต่อหรือไม่?

ตอบ ไม่จำเป็น (ผู้ใช้อาจยกเลิก หรือคลิปไปแก้ไขต่อได้)

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

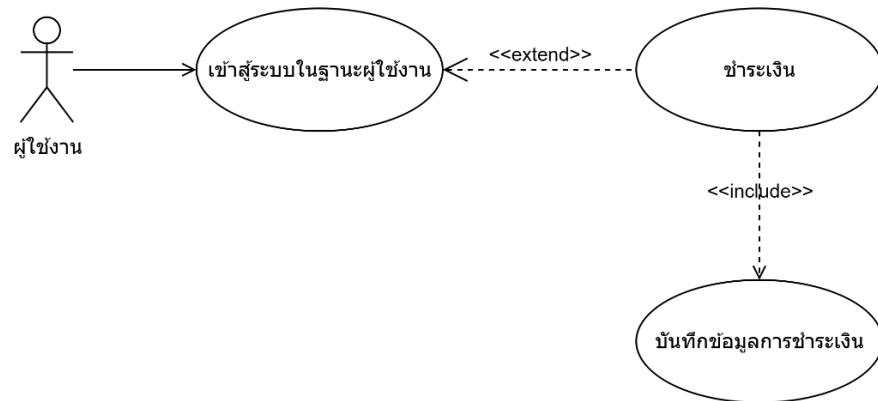
พิจารณาประเด็นที่ 5

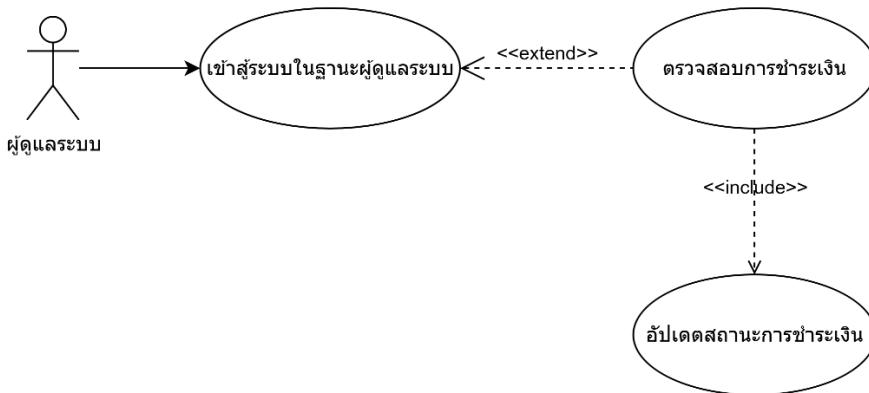
ถ้าสมาชิก “ทำการชำระเงิน” แล้ว จะเป็นต้อง “อัปเดตสถานะคำสั่งซื้อ” ทุกครั้งหรือไม่?

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 ต้องไม่มีอะไร重複 กับ
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 “เข้าระบบในฐาน <Actor>” เท่านั้น
20. การใช้ <--<<extend>>--> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ทางเลือกไปยัง System Use Case หลัก
21. การใช้ <--<<include>>--> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ที่ใหญ่กว่า ไปยัง System Use Case ที่จะถูกรวบเข้ามา

วิเคราะห์ Entity ที่เกี่ยวข้อง

ชื่อ 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

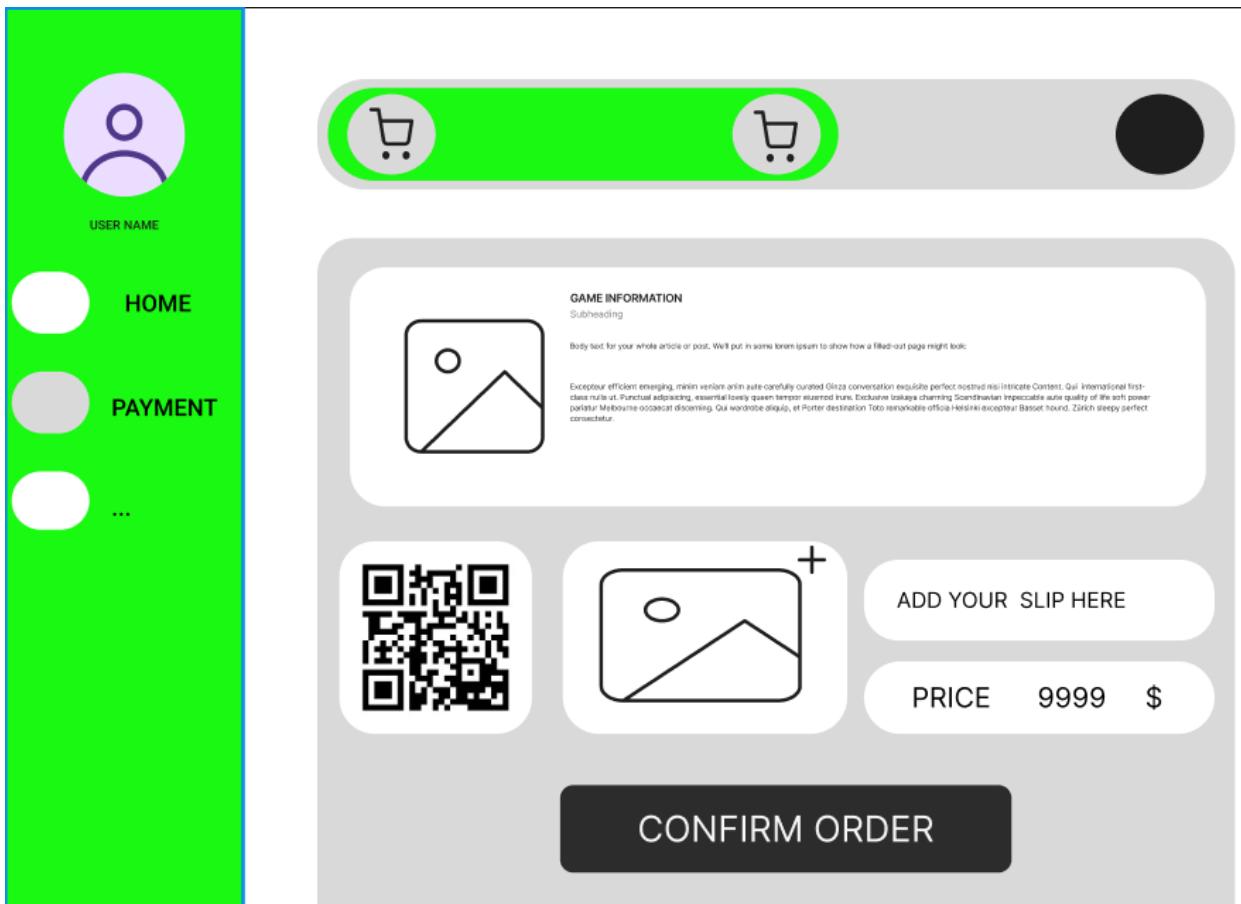
ชื่อ Entity: Payment (การชำระเงิน)

- 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)
 -

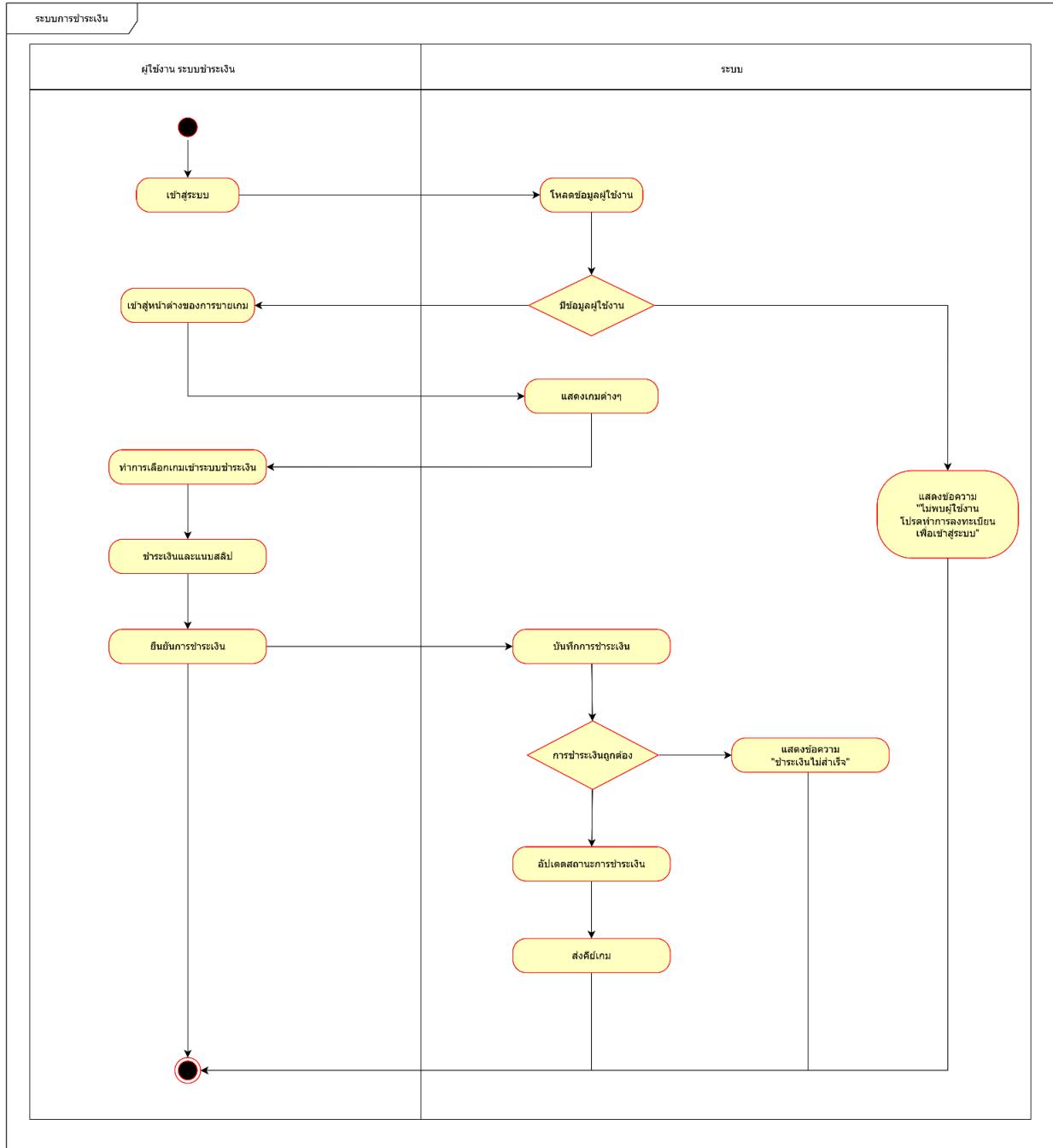
ID INTEGER	Order_ID INTEGER	Payment Date DATETIME	Amount Paid DECIMAL

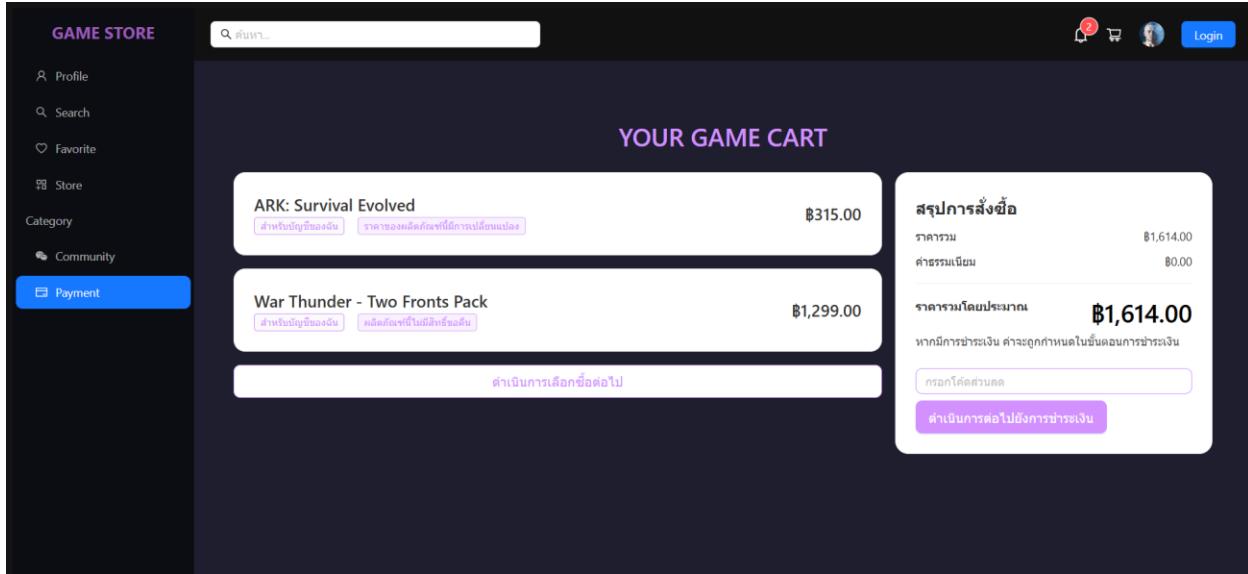
NOT NULL PK	NOT NULL FK	NOT NULL	NOT NULL
2000	501	2025-07-25 09:50:00	999.00
2001	502	2025-07-25 10:50:00	100.00

UI Design



Activity diagram





Payment UI

```
// src/pages/PaymentPage.tsx
import { useMemo, useState } from 'react';
import {
  Row,
  Col,
  Card,
  Button,
  Typography,
  Space,
  Tag,
  Divider,
  Modal,
  QRCode,
  Upload,
  Input,
  message,
} from 'antd';
import type { UploadFile } from 'antd/es/upload/interface';

const THEME_PRIMARY = '#d291ff';

interface CartItem {
  id: string;
  title: string;
  price: number; // THB
  note?: string;
}

const initialItems: CartItem[] = [
  {
    id: '1',
    title: 'ARK: Survival Evolved',
    price: 315.00,
    note: 'สำหรับผู้ใช้แพลตฟอร์ม Steam และชาร์จเดิมพันที่มีการเปลี่ยนแปลง',
  },
  {
    id: '2',
    title: 'War Thunder - Two Fronts Pack',
    price: 1299.00,
    note: 'สำหรับผู้ใช้แพลตฟอร์ม Steam และชาร์จเดิมพันที่มีการเปลี่ยนแปลง',
  },
]
```

```

id: 'ark',
title: 'ARK: Survival Evolved',
price: 315,
note: 'ราคาของผลิตภัณฑ์นี้มีการเปลี่ยนแปลง',
},
{
id: 'wt-two-fronts',
title: 'War Thunder - Two Fronts Pack',
price: 1299,
note: 'ผลิตภัณฑ์นี้ไม่มีลิขสิทธิ์ของคุณ',
},
];
};

const formatTHB = (n: number) => `฿${n.toLocaleString('th-TH', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}`;

const PaymentPage = () => {
const [items] = useState<CartItem[]>(initialItems);
const [payOpen, setPayOpen] = useState(false);
const [files, setFiles] = useState<UploadFile[]>([]);
const [submitting, setSubmitting] = useState(false);
const [discountCode, setDiscountCode] = useState('');

const subtotal = useMemo(() => items.reduce((s, it) => s + it.price, 0), [items]);
const fee = 0;
const discount = useMemo(() => (discountCode.trim().toUpperCase() === 'SAVE10' ? subtotal * 0.1 : 0), [subtotal, discountCode]);
const total = useMemo(() => subtotal + fee - discount, [subtotal, discount]);

const orderId = useMemo(() => `ORD-${Date.now()}`, [payOpen]);

const handleSubmitSlip = async () => {
if (!files.length) {
message.warning('กรุณาแนบไฟล์เอกสารชำระเงิน');
return;
}
try {
setSubmitting(true);
await new Promise(r => setTimeout(r, 900));
message.success('ส่งยืนยันการชำระเงินเรียบร้อย');
setPayOpen(false);
setFiles([]);
} catch (e) {
message.error('ส่งล้มเหลว เรื่อง กรุณาลองใหม่');
} finally {
setSubmitting(false);
}
};

return (
<div style={{ padding: 24 }}>

```

```

<Typography.Title level={2} style={{ color: THEME_PRIMARY, textAlign: 'center', marginBottom: 24 }}>
  YOUR GAME CART
</Typography.Title>

<Row gutter={[16, 16]}>
  <Col xs={24} lg={16}>
    <Space direction="vertical" style={{ width: '100%' }} size={16}>
      {items.map((it) => (
        <Card key={it.id} hoverable style={{ borderRadius: 14, background: '#fff' }}>
          <Row align="middle" gutter={[12, 12]}>
            <Col flex="auto">
              <Typography.Title level={4} style={{ margin: 0, color: '#333' }}>{it.title}</Typography.Title>
              <Space size="small" wrap>
                <Tag color="default" style={{ borderColor: THEME_PRIMARY, color: THEME_PRIMARY }}>สำหรับบัญชีของฉัน</Tag>
                {it.note && (
                  <Tag color="purple" style={{ backgroundColor: `${THEME_PRIMARY}22`, color: THEME_PRIMARY }}>{it.note}</Tag>
                )}
              </Space>
            </Col>
            <Col>
              <Typography.Title level={4} style={{ margin: 0, color: '#333' }}>{formatTHB(it.price)}</Typography.Title>
            </Col>
          </Row>
        </Card>
      ))}
    <Button
      type="default"
      size="large"
      style={{ width: '100%', borderColor: THEME_PRIMARY, color: THEME_PRIMARY, background: '#fff' }}
    >
      ดำเนินการเลือกซื้อต่อไป
    </Button>
  </Space>
</Col>

<Col xs={24} lg={8}>
  <Card style={{ borderRadius: 16, background: '#fff' }}>
    <Typography.Title level={4} style={{ marginTop: 4, color: '#333' }}>สรุปการสั่งซื้อ</Typography.Title>
    <Space direction="vertical" style={{ width: '100%' }}>
      <Row>
        <Col flex="auto" style={{ color: '#333' }}>ราคารวม</Col>
        <Col style={{ color: '#333' }}>{formatTHB(subtotal)}</Col>
      </Row>
      <Row>
        <Col flex="auto" style={{ color: '#333' }}>ค่าธรรมเนียม</Col>
        <Col style={{ color: '#333' }}>{formatTHB(fee)}</Col>
      </Row>
      {discount > 0 && (

```

```

<Row>
  <Col flex="auto" style={{ color: '#333' }}>ส่วนลด</Col>
  <Col style={{ color: 'green' }}>- {formatTHB(discount)}</Col>
</Row>
)}
<Divider style={{ margin: '8px 0' }} />
<Row>
  <Col flex="auto"><strong style={{ color: '#333' }}>ราคารวมโดยประมาณ</strong></Col>
  <Col>
    <Typography.Title level={2} style={{ margin: 0, color: '#000' }}>{formatTHB(total)}</Typography.Title>
  </Col>
</Row>
<Typography.Paragraph type="secondary" style={{ marginTop: -4, color: '#555' }}>
  หากมีการชำระเงิน ค่าจะถูกกำหนดในขั้นตอนการชำระเงิน
</Typography.Paragraph>

<Input
  placeholder="กรอกโค้ดส่วนลด"
  value={discountCode}
  onChange={(e) => setDiscountCode(e.target.value)}
  style={{ borderRadius: 8, border: 1px solid #ccc, color: '#333' }}
/>

<Button
  type="primary"
  size="large"
  style={{ background: '#fff', border: 1px solid #ccc, color: '#333' }}
  onClick={() => setPayOpen(true)}
>
  ดำเนินการต่อไปยังการชำระเงิน
</Button>
</Space>
</Card>
</Col>
</Row>

<Modal
  title={{ color: THEME_PRIMARY }}>ชำระเงินด้วยคิวอาร์โค้ด</span>
  open={payOpen}
  onCancel={() => setPayOpen(false)}
  footer={null}
  centered
>
<Space direction="vertical" style={{ width: '100%' }} size={16}>
  <Card bordered={false} style={{ background: '#fff', border: 1px solid #ccc, padding: 16px }}>
    <Row gutter={[16, 16]} align="middle" justify="center">
      <Col style={{ textAlign: 'center' }}>
        <QRCode
          value={`PAY|${orderId}|AMOUNT:${total}|CCY:THB`}
        />
      </Col>
    </Row>
  </Card>
</Space>

```

```

size={192}
color="#000"
bgColor="#fff"
bordered
/>
<Typography.Paragraph style={{ marginTop: 12, color: '#333' }}>
<strong>จำนวนเงิน:</strong> <span style={{ color: THEME_PRIMARY }}>{formatTHB(total)}</span>
</Typography.Paragraph>
<Typography.Text style={{ color: '#555' }}>สแกนคิวอาร์โค้ดเพื่อชำระเงิน (ระบุอ้างอิง: {orderId})</Typography.Text>
</Col>
</Row>
</Card>

<div>
<Typography.Title level={5} style={{ marginBottom: 8, color: '#333' }}>แนบสลิปการชำระเงิน</Typography.Title>
<Upload.Dragger
multiple={false}
fileList={files}
maxCount={1}
accept="image/*,.pdf"
beforeUpload={() => false}
onChange={({ fileList }) => setFiles(fileList)}
onRemove={() => { setFiles([]); return true; }}
style={{ borderColor: THEME_PRIMARY, background: '#fff' }}
>
<p className="ant-upload-drag-icon">❩ </p>
<p className="ant-upload-text" style={{ color: '#333' }}>ลาก & วางไฟล์ หรือ คลิกเพื่อเลือกไฟล์</p>
<p className="ant-upload-hint" style={{ color: '#555' }}>รองรับไฟล์ภาพหรือ PDF ขนาดไม่เกิน ~10MB</p>
</Upload.Dragger>
</div>

<Space style={{ width: '100%', justifyContent: 'flex-end' }}>
<Button onClick={() => setPayOpen(false)}>ยกเลิก</Button>
<Button
type="primary"
disabled={!files.length}
loading={submitting}
style={{ backgroundColor: THEME_PRIMARY, borderColor: THEME_PRIMARY, color: '#fff' }}
onClick={handleSubmitSlip}
>
ส่งยืนยันการชำระเงิน
</Button>
</Space>
</Space>
</Modal>
</div>
);
};

```

```
export default PaymentPage;
```

Payment Entity

```
// user.go
package entity

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

type User 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"`
    RoleID   uint    `json:"role_id"`

    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"`
}

// order.go
package entity

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

type Order struct {
    gorm.Model
    TotalAmount float64 `json:"total_amount"`
    OrderCreate time.Time `json:"order_create"`
    OrderStatus string `json:"order_status"`

    UserID uint `json:"user_id"`
    User *User `gorm:"foreignKey:UserID" json:"user,omitempty"`

    // relations
    OrderItems []OrderItem `gorm:"foreignKey:OrderID" json:"order_items,omitempty"`
}
```

```

Payments []Payment `gorm:"foreignKey:OrderID" json:"payments,omitempty"`
OrderPromotions []OrderPromotion `gorm:"foreignKey:OrderID" json:"order_promotions,omitempty"`
}

// payment.go
package entity

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

type Payment struct {
    gorm.Model
    PaymentDate time.Time `json:"payment_date"`
    Status      string   `json:"status"`
    Amount      float64  `json:"amount"`

    OrderID uint   `json:"order_id"`
    Order   *Order `gorm:"foreignKey:OrderID" json:"order,omitempty"`

    PaymentSlips []PaymentSlip `gorm:"foreignKey:PaymentID" json:"payment_slips,omitempty"`
}

//PaymentSlip.go
// entity/payment_slip.go
package entity

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

type PaymentSlip struct {
    gorm.Model
    UploadAt time.Time `json:"upload_at"`
    FileURL  string   `json:"file_url"`

    PaymentID uint   `json:"payment_id"`
    Payment   *Payment `gorm:"foreignKey:PaymentID" json:"payment,omitempty"`
}

//PaymentReview.go
package entity

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

```

```

)

// ប័ណ្ណកែវការត្រួសពិនិត្យ/ការចាំខែ (ដើម្បីអាជីវិន)
type PaymentReview struct {
    gorm.Model
    VerifiedAt time.Time `json:"verified_at"`
    Title      string   `json:"title"`
    Result     string   `json:"result"` // e.g. "approved" | "rejected"
    Note       string   `json:"note"`

    UserID uint `json:"user_id"` // ផ្ទុរវគ្គ
    User   *User  `gorm:"foreignKey:UserID" json:"user,omitempty"`

}

//OrderPromotion.go
package entity

import "gorm.io/gorm"

type OrderPromotion struct {
    gorm.Model
    DiscountAmount float64 `json:"discount_amount"`

    OrderID uint `json:"order_id"`
    Order   *Order `gorm:"foreignKey:OrderID" json:"order,omitempty"`

    PromotionID uint `json:"promotion_id"`
    Promotion  *Promotion `gorm:"foreignKey:PromotionID" json:"promotion,omitempty"`
}

//Promotion.go
// entity/promotion.go
package entity

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

type Promotion struct {
    gorm.Model
    Title      string   `json:"title"`
    Description string   `json:"description"`
    DiscountValue int     `json:"discount_value"` // គឺជំនាញរៀងទីនឹងដែលត្រូវបានបញ្ចូន
    StartDate  time.Time `json:"start_date"`
    EndDate   time.Time `json:"end_date"`
    PromoName  string   `json:"promoname"`
}

```

```

CreatedBy uint `json:"createdby"`
Creator *User `gorm:"foreignKey:CreatedBy" json:"creator,omitempty"`

OrderPromotions []OrderPromotion `gorm:"foreignKey:PromotionID" json:"order_promotions,omitempty"`
}

//OrderItem.go
package entity

import "gorm.io/gorm"

type OrderItem struct {
gorm.Model
UnitPrice float64 `json:"unit_price"`
QTY int `json:"qty"`
LineDiscount float64 `json:"line_discount"`
LineTotal float64 `json:"line_total"`

OrderID uint `json:"order_id"`
Order *Order `gorm:"foreignKey:OrderID" json:"order,omitempty"`

// မุกគីមពេលទិន្នន័យ (តាមី)
GameKeyID *uint `json:"game_key_id"`
GameKey *GameKey `gorm:"foreignKey:GameKeyID" json:"game_key,omitempty"`
}

//UserGame.go
package entity

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

type UserGame struct {
gorm.Model
Status string `json:"status" // e.g. "active", "revoked"`
GrantedAt time.Time `json:"granted_at"`
RevokedAt *time.Time `json:"revoked_at" // បើជា nil ដោយមែនកូរយកដើរ
GameID uint `json:"game_id"`
Game *Game `gorm:"foreignKey:GameID" json:"game"`
UserID uint `json:"user_id"`
User *User `gorm:"foreignKey:UserID" json:"user"`
}

//Game.go
package entity

import (

```

```

"gorm.io/gorm"
)

type Game struct {
gorm.Model
GameName string `json:"game_name"`
GamePrice float64 `json:"game_price" // ใช้ float64 จ่าย ๆ ถ้าอยากแม่นยำค่อยเปลี่ยนเป็น decimal lib
Description string `json:"description"`

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

//GameKey.go
package entity

import "gorm.io/gorm"

// คีย์เกมในสต็อก; จะถูกจ่อให้ OrderItem เมื่อขายสำเร็จ
type GameKey struct {
gorm.Model
KeyCode string `json:"key_code"`

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

// ถ้าคีย์ซ้ำกับอีกบอร์ดเดียวให้อัปเดตใหม่
OrderItemID *uint `json:"order_item_id"`
OrderItem *OrderItem `gorm:"foreignKey:OrderItemID" json:"order_item,omitempty"`
}

//Notification.go
package entity

import (
 "gorm.io/gorm"
)

type Notification struct {
gorm.Model
Title string `json:"title"`
Type string `json:"type"` // e.g. "payment", "system", ...
Message string `json:"message"`
UserID uint `json:"user_id"`
User *User `gorm:"foreignKey:UserID" json:"user"`
}

```

```
//Role.go
package entity

import "gorm.io/gorm"

type Role struct {
    gorm.Model
    RoleName string `json:"role_name"`
    Description string `json:"description"`

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

Payment Controller

```
//user.go
package controllers

import (
    "net/http"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

// POST /users
func CreateUser(c *gin.Context) {
    var body entity.User
    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 /users
// optional: ?username=... / ?email=...
func FindUsers(c *gin.Context) {
    var users []entity.User
```

```

db := configs.DB()

username := c.Query("username")
email := c.Query("email")

tx := db.Model(&entity.User{})
if username != "" {
    tx = tx.Where("username = ?", username)
}
if email != "" {
    tx = tx.Where("email = ?", email)
}
if err := tx.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().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"})
}

```

```

// 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"})
}

//Role.go
package controllers

import (
    "net/http"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

func CreateRole(c *gin.Context) {
    var body entity.Role
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request"})
        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)
}

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

func FindRoleByID(c *gin.Context) {
    var row entity.Role
    if tx := configs.DB().First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
}

```

```

c.JSON(http.StatusOK, row)
}

func UpdateRole(c *gin.Context) {
    var payload entity.Role
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    var row entity.Role
    db := configs.DB()
    if tx := db.First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    if err := db.Model(&row).Updates(payload).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "updated successful"})
}

func DeleteRole(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM roles 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.go
package controllers

import (
    "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
    }
}

```

```

// เช็ค User
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
}

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 /notifications (required: ?user_id=...)
func FindNotifications(c *gin.Context) {
    var rows []entity.Notification
    uid := c.Query("user_id")
    if uid == "" {
        c.JSON(http.StatusBadRequest, gin.H{"error": "must be parameter 'user_id'"})
        return
    }

    if err := configs.DB().Preload("User").
        Where("user_id = ?", uid).Find(&rows).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, rows)
}

// GET /notifications/:id
func FindNotificationByID(c *gin.Context) {
    var row entity.Notification
    if tx := configs.DB().Preload("User").First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, row)
}

// PUT /notifications/:id
func UpdateNotification(c *gin.Context) {
    var payload entity.Notification
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    db := configs.DB()

```

```

var row entity.Notification
if tx := db.First(&row, c.Param("id")); tx.RowsAffected == 0 {
    c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
    return
}
if err := db.Model(&row).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 /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": "id not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "deleted successful"})
}

//Order.go
package controllers

import (
    "net/http"
    "time"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

func CreateOrder(c *gin.Context) {
    var body entity.Order
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request"})
        return
    }
    // တော် User
    var user entity.User
    if tx := configs.DB().First(&user, body.UserID); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "user_id not found"})
        return
    }
    if body.OrderCreate.IsZero() {
        body.OrderCreate = time.Now()
    }
    if err := configs.DB().Create(&body).Error; err != nil {

```

```

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

func FindOrders(c *gin.Context) {
    var rows []entity.Order
    db := configs.DB().Preload("User").Preload("OrderItems").Preload("Payments").Preload("OrderPromotions")
    userID := c.Query("user_id")
    status := c.Query("status")
    if userID != "" {
        db = db.Where("user_id = ?", userID)
    }
    if status != "" {
        db = db.Where("order_status = ?", status)
    }
    if err := db.Find(&rows).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, rows)
}

func FindOrderByID(c *gin.Context) {
    var row entity.Order
    if tx := configs.DB().
        Preload("User").
        Preload("OrderItems.GameKey").
        Preload("Payments.PaymentSlips").
        Preload("OrderPromotions.Promotion").
        First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, row)
}

func UpdateOrder(c *gin.Context) {
    var payload entity.Order
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    var row entity.Order
    db := configs.DB()
    if tx := db.First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
}

```

```

}

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

c.JSON(http.StatusOK, gin.H{"message": "updated successful"})
}

func DeleteOrder(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM orders 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"})
}

//OrderItem.go
package controllers

import (
    "math"
    "net/http"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

func CreateOrderItem(c *gin.Context) {
    var body entity.OrderItem
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request"})
        return
    }
    db := configs.DB()

    // ตรวจ Order
    var od entity.Order
    if tx := db.First(&od, body.OrderID); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "order_id not found"})
        return
    }

    // ตรวจ GameKey (ถ้าระบุ)
    if body.GameKeyID != nil {
        var gk entity.GameKey
        if tx := db.First(&gk, *body.GameKeyID); tx.RowsAffected == 0 {
            c.JSON(http.StatusBadRequest, gin.H{"error": "game_key_id not found"})
            return
        }
    }
}

```

```

        }

        if gk.OrderItemID != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error": "game key already assigned"})
            return
        }
    }

    // คำนวณ line total แบบทั่ว
    sub := body.UnitPrice * float64(body.QTY)
    total := sub - body.LineDiscount
    if total < 0 {
        total = 0
    }
    body.LineTotal = math.Round(total*100) / 100

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

    // ต้าງค์ GameKey ให้ตั้ง owner
    if body.GameKeyID != nil {
        db.Model(&entity.GameKey{}).
            Where("id = ?", *body.GameKeyID).
            Update("order_item_id", body.ID)
    }

    c.JSON(http.StatusCreated, body)
}

func FindOrderItems(c *gin.Context) {
    var rows []entity.OrderItem
    db := configs.DB().Preload("Order").Preload("GameKey")
    orderID := c.Query("order_id")
    if orderID != "" {
        db = db.Where("order_id = ?", orderID)
    }
    if err := db.Find(&rows).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, rows)
}

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

```

```

    }

    db := configs.DB()

    var row entity.OrderItem

    if tx := db.First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }

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

    c.JSON(http.StatusOK, gin.H{"message": "updated successful"})
}

func DeleteOrderItem(c *gin.Context) {
    // เคสีร์การอ้างถึง GameKey ก่อน
    db := configs.DB()
    db.Model(&entity.GameKey{}).Where("order_item_id = ?", c.Param("id")).Update("order_item_id", nil)

    if tx := db.Exec("DELETE FROM order_items 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"})
}

//OrderPromotion.go
package controllers

import (
    "net/http"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

func CreateOrderPromotion(c *gin.Context) {
    var body entity.OrderPromotion

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

    db := configs.DB()
    var od entity.Order

    if tx := db.First(&od, body.OrderID); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "order_id not found"})
        return
    }
}

```

```

var pr entity.Promotion
if tx := db.First(&pr, body.PromotionID); tx.RowsAffected == 0 {
    c.JSON(http.StatusBadRequest, gin.H{"error": "promotion_id not found"})
    return
}
if err := db.Create(&body).Error; err != nil {
    c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
    return
}
c.JSON(http.StatusCreated, body)
}

func FindOrderPromotions(c *gin.Context) {
    var rows []entity.OrderPromotion
    db := configs.DB().Preload("Order").Preload("Promotion")
    oid := c.Query("order_id")
    if oid != "" {
        db = db.Where("order_id = ?", oid)
    }
    if err := db.Find(&rows).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, rows)
}

func DeleteOrderPromotion(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM order_promotions 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"})
}

//Promotion.go
package controllers

import (
    "net/http"
    "time"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

func CreatePromotion(c *gin.Context) {
    var body entity.Promotion
    if err := c.ShouldBindJSON(&body); err != nil {

```

```

c.JSON(http.StatusBadRequest, gin.H{"error": "bad request"})
return
}
// ตรวจสอบผู้สร้าง (User)
if body.CreatedBy != 0 {
    var u entity.User
    if tx := configs.DB().First(&u, body.CreatedBy); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "createdby not found"})
        return
    }
}
// guard เวลา
if body.StartDate.IsZero() {
    body.StartDate = time.Now()
}
if err := configs.DB().Create(&body).Error; err != nil {
    c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
    return
}
c.JSON(http.StatusCreated, body)
}

func FindPromotions(c *gin.Context) {
    var rows []entity.Promotion
    db := configs.DB().Preload("Creator")
    active := c.Query("active") // "true" = อยู่ในช่วงวันที่ใช้งาน

    if active == "true" {
        now := time.Now()
        db = db.Where("(start_date IS NULL OR start_date <= ?) AND (end_date IS NULL OR end_date >= ?)", now, now)
    }

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

func FindPromotionByID(c *gin.Context) {
    var row entity.Promotion
    if tx := configs.DB().Preload("Creator").First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, row)
}

func UpdatePromotion(c *gin.Context) {

```

```

var payload entity.Promotion
if err := c.ShouldBindJSON(&payload); err != nil {
    c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
    return
}
db := configs.DB()
var row entity.Promotion
if tx := db.First(&row, c.Param("id")); tx.RowsAffected == 0 {
    c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
    return
}
if err := db.Model(&row).Updates(payload).Error; err != nil {
    c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
    return
}
c.JSON(http.StatusOK, gin.H{"message": "updated successful"})
}

func DeletePromotion(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM promotions 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"})
}

//Payment.go
package controllers

import (
    "net/http"
    "time"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

func CreatePayment(c *gin.Context) {
    var body entity.Payment
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request"})
        return
    }
    // ตรวจสอบ Order
    var od entity.Order
    if tx := configs.DB().First(&od, body.OrderID); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "order_id not found"})
        return
    }
}

```

```

        }

        if body.PaymentDate.IsZero() {
            body.PaymentDate = time.Now()
        }

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

        c.JSON(http.StatusCreated, body)
    }
}

func FindPayments(c *gin.Context) {
    var rows []entity.Payment
    db := configs.DB().Preload("Order").Preload("PaymentSlips")
    orderID := c.Query("order_id")
    status := c.Query("status")
    if orderID != "" {
        db = db.Where("order_id = ?", orderID)
    }
    if status != "" {
        db = db.Where("status = ?", status)
    }
    if err := db.Find(&rows).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, rows)
}

func UpdatePayment(c *gin.Context) {
    var payload entity.Payment
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    db := configs.DB()
    var row entity.Payment
    if tx := db.First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    if err := db.Model(&row).Updates(payload).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "updated successful"})
}

func DeletePayment(c *gin.Context) {
}

```

```

// ลบสลิปที่เกี่ยวข้องก่อน
db := configs.DB()
db.Exec("DELETE FROM payment_slips WHERE payment_id = ?", c.Param("id"))

if tx := db.Exec("DELETE FROM payments 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"})
}

//PaymentSlip.go
package controllers

import (
    "net/http"
    "time"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

func CreatePaymentSlip(c *gin.Context) {
    var body entity.PaymentSlip
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request"})
        return
    }

    var pm entity.Payment
    if tx := configs.DB().First(&pm, body.PaymentID); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "payment_id not found"})
        return
    }

    if body.UploadAt.IsZero() {
        body.UploadAt = time.Now()
    }

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

    c.JSON(http.StatusCreated, body)
}

func FindPaymentSlips(c *gin.Context) {
    var rows []entity.PaymentSlip
    db := configs.DB().Preload("Payment")
    paymentID := c.Query("payment_id")

    if paymentID != "" {
        db = db.Where("payment_id = ?", paymentID)
    }
}

```

```

}

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

func DeletePaymentSlip(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM payment_slips 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"})
}

//PaymentReview.go
package controllers

import (
    "net/http"
    "time"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

func CreatePaymentReview(c *gin.Context) {
    var body entity.PaymentReview
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request"})
        return
    }
    // ตรวจสอบ User (ผู้ดูแล)
    var user entity.User
    if tx := configs.DB().First(&user, body.UserID); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "user_id not found"})
        return
    }
    if body.VerifiedAt.IsZero() {
        body.VerifiedAt = time.Now()
    }
    if err := configs.DB().Create(&body).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusCreated, body)
}

```

```

func FindPaymentReviews(c *gin.Context) {
    var rows []entity.PaymentReview
    db := configs.DB().Preload("User")
    userID := c.Query("user_id")
    if userID != "" {
        db = db.Where("user_id = ?", userID)
    }
    if err := db.Find(&rows).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, rows)
}

func DeletePaymentReview(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM payment_reviews 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"})
}

//Game.go
package controllers

import (
    "net/http"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

// POST /games
func CreateGame(c *gin.Context) {
    var body entity.Game
    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 /games (optional: ?name=...)

```

```

func FindGames(c *gin.Context) {
    var games []entity.Game
    name := c.Query("name")

    tx := configs.DB().Model(&entity.Game{})
    if name != "" {
        tx = tx.Where("game_name LIKE ?", "%" + name + "%")
    }

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

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

// PUT /games/:id
func UpdateGame(c *gin.Context) {
    var payload entity.Game
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    var game entity.Game
    db := configs.DB()
    if tx := db.First(&game, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    if err := db.Model(&game).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 /games/:id
func DeleteGameByID(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM games 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"})
}

//GameKey.go
package controllers

import (
    "net/http"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

func CreateGameKey(c *gin.Context) {
    var body entity.GameKey
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request"})
        return
    }
    // ตรวจ Game
    var g entity.Game
    if tx := configs.DB().First(&g, body.GameID); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "game_id not found"})
        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)
}

func FindGameKeys(c *gin.Context) {
    var rows []entity.GameKey
    db := configs.DB().Preload("Game").Preload("OrderItem")

    gameID := c.Query("game_id")
    available := c.Query("available") // "true" = ยังไม่ถูกจองไว้ OrderItem

    if gameID != "" {
        db = db.Where("game_id = ?", gameID)
    }
    if available == "true" {
        db = db.Where("order_item_id IS NULL")
    }
    if err := db.Find(&rows).Error; err != nil {

```

```

        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, rows)
}

func DeleteGameKey(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM game_keys 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"})
}

//UserGame.go
package controllers

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

// POST /user-games
func CreateUserGame(c *gin.Context) {
    var body entity.UserGame
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request body"})
        return
    }

    db := configs.DB()
    var user entity.User
    if tx := db.Where("id = ?", body.UserID).First(&user); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "user_id not found"})
        return
    }

    var game entity.Game
    if tx := db.Where("id = ?", body.GameID).First(&game); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "game_id not found"})
        return
    }

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

```

```

c.JSON(http.StatusCreated, body)
}

// GET /user-games (optional: ?user_id=... ?game_id=...)
func FindUserGames(c *gin.Context) {
    var rows []entity.UserGame
    db := configs.DB()

    userID := c.Query("user_id")
    gameID := c.Query("game_id")

    tx := db.Preload("User").Preload("Game").Model(&entity.UserGame{})
    if userID != "" {
        tx = tx.Where("user_id = ?", userID)
    }
    if gameID != "" {
        tx = tx.Where("game_id = ?", gameID)
    }

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

// GET /user-games/:id
func FindUserGameByID(c *gin.Context) {
    var row entity.UserGame
    if tx := configs.DB().Preload("User").Preload("Game").First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, row)
}

// PUT /user-games/:id
func UpdateUserGame(c *gin.Context) {
    var payload entity.UserGame
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    db := configs.DB()
    var row entity.UserGame
    if tx := db.First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
}

```

```

if err := db.Model(&row).Updates(payload).Error; err != nil {
    c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
    return
}
c.JSON(http.StatusOK, gin.H{"message": "updated successful"})
}

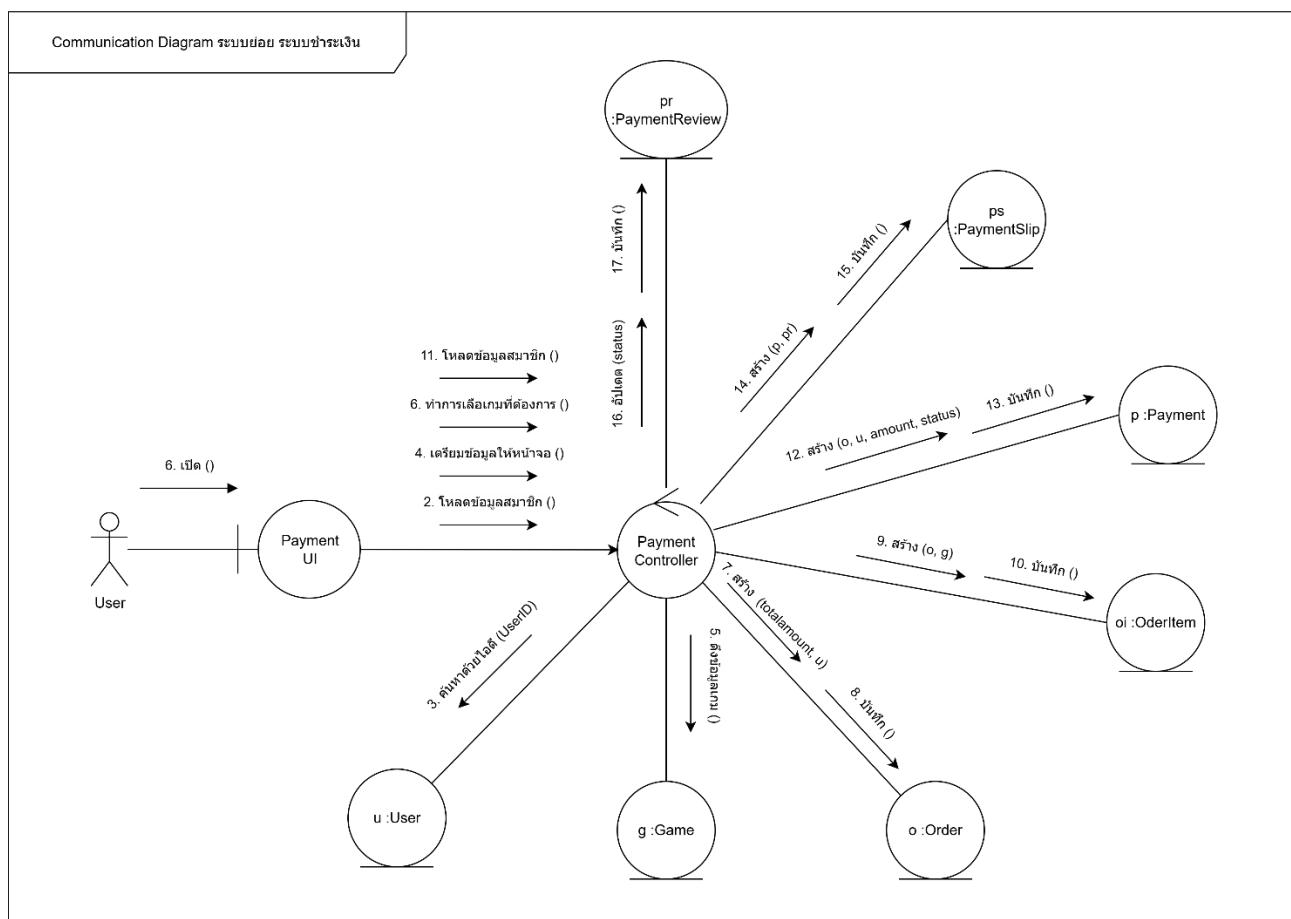
// DELETE /user-games/:id
func DeleteUserGameByID(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM user_games WHERE id = ?", c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusNotFound, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message":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

```

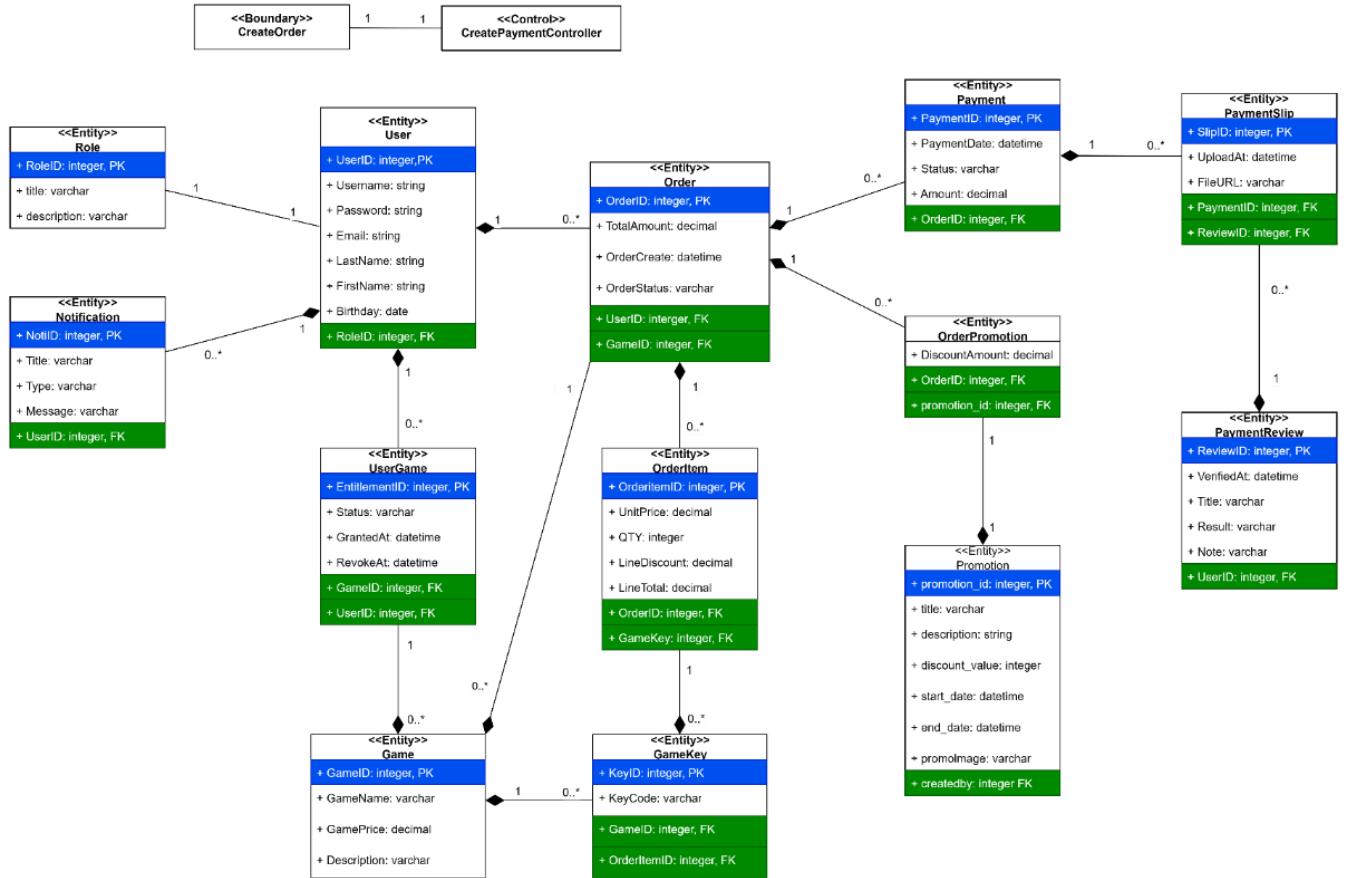
Communication Diagram

Activity	เป็นคำสั่งสำหรับระบบได้ หรือไม่	ถ้าเป็น ,วัตถุที่รับหน้าที่ ทำงาน คืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click “ไปหน้า Payment”	เป็นคำสั่ง สั่งให้ UI โหลดหน้า	:PaymentUI	เปิด ()
โหลดข้อมูลสมาชิก	เป็นคำสั่ง เกิดการสั่งให้โหลด ข้อมูลของสมาชิกในระบบ	:PaymentController	โหลดข้อมูลสมาชิก(userId)
ค้นหาไอดีสมาชิกในฐานข้อมูล	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 เพื่อกำหนดราคาของ เกมรวมกับส่วนลดของโปรโมชัน	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()
เบิดคำขอตรวจสอบสลิป	เป็นคำสั่ง	pr :PaymentReview	สร้าง(paymentId, result, note)

อัปเดต Payment	เป็นคำสั่ง เพื่ออัปเดตสถานะ	p :Payment	อัปเดต (status)
แจ้งผู้ใช้ “รับสลิปแล้ว/กำลังตรวจสอบ”	เป็นคำสั่ง ระบบสร้างแจ้งเตือน	n :Notification	สร้าง (type:'comment_created', targetType:'thread')
แออดมินตรวจสอบ	เป็นคำสั่ง สำหรับการยืนยันว่า ชำระเงินถูกต้อง	pr :PaymentReview	อัปเดต (reviewed)
กรณี Approved ให้เลือกเกม	เป็นคำสั่ง เพื่อให้เลือกการเข้าถึง ระบบต่างๆเมื่อได้รับเงิน	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)
แสดงผลลัพธ์ถูกท้าย (คีย์เกม/ขั้นตอนถัดไป)	ไม่เป็นคำสั่ง	-	-



Class Diagram at Design Level



B6618643 นายกิตตินันท์ ปัจจัยโคงา

ระบบหลัก ระบบร้านขายเกมออนไลน์

ระบบย่อย ระบบคอมมูนิตี้ขนาดเล็ก

ระบบคอมมูนิตี้ขนาดเล็กเป็นระบบที่เปิดให้ผู้ใช้งานที่ชื่อเกมเข้าร่วมพูดคุย และแสดงความคิดเห็น หรือแชร์

ประสบการณ์เกี่ยวกับเกมต่าง ๆ ผ่านกระดานสนทนา และกดถูกใจกระทู้ที่ชอบ ผู้ใช้งานสามารถสร้างกระทู้ใหม่ แก้ไขข้อความของตนเอง หรือตอบกระทู้เมื่อไม่ต้องการเผยแพร่อีกต่อไป ระบบจะบันทึกประวัติการใช้งาน เช่น

วันเวลาที่โพสต์ จำนวนการกดถูกใจ และการตอบกลับจากผู้ใช้งานอื่น ระบบยังเปิดให้มีการแสดงข้อมูลประวัติ กระทู้ที่เคยโพสต์ที่มีส่วนร่วมในคอมมูนิตี้ เช่น กระทู้ที่กดถูกใจไว้

User Story ระบบคอมมูนิตี้ขนาดเล็ก

ในบทบาทของ (As a) ผู้ใช้งาน

ฉันต้องการ (I want to) พูดคุยแลกเปลี่ยนข้อมูลกับคนอื่นในเกม

เพื่อ (So that) นำข้อมูลที่ได้ไปรับใช้ในการเล่นเกม

Output บนหน้าจอ

- สามารถกดสร้างกระทู้ขึ้นมาได้ พร้อมหัวข้อและเนื้อหาที่ต้องการโพสต์ พร้อมกับสามารถแก้ไขหรือลบกระทู้ได้ตามความต้องการ และสามารถสื่อสารกับคนอื่นได้ในกระทู้รวมไปถึงการกดถูกใจกระทู้ที่ชอบ

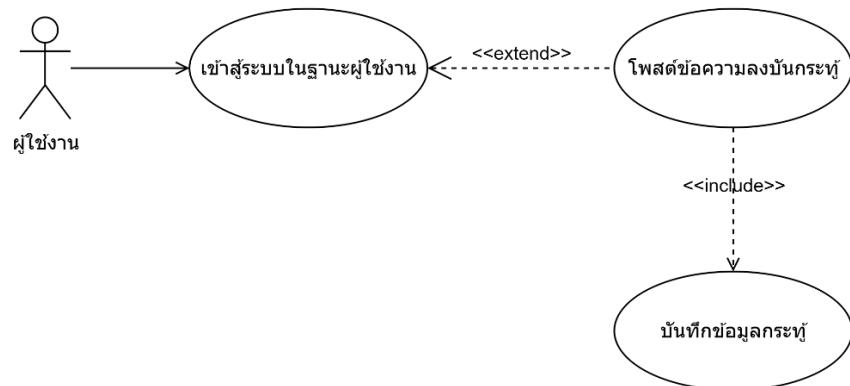
Output ของข้อมูล

- ระบบจะจัดการเก็บข้อมูล เนื้อหา วันเวลาเวลาที่โพสต์ ผู้ตั้งกระทู้ ข้อความการตอบกลับจากผู้ใช้งานคนอื่น จำนวนคอมเมนต์ของกระทู้ หากผู้ใช้ทำการลบกระทู้ก็จะทำการลบข้อมูลออกจากฐานข้อมูลด้วย

คำนำที่อาจจะกลایมมาเป็นตารางในฐานข้อมูลของระบบ

คำนำ	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
ผู้ใช้งาน	เกี่ยวข้องโดยตรง เนื่องจากผู้ใช้งานจะเป็นคนสร้างกระทู้ขึ้นมา
เกม	ไม่เกี่ยวข้องโดยตรง
ถูกใจ	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)” และ จำเป็นต้อง “สร้างกระทู้ใหม่” ทุกครั้งหรือไม่?

ตอบ ไม่จำเป็น

แปลง Use Case “สร้างกระทู้ใหม่” เป็นกิจกรรมทางเลือกหลังจาก “เข้าสู่ระบบ”

พิจารณาประเด็นที่ 2

ถ้าสมาชิก “อ่านกระทู้” และ จำเป็นต้อง “แสดงความคิดเห็น” ทุกครั้งหรือไม่?

ตอบ ไม่จำเป็น (ผู้ใช้อาจอ่านโดย ๆ ได้)

แปลง Use Case “แสดงความคิดเห็น” เป็นกิจกรรมทางเลือกหลังจาก “อ่านกระทู้”

พิจารณาประเด็นที่ 3

ถ้าสมาชิก “แสดงความคิดเห็น” แล้ว จำเป็นต้อง “บันทึกข้อมูลความคิดเห็น” ทุกครั้งหรือไม่?

ตอบ ใช่ จำเป็น

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

พิจารณาประเด็นที่ 5

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

ตอบ ไม่จำเป็น (ผู้ใช้สามารถ Like ได้แม้ไม่ได้แสดงความคิดเห็น)

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

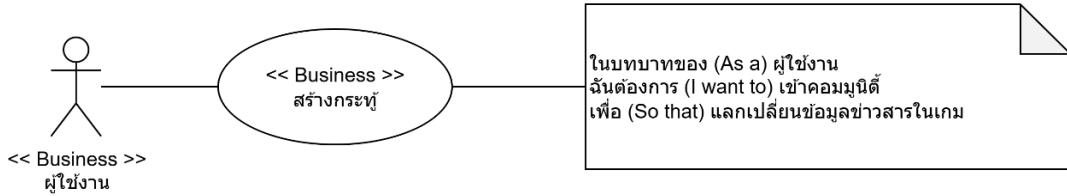
พิจารณาประเด็นที่ 5

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 ต้องไม่มีอะไรรากกับ
2. เส้นโยงจาก System Actor ไปหา System Use Case จะต้องเป็นเส้นทิบ 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 ทางเลือกไปยัง System Use Case หลัก
7. การใช้ <--<<include>>--> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ที่ใหญ่กว่า ไปยัง System Use Case ที่จะรวมเข้ามา

วิเคราะห์ Entity ที่เกี่ยวข้อง

ชื่อ 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	กระตุก...	2002	2025-07-25 15:00:00

ชื่อ Entity: Comment (ความคิดเห็น)

- ข้อมูลที่ต้องจัดเก็บ:
 - ID: Primary Key (INTEGER, NOT NULL, PK)
 - Thread_ID: กระทู้ที่ตอบกลับ (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.

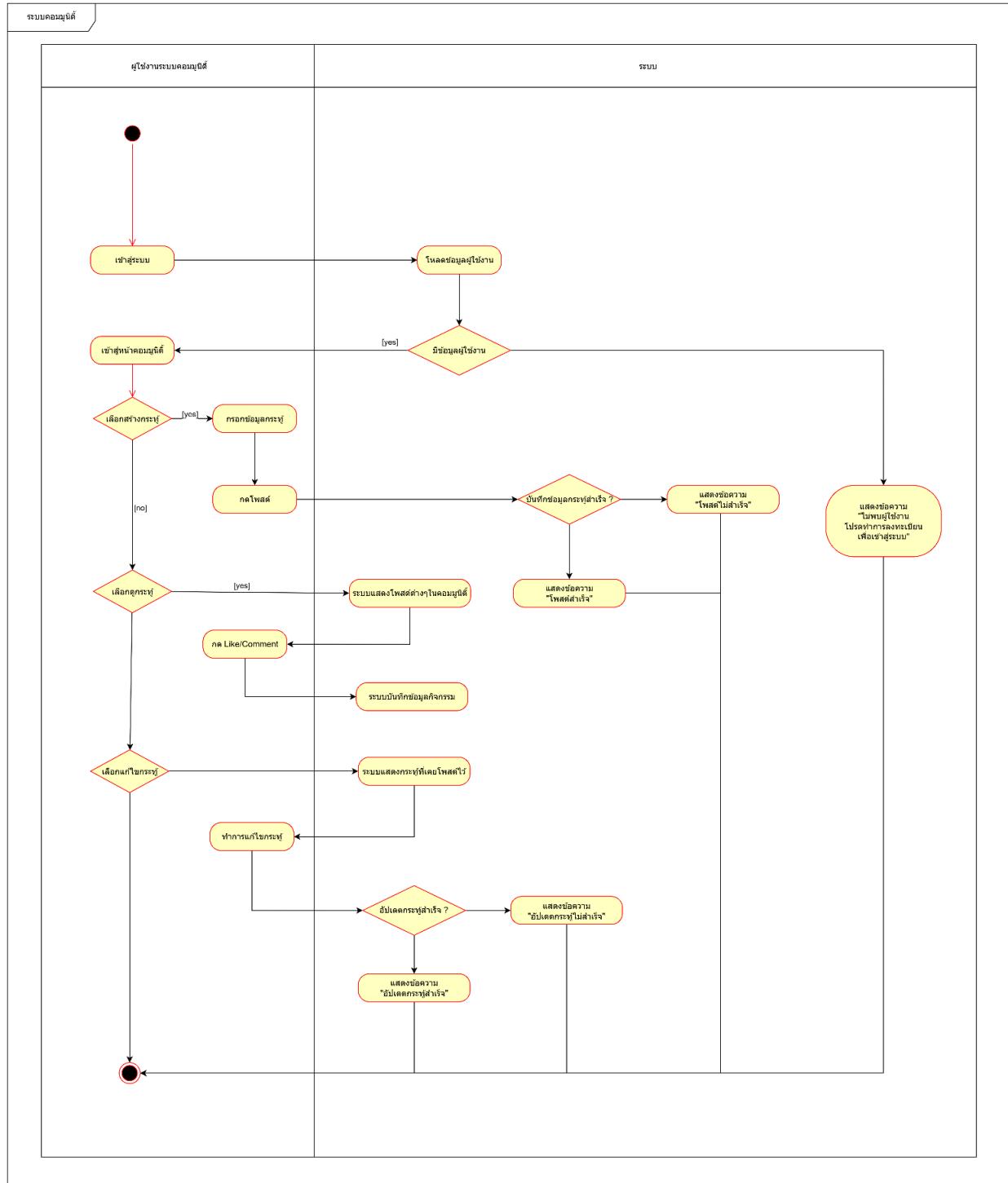
Left Side (Wireframe):

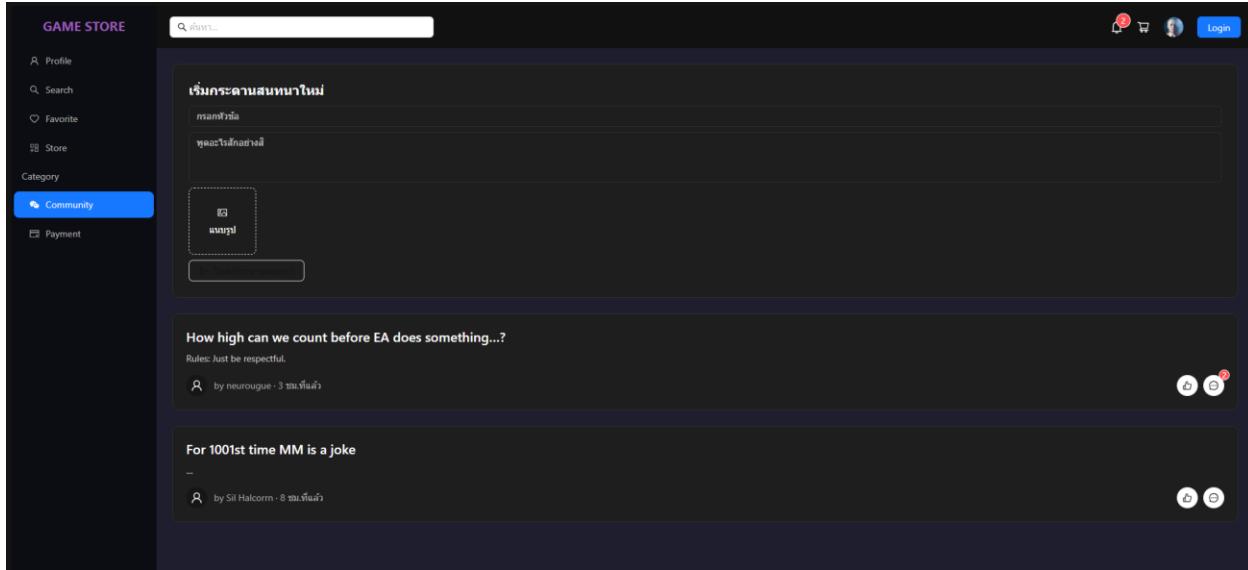
- User Profile:** A large yellow vertical bar on the left containing:
 - A circular profile icon with a person symbol.
 - The text "USER NAME" below it.
- Navigation Bar:** Below the profile, a white horizontal bar with the text "HOME".
- Community Section:** A white rectangular box labeled "Community".
- Game Section:** Two white rectangular boxes, each with a small white square icon and the word "GAME" below it.
- Empty Spacing:** Several thin white horizontal bars and a thick yellow bar at the bottom for visual separation.

Right Side (Filled Out):

- User Profile:** A grey header bar with a user icon and the placeholder text "USER NAME".
- Post Preview:** A large grey area containing a placeholder image with a "G" watermark, a "Like" button, and a "Comment" button.
- Post Content Area:** A grey section with:
 - Heading:** "Heading" followed by "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:" followed by a long block of placeholder text.
- Create Post Section:** A grey area with:
 - A text input field with a speech bubble icon and the placeholder text "anything ?".
 - Two buttons: "Add pictures" and "Add video".
 - A "POST" button on the right.

Activity Diagram





Community UI

```
// CommunityPage.tsx
import { useMemo, useRef, useState } from "react";
import { message, Space } from "antd";
import ThreadList from "./ThreadList";
import ThreadDetail from "./ThreadDetail";
import type { Thread, ThreadComment } from "./types";

export default function CommunityPage() {
  // --- mock data รีบมั้น ---
  const [threads, setThreads] = useState<Thread[]>([
    {
      id: 1,
      title: "How high can we count before EA does something...?",
      body: "Rules: Just be respectful.",
      author: "neuroogue",
      createdAt: "3 ชม. ที่แล้ว",
      likes: 2,
      comments: [
        { id: 11, author: "Yanis_zaky", content: "1", datetime: "3 ชม. @ 8:17am" },
        { id: 12, author: "MaT", content: "2", datetime: "3 ชม. @ 5:06pm" },
      ],
    },
    {
      id: 2,
      title: "For 1001st time MM is a joke",
      body: "...",
      author: "Sil Halcomm",
      createdAt: "8 ชม. ที่แล้ว",
      likes: 1,
      comments: [],
    }
  ]);
}
```

```

    },
  ]);
// ----

const idRef = useRef(1000);
const [activelid, setActiveLid] = useState<number | null>(null);

const activeThread = useMemo(
() => threads.find((t) => t.id === activelid) || null,
[threads, activelid]
);

// สร้างเรดรีดใหม่ (หน้า List เท่านั้น)
const createThread = ({ title, body, images }: { title: string; body: string; images?: string[] }) => {
const n: Thread = {
  id: idRef.current++,
  title,
  body,
  author: "คุณ",
  createdAt: "เพิ่งโพสต์",
  likes: 0,
  comments: [],
  images, // ✓ เก็บรูปไปกับเรดรีด
};
setThreads((prev) => [n, ...prev]);
message.success("สร้างเรดรีดใหม่แล้ว");
};

// เพิ่มคอมเม้นต์ที่ระดับ root ของเรดรีดที่เปิดอยู่ (หน้า Detail เท่านั้น)
const replyRoot = ({ content }: { content: string }) => {
if (!activeThread) return;
const newC: ThreadComment = {
  id: idRef.current++,
  author: "คุณ",
  content,
  datetime: "เพิ่งตอบกลับ",
};
setThreads((prev) =>
  prev.map((t) => (t.id === activeThread.id ? { ...t, comments: [...t.comments, newC] } : t))
);
message.success("ตอบกลับแล้ว");
};

return (
<Space direction="vertical" size="large" style={{ width: "100%" }}>
{activeThread ? (
<ThreadDetail
  thread={activeThread}
  onBack={() => setActiveLid(null)}
)

```

```

        onReplyRoot={replyRoot}
      />
    ) : (
      <ThreadList
        threads={threads}
        onOpen={(id) => setActiveId(id)}
        onCreate={createThread}
      />
    )}
  </Space>
);
}

//CommunityItem.tsx
import { useState } from "react";
import { Avatar, Button, Input, Space, Typography } from "antd";
import { MessageOutlined, SendOutlined, UserOutlined } from "@ant-design/icons";
import type { ThreadComment } from "./types";

const { Text } = Typography;

type Props = {
  data: ThreadComment;
  onReply: (parentId: number, payload: { content: string }) => void;
};

export default function CommentItem({ data, onReply }: Props) {
  const [open, setOpen] = useState(false);
  const [val, setVal] = useState("");

  const submit = () => {
    const t = val.trim();
    if (!t) return;
    onReply(data.id, { content: t });
    setVal("");
    setOpen(false);
  };

  return (
    <div style={{ display: "flex", gap: 12, marginBottom: 12 }}>
      <Avatar icon={<UserOutlined />} />
      <div style={{ flex: 1 }}>
        <div
          style={{
            background: "#191a1f",
            border: "1px solid #2b2b2b",
            borderRadius: 10,
            padding: 10,
          }}>

```

```

>
<div style={{ display: "flex", gap: 8, alignItems: "baseline" }}>
  <Text style={{ color: "#ddd", fontWeight: 500 }}>{data.author}</Text>
  <span style={{ color: "#888", fontSize: 12 }}>{data.datetime}</span>
</div>
<div style={{ color: "#ccc", marginTop: 6 }}>{data.content}</div>

<div style={{ marginTop: 8 }}>
  <Button
    size="small"
    icon={<MessageOutlined />}
    onClick={() => setOpen((s) => !s)}
  >
    ตอบกลับ
  </Button>
</div>

{open && (
  <div style={{ display: "flex", gap: 8, marginTop: 8 }}>
    <Input.TextArea
      value={val}
      onChange={(e) => setVal(e.target.value)}
      autoSize={{ minRows: 1, maxRows: 4 }}
      placeholder="พิมพ์คำตอบของคุณ... (Ctrl+Enter เพื่อส่ง)"
      onKeyDown={(e) => {
        if (e.key === "Enter" && (e.ctrlKey || e.metaKey)) {
          e.preventDefault();
          submit();
        }
      }}
      style={{
        background: "#1e1e1e",
        color: "#fff",
        borderColor: "#303030",
        borderRadius: 8,
      }}
    />
    <Space direction="vertical" size="small">
      <Button type="primary" size="small" icon={<SendOutlined />} onClick={submit}>
        ส่ง
      </Button>
      <Button size="small" type="text" onClick={() => setOpen(false)}>
        ยกเลิก
      </Button>
    </Space>
  </div>
)
</div>

```

```

{!data.children?.length && (
  <div style={{ borderLeft: "1px dashed #303030", marginTop: 10, paddingLeft: 12 }}>
    {data.children!.map((c) => (
      <CommentItem key={c.id} data={c} onReply={onReply} />
    )));
  </div>
</div>
</div>
);
}

//ThreadDetail.tsx
import { useMemo, useState } from "react";
import { Avatar, Badge, Button, Card, Input, Modal, Space, Typography } from "antd";
import { ArrowLeftOutlined, LikeOutlined, MessageOutlined, PictureOutlined, SendOutlined, UserOutlined } from "@ant-design/icons";
import CommentItem from "./CommentItem";
import type { Thread, ThreadComment, CreateReplyPayload } from "./types";

const { Title, Text } = Typography;

type Props = {
  thread: Thread;
  onBack: () => void;
  onReplyRoot: (payload: CreateReplyPayload) => void;
};

export default function ThreadDetail({ thread, onBack, onReplyRoot }: Props) {
  const [text, setText] = useState("");
  const [preview, setPreview] = useState<string | null>(null);

  const canSend = text.trim().length > 0;

  const commentsCount = useMemo(
    () => {
      const dfs = (arr: ThreadComment[]): number =>
        arr.reduce((s, c) => s + 1 + (c.children?.length ? dfs(c.children) : 0), 0);
      return dfs(thread.comments);
    },
    [thread.comments]
  );

  return (
    <Space direction="vertical" size="middle" style={{ width: "100%" }}>
      <Button icon={<ArrowLeftOutlined />} onClick={onBack}>ย้อนกลับ</Button>

      <Card style={{ background: "#1e1e1e", border: "1px solid #303030", borderRadius: 10 }} bodyStyle={{ padding: 24 }}>
        <Space direction="vertical" size="middle" style={{ width: "100%" }}>
          <Title level={3} style={{ color: "#fff", margin: 0 }}>{thread.title}</Title>

```

```

<Text style={{ color: "#ccc" }}>{thread.body}</Text>

<div style={{ display: "flex", justifyContent: "space-between" }}>
  <div style={{ display: "flex", alignItems: "center", gap: 10 }}>
    <Avatar icon={<UserOutlined />} />
    <Text style={{ color: "#aaa" }}>by {thread.author} · {thread.createdAt}</Text>
  </div>
  <Space>
    <Button icon={<LikeOutlined />} shape="circle" />
    <Badge count={commentsCount} size="small">
      <Button icon={<MessageOutlined />} shape="circle" />
    </Badge>
  </Space>
</div>

/* โอนคอมเม้นต์ */
<div style={{ marginTop: 8, padding: 12, border: "1px solid #303030", background: "#161616", borderRadius: 12 }}>
  {thread.comments.map((c) => (
    <CommentItem
      key={c.id}
      data={c}
      onReply={() => onReplyRoot(p)} // เด้งไปให้ parent เพิ่มลง root ที่เหมาะสม
    />
  ))}
</div>

/* กล่องตอบกลับ (ของเรา) */
/* กล่องตอบกลับ (ของเรา) */

<div className="dark-input" style={{ display: "flex", gap: 10 }}>
  <Input.TextArea
    value={text}
    onChange={(e) => setText(e.target.value)}
    autoSize={{ minRows: 2, maxRows: 6 }}
    placeholder="ตอบกลับในช่องนี้... (Ctrl+Enter เพื่อส่ง)" // ✓ placeholder จะเป็นสีขาวตาม CSS
    onKeyDown={(e) => {
      if (e.key === "Enter" && (e.ctrlKey || e.metaKey)) {
        e.preventDefault();
        if (canSend) onReplyRoot({ content: text.trim() }), setText("");
      }
    }}
    style={{ flex: 1, borderRadius: 8 }}
  />
  <Button
    type="primary"
    icon={<SendOutlined />}
    disabled={!canSend}
    onClick={() => {
      onReplyRoot({ content: text.trim() });
      setText("");
    }}
  />
</div>

```

```

    )}
>
  ส่ง
</Button>
</div>
</Space>
</Card>

<Modal open={!preview} onCancel={() => setPreview(null)} footer={null} centered bodyStyle={{ padding: 0, background: "#000" }}>
  {preview && <img src={preview} style={{ width: "100%", display: "block" }} />}
</Modal>
</Space>
);
}

//ThreadList.tsx
// src/pages/Community/ThreadList.tsx
import { useState } from "react";
import {
  Avatar, Badge, Button, Card, Input, Space, Typography, Upload
} from "antd";
import {
  LikeOutlined, MessageOutlined, PictureOutlined, SendOutlined, UserOutlined
} from "@ant-design/icons";
import type { UploadFile } from "antd/es/upload/interface"; // ✓
import type { Thread, CreateThreadPayload } from "./types";

const { Title, Text } = Typography;

type Props = {
  threads: Thread[];
  onOpen: (threadId: number) => void;
  onCreate: (payload: CreateThreadPayload & { images?: string[] }) => void; // ✓
};

export default function ThreadList({ threads, onOpen, onCreate }: Props) {
  const [title, setTitle] = useState("");
  const [body, setBody] = useState("");
  const [files, setFiles] = useState<UploadFile[]>([]); // ✓ เก็บไฟล์ภาพที่เลือก

  const canSubmit = title.trim() && body.trim();

  // แปลงไฟล์เป็น preview URL
  const toUrls = (list: UploadFile[]) =>
    list
      .map((f) => (f.originFileObj ? URL.createObjectURL(f.originFileObj) : (f.url as string)))
      .filter(Boolean) as string[];
}

```

```

const resetForm = () => {
  // cleanup object URLs
  files.forEach((f) => {
    if (f.originFileObj) URL.revokeObjectURL(URL.createObjectURL(f.originFileObj));
  });
  setTitle("");
  setBody("");
  setFiles([]);
};

return (
  <Space direction="vertical" size="large" style={{ width: "100%" }}>
    /* កត់ស្នើសុំឡើង */
    <Card className="community-card">
      <Title level={4} style={{ color: "#fff", marginTop: 0 }}>
        ឯកសារការបង្ហាញថាមពល
      </Title>

      <div className="dark-input">
        <Input
          value={title}
          onChange={(e) => setTitle(e.target.value)}
          placeholder="ក្រុមហ៊្រណ៍"
          style={{ marginBottom: 8 }}
        />
        <Input.TextArea
          value={body}
          onChange={(e) => setBody(e.target.value)}
          autoSize={{ minRows: 3, maxRows: 8 }}
          placeholder="ឯកសារការបង្ហាញថាមពល"
          style={{ marginBottom: 8 }}
        />
      </div>

      /* ✓ ចូលរួមរាយការបង្ហាញថាមពល */
      <Upload
        multiple
        accept="image/*"
        listType="picture-card"
        fileList={files}
        beforeUpload={() => false} // ឲ្យបញ្ចូនឈ្មោះឯកសារដែលបានចូលរួម
        onChange={({ fileList }) => setFiles(fileList)}
        onRemove={(file) => {
          setFiles((prev) => prev.filter((f) => f.uid !== file.uid));
        }}
      >
      /* ចូលរួមរាយការបង្ហាញថាមពល */
      <div style={{ color: "#fff" }}>
        <PictureOutlined /> <div style={{ marginTop: 4 }}>ឯកសារការបង្ហាញថាមពល</div>
      </div>
    </Card>
  </Space>
);

```

```

</div>
</Upload>

<Space style={{ marginTop: 8 }}>
  <Button
    type="primary"
    icon={<SendOutlined />}
    disabled={!canSubmit}
    onClick={() => {
      onCreate({
        title: title.trim(),
        body: body.trim(),
        images: toUrls(files), // ✓ ส่ง URL ของรูปไปเก็บกับ纪录
      });
      resetForm();
    }}
  >
    โพสต์กระดานสนทนา
  </Button>
</Space>
</Card>

/* รายการเรอเดททั้งหมด */
{threads.map((t) => (
  <Card
    key={t.id}
    className="community-card"
    bodyStyle={{ padding: 20 }}
    hoverable
    onClick={() => onOpen(t.id)}
  >
    <Space direction="vertical" size="small" style={{ width: "100%" }}>
      <Title level={4} style={{ color: "#fff", margin: 0 }}>
        {t.title}
      </Title>
      <Text style={{ color: "#ccc" }}>{t.body}</Text>
    </Space>
    /* แสดงรูปแนบ (ถ้ามี) */
    {!!t.images?.length && (
      <div style={{ display: "flex", gap: 8, flexWrap: "wrap", marginTop: 6 }}>
        {t.images.map((src, i) => (
          <img
            key={i}
            src={src}
            alt={'thread-img-' + i}
            style={{ width: 96, height: 96, objectFit: "cover", borderRadius: 8, border: "1px solid #303030" }}
          />
        )));
    )}
  </Card>
))

```

```

        )}

<div style={{ display: "flex", justifyContent: "space-between", marginTop: 6 }}>
  <div style={{ display: "flex", alignItems: "center", gap: 10 }}>
    <Avatar icon={<UserOutlined />} />
    <Text style={{ color: "#aaa" }}>by {t.author} · {t.createdAt}</Text>
  </div>
  <Space>
    <Button icon={<LikeOutlined />} shape="circle" />
    <Badge count={t.comments.length} size="small">
      <Button icon={<MessageOutlined />} shape="circle" />
    </Badge>
  </Space>
</div>
<Space>
</Card>
)});

</Space>
);
}
}

```

Community Entity

```

// User.go
package entity

import (
  "time"

  "gorm.io/gorm"
)

type User 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"`
  RoleID   uint    `json:"role_id"`

  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"`
}

//Notification.go
package entity

import (
    "gorm.io/gorm"
)

type Notification struct {
    gorm.Model
    Title string `json:"title"`
    Type string `json:"type"` // e.g. "payment", "system", ...
    Message string `json:"message"`
    UserID uint `json:"user_id"`
    User *User `gorm:"foreignKey:UserID" json:"user"`
}

//Thread.go
package entity

import (
    "gorm.io/gorm"
)

type Thread struct {
    gorm.Model
    Title string `json:"title"`
    Content string `json:"content"`
    UserID uint `json:"user_id"`
    User *User `gorm:"foreignKey:UserID" json:"user"`
    GameID uint `json:"game_id"`
    Game *Game `gorm:"foreignKey:GameID" json:"game"`

    Comments []Comment `gorm:"foreignKey:ThreadID" json:"comments,omitempty"`
}

//Comment.go
package entity

import (
    "gorm.io/gorm"
)

type Comment struct {
    gorm.Model
    Content string `json:"content"`
    UserID uint `json:"user_id"`
}

```

```

User      *User `gorm:"foreignKey:UserID" json:"user"`
ThreadID  uint   `json:"thread_id"`
Thread     *Thread `gorm:"foreignKey:ThreadID" json:"thread"`
ParentCommentID *uint  `json:"parent_comment_id" // เป็น nil ถ้าเป็นคอมเม้นต์ระดับน
Parent      *Comment `gorm:"foreignKey:ParentCommentID" json:"parent"`

Replies []Comment `gorm:"foreignKey:ParentCommentID" json:"replies,omitempty"`
}

//Reaction.go
package entity

import (
    "gorm.io/gorm"
)

type Reaction struct {
    gorm.Model
    TargetType string `json:"target_type" // "thread" | "comment" | อื่น ๆ`
    TargetID  uint   `json:"target_id"`
    Type      string `json:"type"      // "like", "love", ...`
    UserID    uint   `json:"user_id"`
    User      *User  `gorm:"foreignKey:UserID" json:"user"`
}

//Attachment.go
package entity

import (
    "gorm.io/gorm"
)

type Attachment struct {
    gorm.Model
    TargetType string `json:"target_type" // "thread" | "comment"`
    TargetID  uint   `json:"target_id"`
    FileURL   string `json:"file_url"`
    UserID    uint   `json:"user_id"`
    User      *User  `gorm:"foreignKey:UserID" json:"user"`
}

//Game.go
package entity

import (
    "gorm.io/gorm"
)

type Game struct {

```

```

gorm.Model
GameName string `json:"game_name"`
GamePrice float64 `json:"game_price" // ใช้ float64 จ่าย ๆ ถ้าอยากแม่นยำค่อยเปลี่ยนเป็น decimal lib
Description string `json:"description"`

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

//UserGame.go
package entity

import (
    "time"

    "gorm.io/gorm"
)

type UserGame struct {
    gorm.Model
    Status string `json:"status" // e.g. "active", "revoked"`
    GrantedAt time.Time `json:"granted_at"`
    RevokedAt *time.Time `json:"revoked_at" // เป็น nil ได้ถ้ายังไม่ถูกยกเลิก`
    GameID uint `json:"game_id"`
    Game *Game `gorm:"foreignKey:GameID" json:"game"`
    UserID uint `json:"user_id"`
    User *User `gorm:"foreignKey:UserID" json:"user"`
}

```

Payment Controller

```

//User.go
package controllers

import (
    "net/http"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

// POST /users

```

```

func CreateUser(c *gin.Context) {
    var body entity.User
    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 /users
// optional: ?username=... / ?email=...
func FindUsers(c *gin.Context) {
    var users []entity.User
    db := configs.DB()

    username := c.Query("username")
    email := c.Query("email")

    tx := db.Model(&entity.User{})
    if username != "" {
        tx = tx.Where("username = ?", username)
    }
    if email != "" {
        tx = tx.Where("email = ?", email)
    }
    if err := tx.Find(&users).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, users)
}

// GET /users/:id
func FindUserID(c *gin.Context) {
    var user entity.User
    if tx := configs.DB().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"})
}

// 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.go
package controllers

import (
    "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
    }

    // เช็ค User
    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
}

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 /notifications (required: ?user_id=...)
func FindNotifications(c *gin.Context) {
    var rows []entity.Notification
    uid := c.Query("user_id")
    if uid == "" {
        c.JSON(http.StatusBadRequest, gin.H{"error": "must be parameter 'user_id'"})
        return
    }

    if err := configs.DB().Preload("User").
        Where("user_id = ?", uid).Find(&rows).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, rows)
}

// GET /notifications/:id
func FindNotificationByID(c *gin.Context) {
    var row entity.Notification
    if tx := configs.DB().Preload("User").First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, row)
}

// PUT /notifications/:id
func UpdateNotification(c *gin.Context) {
    var payload entity.Notification
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    db := configs.DB()
    var row entity.Notification
    if tx := db.First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
    }
}

```

```

        return
    }

    if err := db.Model(&row).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 /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": "id not found"})
        return
    }

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

//Thread.go
package controllers

import (
    "net/http"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

// POST /threads
func CreateThread(c *gin.Context) {
    var body entity.Thread

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

    db := configs.DB()
    // เช็ค FK: UserID
    var user entity.User

    if tx := db.Where("id = ?", body.UserID).First(&user); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "user_id not found"})
        return
    }

    // เช็ค FK: GameID
    var game entity.Game

    if tx := db.Where("id = ?", body.GameID).First(&game); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "game_id not found"})
        return
    }
}

```

```

}

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

// GET /threads (optional: ?user_id=... ?game_id=...)
func FindThreads(c *gin.Context) {
    var threads []entity.Thread
    db := configs.DB()

    userID := c.Query("user_id")
    gameID := c.Query("game_id")

    tx := db.Preload("User").Preload("Game").Model(&entity.Thread{})

    if userID != "" {
        tx = tx.Where("user_id = ?", userID)
    }
    if gameID != "" {
        tx = tx.Where("game_id = ?", gameID)
    }

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

// GET /threads/:id
func FindThreadByID(c *gin.Context) {
    var thread entity.Thread
    id := c.Param("id")
    if tx := configs.DB().Preload("User").Preload("Game").First(&thread, id); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, thread)
}

// PUT /threads/:id
func UpdateThread(c *gin.Context) {
    var payload entity.Thread
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
}

```

```

        }

        db := configs.DB()

        var thread entity.Thread

        if tx := db.First(&thread, c.Param("id")); tx.RowsAffected == 0 {
            c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
            return
        }

        if err := db.Model(&thread).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 /threads/:id
func DeleteThreadByID(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM threads 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"})
}

//Comment.go
package controllers

import (
    "net/http"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

// POST /comments
func CreateComment(c *gin.Context) {
    var body entity.Comment

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

    db := configs.DB()

    // เช็ค User
    var user entity.User

    if tx := db.Where("id = ?", body.UserID).First(&user); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "user_id not found"})
        return
    }
}

```

```

}

// เช็ค Thread
var thread entity.Thread
if tx := db.Where("id = ?", body.ThreadID).First(&thread); tx.RowsAffected == 0 {
    c.JSON(http.StatusBadRequest, gin.H{"error": "thread_id not found"})
    return
}

// เช็ค Parent (ถ้ามี)
if body.ParentCommentID != nil {
    var parent entity.Comment
    if tx := db.Where("id = ?", *body.ParentCommentID).First(&parent); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "parent_comment_id not found"})
        return
    }
}

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

// GET /comments (optional: ?thread_id=... ?user_id=... ?parent_id=...)
func FindComments(c *gin.Context) {
    var comments []entity.Comment
    db := configs.DB()

    threadID := c.Query("thread_id")
    userID := c.Query("user_id")
    parentID := c.Query("parent_id")

    tx := db.Preload("User").Preload("Thread").Preload("Parent").Model(&entity.Comment{})
    if threadID != "" {
        tx = tx.Where("thread_id = ?", threadID)
    }
    if userID != "" {
        tx = tx.Where("user_id = ?", userID)
    }
    if parentID != "" {
        tx = tx.Where("parent_comment_id = ?", parentID)
    }

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

```

```

c.JSON(http.StatusOK, comments)
}

// GET /comments/:id
func FindCommentByID(c *gin.Context) {
    var cm entity.Comment
    if tx := configs.DB().Preload("User").Preload("Thread").Preload("Parent").First(&cm, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, cm)
}

// PUT /comments/:id
func UpdateComment(c *gin.Context) {
    var payload entity.Comment
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    db := configs.DB()
    var cm entity.Comment
    if tx := db.First(&cm, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    if err := db.Model(&cm).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 /comments/:id
func DeleteCommentByID(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM comments 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"})
}

//Reaction.go
package controllers

import (
    "net/http"
    "strings"
)

```

```

"example.com/sa-gameshop/configs"
"example.com/sa-gameshop/entity"
"github.com/gin-gonic/gin"
)

func targetExists(targetType string, targetID any) bool {
    db := configs.DB()
    switch strings.ToLower(targetType) {
    case "thread":
        var t entity.Thread
        return db.First(&t, targetID).RowsAffected > 0
    case "comment":
        var m entity.Comment
        return db.First(&m, targetID).RowsAffected > 0
    default:
        return false
    }
}

// POST /reactions
func CreateReaction(c *gin.Context) {
    var body entity.Reaction
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request body"})
        return
    }

    // เช็ค User
    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
    }

    // เช็คเป้าหมาย
    if !targetExists(body.TargetType, body.TargetID) {
        c.JSON(http.StatusBadRequest, gin.H{"error": "invalid target"})
        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 /reactions (required: ?target_type=...&target_id=... ; optional: ?user_id=...)
func FindReactions(c *gin.Context) {
    var rows []entity.Reaction
}

```

```

tt := c.Query("target_type")
tid := c.Query("target_id")
uid := c.Query("user_id")

if tt == "" || tid == "" {
    c.JSON(http.StatusBadRequest, gin.H{"error": "must be parameters 'target_type' and 'target_id'"})
    return
}

tx := configs.DB().Model(&entity.Reaction{})
    Where("target_type = ? AND target_id = ?", tt, tid)
if uid != "" {
    tx = tx.Where("user_id = ?", uid)
}

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

// GET /reactions/id
func FindReactionByID(c *gin.Context) {
    var row entity.Reaction
    if tx := configs.DB().First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, row)
}

// PUT /reactions/:id
func UpdateReaction(c *gin.Context) {
    var payload entity.Reaction
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    db := configs.DB()
    var row entity.Reaction
    if tx := db.First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    // ต้าแก้ target ให้เป็นค่าวาย
    if payload.TargetType != "" || payload.TargetID != 0 {
        tt := payload.TargetType

```

```

if tt == "" {
    tt = row.TargetType
}
tid := payload.TargetID
if tid == 0 {
    tid = row.TargetID
}
if !targetExists(tt, tid) {
    c.JSON(http.StatusBadRequest, gin.H{"error": "invalid target"})
    return
}
if err := db.Model(&row).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 /reactions/:id
func DeleteReactionByID(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM reactions 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"})
}

//Attachment.go
package controllers

import (
    "net/http"
    "strings"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

func attachmentTargetExists(targetType string, targetID any) bool {
    switch strings.ToLower(targetType) {
    case "thread":
        var t entity.Thread
        return configs.DB().First(&t, targetID).RowsAffected > 0
    case "comment":
        var m entity.Comment
        return configs.DB().First(&m, targetID).RowsAffected > 0
    default:

```

```

        return false
    }
}

// POST /attachments
func CreateAttachment(c *gin.Context) {
    var body entity.Attachment
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request body"})
        return
    }

    // เช็ค User
    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
    }

    // เช็ค Target
    if !attachmentTargetExists(body.TargetType, body.TargetID) {
        c.JSON(http.StatusBadRequest, gin.H{"error": "invalid target"})
        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 /attachments (required: ?target_type=...&target_id=... ; optional: ?user_id=...)
func FindAttachments(c *gin.Context) {
    var rows []entity.Attachment
    tt := c.Query("target_type")
    tid := c.Query("target_id")
    uid := c.Query("user_id")

    if tt == "" || tid == "" {
        c.JSON(http.StatusBadRequest, gin.H{"error": "must be parameters 'target_type' and 'target_id'"})
        return
    }

    tx := configs.DB().Preload("User").Model(&entity.Attachment{}).
        Where("target_type = ? AND target_id = ?", tt, tid)
    if uid != "" {
        tx = tx.Where("user_id = ?", uid)
    }
}

```

```

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

// GET /attachments/:id
func FindAttachmentByID(c *gin.Context) {
    var row entity.Attachment
    if tx := configs.DB().Preload("User").First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, row)
}

// PUT /attachments/:id
func UpdateAttachment(c *gin.Context) {
    var payload entity.Attachment
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    db := configs.DB()
    var row entity.Attachment
    if tx := db.First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    // ถ้าแก้ target ให้ตรวจสอบเหมือนเดิม
    if payload.TargetType != "" || payload.TargetID != 0 {
        tt := payload.TargetType
        if tt == "" {
            tt = row.TargetType
        }
        tid := payload.TargetID
        if tid == 0 {
            tid = row.TargetID
        }
        if !attachmentTargetExists(tt, tid) {
            c.JSON(http.StatusBadRequest, gin.H{"error": "invalid target"})
            return
        }
    }
    if err := db.Model(&row).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 /attachments/:id
func DeleteAttachmentByID(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM 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"})
}

//Game.go
package controllers

import (
    "net/http"

    "example.com/sa-gameshop/configs"
    "example.com/sa-gameshop/entity"
    "github.com/gin-gonic/gin"
)

// POST /games
func CreateGame(c *gin.Context) {
    var body entity.Game
    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 /games (optional: ?name=...)
func FindGames(c *gin.Context) {
    var games []entity.Game
    name := c.Query("name")

    tx := configs.DB().Model(&entity.Game{})
    if name != "" {
        tx = tx.Where("game_name LIKE ?", "%" + name + "%")
    }

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

```

```

        return
    }
    c.JSON(http.StatusOK, games)
}

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

// PUT /games/:id
func UpdateGame(c *gin.Context) {
    var payload entity.Game
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    var game entity.Game
    db := configs.DB()
    if tx := db.First(&game, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    if err := db.Model(&game).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 /games/:id
func DeleteGameByID(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM games 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"})
}

//UserGame.go
package controllers

import (
    "net/http"
)

```

```

"example.com/sa-gameshop/configs"
"example.com/sa-gameshop/entity"
"github.com/gin-gonic/gin"
)

// POST /user-games
func CreateUserGame(c *gin.Context) {
    var body entity.UserGame
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "bad request body"})
        return
    }

    db := configs.DB()
    var user entity.User
    if tx := db.Where("id = ?", body.UserID).First(&user); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "user_id not found"})
        return
    }

    var game entity.Game
    if tx := db.Where("id = ?", body.GameID).First(&game); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "game_id not found"})
        return
    }

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

// GET /user-games (optional: ?user_id=... ?game_id=...)
func FindUserGames(c *gin.Context) {
    var rows []entity.UserGame
    db := configs.DB()

    userID := c.Query("user_id")
    gameID := c.Query("game_id")

    tx := db.Preload("User").Preload("Game").Model(&entity.UserGame{})
    if userID != "" {
        tx = tx.Where("user_id = ?", userID)
    }
    if gameID != "" {
        tx = tx.Where("game_id = ?", gameID)
    }
}

```

```

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

// GET /user-games/:id
func FindUserGameByID(c *gin.Context) {
    var row entity.UserGame
    if tx := configs.DB().Preload("User").Preload("Game").First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    c.JSON(http.StatusOK, row)
}

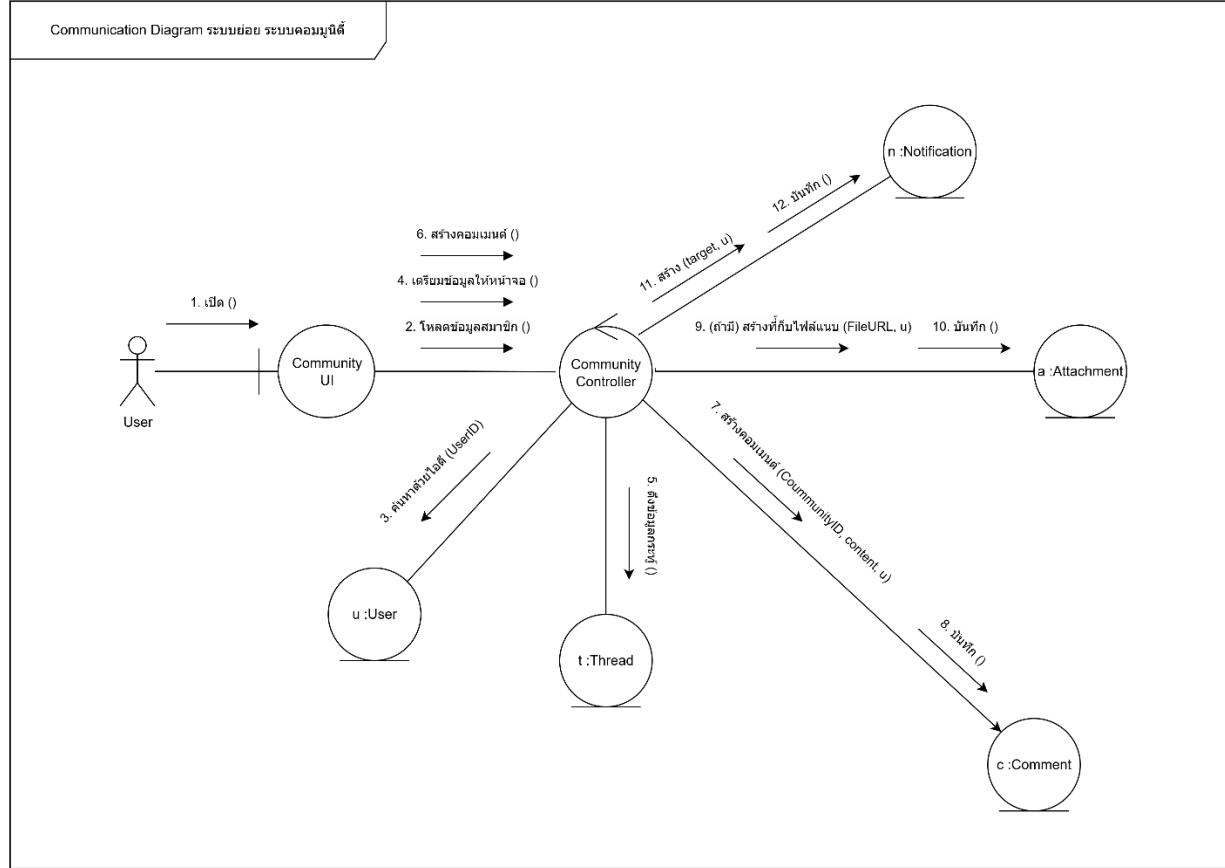
// PUT /user-games/:id
func UpdateUserGame(c *gin.Context) {
    var payload entity.UserGame
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    db := configs.DB()
    var row entity.UserGame
    if tx := db.First(&row, c.Param("id")); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
    if err := db.Model(&row).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 /user-games/:id
func DeleteUserGameByID(c *gin.Context) {
    if tx := configs.DB().Exec("DELETE FROM user_games 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"})
}

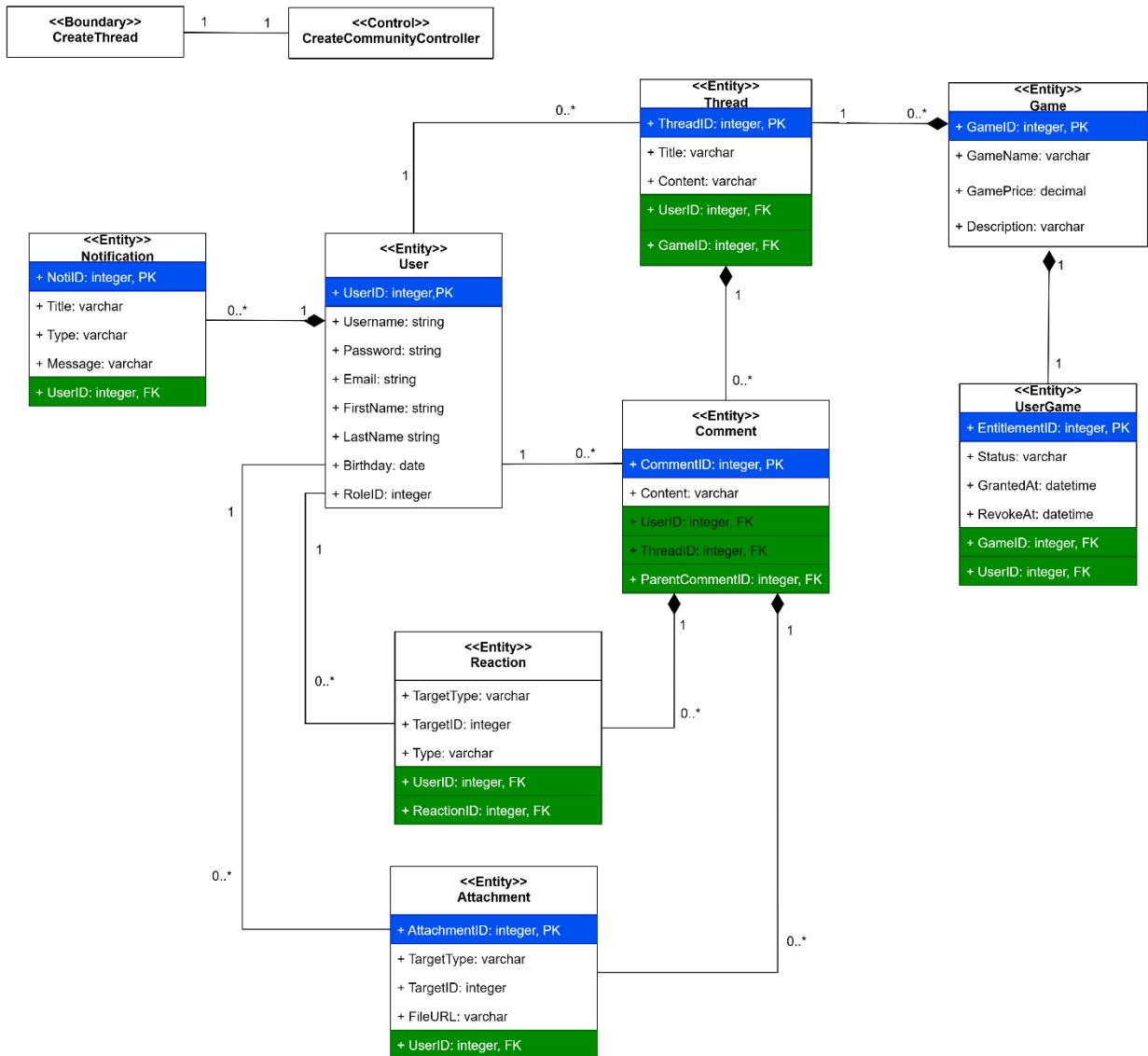
```

Communication Diagram

Activity	เป็นคำสั่งสำหรับระบบได้ หรือไม่	ถ้าเป็น ,วัตถุที่รับหน้าที่ ทำงาน คืออะไร	ชื่อคำสั่ง , ตัวอย่าง parameter
Click “เปิดคอมมูนิตี้”	เป็นคำสั่ง สั่งให้ UI โหลดหน้า	:CommunityUI	เปิด ()
โหลดข้อมูลสมาชิก	เป็นคำสั่ง เกิดการสั่งให้โหลด ข้อมูลของสมาชิกในระบบ	:CommunityController	โหลดข้อมูลสมาชิก(userId)
ค้นหา/oideسمาร์กในฐานข้อมูล	เป็นคำสั่ง	u :User	ค้นหาด้วย/oide (userId)
ขอข้อมูลกระทู้พร้อมคอมเม้นต์	เป็นคำสั่ง	:CommunityController	โหลดข้อมูล (threadId)
ดึงข้อมูลกระทู้	เป็นคำสั่ง เพื่อดึงข้อมูลกระทู้ ออกมานา	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 ระบบรีวิวและให้คะแนน

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

นอกเหนือไปจากนี้ เมื่อมีการลบเกมออกจากระบบ ระบบจะต้องทำการลบรีวิวและคะแนนทั้งหมดที่เกี่ยวข้องกับเกมนั้น โดยอัตโนมัติ ระบบยังรองรับการแสดงผลตัวอย่างสำหรับผู้พัฒนาเกม เช่น จำนวนยอดการสั่งซื้อเกม, จำนวนรีวิวที่ได้รับ และคะแนนเฉลี่ยจากผู้เล่น เพื่อใช้ในการประเมินและปรับปรุงเกมในอนาคต

User Story ระบบรีวิวและให้คะแนน

ในบทบาทของ (As a) ผู้ใช้งาน

(I want to) เขียนรีวิวและให้คะแนนเกมที่ซื้อแล้ว

(So that) ฉันจะสามารถแบ่งปันประสบการณ์กับผู้ใช้คนอื่น

Output บนหน้าจอ

- ผู้ใช้งานเลือกเกมที่ตนเคยซื้อไปแล้วกดสร้างรีวิว และใส่ข้อมูลการรีวิวและคะแนน-> จากนั้นระบบทำการบันทึกข้อมูลคะแนนที่ให้และรีวิวที่เขียน แล้วแสดง Indicator บางอย่างที่แสดงให้เห็นว่าได้บันทึกข้อมูลเรียบร้อยแล้ว

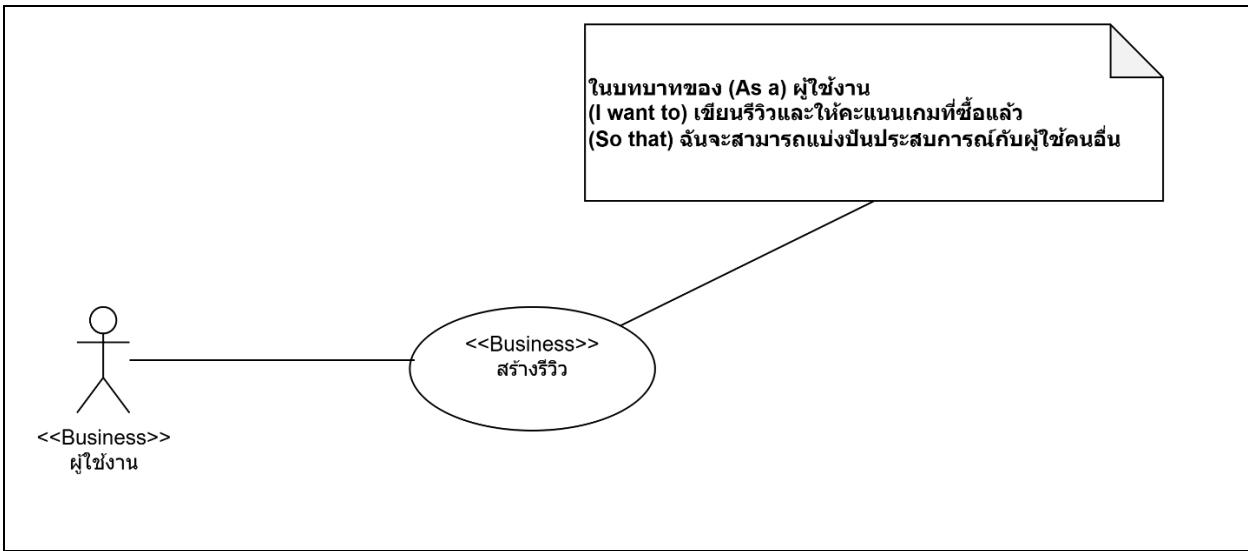
Output ของข้อมูล

- ระบบเรียกข้อมูลเกมเพื่อให้ผู้ใช้งานกดสร้างรีวิวของเกมนั้น -> ระบบทำการบันทึกข้อมูลรีวิว กับเกมที่ผู้ใช้งานเลือก

คำนามที่อาจจะกล้ายมาเป็นตารางในฐานข้อมูลของระบบ

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

Business Use Case Diagram (แบบเดี่ยว)



Checklist: Business Use Case Diagram

Business Actor มี <<Business>> กำกับ

Business Actor เป็นชื่อบทบาทเท่านั้น ไม่ใช่ชื่อคน ไม่เป็นชื่อบุคคล

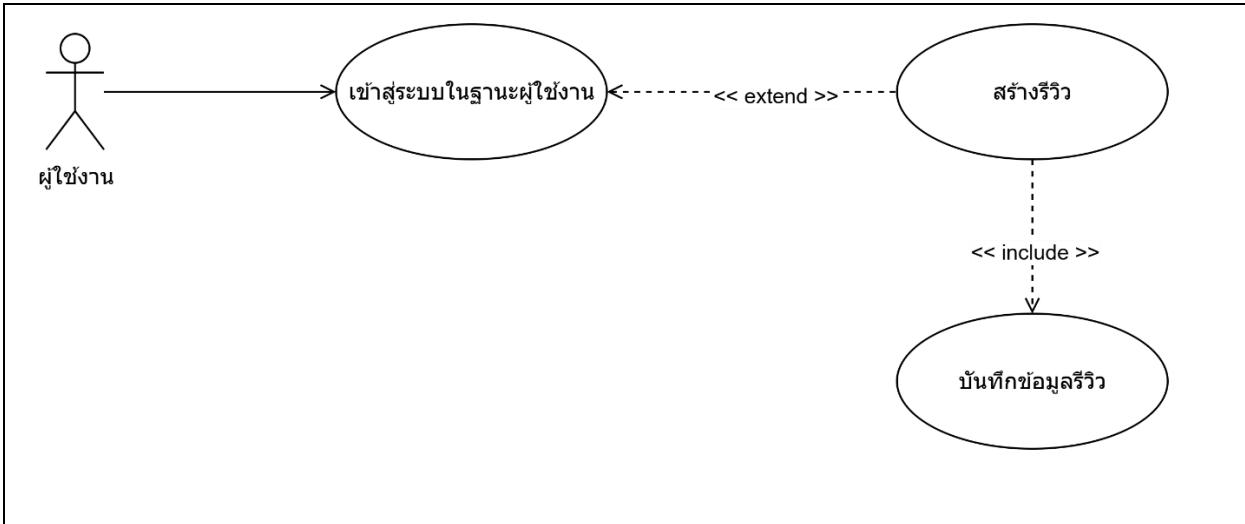
Business Use Case มี <<Business>> กำกับ

Business Use Case จัดตั้งด้วยคำที่แสดงพฤติกรรม “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 "ผู้ใช้งาน" สามารถเป็นผู้ใช้ระบบคอมพิวเตอร์ที่ระบบงานได้โดยตรงหรือไม่

ตอบ ได้ สมาชิก 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 จะยังไม่ครอบคลุม 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” ทุกครั้งหรือไม่
ตอบ ไม่

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

พิจารณาประเด็นที่ 4

ถ้าผู้ใช้งาน “เข้าระบบในฐานะผู้ใช้งาน” มาแล้วจำเป็นที่ต้อง “บันทึกข้อมูลรีวิว” ทุกครั้งหรือไม่
ตอบ ไม่ และเป็น Use Case ที่ไม่เกี่ยวข้องกัน

พิจารณาประเด็นที่ 5

ถ้าสมาชิก “สร้าง รีวิว” และ

จำเป็นต้อง "บันทึกข้อมูล รีวิว" ทุกรังหรือไม่

ตอบ ใช่ จำเป็น

แปลว่า System Use Case “สร้าง รีวิว” จะต้องรวมขั้นตอนของ

System Use Case “บันทึกข้อมูล รีวิว” 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>>---> 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) ที่ใช้เก็บข้อมูลซึ่งหมายความตามลักษณะของระบบอย่างที่ออกแบบ

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

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

วิเคราะห์ Entity ที่เกี่ยวข้อง

ReviewLike

ชื่อ Entity: Purchase

ข้อมูลที่ต้องจัดเก็บ

- Review_id_ID: FK จัดอิงไปที่ Review_ID
- User_ID: FK จัดอิงไปที่ User_ID

Review_ID	User_ID
INTEGER NOT NULL, PK	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
- Rating: integer (1-5) และไม่เป็น Null
- ReviewText: String (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 NULL	ReviewText TEXT, NOT NULL	CreatedAt DATETIME, NOT NULL
4001	1001	2001	5	สนุกมาก ระบบ ดีมาก	2025-07-24 15:00:00
4002	1002	2002	3	กราฟฟิกสวยแต่ บักเบิก	2025-07-24 15:30:00

หลักการออกแบบ User Interface

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



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

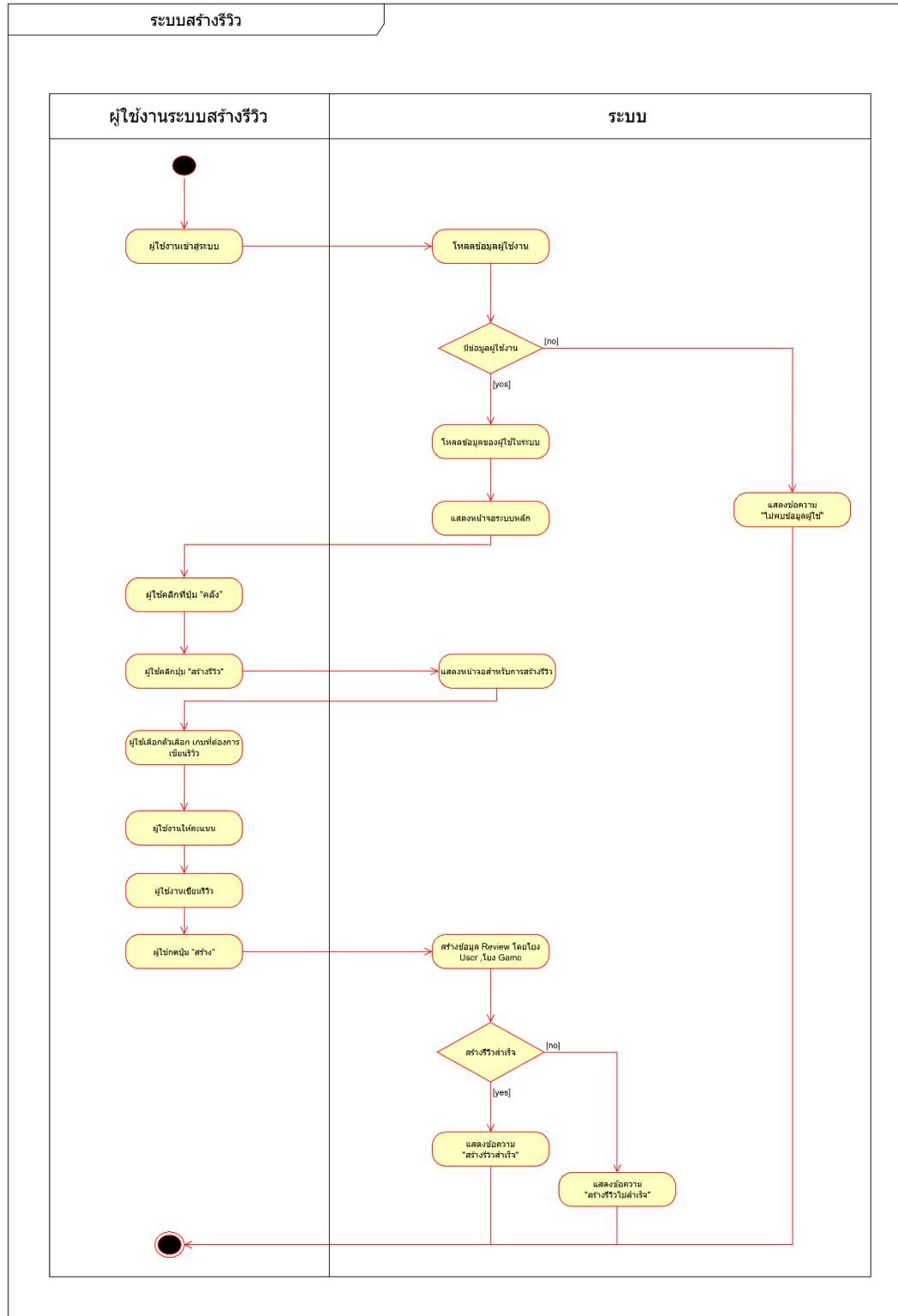
หลักสำคัญ ของการออกแบบ System Activity Diagram ได้แก่

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

System Activity Diagram ต้องสื้นสุดที่การสร้าง Output ดังนี้

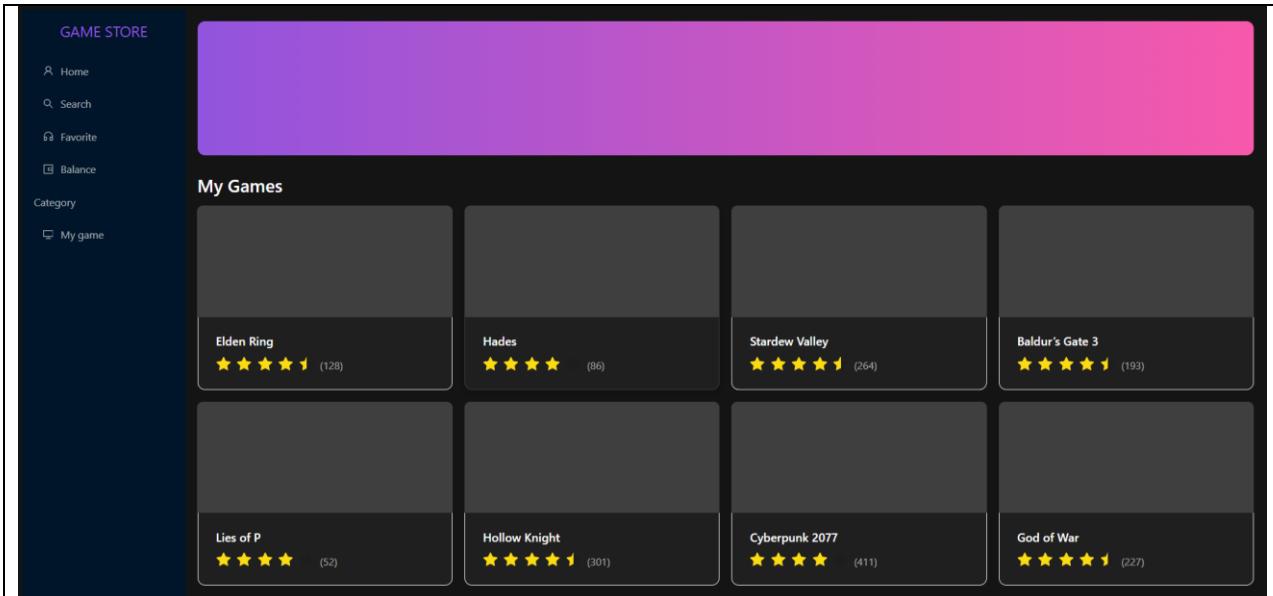
1. บันทึกข้อมูลลงใน ฐานข้อมูล
2. แสดงผลข้อมูลผ่าน หน้าจอ (User Interface) โดยต้องสอดคล้องกับ User Story ที่กำหนดไว้

Activity Diagram



Frontend

Review UI



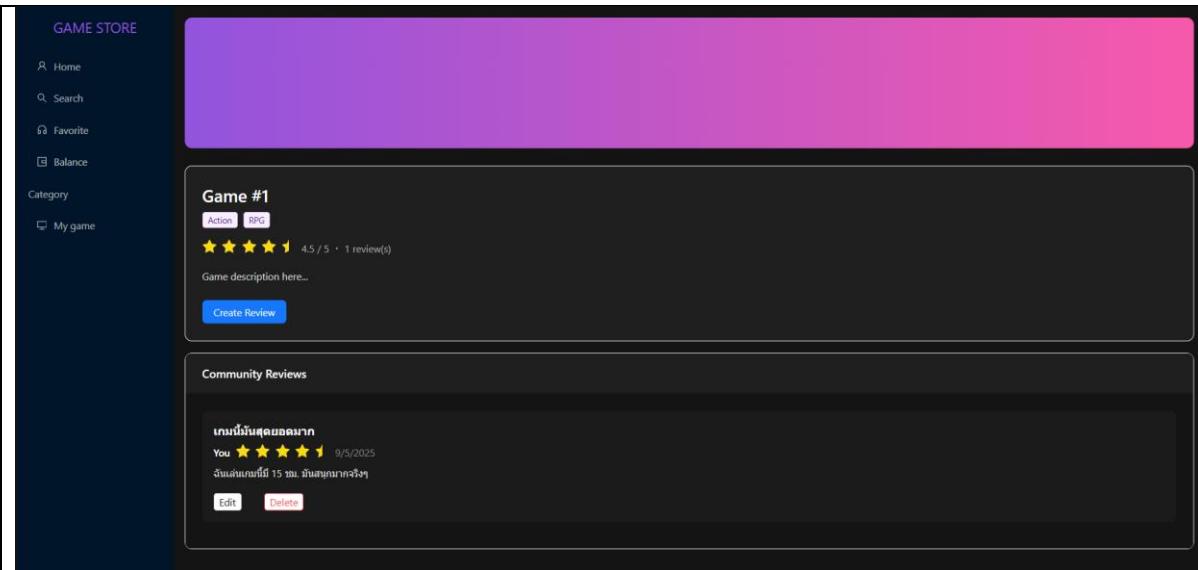
```
import { Layout, Space, Typography } from "antd";
import Sidebar from "../components/Sidebar";
import ProductGrid from "../components/ProductGrid_mygame";
import ProductGrid_mygame from "../components/ProductGrid_mygame";
const { Content } = Layout;
const { Title } = Typography;
const Mygame = () => {
  return (
    <Layout style={{ minHeight: "100vh", background: "#0f0f0f" }}>
      /* Sidebar ဘဏ္ဍာယ */
      <Sidebar />
      /* အော်လုပ် */
      <Content style={{ background: '#141414' }}>
        <div style={{ padding: 16 }}>
          <div style={{
            background: 'linear-gradient(90deg, #9254de 0%, #f759ab 100%)',
            height: 180,
            borderRadius: 10,
            marginBottom: 24,
          }}
          />
          <Title level={3} style={{ color: 'white' }}>
            My Games
          </Title>
        </div>
      </Content>
    </Layout>
  );
}
```

```

</Title>
<Space style={{ marginBottom: 16 }}>
  /* บูมพิลเตอร์อีน ๆ ใส่เพื่อได้ตามต้องการ */
</Space>
<ProductGrid_mygame />
</div>
</Content>
</Layout>
);
};

export default Mygame;

```



```

// src/pages/GameDetail.tsx
import { useMemo, useState } from "react";
import { useNavigate, useParams } from "react-router-dom";
import { Layout, Typography, Button, Card, Rate, Space, Tag } from "antd";
import Sidebar from "../components/Sidebar";
import GameReviews from "../components/GameReviews";
const { Content } = Layout;
const { Title, Paragraph, Text } = Typography;

const mockGameById = (id?: string) => {
  const base = {
    id,
    title: `Game ${id}`,
    genres: ["Action", "RPG"],
    ratingAvg: 4.5,
    description: "Game description here...",
  };
  return base;
}

```

```

return base;
};

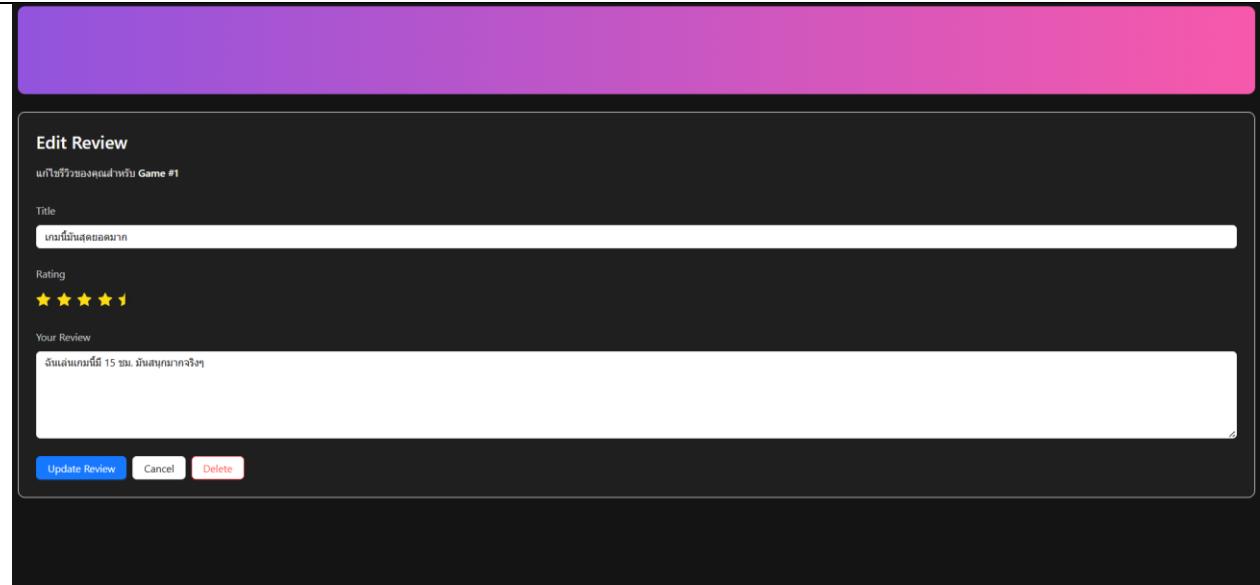
export default function GameDetail() {
  const { id: gameId } = useParams();
  const navigate = useNavigate();
  const game = useMemo(() => mockGameById(gameId), [gameId]);
  // รับสถิติรีวิวจากคอมเพเนนท์ลูก
  const [reviewStats, setReviewStats] = useState<{ count: number; avg: number }>({
    count: 0,
    avg: game.ratingAvg,
  });
  if (!gameId) return null;
  return (
    <Layout style={{ minHeight: "100vh", background: "#0f0f0f" }}>
      <Sidebar />
      <Content style={{ background: "#141414" }}>
        <div style={{ padding: 16 }}>
          {/* Banner */}
          <div
            style={{
              background: "linear-gradient(90deg, #9254de 0%, #f759ab 100%)",
              height: 180,
              borderRadius: 10,
              marginBottom: 24,
            }}
          />
          {/* ข้อมูลเกม */}
          <Card style={{ background: "#1f1f1f", color: "white", borderRadius: 10, marginBottom: 16 }}>
            <Space direction="vertical" size="small" style={{ width: "100%" }}>
              <Title level={3} style={{ color: "white", margin: 0 }}>
                {game.title}
              </Title>
              <div>
                {game.genres.map((g) => (
                  <Tag key={g} color="purple" style={{ marginBottom: 8 }}>
                    {g}
                  </Tag>
                )))
              </div>
              <div>
                <Rate allowHalf disabled value={reviewStats.avg || game.ratingAvg} />
                <Text style={{ marginLeft: 8, color: "#aaa" }}>
                  {(reviewStats.avg || game.ratingAvg).toFixed(1)} ▪ {reviewStats.count} review(s)
                </Text>
              </div>
              <Paragraph style={{ color: "#ddd", marginTop: 8 }}>{game.description}</Paragraph>
              {/* ปุ่มสร้างรีวิว (ถ้าขอบอยู่ตรงหัวก็เป็นรีวิว) */}
              <div>

```

```

<Button type="primary" onClick={() => navigate('/game/${gameId}/review/new')}>
  Create Review
</Button>
</div>
</Space>
</Card>
/* 🔍 ส่วนรีวิวที่แยกเป็นคอมโพเนนต์แล้ว */
<GameReviews
  gameId={gameId}
  onStatsChange={setReviewStats}
  // รักษาสถานะการ Create อยู่ในทั้งหมด ให้เปิดออกป๊อปอัปหน้าจอ:
  // showCreateButton
/>
</div>
</Content>
</Layout>
);
}

```



```

// src/pages/ReviewForm.tsx
import { useEffect, useMemo, useState } from "react";
import { useNavigate, useParams } from "react-router-dom";
import { Typography, Card, Form, Rate, Input, Button, Space, message, Popconfirm } from "antd";
import { create, getById, update, remove } from "../services/reviewsApi";
const { Title, Paragraph } = Typography;

```

```

const { TextArea } = Input;
export default function ReviewForm() {
  const { gameId, reviewId } = useParams();
  const navigate = useNavigate();
  const isEdit = Boolean(reviewId);
  const [form] = Form.useForm();
  const [loading, setLoading] = useState(false);
  const gameTitle = useMemo(() => `Game #${gameId}`, [gameId]);
  useEffect(() => {
    if (isEdit && reviewId) {
      const data = getById(reviewId);
      if (!data) {
        message.error("ไม่พบรีวิวนี้");
        navigate(`/game/${gameId}`);
        return;
      }
      form.setFieldsValue({
        title: data.title,
        rating: data.rating,
        text: data.text,
      });
    }
  }, [isEdit, reviewId, form, navigate, gameId]);
  const handleSubmit = async () => {
    try {
      setLoading(true);
      const { title, rating, text } = await form.validateFields();
      const user = "You"; // mock user
      if (!gameId) { message.error("ไม่พบ gameId"); return; }

      if (isEdit && reviewId) {
        const updated = update(reviewId, { title, rating, text });
        if (!updated) throw new Error("อัปเดตไม่สำเร็จ");
        message.success("แก้ไขรีวิวสำเร็จ");
      } else {
        create({ gameId, user, title, rating, text });
        message.success("สร้างรีวิวสำเร็จ");
      }
      navigate(`/game/${gameId}`);
    } catch (err: any) {
      if (err?.errorFields) return; // validation error
      message.error(err?.message || "เกิดข้อผิดพลาด");
    } finally { setLoading(false); }
  };
  const handleDelete = () => {
    if (!reviewId) return;
    const ok = remove(reviewId);
    if (ok) { message.success("ลบรีวิวสำเร็จ"); navigate(`/game/${gameId}`); }
    else { message.error("ลบไม่สำเร็จ"); }
  };
}

```

```

};

return (
<div>
<div style={{
background: "linear-gradient(90deg, #9254de 0%, #f759ab 100%)",
height: 120, borderRadius: 10, marginBottom: 24
}} />
<Card style={{ background: "#1f1f1f", color: "white", borderRadius: 10 }}>
<Space direction="vertical" style={{ width: "100%" }} size="middle">
<Title level={3} style={{ color: "white", margin: 0 }}>
{isEdit ? "Edit Review" : "Create Review"}
</Title>
<Paragraph style={{ color: "#ddd" }}>
{isEdit ? "ແກ້ເຂົ້າວິວຂອງຄູນສໍາຫັບ" : "ເພີ່ມເຂົ້າວິວສໍາຫັບ"} {" " }
<span style={{ color: "#fff", fontWeight: 600 }}>{gameTitle}</span>
</Paragraph>
<Form form={{form}} layout="vertical" requiredMark={false}>
{/*  Title */}
<Form.Item
name="title"
label={{<span style={{ color: "#ccc" }}>Title</span>}}
rules={[
{ required: true, message: "ກະລຸນາກໍາຕ່ອງສັບດີນ ແຕ່ performance ສະດຸດ" },
{ min: 4, message: "ອ່ານັ້ນຕ້ອງ 4 ຕົວອັກຊາ" },
]}
>
<Input placeholder="ເຊັ່ນ ສາມາດ ຮະບັບຕ່ອງສັບດີນ ແຕ່ performance ສະດຸດ" />
</Form.Item>
<Form.Item
name="rating"
label={{<span style={{ color: "#ccc" }}>Rating</span>}}
rules={[{ required: true, message: "ໃຫ້ຄະແນນຕ້ອງຄັບ" }]}
>
<Rate allowHalf />
</Form.Item>
<Form.Item
name="text"
label={{<span style={{ color: "#ccc" }}>Your Review</span>}}
rules={[
{ required: true, message: "ກິນເກົ່າວິວທັງຄັບ" },
{ min: 10, message: "ອ່ານັ້ນຕ້ອງ 10 ຕົວອັກຊາ" },
]}
>
<TextArea rows={5} placeholder="ບອກເລົາປະສົບການຟ້າ ຄວາມເຫັນ ຈຸດເດັ່ນ/ດ້ອຍ ແລະ" />
</Form.Item>
<Space>
<Button type="primary" loading={loading} onClick={handleSubmit}>
{isEdit ? "Update Review" : "Submit Review"}
</Button>

```

```
<Button onClick={() => navigate('/game/${gameId}')}>Cancel</Button>
{isEdit && (
  <Popconfirm
    title="ລັບຮືວໃຈ?"
    okText="ລັບ"
    cancelText="ຍົກເລີກ"
    okButtonProps={{ danger: true }}
    onConfirm={handleDelete}
  >
    <Button danger>Delete</Button>
  </Popconfirm>
)
</Space>
</Form>
</Space>
</Card>
</div>
);
}
```

Backend

Game Entity

```
package entity
import "gorm.io/gorm"
type Game struct {
    gorm.Model
    GameName string `json:"game_name" gorm:"type:varchar(100);not null"`
    KeyGame string `json:"key_Game" gorm:"type:varchar(100);uniqueIndex;not null"`

    // One-to-Many: Game -> Reviews
    Reviews []Review `gorm:"foreignKey:GameID" json:"reviews"`

    // Many-to-Many: Game <-> Promotions (join table: promotion_games)
    Promotions []Promotion `gorm:"many2many:promotion_games" json:"promotions"`
}
```

Review_Like Entity

```
package entity
import "gorm.io/gorm"
type Review_Like struct {
    gorm.Model
    ReviewID uint `json:"review_id" gorm:"index:idx_review_user,unique"`
    Review *Review `gorm:"foreignKey:ReviewID" json:"review"`
    UserID uint `json:"user_id" gorm:"index:idx_review_user,unique"`
    User *User `gorm:"foreignKey:UserID" json:"user"`
}
```

Review Entity

```
package entity
import "gorm.io/gorm"
type Review struct {
    gorm.Model
    Rating int `json:"rating" gorm:"not null"`
    ReviewText string `json:"review_text" gorm:"type:text"`
    // FK → Users
    UserID uint `json:"user_id"`
    User *User `gorm:"foreignKey:UserID" json:"user"`
    // FK → Games
}
```

```

GameID uint `json:"game_id"`
Game *Game `gorm:"foreignKey:GameID" json:"game"`
// 1:N ReviewLike
Likes []Review_Like `gorm:"foreignKey:ReviewID" json:"likes"`
}

```

User Entity

```

package entity
import (
    "time"
    "gorm.io/gorm"
)
type User struct {
    gorm.Model
    Username string `json:"username" gorm:"type:varchar(50);uniqueIndex;not null"`
    Password string `json:"password" gorm:"type:varchar(255);not null" // เก็บรหัสผ่านแบบ hash`
    Email string `json:"email" gorm:"type:varchar(120);uniqueIndex;not null"`
    FirstName string `json:"first_name" gorm:"type:varchar(100)"`
    LastName string `json:"last_name" gorm:"type:varchar(100)"`
    Birthday *time.Time `json:"birthday" // ใช้ pointer เพื่อให้เป็น NULL ได้`

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

    Reviews []Review `gorm:"foreignKey:UserID;constraint:OnUpdate:CASCADE,OnDelete:SET NULL;" json:"reviews"`
    Promotions []Promotion `gorm:"foreignKey:UserID;constraint:OnUpdate:CASCADE,OnDelete:SET NULL;" json:"promotions"`
}

```

Controller review_controller

```

package controllers
import (
    "net/http"
    "strconv"
    "github.com/gin-gonic/gin"
    "projectSA/configs"
    "projectSA/entity"
)
type ReviewRequest struct {
    Rating int `json:"rating" binding:"required,gte=1,lte=5"`
    ReviewText string `json:"review_text" binding:"required"`
    UserID uint `json:"user_id" binding:"required"`
}

```

```

GameID  uint  `json:"game_id"  binding:"required"
}

// GET /api/v1/reviews?limit=20&offset=0&search=...
func ListReviews(c *gin.Context) {
    limit, offset := 20, 0
    if v, err := strconv.Atoi(c.DefaultQuery("limit", "20")); err == nil && v > 0 {
        limit = v
    }
    if v, err := strconv.Atoi(c.DefaultQuery("offset", "0")); err == nil && v >= 0 {
        offset = v
    }
    search := c.Query("search")

    q := configs.GetDB().Model(&entity.Review{}).
        Preload("User").Preload("Game").Preload("Likes")
    if search != "" {
        like := "%" + search + "%"
        q = q.Where("review_text LIKE ?", like)
    }
    var reviews []entity.Review
    if err := q.Limit(limit).Offset(offset).Find(&reviews).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"items": reviews, "limit": limit, "offset": offset})
}

// POST /api/v1/reviews
func CreateReview(c *gin.Context) {
    var req ReviewRequest
    if err := c.ShouldBindJSON(&req); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    r := entity.Review{
        Rating:   req.Rating,
        ReviewText: req.ReviewText,
        UserID:   req.UserID,
        GameID:   req.GameID,
    }
    if err := configs.GetDB().Create(&r).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusCreated, r)
}

// GET /api/v1/reviews/:id
func GetReview(c *gin.Context) {
}

```

```

id := c.Param("id")
var r entity.Review
if err := configs.GetDB().
    Preload("User").
    Preload("Game").
    Preload("Likes").
    First(&r, id).Error; err != nil {
    c.JSON(http.StatusNotFound, gin.H{"error": "review not found"})
    return
}
c.JSON(http.StatusOK, r)
}

// PUT /api/v1/reviews/:id
func UpdateReview(c *gin.Context) {
    id := c.Param("id")
    var body struct {
        Rating     *int     `json:"rating"`
        ReviewText *string  `json:"review_text"`
    }
    if err := c.ShouldBindJSON(&body); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    db := configs.GetDB()
    var r entity.Review
    if err := db.First(&r, id).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": "review not found"})
        return
    }
    if body.Rating != nil {
        r.Rating = *body.Rating
    }
    if body.ReviewText != nil {
        r.ReviewText = *body.ReviewText
    }
    if err := db.Save(&r).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, r)
}

// DELETE /api/v1/reviews/:id
func DeleteReview(c *gin.Context) {
    id := c.Param("id")
    if err := configs.GetDB().Delete(&entity.Review{}, id).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
}

```

```

        }

        c.Status(http.StatusNoContent)
    }

// POST /api/v1/reviews/:id/like
func LikeReview(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": err.Error()})
        return
    }
    rid, _ := strconv.ParseUint(id, 10, 64)
    like := entity.Review_Like{ReviewID: uint(rid), UserID: body.UserID}
    if err := configs.GetDB().
        Where("review_id=? AND user_id=?", like.ReviewID, like.UserID).
        FirstOrCreate(&like).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"liked": true})
}

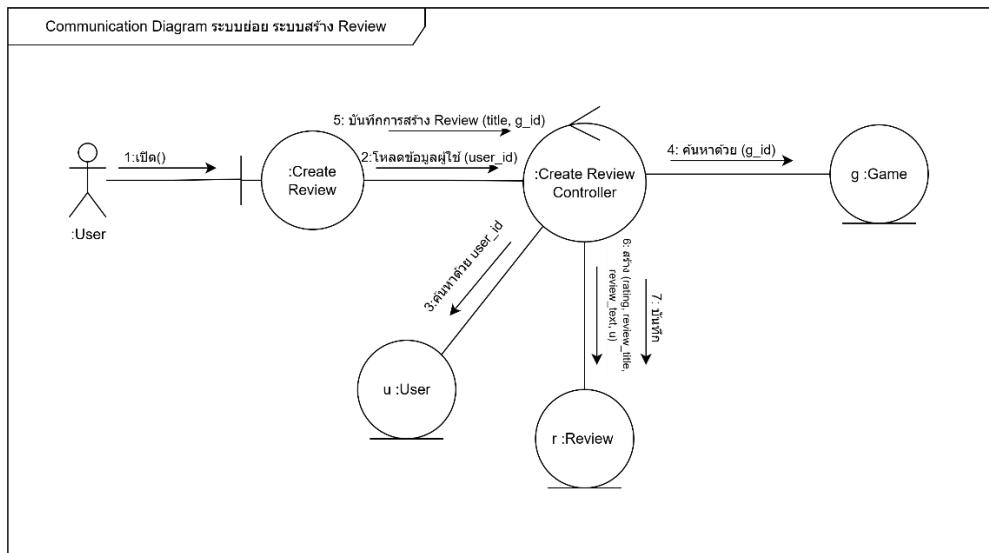
// DELETE /api/v1/reviews/:id/like?user_id=123
func UnlikeReview(c *gin.Context) {
    id := c.Param("id")
    uidStr := c.Query("user_id")
    uid, _ := strconv.ParseUint(uidStr, 10, 64)

    if err := configs.GetDB().
        Where("review_id=? AND user_id=?", id, uid).
        Delete(&entity.Review_Like{}).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"liked": false})
}

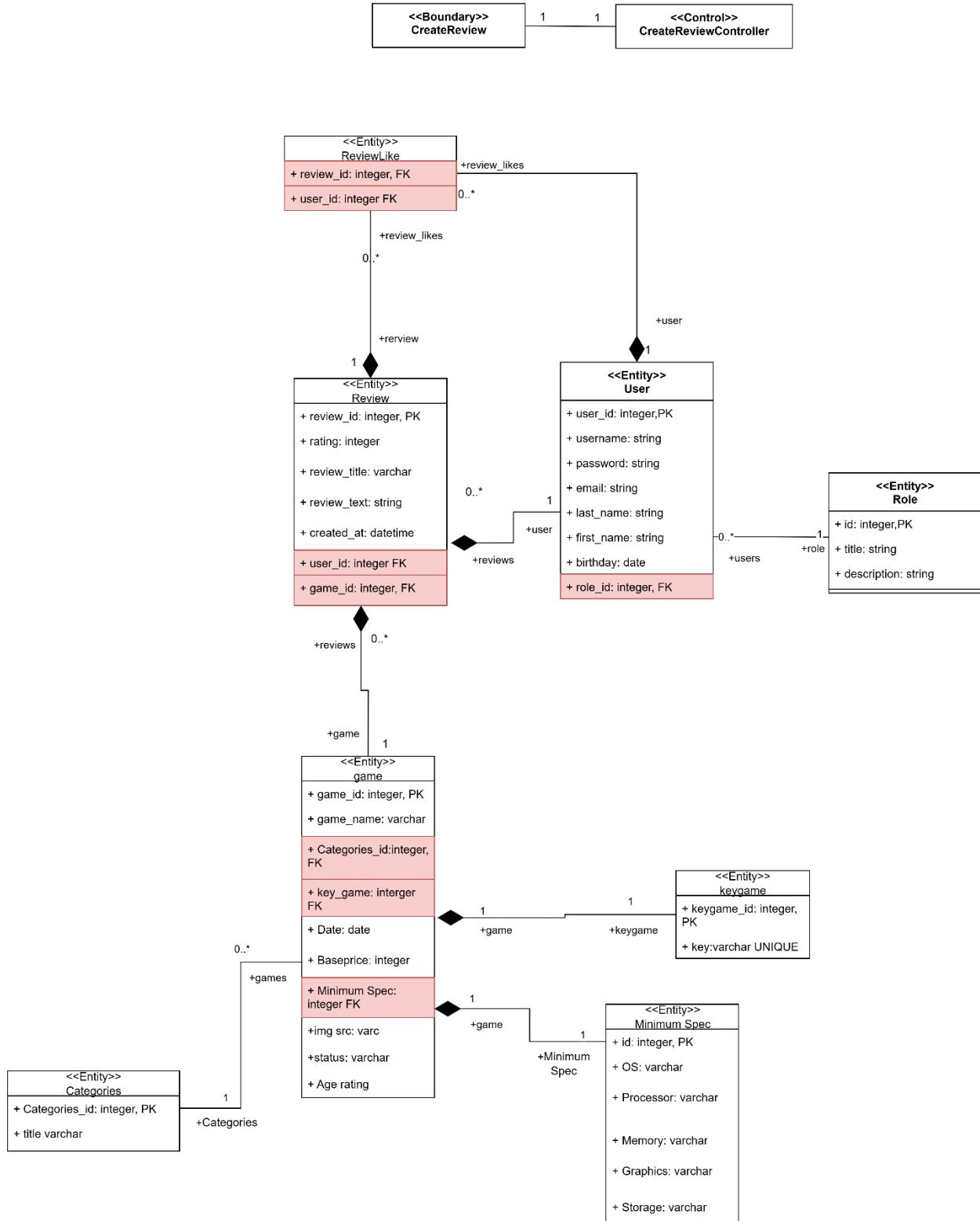
```

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)
ค้นหาโดยสมาชิกในฐานข้อมูล	เป็นคำสั่ง	u :user	ค้นหาด้วยไอดี (user_id)
โหลดข้อมูลรายการเกมที่เคยซื้อ ไปแล้ว	เป็นคำสั่ง	g :Game	ค้นหาด้วยไอดี(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 เพื่อป้องกันข้อมูลพิດพลาดหรือโปรโมชั่นที่ไม่มีสินค้า

User Story ระบบจัดการโปรโมชั่น

ในบทบาทของ (As a) ผู้ดูแลระบบ (Admin)

(I want to) สร้างโปรโมชั่นสำหรับเกมต่าง ๆ

(So that) ฉันจะสามารถกระตุ้นยอดขายและดึงดูดผู้ใช้งานให้ซื้อเกมมากขึ้นได้

Output บนหน้าจอ

- ผู้ดูแลระบบ (Admin) กดสร้างโปรโมชั่นเลือกเกมที่อยากจะให้ร่วมโปรโมชั่นแล้วเพิ่มส่วนลด-> จากนั้นระบบทำการบันทึกข้อมูลโปรโมชั่น และแสดงโปรโมชั่นบนหน้าหลักของร้านค้า

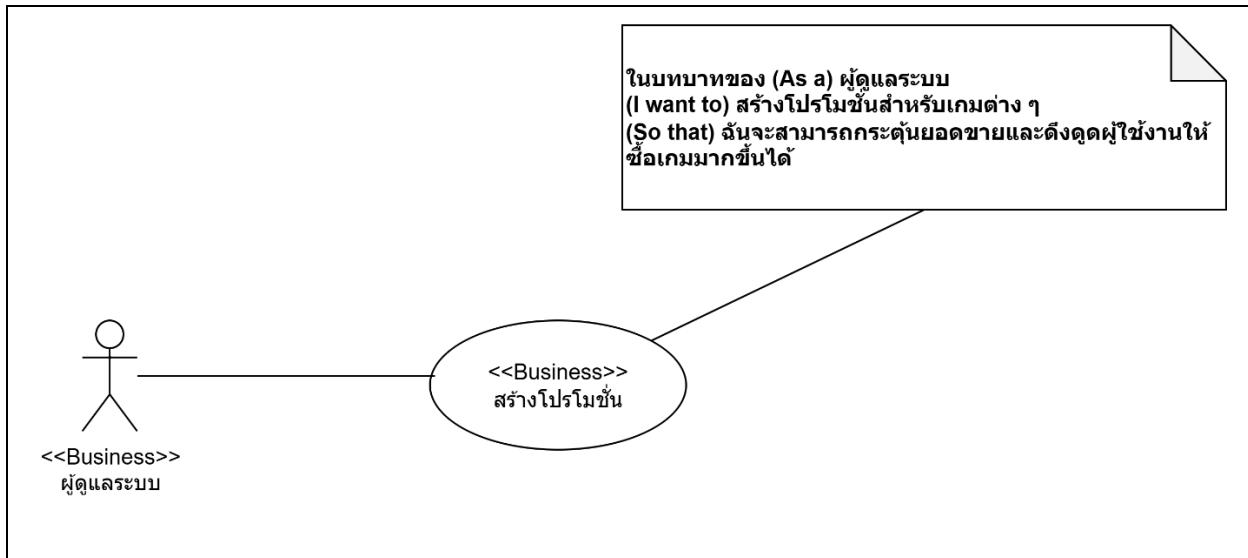
Output ของข้อมูล

- 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)



Checklist: Business Use Case Diagram

Business Actor มี <<Business>> กำກັບ

Business Actor เป็นຊື່ອບທບາຫທ່ານັ້ນ ໄນໃຫ້ຊື່ອົກ ໄນເປັນຊື່ອບຸຄຄລ

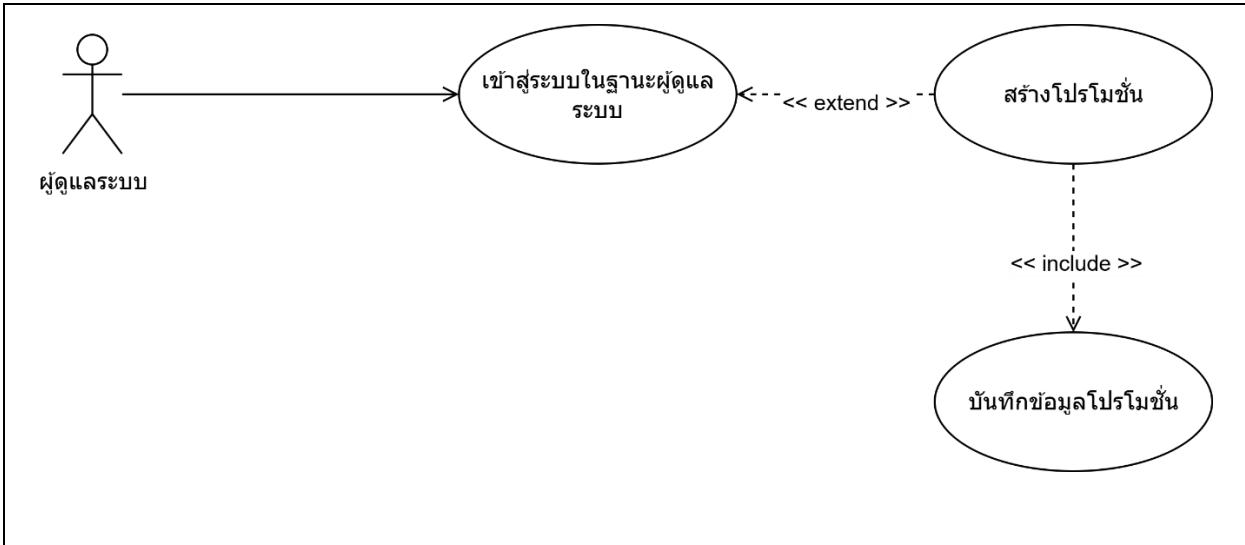
Business Use Case มี <<Business>> กำກັບ

Business Use Case ຂຶ້ນຕັ້ນດ້ວຍຄຳທີ່ແສດງພຸຕິກຣມ “ຄໍາກີ່າຍາ”

ພຸຕິກຣມຂອງ Business Use Case ຕ້ອງມາຈາກ User Story

ມີ Note ທີ່ແສດງ User Story ອູ້ໃນ Diagram

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

เพราฉะนั้น ผู้ดูแลระบบ จะถูกจัดเป็น System Actor ได้

พิจารณาประเด็นที่ 2

Business Use Case “สร้างໂປຣມີ່ນ” สามารถถูกลายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบคอมพิวเตอร์ได้หรือไม่
ต้องมีขั้นตอนทาง Security มาเกี่ยวหรือไม่

ตอบ Business Use Case “สร้างໂປຣໂມໜັນ” ສາມາຄະລາຍໄປເປັນ ກິຈกรรมທີ່ເກີ່ວຂຶ້ອງກັບຮບຄວມພິວເຕອີ່ໄດ້
ແລະຕ້ອງການຂັ້ນຕອນທາງ Security (ມີ Privacy ມາເກີ່ວຂຶ້ອງ)

ເພົ່າຈະນັ້ນ Business Use Case “ສ້າງໂປຣໂມໜັນ” ຈະກລາຍເປັນ System Use Case ມາກກວ່າ 1 Use Case
ຈະປະກອບປັບປຸງ

- 1 System Use Case ສໍາຮັບ “ເຂົ້າຮບບໃນຮູ້ານຸ້າແລຮບບ”
 - 2 System Use Case ສໍາຮັບ “ສ້າງໂປຣໂມໜັນ” ແຕ່ກິຈกรรมຂອງຮບບ ຈະຢັ້ງໄມ່ຄຽບຕາມ Requirements
ທີ່ຕ້ອງການ
 - 3 System Use Case ສໍາຮັບ “ບັນທຶກຂໍ້ມູນໂປຣໂມໜັນ”
-
- System Use Case ສໍາຮັບ “ເຂົ້າຮບບໃນຮູ້ານຸ້າແລຮບບ”
 - System Use Case ສໍາຮັບ “ສ້າງໂປຣໂມໜັນ”
 - System Use Case ສໍາຮັບ “ບັນທຶກຂໍ້ມູນໂປຣໂມໜັນ”

ພິຈານາປະເດືອນທີ 3

ດ້າສາມາຊີກ “ເຂົ້າຮບບໃນຮູ້ານຸ້າສາມາຊີກ” ມາແລ້ວຈຳເປັນທີ່ຕ້ອງ “ສ້າງໂປຣໂມໜັນ” ທຸກຄັ້ງທີ່ອຳນິ່ມ
ຕອບໄມ່

ແປລວ່າ System Use Case “ສ້າງໂປຣໂມໜັນ” ເປັນກິຈกรรมທາງເລືອກຫລັງຈາກທຳ System Use Case “ເຂົ້າຮບບ
ໃນຮູ້ານຸ້າແລຮບບ”

ພິຈານາປະເດືອນທີ 4

ດ້າສາມາຊີກ “ເຂົ້າຮບບໃນຮູ້ານຸ້າສາມາຊີກ” ມາແລ້ວຈຳເປັນທີ່ຕ້ອງ “ບັນທຶກຂໍ້ມູນໂປຣໂມໜັນ” ທຸກຄັ້ງທີ່ອຳນິ່ມ
ຕອບໄມ່ ແລະເປັນ Use Case ທີ່ໄມ່ເກີ່ວຂຶ້ອງກັນ

พิจารณาประเด็นที่ 5

ถ้าผู้ดูแลระบบ “สร้างโปรโมชั่น” และ

จำเป็นต้อง “บันทึกข้อมูลโปรโมชั่น” ทุกครั้งหรือไม่

ตอบ ใช่ จำเป็น

แปลว่า System Use Case “สร้างโปรโมชั่น” จะต้องรวมขั้นตอนของ

System Use Case “บันทึกข้อมูลโปรโมชั่น” 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

Checklist สำหรับการทวนสอบความถูกต้องของ Requirements

1. งาน (ระบบย่อยของแต่ละคน) ต้องไม่ซ้ำกับเพื่อนในกลุ่ม
2. งาน (ระบบย่อยของแต่ละคน) ต้องสอดคล้องกับ Business Domain ของระบบหลัก
3. ระบบย่อยของแต่ละคน ต้องเชื่อมโยงและต่อเนื่องกับระบบย่อยของคนอื่นๆ ในทีม ไม่สามารถเป็นงานเดี่ยวที่ไม่เกี่ยวข้องกับใครเลยได้

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) ที่ใช้เก็บข้อมูลซึ่งหมายความตามลักษณะของระบบโดยที่ออกแบบ

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
- CreatedBy: FK อ้างอิงไปยัง User_ID และไม่เป็นค่า Null

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	CreatedBy 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	9002	/images/8001.jpg
8002	Weekend Sale	ลดเกมพิเศษ สุดสัปดาห์	10	2025-08- 05 00:00:00	2025-08- 07 23:59:59	9001	/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	PromotionID	GameID
INTEGER NOT NULL,PK	INTEGER NOT NULL,FK	INTEGER NOT NULL,FK
1001	8001	2001
1002	8002	2002

หลักการออกแบบ User Interface

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



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

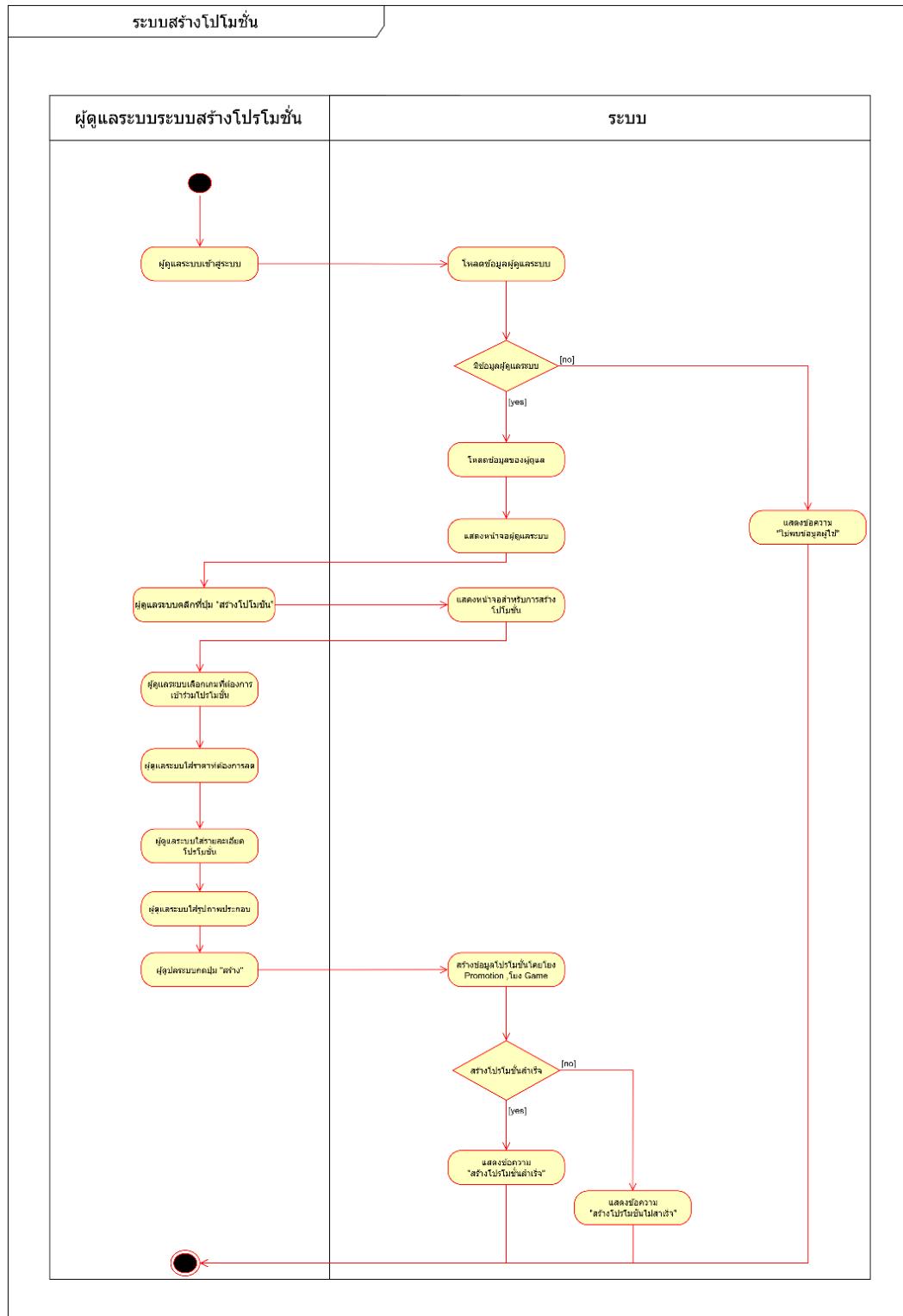
หลักสำคัญ ของการออกแบบ System Activity Diagram ได้แก่

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

System Activity Diagram ต้องสื้นสุดที่การสร้าง Output ดังนี้

3. บันทึกข้อมูลลงใน ฐานข้อมูล
4. แสดงผลข้อมูลผ่าน หน้าจอ (User Interface) โดยต้องสอดคล้องกับ User Story ที่กำหนดไว้

Activity Diagram



Frontend

Promotion UI

```
// src/pages/PromotionManager.tsx
import { useEffect, useMemo, useState } from "react";
import {
  Layout,
  Card,
  Form,
  Input,
  InputNumber,
  DatePicker,
  Switch,
  Button,
  Space,
  Table,
  Tag,
  Popconfirm,
  message,
  Typography,
  Select,
  Upload,
  Image as AntImage, // ✅ 替換นี้อิมпорт กั้มชนกับ window.Image
} from "antd";
import type { UploadFile } from "antd/es/upload/interface";
import { PlusOutlined } from "@ant-design/icons";
import dayjs, { Dayjs } from "dayjs";
import Sidebar from "../components/Sidebar";
import { list, create, update, remove, getById } from "../services/promotionsApi";
import type { Promotion } from "../services/promotionsApi";
import { listGames, mapGameTitles } from "../services/gamesApi";

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

type FormValues = {
  title: string;
}
```

```

description?: string;
discountPercent: number;
dateRange?: [Dayjs, Dayjs];
active: boolean;
gameIds: string[];
imageUrl?: string; // base64 (ບົບອັດແລ້ວ)
};

/* ===== Helpers: compress & size ===== */
function dataUrlBytes(dataUrl: string) {
  const i = dataUrl.indexOf(",");
  const b64 = i >= 0 ? dataUrl.slice(i + 1) : dataUrl;
  return Math.ceil((b64.length * 3) / 4);
}

async function compressImageToDataURL(
  file: File,
  { maxWidth = 1280, maxHeight = 1280, maxBytes = 300 * 1024 } = {}
): Promise<string> {
  const img = await new Promise<HTMLImageElement>((resolve, reject) => {
    const url = URL.createObjectURL(file);
    const image = new window.Image(); // ✅ ໃຊ້ window.Image ຂັດເຈນ
    image.onload = () => {
      URL.revokeObjectURL(url);
      resolve(image);
    };
    image.onerror = (e) => {
      URL.revokeObjectURL(url);
      reject(e);
    };
    image.src = url;
  });

  let { width, height } = img;
  const scale = Math.min(maxWidth / width, maxHeight / height, 1);
  width = Math.round(width * scale);
  height = Math.round(height * scale);

  const canvas = document.createElement("canvas");
  canvas.width = width;
  canvas.height = height;
  const ctx = canvas.getContext("2d")!;
  ctx.drawImage(img, 0, 0, width, height);

  let quality = 0.8;
  let out = canvas.toDataURL("image/jpeg", quality);
  while (dataUrlBytes(out) > maxBytes && quality > 0.35) {
    quality -= 0.1;
    out = canvas.toDataURL("image/jpeg", quality);
  }
}

```

```

    }

    return out;
}

export default function PromotionManager() {
  const [form] = Form.useForm<FormValues>();
  const [data, setData] = useState<Promotion[]>([]);
  const [editingId, setEditingId] = useState<string | null>(null);
  const [fileList, setFileList] = useState<UploadFile[]>([]);
  const [submitting, setSubmitting] = useState(false);
  const isEdit = useMemo(() => Boolean(editingId), [editingId]);

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

  const reload = () => {
    const items = list();
    console.log("[PromotionManager] reload ->", items);
    setData(items);
  };
  useEffect(() => { reload(); }, []);

  const sync fileListWithForm = () => {
    const url = form.getFieldValue("imageUrl") as string | undefined;
    setFileList(
      url ? [
        {
          uid: "1",
          name: "promo.jpg",
          status: "done",
          url,
        }
      ] : []
    );
  };

  const cancelEdit = () => {
    setEditingId(null);
    form.resetFields();
    form.setFieldsValue({
      discountPercent: 10,
      active: true,
      gameIds: [],
      imageUrl: undefined,
    });
    sync fileListWithForm();
  };
  useEffect(() => { cancelEdit(); }, []);
}

```

```

const onFinish = async (v: FormValues) => {
  setSubmitting(true);
  const payload: Omit<Promotion, "id" | "createdAt" | "updatedAt"> = {
    title: v.title.trim(),
    description: v.description?.trim(),
    discountPercent: v.discountPercent,
    gameIds: v.gameIds,
    startDate: v.dateRange?.[0]?.toDate().toISOString(),
    endDate: v.dateRange?.[1]?.toDate().toISOString(),
    active: v.active,
    imageUrl: v.imageUrl,
  };
  console.log("[PromotionManager] submit payload ->", payload);

  try {
    if (isEdit && editingId) {
      const updated = update(editingId, payload);
      console.log("[PromotionManager] updated ->", updated);
      if (!updated) throw new Error("Update failed");
      message.success("อัปเดตโปรโมชันสำเร็จ");
      setData(prev => prev.map(p => (p.id === editingId ? updated : p)));
    } else {
      const created = create(payload);
      console.log("[PromotionManager] created ->", created);
      message.success("สร้างโปรโมชันสำเร็จ");
      setData(prev => [created, ...prev]);
    }
    cancelEdit();
    reload();
  } catch (e: any) {
    console.error("[PromotionManager] submit error:", e);
    message.error(e?.message || "เกิดข้อผิดพลาดระหว่างบันทึก");
  } finally {
    setSubmitting(false);
  }
};

const onFinishFailed = ({ errorFields }: any) => {
  if (errorFields?.length) form.scrollToField(errorFields[0].name);
};

const onEditRow = (id: string) => {
  const p = getById(id);
  console.log("[PromotionManager] edit row ->", p);
  if (!p) { message.error("ไม่พบโปรโมชันนี้"); return; }
  setEditingId(id);
  form.setFieldsValue({
    title: p.title,
    description: p.description,
  });
};

```

```

discountPercent: p.discountPercent,
gameIds: p.gameIds,
dateRange: (p.startDate && p.endDate) ? [dayjs(p.startDate), dayjs(p.endDate)] : undefined,
active: p.active,
imageUrl: p.imageUrl,
});
syncFileListWithForm();
};

const onDeleteRow = (id: string) => {
try {
const ok = remove(id);
console.log("[PromotionManager] removed? ->", ok, "id:", id);
if (ok) {
message.success("ลบໂປຣໂມຫັນສໍາເລົງ");
setData(prev => prev.filter(p => p.id !== id));
} else {
message.error("ລບໄນ້ສໍາເລົງ");
}
} catch (e: any) {
console.error("[PromotionManager] remove error:", e);
message.error(e?.message || "ລບໄນ້ສໍາເລົງ");
}
if (editingId === id) cancelEdit();
};

/* ===== Upload with compression ===== */
const handleUploadChange = async (info: { file: UploadFile; fileList: UploadFile[] }) => {
const { file } = info;

if (file.status === "removed") {
form.setFieldValue("imageUrl", undefined);
setFileList([]);
return;
}

const raw = file.originFileObj;
if (raw) {
try {
const compressed = await compressImageToDataURL(raw, {
maxWidth: 1280,
maxHeight: 1280,
maxBytes: 300 * 1024, // ≤ 300KB
});
if (dataUrlBytes(compressed) > 320 * 1024) {
message.error("ຮູບໄທຜູ້ເກີນໄປ ກຽມາເລືອກໄຟເລື້ອກລົງ (≤ 300KB ທີ່ຈຳປັບອັດ)");
return;
}
}
}
}

```

```

form.setFieldValue("imageUrl", compressed);
setFileList([{ uid: "1", name: raw.name, status: "done" , url: compressed }]);
message.success(`อัปโหลดสำเร็จ (≈ ${dataUrlBytes(compressed) / 1024).toFixed(0)} KB}`);
} catch (e) {
  console.error(e);
  message.error("อัปโหลดรูปไม่สำเร็จ");
}
};

const dummyRequest = ({ onSuccess }: any) => setTimeout(() => onSuccess?("ok"), 0);

const columns = [
{
  title: "Banner",
  dataIndex: "imageUrl",
  key: "imageUrl",
  width: 120,
  render: (url?: string) =>
    url ? (
      <AntImage // ✅ ใช้ AntImage
        src={url}
        alt="promo"
        width={100}
        height={56}
        style={{ objectFit: "cover", borderRadius: 8 }}
      />
    ) : (
      <Tag>no image</Tag>
    ),
},
{
  title: "Title",
  dataIndex: "title",
  key: "title",
  render: (t: string, r: Promotion) =>
    <Space direction="vertical" size={0}>
      <Text strong>{t}</Text>
      {r.description && <Text type="secondary">{r.description}</Text>}
    </Space>
),
},
{
  title: "Discount", dataIndex: "discountPercent", key: "discountPercent", render: (v: number) => <Tag color="purple">{v}%</Tag> },
{
  title: "Games",
  dataIndex: "gameds",
  key: "gameds",
  render: (ids?: string[]) =>
    mapGameTitles(ids).map(name => (

```

```

<Tag key={name} color="geekblue" style={{ marginBottom: 4 }}>
  {name}
</Tag>
)),
},
},
{ title: "Start", dataIndex: "startDate", key: "startDate", render: (s?: string) => s ? dayjs(s).format("YYYY-MM-DD") : "-" },
{ title: "End", dataIndex: "endDate", key: "endDate", render: (s?: string) => s ? dayjs(s).format("YYYY-MM-DD") : "-" },
{ title: "Active", dataIndex: "active", key: "active", render: (a: boolean) => a ? <Tag color="green">Active</Tag> : <Tag>Inactive</Tag> },
{
  title: "Actions",
  key: "actions",
  render: (_: unknown, r: Promotion) => (
    <Space>
      <Button size="small" onClick={() => onEditRow(r.id)}>Edit</Button>
      <Popconfirm title="ລັບໄປໂນໜີ້ນີ້?" okText="ລັບ" cancelText="ຍົກເລີກ" okButtonProps={{ danger: true }} onConfirm={() => onDeleteRow(r.id)}>
        <Button size="small" danger>Delete</Button>
      </Popconfirm>
    </Space>
  ),
),
],
);
return (
  <Layout style={{ minHeight: "100vh", background: "#0f0f0f" }}>
    <Sidebar />
    <Content style={{ background: "#141414" }}>
      <div style={{ padding: 16 }}>
        <div style={{ background: "linear-gradient(90deg,#9254de 0%,#f759ab 100%)", height: 120, borderRadius: 10, marginBottom: 24 }} />
        <Card style={{ background: "#1f1f1f", color: "white", borderRadius: 10, marginBottom: 16 }}>
          <Space direction="vertical" style={{ width: "100%" }}>
            <Title level={3} style={{ color: "white", margin: 0 }}>
              {isEdit ? "Edit Promotion" : "Create Promotion"}
            </Title>
            <Form
              form={form}
              layout="vertical"
              requiredMark={false}
              initialValues={{ discountPercent: 10, active: true, gameIds: [], imageUrl: undefined }}
              onFinish={onFinish}
              onFinishFailed={onFinishFailed}
            >
              {/* hidden field ເນື່ອ base64 */}
              <Form.Item name="imageUrl" hidden>
                <input type="hidden" />
              </Form.Item>
              <Form.Item label={<span style={{ color: "#ccc" }}>Banner Image</span>}>
                <Upload
                  listType="picture-card"
                  fileList={fileList}
                </Upload>
              </Form.Item>
            </Form>
          </Space>
        </Card>
      </div>
    </Content>
  </Layout>
)

```

```

onChange={handleUploadChange}
customRequest={dummyRequest}
maxCount={1}
accept="image/*"
onRemove={() => {
  form.setFieldValue("imageUrl", undefined);
  setFileList([]);
}}
>
{fileList.length >= 1 ? null : (
  <div style={{ color: "#ccc" }}>
    <PlusOutlined />
    <div style={{ marginTop: 8 }}>Upload</div>
  </div>
)}
</Upload>
</Form.Item>
<Form.Item
  name="title"
  label={<span style={{ color: "#ccc" }}>Title</span>}
  rules={[{ required: true, message: "กรอกชื่อโปรดเมื่อชั้น" }, { min: 3, message: "อย่างน้อย 3 ตัวอักษร" }]}
>
  <Input placeholder="เช่น Mid-Year Sale" />
</Form.Item>
<Form.Item name="description" label={<span style={{ color: "#ccc" }}>Description</span>}>
  <Input.TextArea rows={3} placeholder="รายละเอียดเงื่อนไข / ข้อจำกัด" />
</Form.Item>
<Space style={{ display: "flex" }} size="large" wrap>
  <Form.Item
    name="discountPercent"
    label={<span style={{ color: "#ccc" }}>Discount (%)</span>}
    rules={[{ required: true, message: "กรอกเบอร์เช่นตัวลบนคด" }]}
  >
    <InputNumber style={{ width: 160 }} min={0} max={100} placeholder="0–100" />
  </Form.Item>
  <Form.Item
    name="gamelds"
    label={<span style={{ color: "#ccc" }}>Games</span>}
    rules={[{ required: true, message: "เลือกอย่างน้อย 1 เกม" }]}
  >
    <Select mode="multiple" style={{ minWidth: 280 }} options={gameOptions} placeholder="เลือกเกมที่เข้าร่วมปี" allowClear />
  </Form.Item>
  <Form.Item name="dateRange" label={<span style={{ color: "#ccc" }}>Date Range</span>}>
    <RangePicker />
  </Form.Item>
  <Form.Item name="active" label={<span style={{ color: "#ccc" }}>Active</span>} valuePropName="checked">
    <Switch />
  </Form.Item>
</Space>

```

```

<Space>
  <Button type="primary" htmlType="submit" loading={submitting}>
    {isEdit ? "Update" : "Create"}
  </Button>
  {isEdit && (
    <Button onClick={cancelEdit} disabled={submitting}>
      Cancel
    </Button>
  )}
</Space>
</Form>
</Space>
</Card>
<Card
  title={<span style={{ color: "white" }}>Promotions</span>}
  headStyle={{ background: "#1f1f1f" }}
  bodyStyle={{ background: "#141414" }}
  style={{ background: "#1f1f1f", color: "white", borderRadius: 10 }}
>
  <Table rowKey="id" columns={columns as any} dataSource={data} pagination={{ pageSize: 8 }} />
</Card>
</div>
</Content>
</Layout>
);
}

```

The screenshot shows a user interface for a game store. On the left, there's a sidebar with navigation links: Home, Search, Favorite, Balance, Category, and My game. The main content area features a large, vibrant banner at the top with the text "WINTER GAMES EXTRAVAGANZA" in large, stylized blue letters, set against a background of snow-covered trees and falling snowflakes. Below the banner, a section titled "winter sale" is displayed, showing a 10% discount offer from September 2nd to October 15th, which is currently active. A unique identifier "99999999" is also shown. Further down, a list of games included in the promotion is listed, each with a "View" button to its right. The games listed are Baldur's Gate 3, Hollow Knight, and Cyberpunk 2077.

```

// src/pages/PromotionDetail.tsx
import { useMemo } from "react";
import { useParams, Link } from "react-router-dom";
import { Layout, Typography, Card, Tag, List, Button, Empty } from "antd";
import Sidebar from "../components/Sidebar";
import { getByld } from "../services/promotionsApi";
import { listGames } from "../services/gamesApi";
import dayjs from "dayjs";
const { Content } = Layout;
const { Title, Text } = Typography;
export default function PromotionDetail() {
  const { id } = useParams<{ id: string }>();
  const promotion = useMemo(() => (id ? getByld(id) : undefined), [id]);

  const games = useMemo(() => {
    if (!promotion) return [];
    const all = listGames();
    return all.filter(g => promotion.gameIds?.includes(g.id));
  }, [promotion]);
  return (
    <Layout style={{ minHeight: "100vh", background: "#0f0f0f" }}>
      <Sidebar />
      <Content style={{ background: "#141414" }}>
        {!promotion ? (
          <div style={{ padding: 24 }}>
            <Empty description=<span style={{ color: "#bbb" }}>Promotion not found</span> />
          </div>
        ) : (
          <div style={{ padding: 16 }}>
            {/* Banner */}
            {promotion.imageUrl && (
              <div
                style={{
                  height: 240,
                  borderRadius: 12,
                  backgroundImage: `url(${promotion.imageUrl})`,
                  backgroundSize: "cover",
                  backgroundPosition: "center",
                  marginBottom: 16,
                }}
              />
            )}
            {/* Head */}
            <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 color="magenta" style={{ marginRight: 8 }}>

```

```

{promotion.discountPercent}% OFF
</Tag>
{promotion.startDate || promotion.endDate) && (
  <Tag color="geekblue">
    {promotion.startDate ? dayjs(promotion.startDate).format("YYYY-MM-DD") : "now"} →{" "}
    {promotion.endDate ? dayjs(promotion.endDate).format("YYYY-MM-DD") : "open"}
  </Tag>
)
<Tag color={promotion.active ? "green" : ""}>{promotion.active ? "Active" : "Inactive"}</Tag>
</div>
{promotion.description && (
  <Text style={{ color: "rgba(255,255,255,0.85)" }}>{promotion.description}</Text>
)
}
</Card>
/* Games in this promotion */
<Card
  title={<span style={{ color: "white" }}>Games in this promotion</span>}
  headStyle={{ background: "#1f1f1f" }}
  bodyStyle={{ background: "#141414" }}
  style={{ background: "#1f1f1f", borderRadius: 10 }}
>
{!games.length ?
  <Empty description={<span style={{ color: "#bbb" }}>No games selected</span>} />
} : (
  <List
    itemLayout="horizontal"
    dataSource={games}
    renderItem={({g: any}) => (
      <List.Item
        actions={[
          <Link key="view" to={`/gamedetail/${g.id}`}>
            <Button type="link">View</Button>
          </Link>,
        ]}
      >
        <List.Item.Meta
          title={<span style={{ color: "white" }}>{g.title}</span>}
          description={g.genre ? <span style={{ color: "#bbb" }}>{g.genre}</span> : null}
        />
      </List.Item>
    )}
  />
)
</Card>
</div>
)
</Content>
</Layout>
);

```

```
}
```

Backend

Game Entity

```
package entity
import "gorm.io/gorm"
type Game struct {
    gorm.Model
    GameName string `json:"game_name" gorm:"type:varchar(100);not null"`
    KeyName string `json:"key_name" gorm:"type:varchar(100);uniqueIndex;not null"`
    // One-to-Many: Game -> Reviews
    Reviews []Review `gorm:"foreignKey:GameID" json:"reviews"`
    // Many-to-Many: Game <-> Promotions (join table: promotion_games)
    Promotions []Promotion `gorm:"many2many:promotion_games" json:"promotions"`
}
```

Promotion_game Entity

```
package entity
import "gorm.io/gorm"
// ตารางสถานะ Promotion <-> Game
type Promotion_Game struct {
    gorm.Model
    PromotionID uint `json:"promotion_id" gorm:"index:idx_promo_game,unique"`
    Promotion *Promotion `gorm:"foreignKey:PromotionID" json:"promotion"`
    GameID uint `json:"game_id" gorm:"index:idx_promo_game,unique"`
    Game *Game `gorm:"foreignKey:GameID" json:"game"`
}
// กำหนดชื่อตารางขัดเจน (ไม่ใช่ GORM จะต้องเป็น promotion_games อัตโนมัติอยู่แล้ว)
func (Promotion_Game) TableName() string {
    return "promotion_games"
}
```

Promotion Entity

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

type Promotion struct {
    gorm.Model
    Title      string `json:"title"      gorm:"type:varchar(120);not null"`
    Description string `json:"description"  gorm:"type:text"`
    DiscountValue int    `json:"discount_value" gorm:"not null"`
    StartDate   time.Time `json:"start_date"   gorm:"not null"`
    EndDate    time.Time `json:"end_date"    gorm:"not null"`
    Promolmage string `json:"promo_image"`
    // สถานะการใช้งานโปรโมชั่น (true = เปิดใช้งาน, false = ปิดการใช้งาน)
    Status bool `json:"status" gorm:"default:true"`
    // ใครเป็นคนสร้างโปรโมชั่น
    UserID uint `json:"user_id"`
    User  *User `gorm:"foreignKey:UserID" json:"user"`
    // ความสัมพันธ์ Many-to-Many: Promotion <-> Games
    Games []Game `gorm:"many2many:promotion_games" json:"games"`
}
```

User Entity

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

type User struct {
    gorm.Model
    Username string `json:"username"  gorm:"type:varchar(50);uniqueIndex;not null"`
    Password string `json:"password"  gorm:"type:varchar(255);not null;" // เก็บรหัสผ่านแบบ hash`
    Email    string `json:"email"     gorm:"type:varchar(120);uniqueIndex;not null"`
    FirstName string `json:"first_name" gorm:"type:varchar(100)"`
    LastName  string `json:"last_name"  gorm:"type:varchar(100)"`
    Birthday  *time.Time `json:"birthday" // ใช้ pointer เพื่อให้เป็น NULL ได้`

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

    Reviews []Review `gorm:"foreignKey:UserID;constraint:OnUpdate:CASCADE,OnDelete:SET NULL;" json:"reviews"`
    Promotions []Promotion `gorm:"foreignKey:UserID;constraint:OnUpdate:CASCADE,OnDelete:SET NULL;" json:"promotions"`
}
```

Controller promotion_controller

```
package controllers

import (
    "net/http"
    "strconv"
    "time"
    "github.com/gin-gonic/gin"
    "projectSA/configs"
    "projectSA/entity"
)

type PromotionRequest struct {
    Title      string `json:"title"        binding:"required"`
    Description string `json:"description"`
    DiscountValue int    `json:"discount_value" binding:"required"`
    StartDate   time.Time `json:"start_date"    binding:"required"`
    EndDate     time.Time `json:"end_date"      binding:"required"`
    Promolmage  string  `json:"promo_image"`
    Status      *bool   `json:"status"`
    UserID      uint    `json:"user_id"`
}

type AssignGamesRequest struct {
    GameIDs []uint `json:"game_ids" binding:"required,min=1,dive,gt=0"`
}

// GET /api/v1/promotions?limit=20&offset=0&search=...
func ListPromotions(c *gin.Context) {
    // pagination
    limit, offset := 20, 0
    if v, err := strconv.Atoi(c.DefaultQuery("limit", "20")); err == nil && v > 0 {
        limit = v
    }
    if v, err := strconv.Atoi(c.DefaultQuery("offset", "0")); err == nil && v >= 0 {
        offset = v
    }
    // search
    search := c.Query("search")
    db := configs.GetDB()
    q := db.Model(&entity.Promotion{}).Preload("Games").Preload("User")
    if search != "" {
        like := "%" + search + "%"
        q = q.Where("title LIKE ? OR description LIKE ?", like, like)
    }

    var promos []entity.Promotion
    if err := q.Limit(limit).Offset(offset).Find(&promos).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    }
}
```

```

        return
    }
    c.JSON(http.StatusOK, gin.H{"items": promos, "limit": limit, "offset": offset})
}

// POST /api/v1/promotions
func CreatePromotion(c *gin.Context) {
    var req PromotionRequest
    if err := c.ShouldBindJSON(&req); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if !req.EndDate.After(req.StartDate) {
        c.JSON(http.StatusBadRequest, gin.H{"error": "end_date must be after start_date"})
        return
    }
    p := entity.Promotion{
        Title:      req.Title,
        Description: req.Description,
        DiscountValue: req.DiscountValue,
        StartDate:   req.StartDate,
        EndDate:     req.EndDate,
        Promolmage:  req.Promolmage,
        Status:      true,
        UserID:      req.UserID,
    }
    if req.Status != nil {
        p.Status = *req.Status
    }
    if err := configs.GetDB().Create(&p).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusCreated, p)
}

// GET /api/v1/promotions/:id
func GetPromotion(c *gin.Context) {
    id := c.Param("id")
    var p entity.Promotion
    if err := configs.GetDB().Preload("Games").Preload("User").First(&p, id).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": "promotion not found"})
        return
    }
    c.JSON(http.StatusOK, p)
}

// PUT /api/v1/promotions/:id
func UpdatePromotion(c *gin.Context) {
    id := c.Param("id")
    var req PromotionRequest

```

```

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

if !req.EndDate.After(req.StartDate) {
    c.JSON(http.StatusBadRequest, gin.H{"error": "end_date must be after start_date"})
    return
}

db := configs.GetDB()
var p entity.Promotion
if err := db.First(&p, id).Error; err != nil {
    c.JSON(http.StatusNotFound, gin.H{"error": "promotion not found"})
    return
}

p.Title = req.Title
p.Description = req.Description
p.DiscountValue = req.DiscountValue
p.StartDate = req.StartDate
p.EndDate = req.EndDate
p.PromoImage = req.PromoImage
if req.Status != nil {
    p.Status = *req.Status
}
if req.UserID != 0 {
    p.UserID = req.UserID
}
if err := db.Save(&p).Error; err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}
c.JSON(http.StatusOK, p)
}

// DELETE /api/v1/promotions/:id
func DeletePromotion(c *gin.Context) {
    id := c.Param("id")
    if err := configs.GetDB().Delete(&entity.Promotion{}, id).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.Status(http.StatusNoContent)
}

// POST /api/v1/promotions/:id/assign-games
func AssignGamesToPromotion(c *gin.Context) {
    id := c.Param("id")
    var req AssignGamesRequest
    if err := c.ShouldBindJSON(&req); err != nil {

```

```

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

return
}

db := configs.GetDB()
var promo entity.Promotion
if err := db.First(&promo, id).Error; err != nil {
    c.JSON(http.StatusNotFound, gin.H{"error": "promotion not found"})
    return
}

var games []entity.Game
if err := db.Where("id IN ?", req.GameIDs).Find(&games).Error; err != nil {
    c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
    return
}

if len(games) == 0 {
    c.JSON(http.StatusBadRequest, gin.H{"error": "no valid games provided"})
    return
}

if err := db.Model(&promo).Association("Games").Replace(&games); err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}

if err := db.Preload("Games").Preload("User").First(&promo, promo.ID).Error; err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}

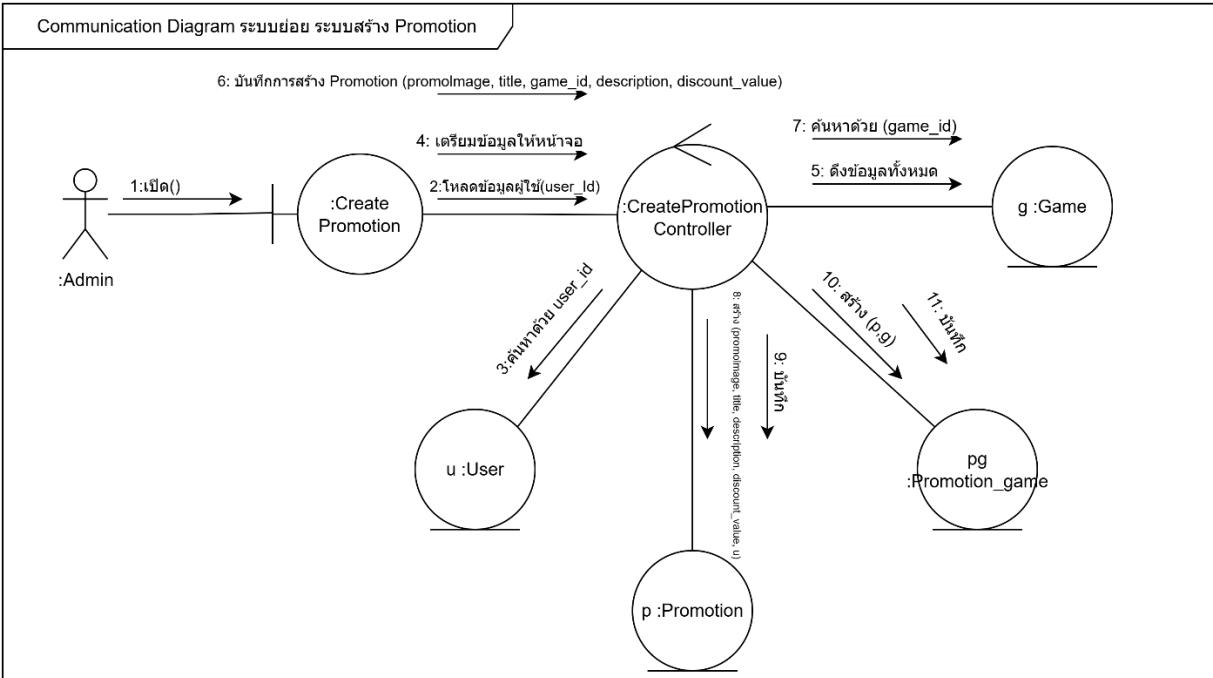
c.JSON(http.StatusOK, promo)
}

// POST /api/v1/promotions/:id/toggle-status
func TogglePromotionStatus(c *gin.Context) {
    id := c.Param("id")
    var p entity.Promotion
    if err := configs.GetDB().First(&p, id).Error; err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": "promotion not found"})
        return
    }
    p.Status = !p.Status
    if err := configs.GetDB().Save(&p).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"id": p.ID, "status": p.Status})
}

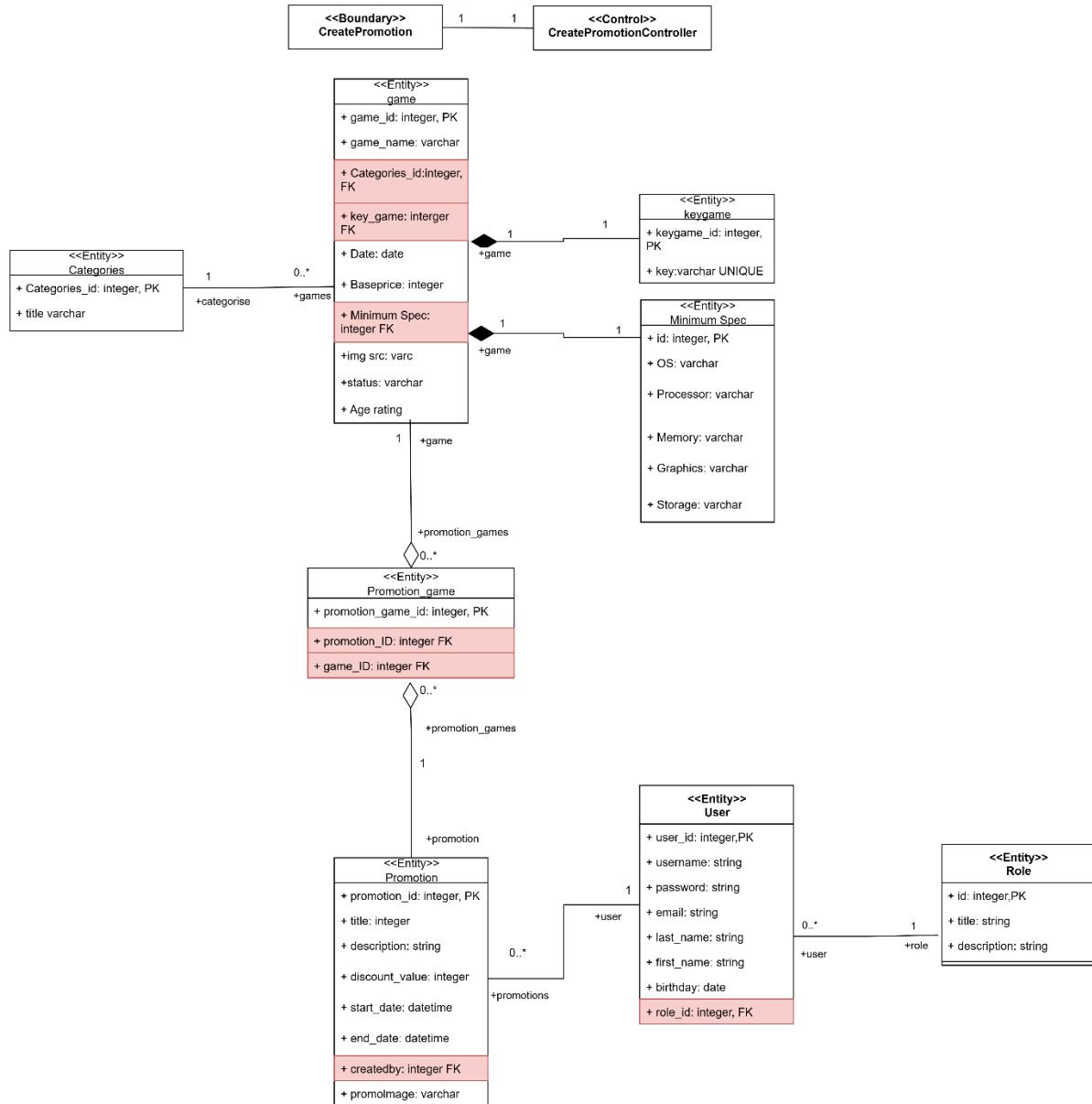
```

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(promolmage)	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 (promolmage, 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 เช็คค่า rating, review_title,review_text	เป็นคำสั่ง	p :promotion	สร้าง (promolmage, 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 ระบบร้านขายเกมออนไลน์

ระบบย่อย ระบบยื่นคำร้องขอคืนเงิน

ระบบคืนเงินเป็นระบบที่ช่วยสร้างความมั่นใจให้ลูกค้าในการซื้อ **เกม** จากร้านของเรา โดยลูกค้าสามารถส่งคำร้องขอคืนเงินได้หากพบปัญหาหลังการซื้อ เช่น คีย์เกมไม่สามารถใช้งานได้ เกมไม่ตรงตามที่แสดงไว้

หรือซื้อเกมผิดประเภท โดย**ลูกค้า**สามารถเลือกเกมจากประวัติการซื้อและส่งคำร้องขอคืนเงินได้ทันที

ระบบจะให้ลูกค้าแนบเหตุผลการคืนเงิน และหลักฐานประกอบคำร้อง ซึ่งข้อมูลจะถูกส่งไปยังแอดมินเพื่อตรวจสอบ

หากแอดมินอนุมัติ ระบบจะดำเนินการคืนเงินให้ตามช่องทางที่ลูกค้าใช้ชำระเงินเป็นช่องทางการคืนเงิน เช่น

บัตรเครดิต วอลเล็ต หรือโอนเข้าบัญชีธนาคาร นอกจากนี้ ระบบจะเก็บประวัติการคืนเงินไว้ในระบบ

เพื่อให้ง่ายต่อการตรวจสอบย้อนหลัง และใช้เป็นข้อมูลวิเคราะห์พัฒนารูปแบบของลูกค้าในอนาคต

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

4. แออดมินตรวจสอบและกด “อนุมัติ” หรือ “ปฏิเสธ” คำร้อง
5. หากอนุมัติ ระบบแสดงสถานะ “คืนเงินแล้ว” พร้อมวันที่และช่องทางที่คืนเงิน

Output ของข้อมูล

- บันทึกคำร้องพร้อมรายละเอียด เช่น เกม, เหตุผล, วันที่ร้องขอ
- บันทึกผลการอนุมัติ และช่องทางคืนเงิน
- สถานะการคืนเงินมี: รอตรวจสอบ, อนุมัติ, ปฏิเสธ, คืนเงินแล้ว
- เก็บประวัติคำร้องทั้งหมดไว้ในระบบฐานข้อมูลคำนा�ມที่อาจถูกยกเป็นตรางในฐานข้อมูล

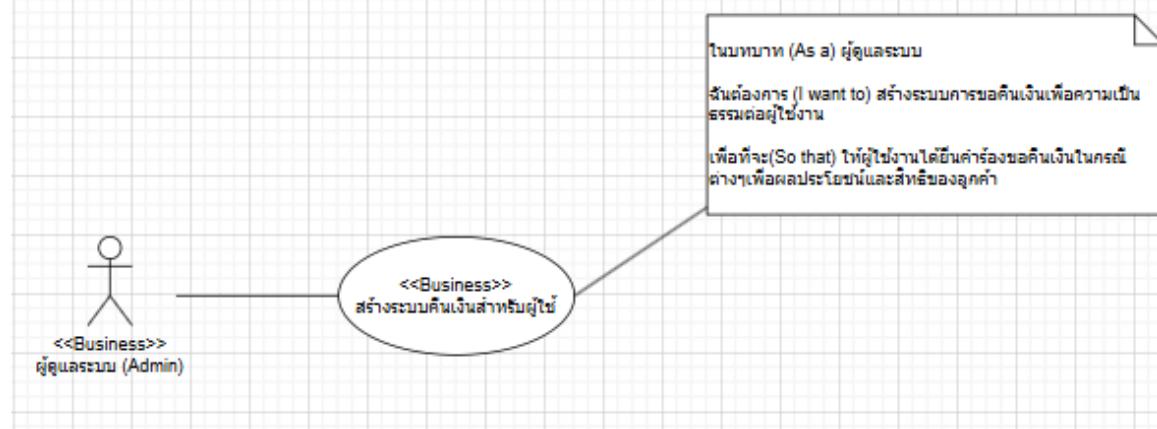
คำนำที่อาจจะกล่าวมาเป็นตารางในฐานข้อมูลของระบบ

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)

ระบบยืนคำร้องขอคืนเงิน

Business Use Case Diagram (แบบเดี่ยว)



Checklist: Business Use Case Diagram Business

Actor มี
> กำกับ Business Actor เป็นชื่อบทบาทเท่านั้น ไม่ใช่ชื่อคน ไม่เป็นชื่อบุคคล

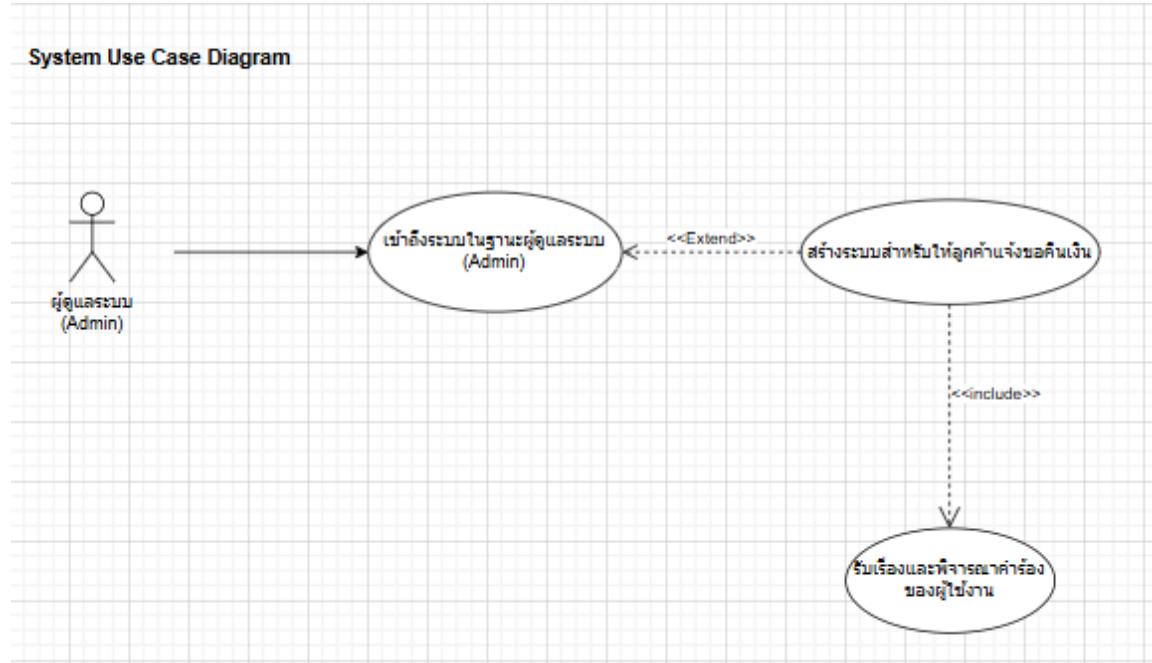
Business Use Case มี
<> กำกับ

Business Use Case ขึ้นต้นด้วยคำว่า “จะ” และแสดงพฤติกรรม “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 “ระบบยื่นคำร้องขอคืนเงิน” สามารถถูกถอดไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบ
คอมพิวเตอร์ได้หรือไม่ต้องมีขั้นตอนทาง Security มาเกี่ยวหรือไม่
ตอบ Business Use Case “ระบบยื่นคำร้องขอคืนเงิน” สามารถถูกถอดไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบ
คอมพิวเตอร์ได้และต้องการขั้นตอนทาง Security (มี Privacy มาเกี่ยวข้อง)

เพราะฉะนั้น Business Use Case “ระบบยื่นคำร้องขอคืนเงิน” จะถูกจัดเป็น 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” ทุกรสั่ง หรือไม่

ตอบ ไม่

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

พิจารณาประเด็นที่ 4

ถ้าสมาชิก “เข้าระบบในฐานะผู้ดูแลระบบ” มาแล้วจำเป็นต้อง “บันทึกข้อมูลคำร้องขอคืนเงิน” ทุกรสั่งหรือไม่ 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. เสน้ยิงจาก System Actor ไปหา System Use Case จะต้องเป็นเสนทิบ 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>>--> ต้องเป็นเสนประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ทางเลือกไปยัง System Use Case หลัก
7. การใช้ <--<<include>>--> ต้องเป็นเสนประ หัวลูกศรปลายเปิด ซึ่งจาก System Use Case ที่ใหญ่กว่า ไปยัง System Use Case ที่จะถูกรวบเข้ามา

Entity: User

ข้อมูลที่ต้องจัดเก็บ

- UserID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- Username: varchar, ไม่เป็น Null
- Password: varchar, ไม่เป็น Null
- Email: varchar, Null ได้
- LastName: varchar, Null ได้
- FirstName: varchar, Null ได้
- Birthday: date, Null ได้
- RoleID: integer, Foreign Key, ไม่เป็น Null

Userl D	Usernam e	Password	Email	LastNam e	FirstNam e	Birthda y	Rolel D
1001	user01	encrypt(123 4)	user01@mail.com	Kim	Minho	2000- 05-14	1
1002	gamer02	encrypt(567 8)	gamer02@mail.co m	Park	Jisoo	1999- 11-23	2

Entity: Role

ข้อมูลที่ต้องจัดเก็บ

- RoleID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- RoleName: varchar, ไม่เป็น Null

RoleID	RoleName
1	Admin
2	Customer

Entity: Game

ข้อมูลที่ต้องจัดเก็บ

- GameID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- GameName: varchar, ไม่เป็น Null
- GamePrice: decimal, ไม่เป็น Null
- Description: varchar, Null ได้

GameID	GameName	GamePrice	Description
2001	RPG Quest	1500.00	เกมผจญภัยแฟนตาซี
2002	Racing Pro	900.00	เกมแข่งรถสมจริง

Entity: Notification

ข้อมูลที่ต้องจัดเก็บ

- - NoteID: 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	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	ເກມເປີດໄໝ່ຕິດ	ເກມຄ້າງຕອນໂທລດ	2025-09-01 10:00:00	2025-09-02 15:00:00	1
4002	1002	2002	None	ກາຮໍາຮະເງິນລ້ຳມະຫວາງ	ຮະບປໄໝ່ຕັດບັດຮເຄຣດີຕ	2025-09-03 09:30:00	None	2

Entity: Redund_Status

ຂໍ້ມູນທີ່ຕ້ອງຈັດເກີບ

- ProblemStatusID: Primary Key, integer, ໄມ້ສໍາກັນ, ໄມ້ເປັນ Null
- StatusName: varchar, ໄມ້ເປັນ 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

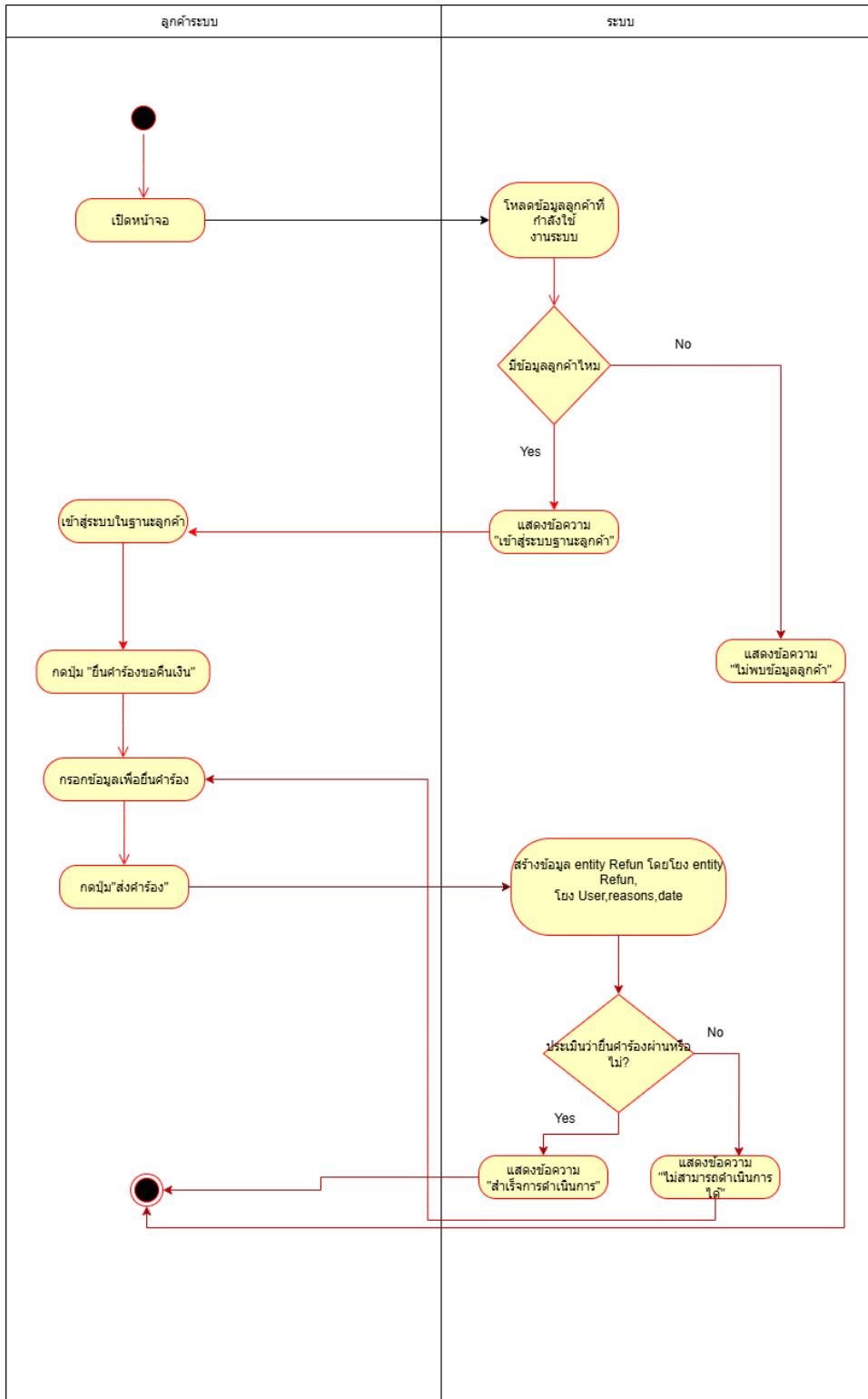
- เมื่อได้กีตามที่มี Foreign Key (FK) ซึ่งจากตารางหลักกลับไปยังตารางสนับสนุน
ให้ใช้ ComboBox เป็นตัวเลือกเพื่อเข้ามายโถงข้อมูลใน User Interface
- สำหรับ Field ประเภทไหนๆ ให้สร้างตามประเภทของข้อมูล เช่น
 - Textbox สำหรับข้อความทั่วไป
 - Password สำหรับข้อมูลรหัสผ่าน
 - Datetime Picker สำหรับเลือกวันและเวลา
 - Numeric Input สำหรับป้อนตัวเลข

UI Design

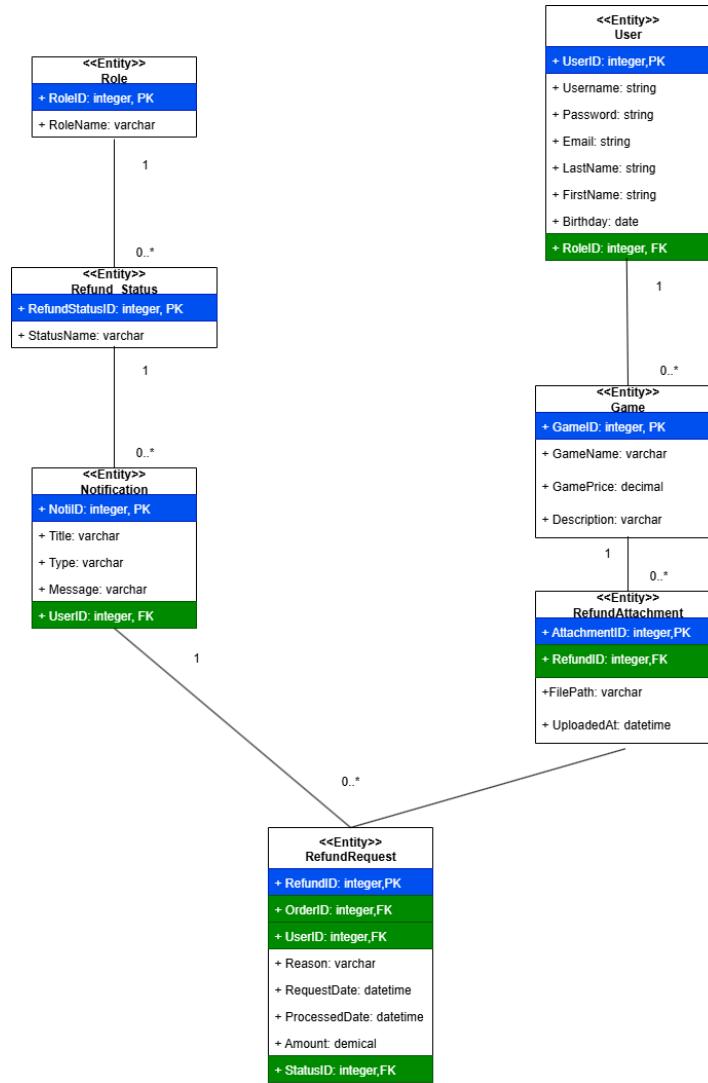
The wireframe illustrates a user interface for a game store. At the top left is a logo of a video game controller, and at the top center is the text "ระบบร้านขายเกมออนไลน์". On the top right is a user icon. The main area is divided into several sections:

- บันทึกด้วย**: A section containing four input fields labeled "ชื่อเกม", "รหัสส่วนลด", "วันที่有效", and "ราคาที่จ่าย".
- เหตุผลในการขอคืนเงิน**: A section containing two input fields labeled "เลือกเหตุผล" and "อธิบายเพิ่มเติม". Below this is a large rectangular area for a file upload.
- แนบหลักฐาน**: A section with a "choose file" button and a message "No file".
- ข้อกำหนดและเงื่อนไข**: A section containing a checkbox with the text "ข้อกำหนดและเงื่อนไขการซื้อขายพิจารณาภายใน 7 วันทำการ".
- Action Buttons**: At the bottom are two buttons: "ยกเลิก" (Cancel) and "ดำเนินการ" (Proceed).

Activity Diagram



Class Diagram At analysis Level



Frontend

ระบบยื่นคำร้องขอคืนเงิน

The screenshot shows a dark-themed user interface for a refund request system. On the left, a sidebar contains links: Balance, Report Problem, Refund Request (which is highlighted in blue), and Admin Dashboard. The main content area is a form for a refund request. It includes the following fields:

- Order ID: A text input field containing "e.g. #621da7ff".
- Reason for Refund: A dropdown menu labeled "Select reason".
- Purchase Date: A date picker labeled "Select date".
- Bank: A dropdown menu labeled "Select bank".
- Account Number: A text input field with placeholder text "Enter your account number".
- Additional Comments: A text area labeled "Provide any additional details".
- Upload Proof: A file upload button with a dashed border and the word "Upload" inside.

```
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),
      });
    }
  };

  await new Promise((resolve) => setTimeout(resolve, 800));

  message.success("ส่งคำขอคืนเงินเรียบร้อย! (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>

```

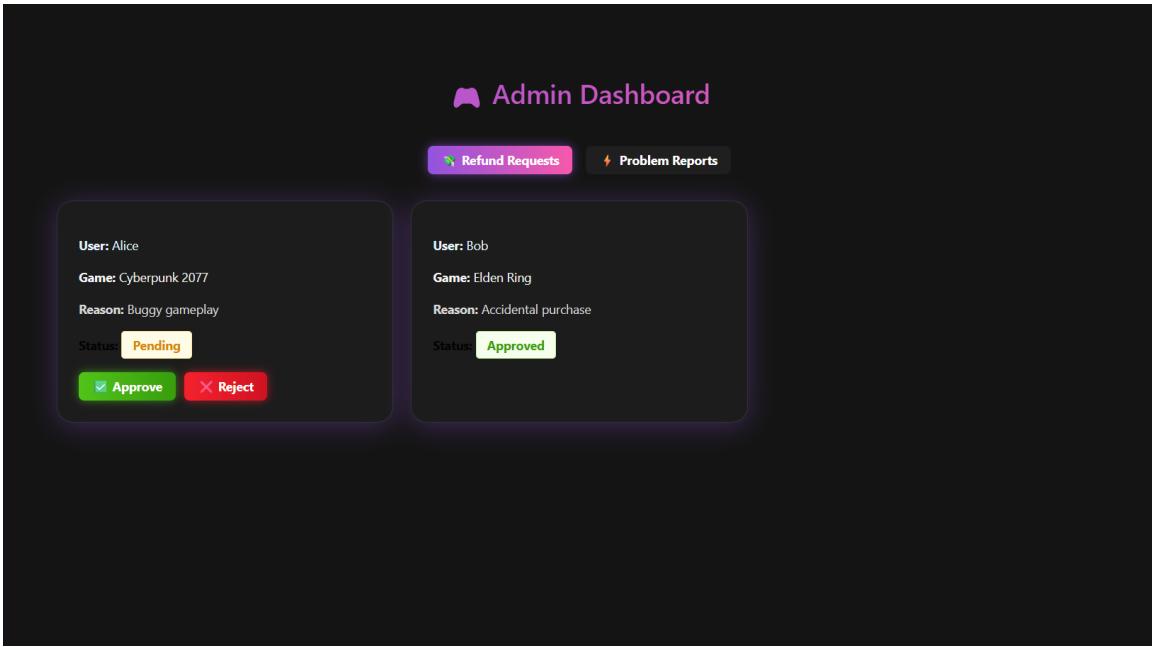
```
<Form.Item label="Additional Comments" name="comments">
  <Input.TextArea
    rows={3}
    placeholder="Provide any additional details"
    style={{ background: "#ffffff", color: "black" }}
  />
</Form.Item>

<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>
  )}
</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, #92254de 0%, #f759ab 100%)",
      border: "none",
      color: "white",
      width: "100%",
    }}
  >
    Submit Request
  </Button>
</Form.Item>
</Form>
</Card>
</div>
);
}
```



```
// src/pages/AdminPage.tsx

import { useState, useEffect } from "react";
import { Card, Button, Typography, Tag, Input, Upload, message, Modal } from "antd";
import { UploadOutlined } from "@ant-design/icons";

const { Title } = Typography;
const { TextArea } = Input;

interface RefundRequest {
  id: number;
  user: string;
  game: string;
  reason: string;
  status: "Pending" | "Approved" | "Rejected";
}

interface ProblemReport {
  id: number;
  user: string;
  category: string;
  title: string;
}
```

```
description: string;
resolved: boolean;
}

export default function AdminPage() {
const [refunds, setRefunds] = useState<RefundRequest[]>([]);
const [problems, setProblems] = useState<ProblemReport[]>([]);
const [activeTab, setActiveTab] = useState<"refunds" | "problems">("refunds");

// reply state เก็บข้อมูลตอบกลับต่อ report
const [replies, setReplies] = useState<Record<number, { text: string; fileList: any[] }>>({});

const [previewImage, setPreviewImage] = useState<string>(""); // สำหรับ preview
const [previewOpen, setPreviewOpen] = useState(false);

useEffect(() => {
  // mock data
  setRefunds([
    { id: 1, user: "Alice", game: "Cyberpunk 2077", reason: "Buggy gameplay", status: "Pending" },
    { id: 2, user: "Bob", game: "Elden Ring", reason: "Accidental purchase", status: "Approved" },
  ]);

  setProblems([
    { id: 1, user: "Charlie", category: "Login Issue", title: "Can't login", description: "Tried multiple times but login fails.", resolved: false },
    { id: 2, user: "Diana", category: "Payment", title: "Card declined", description: "Even though balance is enough.", resolved: true },
  ]);
}, []);

const handleRefundAction = (id: number, action: "Approved" | "Rejected") => {
  setRefunds((prev) =>
    prev.map((req) =>
      req.id === id ? { ...req, status: action } : req
    )
  );
}
```

```

};

};

const handleResolveProblem = (id: number) => {
  setProblems((prev) =>
    prev.map((rep) =>
      rep.id === id ? { ...rep, resolved: true } : rep
    )
  );
};

const handleSendReply = (id: number) => {
  const reply = replies[id];
  if (!reply || (!reply.text && reply.fileList.length === 0)) {
    message.warning("กรุณาพิมพ์ข้อความหรือติดไฟล์อีเมลข้อความที่ 1 อย่างน้อย");
    return;
  }

  console.log("Reply sent:", { problemId: id, ...reply });
  message.success("ส่งคำตอบกลับไปยังลูกค้าเรียบร้อย!");

  // clear reply state
  setReplies((prev) => ({ ...prev, [id]: { text: "", fileList: [] } }));
};

// ฟังก์ชันสำหรับ preview รูป
const handlePreview = async (file: any) => {
  setPreviewImage(file.thumbUrl || file.url || "");
  setPreviewOpen(true);
};

return (
  <div style={{ background: "#141414", minHeight: "100vh", padding: "40px", color: "#fff" }}>
    <Title
      level={2}

```

```
style={{
  textAlign: "center",
  background: "linear-gradient(90deg, #9254de, #f759ab)",
  WebkitBackgroundClip: "text",
  color: "transparent",
  marginBottom: "40px",
}}
>
 Admin Dashboard
</Title>

/* Tabs */
<div style={{ textAlign: "center", marginBottom: "30px" }}>
  <Button
    type={activeTab === "refunds" ? "primary" : "default"}
    onClick={() => setActiveTab("refunds")}
    style={{
      marginRight: "15px",
      background: activeTab === "refunds" ? "linear-gradient(90deg, #9254de, #f759ab)" :
      "#1f1f1f",
      border: "none",
      color: "#fff",
      fontWeight: "bold",
      boxShadow: activeTab === "refunds" ? "0 0 12px rgba(146, 84, 222, 0.6)" : "none",
    }}
  >
 Refund Requests
  </Button>
  <Button
    type={activeTab === "problems" ? "primary" : "default"}
    onClick={() => setActiveTab("problems")}
    style={{
      background: activeTab === "problems" ? "linear-gradient(90deg, #f759ab, #9254de)" :
      "#1f1f1f",
    }}
  >
```

```
border: "none",
color: "#fff",
fontWeight: "bold",
boxShadow: activeTab === "problems" ? "0 0 12px rgba(247, 89, 171, 0.6)" : "none",
}}
>
⚡ Problem Reports
</Button>
</div>

/* Refund Requests */
{activeTab === "refunds" && (
<div style={{ display: "grid", gridTemplateColumns: "repeat(auto-fill, minmax(350px, 1fr))", gap: "20px" }}>
{refunds.map((req) => (
<Card
key={req.id}
bordered={false}
hoverable
style={{
background: "#1c1c1c",
borderRadius: "18px",
border: "1px solid rgba(255,255,255,0.1)",
boxShadow: "0 4px 30px rgba(146, 84, 222, 0.35)",
transition: "transform 0.2s ease, box-shadow 0.2s ease",
}}
>
<p style={{ color: "#e6f7ff" }}><b>User:</b> {req.user}</p>
<p style={{ color: "#f5f5f5" }}><b>Game:</b> {req.game}</p>
<p style={{ color: "#d9d9d9" }}><b>Reason:</b> {req.reason}</p>
<p>
<b>Status:</b>{" "}
<Tag
style={{ fontWeight: "bold", padding: "5px 12px", fontSize: "14px" }}
color={
```

```
req.status === "Pending" ? "gold" :  
  req.status === "Approved" ? "green" : "red"  
}  
>  
{req.status}  
</Tag>  
</p>  
  
{req.status === "Pending" && (  
  <div style={{ marginTop: "15px" }}>  
    <Button  
      onClick={() => handleRefundAction(req.id, "Approved")}  
      style={{  
        marginRight: "10px",  
        background: "linear-gradient(90deg, #52c41a, #389e0d)",  
        border: "none",  
        color: "white",  
        fontWeight: "bold",  
        boxShadow: "0 0 10px rgba(82,196,26,0.6)",  
      }}  
    >  
       Approve  
    </Button>  
    <Button  
      onClick={() => handleRefundAction(req.id, "Rejected")}  
      style={{  
        background: "linear-gradient(90deg, #f5222d, #cf1322)",  
        border: "none",  
        color: "white",  
        fontWeight: "bold",  
        boxShadow: "0 0 10px rgba(245,34,45,0.6)",  
      }}  
    >  
       Reject  
    </Button>  
  </div>  
)}
```

```

        </Button>
    </div>
)
</Card>
))
</div>
)

/* Problem Reports */
{activeTab === "problems" && (
<div style={{ display: "grid", gridTemplateColumns: "repeat(auto-fill, minmax(350px, 1fr))", gap: "20px" }}>
{problems.map((rep) => (
<Card
key={rep.id}
bordered={false}
hoverable
style={{
background: "#1c1c1c",
borderRadius: "18px",
border: "1px solid rgba(255,255,255,0.1)",
boxShadow: "0 4px 30px rgba(247, 89, 171, 0.35)",
transition: "transform 0.2s ease, box-shadow 0.2s ease",
}}
>
<p style={{ color: "#e6f7ff" }}><b>User:</b> {rep.user}</p>
<p style={{ color: "#f5f5f5" }}><b>Category:</b> {rep.category}</p>
<p style={{ color: "#f5f5f5" }}><b>Title:</b> {rep.title}</p>
<p style={{ color: "#d9d9d9" }}><b>Description:</b> {rep.description}</p>
<p>
<b>Status:</b>{" "}
{rep.resolved ? (
<Tag style={{ fontWeight: "bold", padding: "5px 12px", fontSize: "14px" }}
color="green">

```



Resolved

```
</Tag>
) : (
  <Tag style={{ fontWeight: "bold", padding: "5px 12px", fontSize: "14px" }}
color="gold">
   Pending
  </Tag>
)
</p>

{!rep.resolved && (
<div style={{ marginTop: "15px" }}>
  /* Reply Box */
  <TextArea
    rows={3}
    placeholder="ພິເພີ້ນຂໍ້ອະນາຄາມຕອບກລັບລູກຄ້າ..."
    value={replies[rep.id]?.text || ""}
    onChange={(e) =>
      setReplies((prev) => ({
        ...prev,
        [rep.id]: { text: e.target.value, fileList: prev[rep.id]?.fileList || [] },
      }))
    }
    style={{
      marginBottom: "10px",
      background: "#141414",
      color: "white",
      borderRadius: "10px",
      border: "1px solid #434343",
    }}
  />

  /* Upload File + Preview */
  <Upload
    fileList={replies[rep.id]?.fileList || []}
    beforeUpload={() => false}
  
```

```
onChange={({ fileList }) =>
  setReplies((prev) => ({
    ...prev,
    [rep.id]: { text: prev[rep.id]?.text || "", fileList },
  }))
}

onPreview={handlePreview}
listType="picture-card" //  แสดงเป็น thumbnail
multiple

>
<Button
  icon={<UploadOutlined />}
  style={{ marginBottom: "10px", background: "#2a2a2a", color: "white", border:
"none" }}
>
  แนบไฟล์/รูปภาพ
</Button>
</Upload>

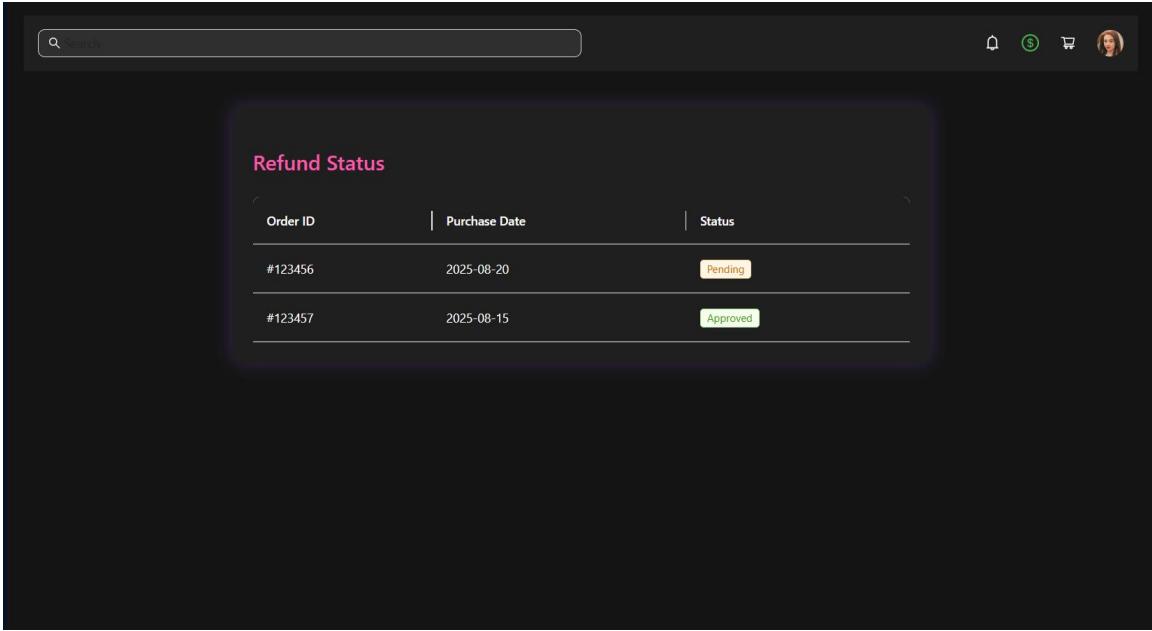
<Button
  onClick={() => handleSendReply(rep.id)}
  style={{
    marginRight: "10px",
    background: "linear-gradient(90deg, #52c41a, #389e0d)",
    border: "none",
    color: "white",
    fontWeight: "bold",
    boxShadow: "0 0 10px rgba(82,196,26,0.6)",
  }}
>
   Send Reply
</Button>

<Button
```

```
    onClick={() => handleResolveProblem(rep.id)}
    style={{
      background: "linear-gradient(90deg, #f759ab, #9254de)",
      border: "none",
      color: "white",
      fontWeight: "bold",
      boxShadow: "0 0 10px rgba(247, 89, 171, 0.6)",
    }}
  >
  ✓ Mark as Resolved
</Button>
</div>
)}
</Card>
))}

</div>
)}

/* Modal Preview */
<Modal open={previewOpen} footer={null} onCancel={() => setPreviewOpen(false)}>
  <img alt="preview" style={{ width: "100%" }} src={previewImage} />
</Modal>
</div>
);
}
```



```
import { Card, Table, Tag, Typography } from "antd";
import Navbar from "../components/Navbar";

const { Title } = Typography;

const columns = [
  {
    title: <span style={{ color: "white" }}>Order ID</span>,
    dataIndex: "orderId",
    key: "orderId",
    render: (text: string) => <span style={{ color: "white" }}>{text}</span>,
  },
  {
    title: <span style={{ color: "white" }}>Purchase Date</span>,
    dataIndex: "purchaseDate",
    key: "purchaseDate",
    render: (text: string) => <span style={{ color: "white" }}>{text}</span>,
  },
  {
    title: <span style={{ color: "white" }}>Status</span>,
    dataIndex: "status",
  },
]
```

```
key: "status",
render: (status: string) => {
  let color = "blue";
  if (status === "Approved") color = "green";
  else if (status === "Rejected") color = "red";
  else if (status === "Pending") color = "orange";
  return <Tag color={color}>{status}</Tag>;
},
},
];
];

const data = [
{
  key: "1",
  orderId: "#123456",
  purchaseDate: "2025-08-20",
  status: "Pending",
},
{
  key: "2",
  orderId: "#123457",
  purchaseDate: "2025-08-15",
  status: "Approved",
},
];
}

export default function RefundStatusPage() {
  return (
    <div style={{ background: "#141414", minHeight: "100%", color: "#fbfbfbff" }}>
      /* ✓ ใช้ Navbar เดียวกับหน้าอื่น */
      <Navbar />

      <Card
        style={{
          backgroundColor: "#1f1f1f",

```

```

borderRadius: 12,
color: "white",
maxWidth: 800,
margin: "40px auto",
boxShadow: "0 0 20px rgba(146, 84, 222, 0.3)",
}}
bordered={false}
>
<Title level={3} style={{ color: "#f759ab", marginBottom: 24 }}>
  Refund Status
</Title>

<Table
  columns={columns}
  dataSource={data}
  pagination={false}
  style={{ background: "#1f1f1f" }}
  rowClassName={() => "dark-row"}
/>
</Card>

```

```

/* ✓ custom CSS สำหรับ row สีเข้ม */
<style>
{
  .dark-row td {
    background-color: #1f1f1f !important;
    color: white !important;
  }
  .ant-table-thead > tr > th {
    background-color: #1f1f1f !important;
    color: #f0f0f0 !important;
  }
}
</style>
</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"
)

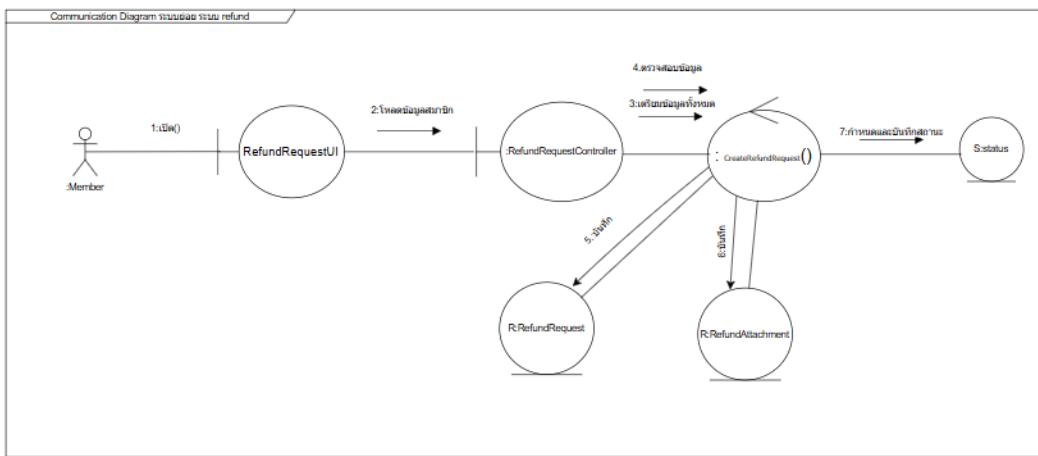
func GetRefundStatuses(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var statuses []models.RefundStatus
        if err := db.Find(&statuses).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
            return
        }
        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)
    }
}
```

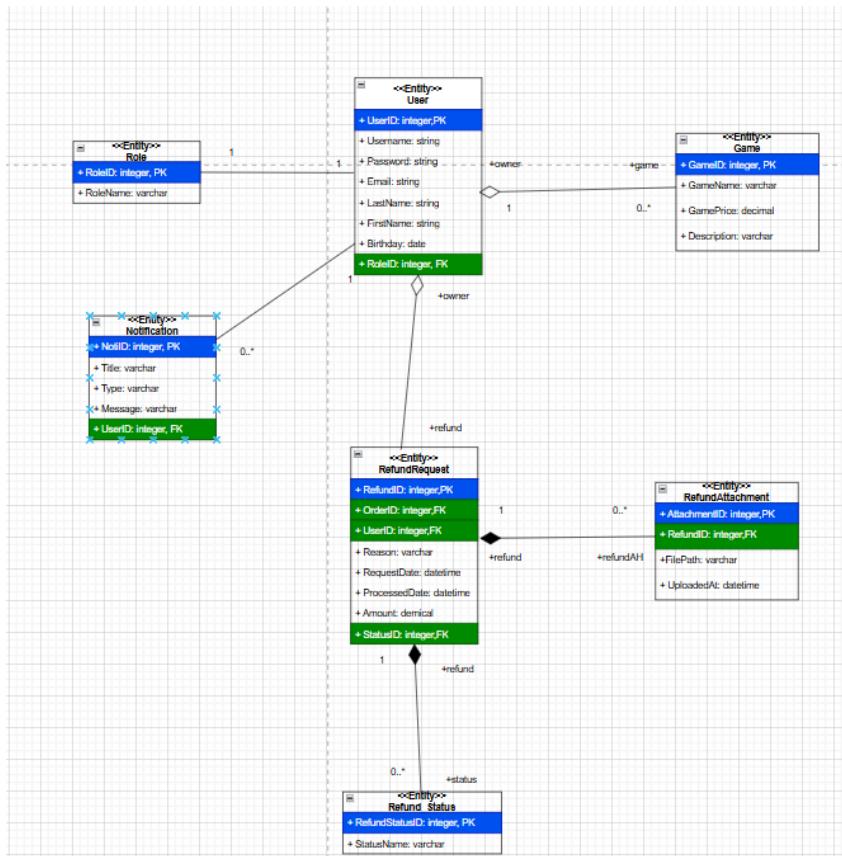
1) Communication Diagram: Refund Request

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 “ยื่นยันคำร้อง 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(r_status)
อัปเดต RefundRequest.StatusID	ไม่เป็นคำสั่ง	-	-
แสดงข้อความ “ส่งคำร้องเรียบร้อย”	ไม่เป็นคำสั่ง	-	



Communication Class Diagram



B6641054 นายณัฐนันท์ จันทร์สุริยา

ระบบหลัก ระบบบร้านขายเกมออนไลน์

ระบบบอท ระบบแจ้งรายงานปัญหา

ระบบ **รายงานปัญหา** เป็นฟังก์ชันที่สำคัญที่เปิดโอกาสให้ลูกค้าสามารถแจ้งปัญหาที่พบจากการใช้งาน เช่น เกมไม่สามารถดาวน์โหลดได้ คีย์เกมไม่ถูกต้อง เว็บไซต์ล่ม หรือชำระเงินไม่สำเร็จ ลูกค้าสามารถกรอกฟอร์มรายงานปัญหาผ่านระบบ โดยระบุหัวข้อ รายละเอียดปัญหา และแนบภาพหน้าจอประกอบ เป็นไฟล์แนบ หลังจากส่งคำร้องแล้ว ระบบจะแจ้งเตือนทีมซัพพอร์ตหรือแอดมินเพื่อดำเนินการตรวจสอบ เมื่อมีความคืบหน้า ระบบจะแจ้ง **ลูกค้า** ผ่านทางอีเมลหรือระบบภายในเว็บ เช่น แสดงสถานะคำร้องว่า 'กำลังตรวจสอบ', 'ดำเนินการแล้ว', 'แก้ไขแล้ว' ระบบนี้ช่วยให้สามารถติดตามและจัดการปัญหาได้อย่างเป็นระบบ และยังสร้างความมั่นใจให้กับลูกค้าอีกด้วย

User Story ระบบรายงานปัญหา

ในบทบาทของ (As a) **ลูกค้า** (Customer)

ฉันต้องการ (I want to) **แจ้งปัญหา** ที่เกิดขึ้นระหว่างใช้งาน

เพื่อ (So that) ทีมงานจะได้ทราบและดำเนินการแก้ไขได้อย่างรวดเร็วและมีประสิทธิภาพ

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

ฉันต้องการ (I want to) **แก้ไขปัญหา** ให้ลูกค้าแบบรวดเร็วและสามารถแก้ปัญหาโดยไม่ให้เกิดปัญหานั้นขึ้นอีก

เพื่อ (So that) **ให้ลูกค้าได้รับการบริการที่สะดวกและราบรื่น** และเกิดปัญหาน้อยที่สุด

Output บนหน้าจอ

- ลูกค้าเข้าเมนู “แจ้งปัญหา” → กรอกหัวข้อและรายละเอียดปัญหา
- แนบภาพหลักฐาน (ถ้ามี) → กด “ส่งคำร้อง”
- ระบบแสดงข้อความยืนยันพร้อมหมายเลข Ticket ID

4. แออดมินเข้าตรวจสอบ → ตอบกลับพร้อมเปลี่ยนสถานะคำร้อง
5. ลูกค้าสามารถดูสถานะล่าสุด และย้อนดูประวัติได้จากหน้า 'รายการปัญหาของฉัน'

Output ของข้อมูล

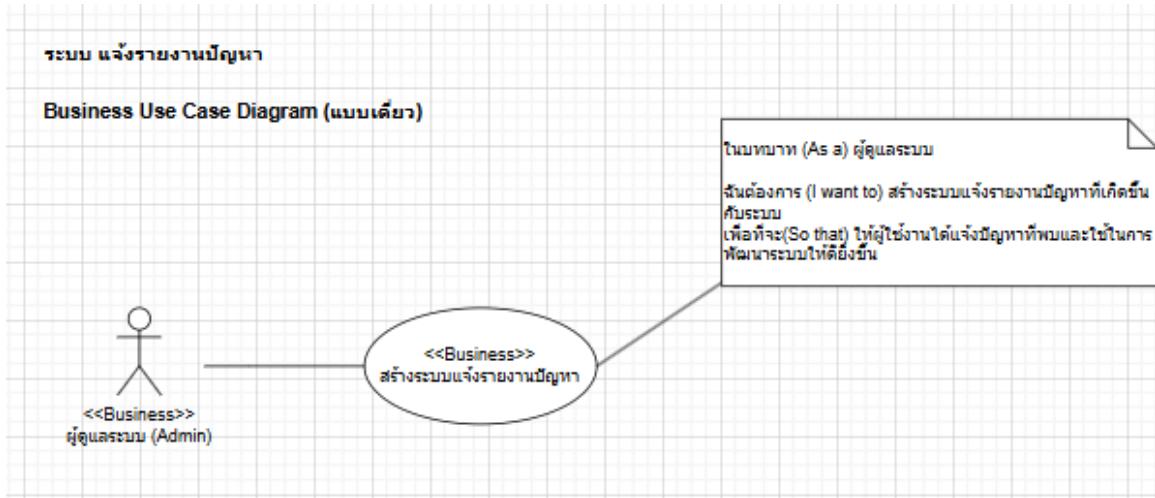
- ระบบเก็บรายละเอียดคำร้องปัญหา เช่น หมวดหมู่, วันที่, รายละเอียด, ไฟล์แนบ
- ระบบบันทึกสถานะของคำร้อง และเวลาอัปเดตล่าสุด
- ระบบแสดง Ticket ID สำหรับติดตาม
- สามารถดูประวัติการสื่อสารระหว่างลูกค้าและแออดมินได้ภายในคำร้องเดียวกัน

คำนามที่อาจจะถูกนำมาเป็นตารางในฐานข้อมูลของระบบ

คำนาม	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
รายงานปัญหา	เกี่ยวข้องโดยตรง ใช้เก็บคำร้องจากลูกค้า
รายละเอียดปัญหา	เกี่ยวข้องโดยตรง สำหรับการวิเคราะห์และแก้ไขปัญหา
ลูกค้า	เกี่ยวข้องโดยตรง เป็นผู้แจ้งปัญหา

สถานะคำร้อง	เกี่ยวข้องโดยตรง เพื่อแสดงความคืบหน้าการดำเนินการ
แออดมิน	เกี่ยวข้องโดยตรง เป็นผู้ตรวจสอบและตอบกลับคำร้อง
ไฟล์แนบ	เกี่ยวข้องโดยตรง ใช้เป็นหลักฐานในการพิจารณาแก้ปัญหา

Business Use Case Diagram (แบบเดียว)



Checklist: Business Use Case Diagram Business

Actor มี
> กำกับ Business Actor เป็นชื่อบทบาทเท่านั้น ไม่ใช่ชื่อคน ไม่เป็นชื่อบุคคล

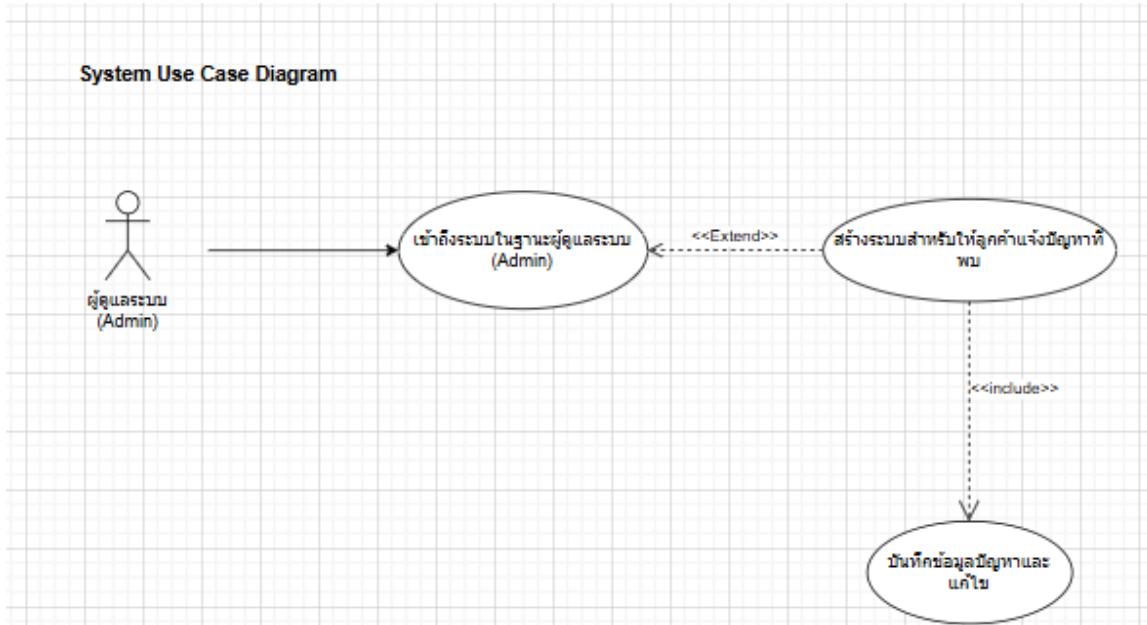
Business Use Case มี
> กำกับ

Business Use Case ขึ้นต้นด้วยคำว่า “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 “ระบบแจ้งรายงานปัญหา” สามารถถูกถ่ายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบ

คอมพิวเตอร์ได้หรือไม่ ต้องมีขั้นตอนทาง Security มาเกี่ยวหรือไม่

ตอบ Business Use Case “ระบบแจ้งรายงานปัญหา” สามารถถูกถ่ายไปเป็น กิจกรรมที่เกี่ยวข้องกับระบบ
คอมพิวเตอร์ได้ และต้องการขั้นตอนทาง Security (มี Privacy มาเกี่ยวข้อง)

เพราะฉะนั้น Business Use Case “ระบบแจ้งรายงานปัญหา” จะกลายเป็น 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” และ

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. เส้นโยงจาก System Actor ไปหา System Use Case จะต้องเป็นเส้นทึบ หัวลูกศรปลายเปิดซึ่งไปหา System Use Case
10. System Actor ต้องเป็นบทบาทของคนที่ใช้ระบบจริง ๆ
11. System Use Case ต้องเป็นคำที่แสดงพฤติกรรม
12. ถ้ามีกลไกทาง Security มาเกี่ยว ชื่อ System Use Case จะอยู่ในรูปแบบ “เข้าระบบในฐาน <Actor>” เท่านั้น
13. การใช้ <--<<extend>>---> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ชี้จาก System Use Case ทางเดียวไปยัง System Use Case หลัก
14. การใช้ <--<<include>>---> ต้องเป็นเส้นประ หัวลูกศรปลายเปิด ชี้จาก System Use Case ที่ใหญ่กว่า ไปยัง System Use Case ที่จะถูกรวบเข้ามา

Entity: User

ข้อมูลที่ต้องจัดเก็บ

- UserID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- Username: varchar, ไม่เป็น Null
- Password: varchar, ไม่เป็น Null
- Email: varchar, Null ได้
- LastName: varchar, Null ได้
- FirstName: varchar, Null ได้
- Birthday: date, Null ได้
- RoleID: integer, Foreign Key, ไม่เป็น Null

Userl D	Usernam e	Password	Email	LastNam e	FirstNam e	Birthda y	Rolel D
1001	user01	encrypt(123 4)	user01@mail.com	Kim	Minho	2000- 05-14	1
1002	gamer02	encrypt(567 8)	gamer02@mail.co m	Park	Jisoo	1999- 11-23	2

Entity: Role

ข้อมูลที่ต้องจัดเก็บ

- RoleID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- RoleName: varchar, ไม่เป็น Null

RoleID	RoleName
1	Admin
2	Customer

Entity: Game

ข้อมูลที่ต้องจัดเก็บ

- GameID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- GameName: varchar, ไม่เป็น Null
- GamePrice: decimal, ไม่เป็น Null
- Description: varchar, Null ได้

GameID	GameName	GamePrice	Description
2001	RPG Quest	1500.00	เกมผจญภัยแฟนตาซี
2002	Racing Pro	900.00	เกมแข่งรถสมจริง

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	Userl D	Game ID	Order ID	Title	Description	Created At	Resolve dAt	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	ระบบไม่ตัดบัตรฯ ครดิต	2025-09-03	None	2

						09:30:0 0		
--	--	--	--	--	--	--------------	--	--

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: Issue Category

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: ProblemAttachment

ข้อมูลที่ต้องจัดเก็บ

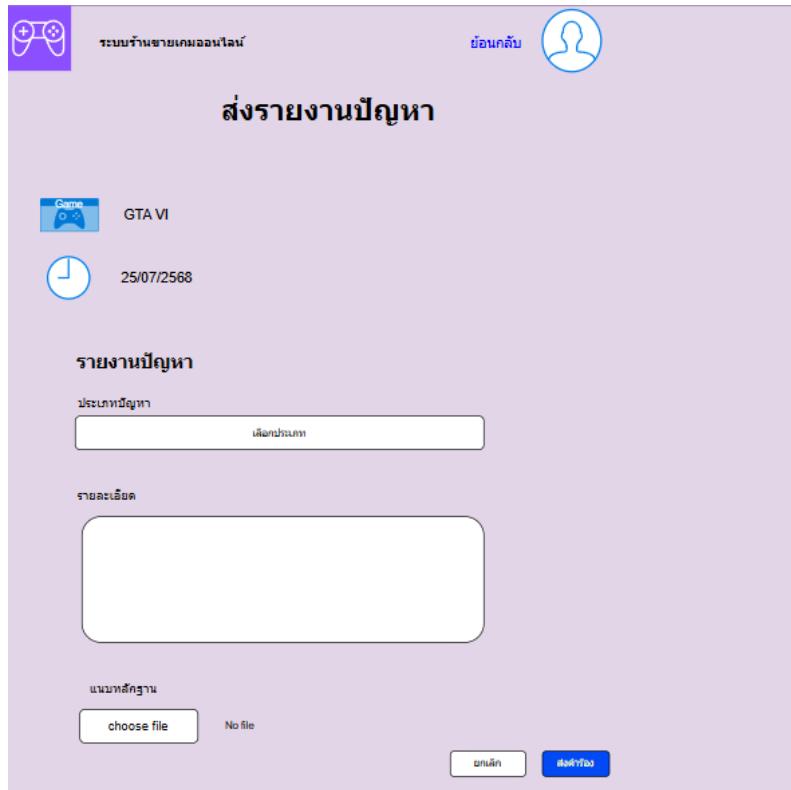
- - AttachmentID: Primary Key, integer, ไม่ซ้ำกัน, ไม่เป็น Null
- - ReportID: integer, Foreign Key, ไม่เป็น Null
- - FilePath: varchar, ไม่เป็น Null
- - UploadedAt: datetime, ไม่เป็น Null

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

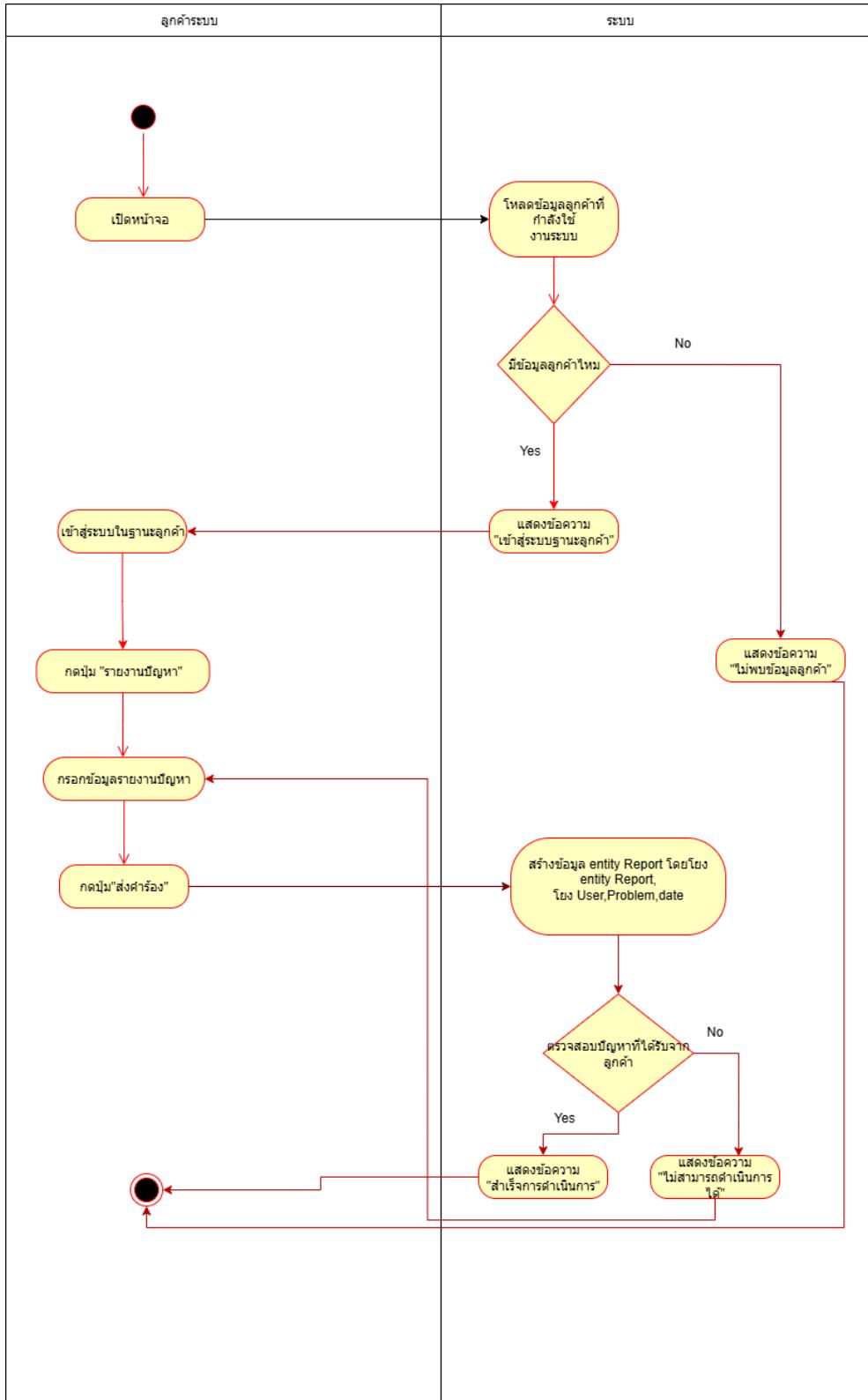
หลักการออกแบบ User Interface

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

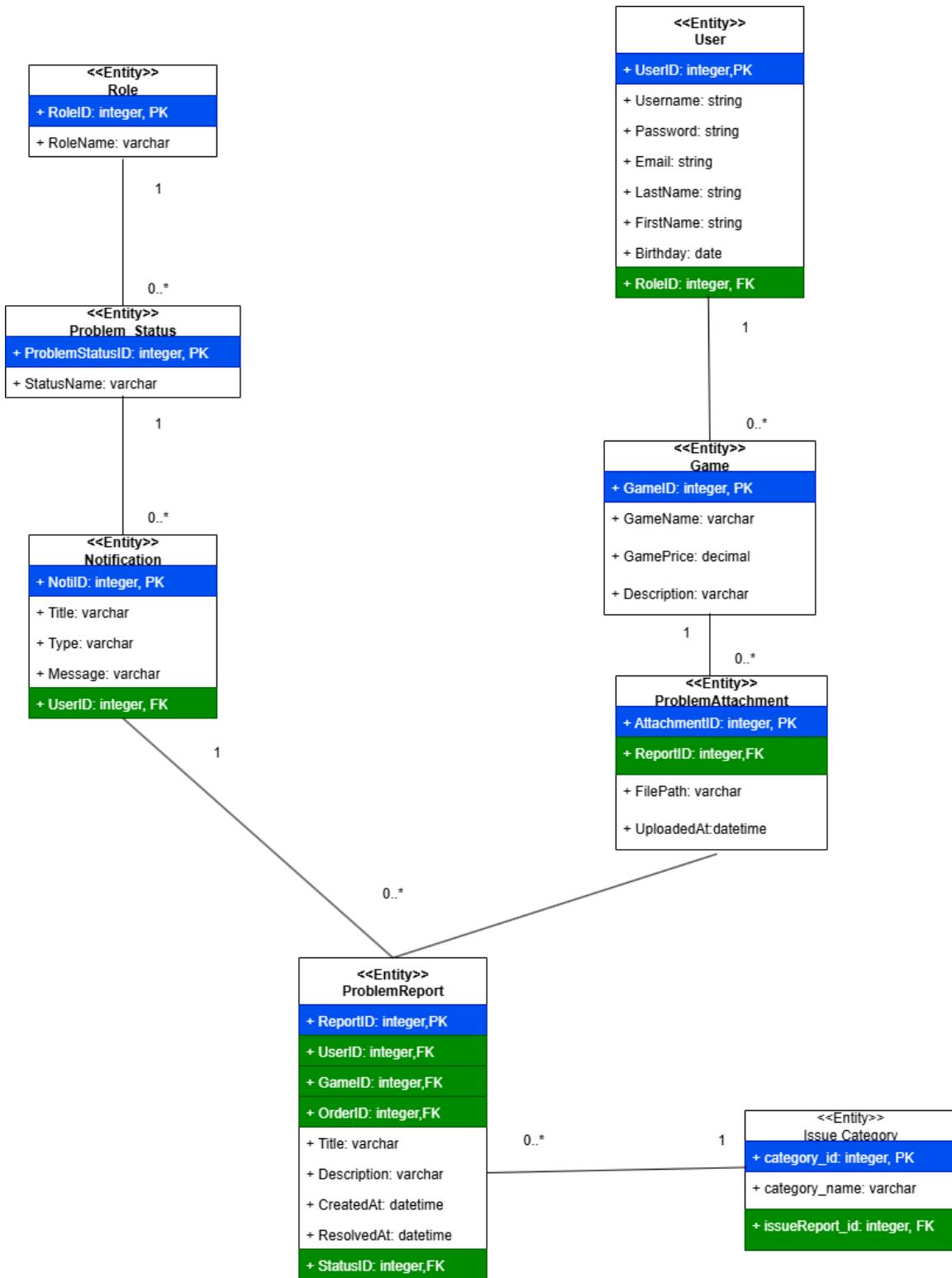
UI Design



Activity Diagram



Class Diagram ระดับ Analysis



ระบบรายงานปัญหา

The screenshot shows a dark-themed web application interface. At the top, there is a search bar with a magnifying glass icon and some small icons on the right. Below the header, a modal window titled "Report a Problem" is displayed. The modal contains several input fields: a dropdown menu for "Problem Category" with the placeholder "Select a category"; a text input field for "Title" with the placeholder "Short description"; and a larger text area for "Description" with the placeholder "Provide detailed information about the problem". Below these fields is a button labeled "Upload Screenshot / Proof" with a small camera icon. At the bottom of the modal is a prominent purple "Submit Report" button.

```
import { Card, Form, Input, Button, Select, Upload, message } from "antd";
import { UploadOutlined } from "@ant-design/icons";
import Navbar from "../components/Navbar";

export default function ReportPage() {
  const [form] = Form.useForm();

  const handleSubmit = (values: any) => {
    console.log("Report Submitted:", values);
  };

  return (
    <div style={{ background: "#141414", minHeight: "100vh", color: "#fff" }}>
      <Navbar />
      <Card
        title=<span style={{ color: "#fff" }}>Report a Problem</span>
        bordered={false}
        style={{
          maxWidth: 600,
          margin: "40px auto",
        }}
      >
        <Form
          {...form}
          layout="vertical"
          onFinish={handleSubmit}
        >
          <Form.Item
            label="Problem Category"
            name="category"
            rules={[{ required: true, message: "Please select a category" }]}
          >
            <Select>
              <Option value="1">Category 1</Option>
              <Option value="2">Category 2</Option>
              <Option value="3">Category 3</Option>
            </Select>
          </Form.Item>
          <Form.Item
            label="Title"
            name="title"
            rules={[{ required: true, message: "Please enter a title" }]}
          >
            <Input placeholder="Short description" />
          </Form.Item>
          <Form.Item
            label="Description"
            name="description"
            rules={[{ required: true, message: "Please provide detailed information" }]}
          >
            <Textarea placeholder="Provide detailed information about the problem" />
          </Form.Item>
          <Form.Item
            label="Upload Screenshot / Proof"
            name="proof"
            rules={[{ required: true, message: "Please upload a screenshot or proof" }]}
          >
            <UploadOutlined /> Click to Upload
          </Form.Item>
        </Form>
      </Card>
    </div>
  );
}
```

```

background: "#1f1f1f",
borderRadius: 12,
boxShadow: "0 0 10px rgba(255, 0, 255, 0.2)",
}}
>
<Form form={form} layout="vertical" onFinish={handleSubmit}>
 {/* Category */}
<Form.Item
  label={<span style={{ color: "#fff" }}>Problem Category</span>}
  name="category"
  rules={[{ required: true, message: "Please select a category" }]}
>
<Select
  placeholder="Select a category"
  style={{ background: "#2a2a2a", color: "#fff" }}
  dropdownStyle={{ background: "#1f1f1f", color: "#fff" }}
>
  <Select.Option value="technical">⚙️ Technical Issue</Select.Option>
  <Select.Option value="billing">💳 Billing Problem</Select.Option>
  <Select.Option value="login">🔒 Login/Authentication Issue</Select.Option>
  <Select.Option value="ui">💻 UI/UX Bug</Select.Option>
  <Select.Option value="performance">🚀 Performance Problem</Select.Option>
  <Select.Option value="crash">💥 App Crash / Freezing</Select.Option>
  <Select.Option value="purchase">🛒 Purchase/Payment Error</Select.Option>
  <Select.Option value="content">📄 Missing or Incorrect Content</Select.Option>
  <Select.Option value="feedback">💬 Suggestion/Feedback</Select.Option>
  <Select.Option value="other">❓ Other</Select.Option>
</Select>
</Form.Item>

 {/* Title */}
<Form.Item
  label={<span style={{ color: "#fff" }}>Title</span>}

```

```

name="title"
rules={[{ required: true, message: "Please enter a title" }]}
>
<Input
  placeholder="Short description"
  style={{ background: "#fefefeff", color: "#070707ff" }}
/>
</Form.Item>

/* Description */
<Form.Item
  label={<span style={{ color: "#fffffcff" }}>Description</span>}
  name="description"
  rules={[{ required: true, message: "Please enter a description" }]}
>
<Input.TextArea
  rows={4}
  placeholder="Provide detailed information about the problem"
  style={{ background: "#ffffffff", color: "#010101ff" }}
/>
</Form.Item>

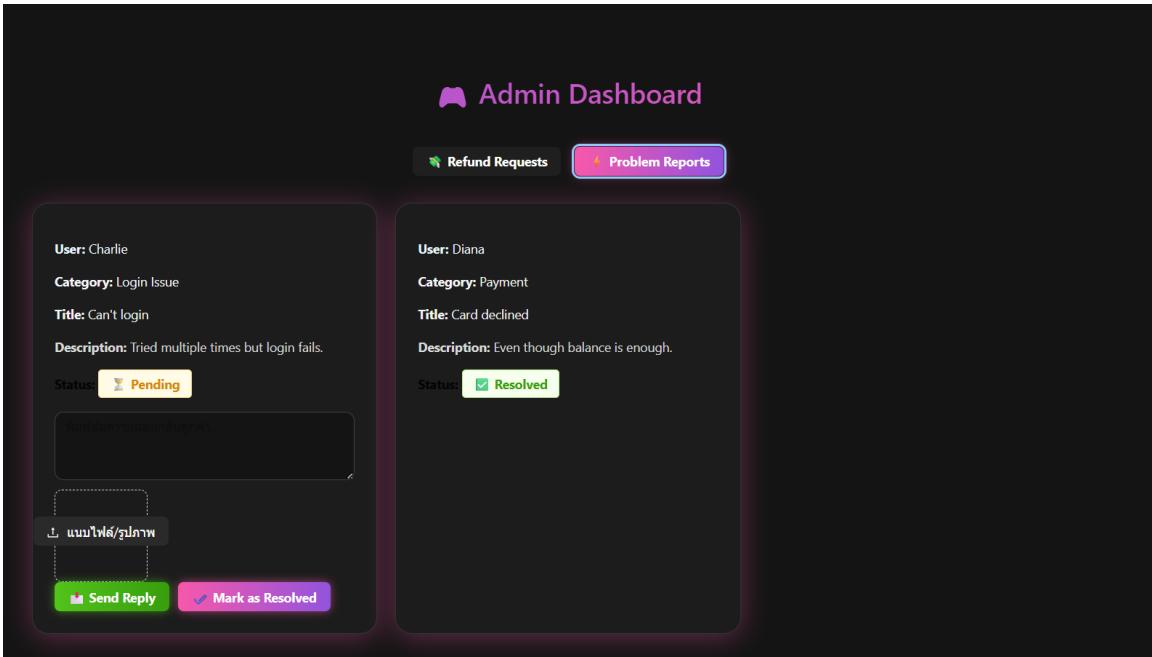
/* ✅ Upload Screenshot */
<Form.Item
  label={<span style={{ color: "#fff" }}>Upload Screenshot / Proof</span>}
  name="attachments"
  valuePropName="fileList"
  getValueFromEvent={(e) => (Array.isArray(e) ? e : e?.fileList)}
>
<Upload
  name="file"
  listType="picture"
  maxCount={3}
  beforeUpload={(file) => {
    const isImage = file.type.startsWith("image/");
  }
}}
</Form.Item>

```

```
        if (!isImage) {
            message.error("You can only upload image files!");
        }
        return isImage || Upload.LIST_IGNORE;
    }
}

<Form.Item>
    <Upload icon={<UploadOutlined />}>Click to Upload</Upload>
</Upload>
</Form.Item>

/* Submit Button */
<Form.Item>
    <Button
        type="primary"
        htmlType="submit"
        style={{
            background: "linear-gradient(90deg, #a34af2, #ff5ca8)",
            border: "none",
            color: "#fff",
            width: "100%",
        }}
    >
        Submit Report
    </Button>
</Form.Item>
</Form>
</Card>
</div>
);
}
```



```
// src/pages/AdminPage.tsx

import { useState, useEffect } from "react";
import { Card, Button, Typography, Tag, Input, Upload, message, Modal } from "antd";
import { UploadOutlined } from "@ant-design/icons";

const { Title } = Typography;
const { TextArea } = Input;

interface RefundRequest {
  id: number;
  user: string;
  game: string;
  reason: string;
  status: "Pending" | "Approved" | "Rejected";
}

interface ProblemReport {
  id: number;
  user: string;
  category: string;
  title: string;
}
```

```

description: string;
resolved: boolean;
}

export default function AdminPage() {
const [refunds, setRefunds] = useState<RefundRequest[]>([]);
const [problems, setProblems] = useState<ProblemReport[]>([]);
const [activeTab, setActiveTab] = useState<"refunds" | "problems">("refunds");

// reply state เก็บข้อมูลตอบกลับต่อ report
const [replies, setReplies] = useState<Record<number, { text: string; fileList: any[] }>>({});

const [previewImage, setPreviewImage] = useState<string>(""); // สำหรับ preview
const [previewOpen, setPreviewOpen] = useState(false);

useEffect(() => {
  // mock data
  setRefunds([
    { id: 1, user: "Alice", game: "Cyberpunk 2077", reason: "Buggy gameplay", status: "Pending" },
    { id: 2, user: "Bob", game: "Elden Ring", reason: "Accidental purchase", status: "Approved" },
  ]);

  setProblems([
    { id: 1, user: "Charlie", category: "Login Issue", title: "Can't login", description: "Tried multiple times but login fails.", resolved: false },
    { id: 2, user: "Diana", category: "Payment", title: "Card declined", description: "Even though balance is enough.", resolved: true },
  ]);
}, []);

const handleRefundAction = (id: number, action: "Approved" | "Rejected") => {
  setRefunds((prev) =>
    prev.map((req) =>
      req.id === id ? { ...req, status: action } : req
    )
  );
}

```

```

};

};

const handleResolveProblem = (id: number) => {
  setProblems((prev) =>
    prev.map((rep) =>
      rep.id === id ? { ...rep, resolved: true } : rep
    )
  );
};

const handleSendReply = (id: number) => {
  const reply = replies[id];
  if (!reply || (!reply.text && reply.fileList.length === 0)) {
    message.warning("กรุณาพิมพ์ข้อความหรือติดไฟล์อีเมลข้อความที่ 1 อย่างน้อย");
    return;
  }

  console.log("Reply sent:", { problemId: id, ...reply });
  message.success("ส่งคำตอบกลับไปยังลูกค้าเรียบร้อย!");

  // clear reply state
  setReplies((prev) => ({ ...prev, [id]: { text: "", fileList: [] } }));
};

// ฟังก์ชันสำหรับ preview รูป
const handlePreview = async (file: any) => {
  setPreviewImage(file.thumbUrl || file.url || "");
  setPreviewOpen(true);
};

return (
  <div style={{ background: "#141414", minHeight: "100vh", padding: "40px", color: "#fff" }}>
    <Title
      level={2}

```

```
style={{
  textAlign: "center",
  background: "linear-gradient(90deg, #9254de, #f759ab)",
  WebkitBackgroundClip: "text",
  color: "transparent",
  marginBottom: "40px",
}}
>
 Admin Dashboard
</Title>

/* Tabs */
<div style={{ textAlign: "center", marginBottom: "30px" }}>
  <Button
    type={activeTab === "refunds" ? "primary" : "default"}
    onClick={() => setActiveTab("refunds")}
    style={{
      marginRight: "15px",
      background: activeTab === "refunds" ? "linear-gradient(90deg, #9254de, #f759ab)" :
      "#1f1f1f",
      border: "none",
      color: "#fff",
      fontWeight: "bold",
      boxShadow: activeTab === "refunds" ? "0 0 12px rgba(146, 84, 222, 0.6)" : "none",
    }}
  >
 Refund Requests
  </Button>
  <Button
    type={activeTab === "problems" ? "primary" : "default"}
    onClick={() => setActiveTab("problems")}
    style={{
      background: activeTab === "problems" ? "linear-gradient(90deg, #f759ab, #9254de)" :
      "#1f1f1f",
    }}
  >
```

```
border: "none",
color: "#fff",
fontWeight: "bold",
boxShadow: activeTab === "problems" ? "0 0 12px rgba(247, 89, 171, 0.6)" : "none",
}}
>
⚡ Problem Reports
</Button>
</div>

/* Refund Requests */
{activeTab === "refunds" && (
<div style={{ display: "grid", gridTemplateColumns: "repeat(auto-fill, minmax(350px, 1fr))", gap: "20px" }}>
{refunds.map((req) => (
<Card
key={req.id}
bordered={false}
hoverable
style={{
background: "#1c1c1c",
borderRadius: "18px",
border: "1px solid rgba(255,255,255,0.1)",
boxShadow: "0 4px 30px rgba(146, 84, 222, 0.35)",
transition: "transform 0.2s ease, box-shadow 0.2s ease",
}}
>
<p style={{ color: "#e6f7ff" }}><b>User:</b> {req.user}</p>
<p style={{ color: "#f5f5f5" }}><b>Game:</b> {req.game}</p>
<p style={{ color: "#d9d9d9" }}><b>Reason:</b> {req.reason}</p>
<p>
<b>Status:</b>{" "}
<Tag
style={{ fontWeight: "bold", padding: "5px 12px", fontSize: "14px" }}
color={
```

```
req.status === "Pending" ? "gold" :  
  req.status === "Approved" ? "green" : "red"  
}  
>  
{req.status}  
</Tag>  
</p>  
  
{req.status === "Pending" && (  
  <div style={{ marginTop: "15px" }}>  
    <Button  
      onClick={() => handleRefundAction(req.id, "Approved")}  
      style={{  
        marginRight: "10px",  
        background: "linear-gradient(90deg, #52c41a, #389e0d)",  
        border: "none",  
        color: "white",  
        fontWeight: "bold",  
        boxShadow: "0 0 10px rgba(82,196,26,0.6)",  
      }}  
    >  
       Approve  
    </Button>  
    <Button  
      onClick={() => handleRefundAction(req.id, "Rejected")}  
      style={{  
        background: "linear-gradient(90deg, #f5222d, #cf1322)",  
        border: "none",  
        color: "white",  
        fontWeight: "bold",  
        boxShadow: "0 0 10px rgba(245,34,45,0.6)",  
      }}  
    >  
       Reject  
    </Button>  
  </div>  
)}
```

```

        </Button>
    </div>
)
</Card>
))
</div>
)

/* Problem Reports */
{activeTab === "problems" && (
<div style={{ display: "grid", gridTemplateColumns: "repeat(auto-fill, minmax(350px, 1fr))", gap: "20px" }}>
{problems.map((rep) => (
<Card
key={rep.id}
bordered={false}
hoverable
style={{
background: "#1c1c1c",
borderRadius: "18px",
border: "1px solid rgba(255,255,255,0.1)",
boxShadow: "0 4px 30px rgba(247, 89, 171, 0.35)",
transition: "transform 0.2s ease, box-shadow 0.2s ease",
}}
>
<p style={{ color: "#e6f7ff" }}><b>User:</b> {rep.user}</p>
<p style={{ color: "#f5f5f5" }}><b>Category:</b> {rep.category}</p>
<p style={{ color: "#f5f5f5" }}><b>Title:</b> {rep.title}</p>
<p style={{ color: "#d9d9d9" }}><b>Description:</b> {rep.description}</p>
<p>
<b>Status:</b>{" "}
{rep.resolved ? (
<Tag style={{ fontWeight: "bold", padding: "5px 12px", fontSize: "14px" }}
color="green">

```



Resolved

```

        </Tag>
    ) : (
        <Tag style={{ fontWeight: "bold", padding: "5px 12px", fontSize: "14px" }}
color="gold">
     Pending
    </Tag>
)
</p>

{!rep.resolved && (
<div style={{ marginTop: "15px" }}>
    /* Reply Box */
    <TextArea
        rows={3}
        placeholder="ພິເພີ້ນຂໍ້ອະນາຄາມຕອບກລັບລູກຄ້າ..."
        value={replies[rep.id]?.text || ""}
        onChange={(e) =>
            setReplies((prev) => ({
                ...prev,
                [rep.id]: { text: e.target.value, fileList: prev[rep.id]?.fileList || [] },
            }))
        }
        style={{
            marginBottom: "10px",
            background: "#141414",
            color: "white",
            borderRadius: "10px",
            border: "1px solid #434343",
        }}
    />

    /* Upload File + Preview */
    <Upload
        fileList={replies[rep.id]?.fileList || []}
        beforeUpload={() => false}

```

```
onChange={({ fileList }) =>
  setReplies((prev) => ({
    ...prev,
    [rep.id]: { text: prev[rep.id]?.text || "", fileList },
  }))
}

onPreview={handlePreview}
listType="picture-card" //  แสดงเป็น thumbnail
multiple

>
<Button
  icon={<UploadOutlined />}
  style={{ marginBottom: "10px", background: "#2a2a2a", color: "white", border:
"none" }}
>
  แนบไฟล์/รูปภาพ
</Button>
</Upload>

<Button
  onClick={() => handleSendReply(rep.id)}
  style={{
    marginRight: "10px",
    background: "linear-gradient(90deg, #52c41a, #389e0d)",
    border: "none",
    color: "white",
    fontWeight: "bold",
    boxShadow: "0 0 10px rgba(82,196,26,0.6)",
  }}
>
   Send Reply
</Button>

<Button
```

```
    onClick={() => handleResolveProblem(rep.id)}
```

```
    style={{
```

```
        background: "linear-gradient(90deg, #f759ab, #9254de)",
```

```
        border: "none",
```

```
        color: "white",
```

```
        fontWeight: "bold",
```

```
        boxShadow: "0 0 10px rgba(247, 89, 171, 0.6)",
```

```
    }}
```

```
>
```

```
    ✓ Mark as Resolved
```

```
</Button>
```

```
</div>
```

```
)}
```

```
</Card>
```

```
))}
```

```
</div>
```

```
)}
```

```
/* Modal Preview */
```

```
<Modal open={previewOpen} footer={null} onCancel={() => setPreviewOpen(false)}>
```

```
    <img alt="preview" style={{ width: "100%" }} src={previewImage} />
```

```
</Modal>
```

```
</div>
```

```
);
```

```
}
```

Backend

Entity Users

```
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"`
}

}
```

Entity 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"`
}
```

Entity Role

```
package models

type Role struct {
    RoleID  uint `gorm:"primaryKey" json:"role_id"`
    RoleName string `json:"role_name"`

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

Entity Problem_status

```
package models

type ProblemStatus struct {
    ProblemStatusID uint `gorm:"primaryKey" json:"problem_status_id"`
    StatusName     string `json:"status_name"`
}
```

Entity 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"`

}
```

Entity Problem_attachment

```
package models

import "time"

type ProblemAttachment struct {
    AttachmentID uint     `gorm:"primaryKey" json:"attachment_id"`
    ReportID     uint     `json:"report_id"`
    FilePath     string   `json:"file_path"`
    UploadedAt   time.Time `json:"uploaded_at"`
}
```

Entity:Problem_report

```
package models

import "time"

type ProblemReport struct {
```

```

ReportID uint `gorm:"primaryKey" json:"report_id"`
UserID uint `json:"user_id"`
GameID uint `json:"game_id"`
OrderID uint `json:"order_id"`
Title string `json:"title"`
Description string `json:"description"`
CreatedAt time.Time `json:"created_at"`
ResolvedAt time.Time `json:"resolved_at"`

Attachments []ProblemAttachment `gorm:"foreignKey:ReportID"`
}

```

Entity:issue_catagory

```

package models

type IssueCategory struct {
    CategoryID uint `gorm:"primaryKey" json:"category_id"`
    CategoryName string `json:"category_name"`
    IssueReportID uint `json:"issue_report_id"`
}

```

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)
    }
}
```

Problem_controller

```
package controllers

import (
    "net/http"
    "sa-gameShop-backend/models"
    "time"

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

func GetProblems(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var problems []models.ProblemReport
        if err := db.Preload("Attachments").Find(&problems).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
            return
        }
        c.JSON(http.StatusOK, problems)
    }
}

func GetProblemByID(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var problem models.ProblemReport
        if err := db.Preload("Attachments").First(&problem, c.Param("id")).Error; err != nil {
            c.JSON(http.StatusNotFound, gin.H{"error": "Problem not found"})
            return
        }
        c.JSON(http.StatusOK, problem)
    }
}

func CreateProblem(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var problem models.ProblemReport
        if err := c.ShouldBindJSON(&problem); err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
            return
        }
        problem.CreatedAt = time.Now()
```

```
problem.StatusID = 1 // Open
if err := db.Create(&problem).Error; err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}
c.JSON(http.StatusCreated, problem)
}

func UpdateProblemStatus(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var problem models.ProblemReport
        if err := db.First(&problem, c.Param("id")).Error; err != nil {
            c.JSON(http.StatusNotFound, gin.H{"error": "Problem 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
        }
        problem.StatusID = req.StatusID
        problem.ResolvedAt = time.Now()
        db.Save(&problem)
        c.JSON(http.StatusOK, problem)
    }
}
```

Status_controller

```
package controllers

import (
    "net/http"
    "sa-gameShop-backend/models"

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

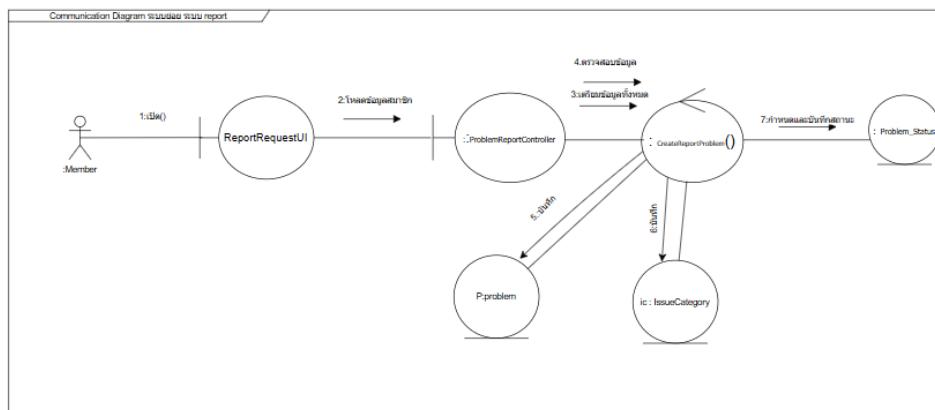
func GetRefundStatuses(db *gorm.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        var statuses []models.RefundStatus
        if err := db.Find(&statuses).Error; err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
            return
        }
        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)
    }
}
```

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	



Communication Class diagram

