



โครงการ วิชา 1101213 Project in Blockchain and Cryptocurrencies

1. ชื่อหัวข้อโครงการ

Blockchain for Condominium Business

2. สมาชิกในกลุ่ม

รหัสนักศึกษา	ชื่อ - นามสกุล
B6501495	นายสติพล เสน่ห์ใหม่
B6501761	นางสาวธิติกานต์ ไวยสุวรรณ
B6501785	นางสาวพรทิวา กาละนิโຍ
B6523909	นายปัณฑวัฒน์ เจริญบุญญสกิตย์
B6534028	นางสาวสิริยากร ศรีสวัสดิ์

รายละเอียดการแบ่งงาน

ลำดับ	รหัสนักศึกษา	ชื่อ - นามสกุล	หน้าที่/ความรับผิดชอบ	สัดส่วนการ ทำงาน
12	B6501495	นายสติพล เสน่ห์ใหม่	<ul style="list-style-type: none"> — Suitability Evaluation Framework — Flowchart — รูปเล่มรายงาน 	20%
21	B6501761	นางสาวธิติกานต์ ไวยสุวรรณ	<ul style="list-style-type: none"> — Suitability Evaluation Framework — พังก์ชันการฟ่อนคอนโดยี่ — Flowchart — รูปเล่มรายงาน 	20%
22	B6501785	นางสาวพรทิวา กะละนิโย	<ul style="list-style-type: none"> — พังก์ชันการเข่าคอนโดยี่ — Flowchart — รูปเล่มรายงาน 	20%
35	B6523909	นายปัณรวัฒน์ เจริญบุญญสติตย์	<ul style="list-style-type: none"> — พังก์ชันการซื้อขายคอนโดยี่ — Flowchart — รูปเล่มรายงาน 	20%
46	B6534028	นางสาวสิริยากร ศรีสวัสดิ์	<ul style="list-style-type: none"> — Flowchart — รูปเล่มรายงาน 	20%
			ร้อยละห้าหมื่น	100%

ที่มาและความสำคัญของปัญหา

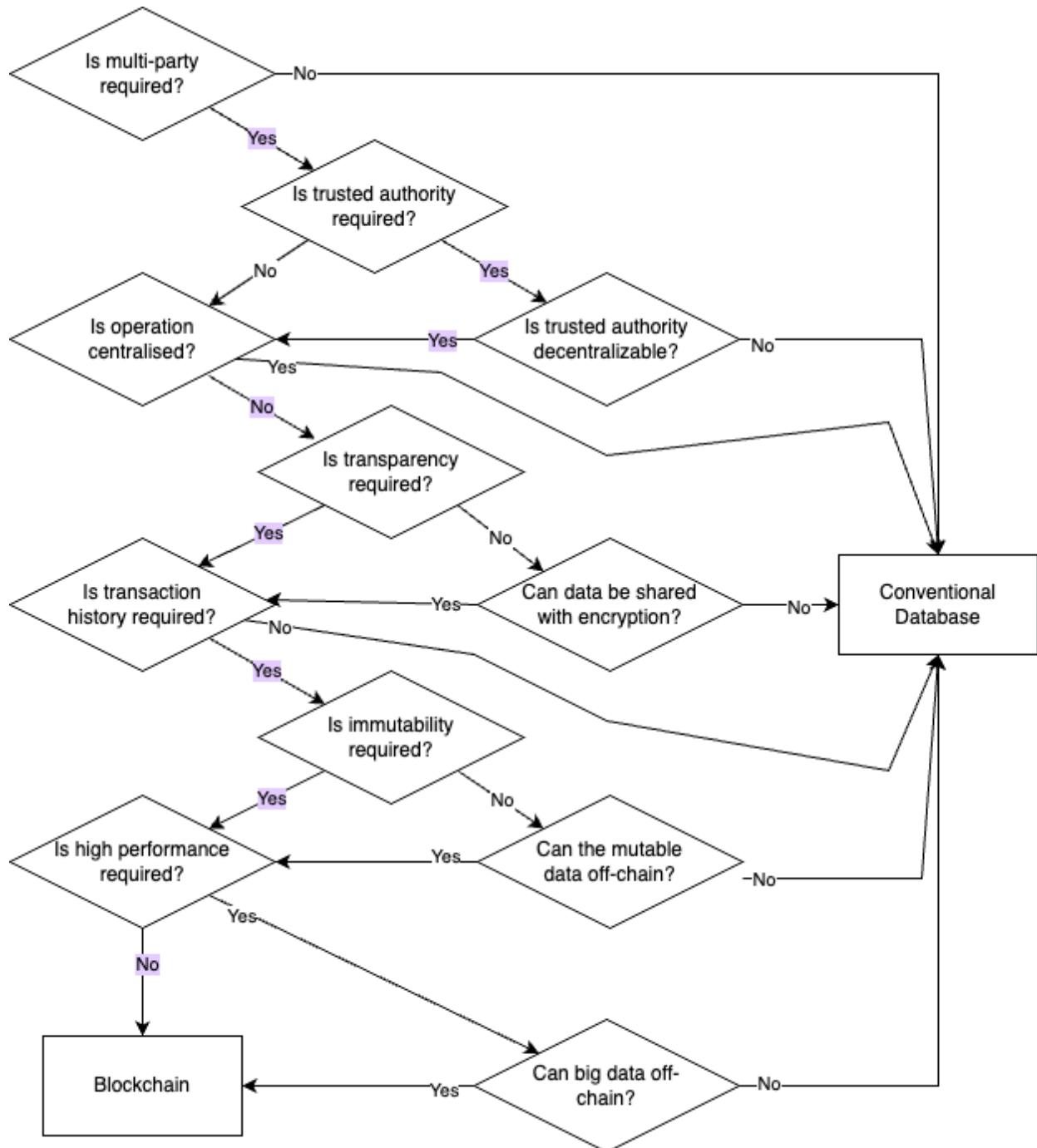
ในปัจจุบันการทำธุรกรรมของธุรกิจค่อนโดยมีเนียมได้ใช้ธนาคารเป็นสื่อกลางในการทำธุรกรรมและในบางครั้งอาจมีความยุ่งยากในการติดต่อทำธุรกรรมกับธนาคารและไฟแนนซ์และอาจมีการเปลี่ยนแปลงในสัญญาที่ธนาคารกำหนด ทางคนจะจัดทำจึงอย่างใช้ Smart contract มาปรับปรุงและแก้ไขปัญหาเดิมที่มีอยู่โดยการใช้ Smart Contract ในการทำสัญญาทางการเงินไม่ว่าจะเป็นการขายขาด การให้เช่า และการผ่อน เพื่อความปลอดภัยใน สัญญาและป้องกันการโงงและละเมิดสัญญา ในส่วนของเรื่องความปลอดภัยของผู้อยู่อาศัย ทางเราจะพัฒนาที่บันทึกการเข้าออกของคนภายนอก โดยนำหลักการของ Blockchain มาใช้ในการพัฒนา

วัตถุประสงค์

- เพื่อลดความยุ่งยากในการดำเนินการกับ Third Party
- เพื่อให้สัญญามีความแน่นอนและไม่สามารถเปลี่ยนแปลงได้
- เพื่อป้องกันการโงง
- เพื่อตรวจสอบประวัติอสังหาริมทรัพย์

แผนกราฟ

Suitability Evaluation Framework



Condominium Business	
Is multi-party required?	Required
Is trusted authority required?	Required
Is trusted authority decentralizable?	Required (Dec)
Is operation centralized?	Not Required
Is transparency required?	Required
Is transaction history required?	Required
Is immutability required?	Required
Is high performance required?	Not Required
Result	Blockchain

Reason

1. Is multi-party required ? (Required)

จำเป็นต้องมีหลายฝ่ายในการทำธุรกรรมอสังหาริมทรัพย์ โดยทั่วไปแล้ว ธุรกรรมเหล่านี้เกี่ยวข้องกับ ผู้ซื้อ ผู้เช่า นิติบุคคล และเจ้าของคอนโด

2. Is trusted authority required ? (Required)

ธุรกรรมอสังหาริมทรัพย์ต้องผ่านตัวกลางที่เชื่อถือได้ เพื่อให้แน่ใจว่าผู้ที่มีอำนาจในการอนุมัติการทำธุรกรรมได้อย่างถูกต้อง และป้องกันการปลอมแปลง

3. Is trusted authority decentralizable ? (Required)

สามารถให้สิทธิ์แก่บุคคลการที่เชื่อถือได้เพื่อตรวจสอบ และยืนยันธุรกรรม โดยไม่ต้องพึ่งพาบุคคลคนเดียว

4. Is operation centralized ? (No required)

เพราะต้องการให้มีหลายฝ่ายสามารถบันทึกข้อมูลได้ และสามารถให้บุคคลทั่วไปสามารถตรวจสอบได้ เช่น รายละเอียดของสัญญา ราคาของคอนโด

5. Is transparency required ? (Required)

ธุรกรรมอสังหาริมทรัพย์ ต้องมีความโปร่งใส เพื่อให้ผู้ซื้อสามารถตรวจสอบข้อมูลต่าง ๆ เกี่ยวกับคอนโดได้

6. Is transaction history required ? (Required)

จำเป็นต้องมีประวัติธุรกรรม เพื่อติดตามการเปลี่ยนแปลงกรรมสิทธิ์ และป้องกันการปลอมแปลง โดยธุรกรรมที่ต้องการเก็บประวัติ จะเป็นธุรกรรมการเก็บค่าเช่า ธุรกรรมการเก็บค่างวด(ผ่อน) ธุรกรรมการซื้อขาย

7. Is immutability required ? (Required)

จำเป็นต้องมีการเปลี่ยนแปลงไม่ได้ เพื่อป้องกันการปลอมแปลง

8. Is high performance required ? (No Required)

ธุรกรรมอสังหาริมทรัพย์โดยทั่วไปไม่จำเป็นต้องมีประสิทธิภาพสูง เนื่องจากไม่ได้เกิดขึ้นบ่อยนักที่จะเกิดการซื้อ และเช่าคอนโด

พังก์ชันในการผ่อนคุณโดย

1. พังก์ชันกรอกข้อมูลส่วนตัว

ให้ผู้ใช้กรอก ชื่อ นามสกุล เลขบัตรประจำตัวประชาชน ที่อยู่ เบอร์โทรศัพท์

2. พังก์ชันการกรอกข้อมูลการซื้อ

กรอกราคาห้อง, ราคากារน์, ระยะเวลาการผ่อน(ปี)

3. พังก์ชันตรวจสอบเงินในบัญชี

เช็คเงินใน wallet ของผู้ซื้อว่ามีพอหรือไม่ในการวางแผนคุณโดย + เงินวงแหวก

4. พังก์ชันการคิดอัตราดอกเบี้ยต่อปี

การคิดอัตราดอกเบี้ยต่อปีจะมีเงื่อนไขการคิด โดยมีปัจจัยมาจากการวางแผนคุณโดย + เงินวงแหวก

มีเงื่อนไขดังนี้

- วางแผนน้อยกว่า 10%, ระยะเวลาผ่อนน้อยกว่า 10 ปี = อัตราดอกเบี้ย 5%
 - วางแผนน้อยกว่า 10%, ระยะเวลาผ่อนมากกว่าหรือเท่ากับ 10 ปี และน้อยกว่า 20 ปี = อัตราดอกเบี้ย 6%
 - วางแผนน้อยกว่า 10%, ระยะเวลามากกว่าหรือเท่ากับ 20 ปี = อัตราดอกเบี้ย 7%
 - วางแผนน้อยกว่า 20%, ระยะเวลาผ่อนน้อยกว่า 10 ปี = อัตราดอกเบี้ย 4%
 - วางแผนน้อยกว่า 20%, ระยะเวลาผ่อนมากกว่าหรือเท่ากับ 10 ปี และน้อยกว่า 20 ปี = อัตราดอกเบี้ย 5%
 - วางแผนน้อยกว่า 20%, ระยะเวลามากกว่าหรือเท่ากับ 20 ปี = อัตราดอกเบี้ย 6%
 - วางแผนมากกว่า 20%, ระยะเวลาผ่อนน้อยกว่า 10 ปี = อัตราดอกเบี้ย 3%
 - วางแผนมากกว่า 20%, ระยะเวลาผ่อนมากกว่าหรือเท่ากับ 10 ปี และน้อยกว่า 20 ปี = อัตราดอกเบี้ย 4%
 - วางแผนมากกว่า 20%, ระยะเวลามากกว่าหรือเท่ากับ 20 ปี = อัตราดอกเบี้ย 5%
5. พังก์ชันแสดงราคาคุณโดยรวมดอกเบี้ยทั้งหมด
 6. พังก์ชันแสดง/เก็บเงินค่าดาวน์คุณโดย
 7. พังก์ชันแสดง/เก็บเงินค่างวดต่อเดือน
 8. พังก์ชันการแปลงค่าเงินโดยใช้ Oracle

สิทธิ์การเข้าถึงของระบบผ่อน

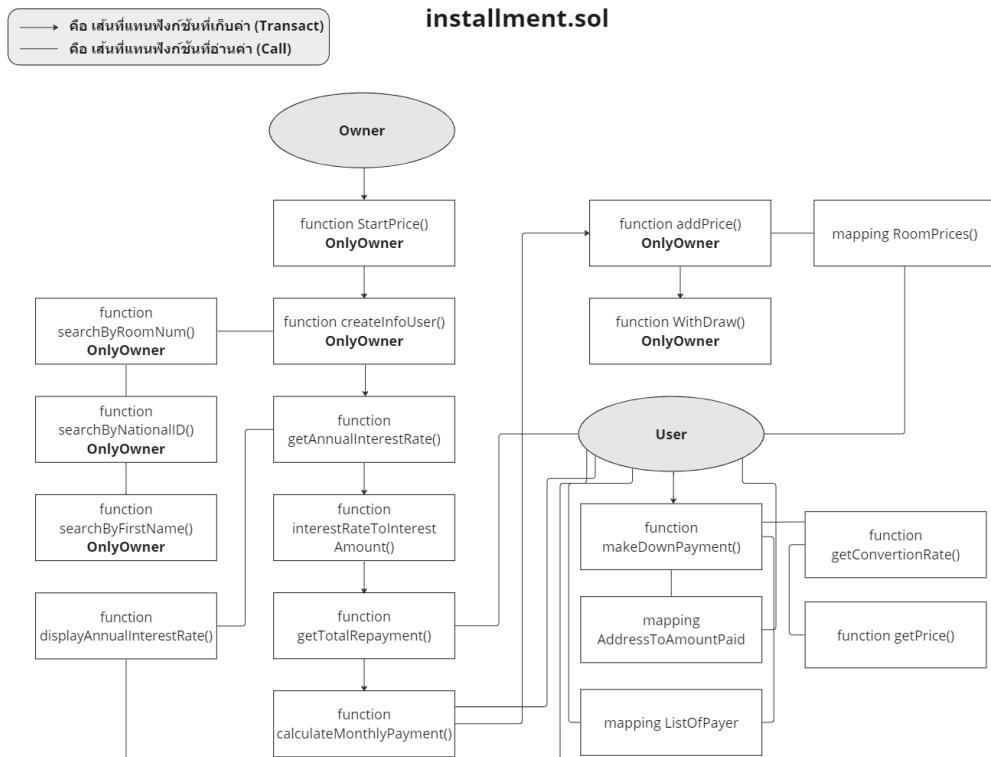
Installment.sol

Function	คำอธิบาย	เจ้าของ	ผู้ใช้/ผู้ผ่อน
StartPrice()	เพิ่มค่าราคาห้อง, ราคากារน์, ระยะเวลา		
createInfoUser()	สร้างข้อมูลผู้เชื่อ		
addPrice()	เพิ่มราคาห้องทั้งหมดรวมดอกเบี้ยและราคางวดรายเดือน		
getAnnualInterestRate()	คำนวณอัตราดอกเบี้ย		
displayAnnualInterestRate()	แสดงอัตราดอกเบี้ย		
interestRateToInterestAmount()	คำนวณดอกเบี้ยต่อปี		
getTotalRepayment()	ราคาเต็มรวมดอกเบี้ย		
calculateMonthlyPayment()	คำนวณค่างวดรายเดือนดอกเบี้ย		
makeDownPayment()	ชำระเงินค่าห้อง		
WithDraw()	ถอนเงิน		
searchByRoomNum()	ค้นหาข้อมูลผู้เชื่อโดยเลขที่ห้อง		
searchByNationalID	ค้นหาข้อมูลผู้เชื่อโดยเลขบัตรประชาชน		
searchByFirstName()	ค้นหาข้อมูลผู้เชื่อโดยชื่อผู้เชื่อ		
getPrice()	แปลงค่าเงิน USD กับ ETH		
getConversionRate()	แปลงค่าเงิน USD กับ ETH		

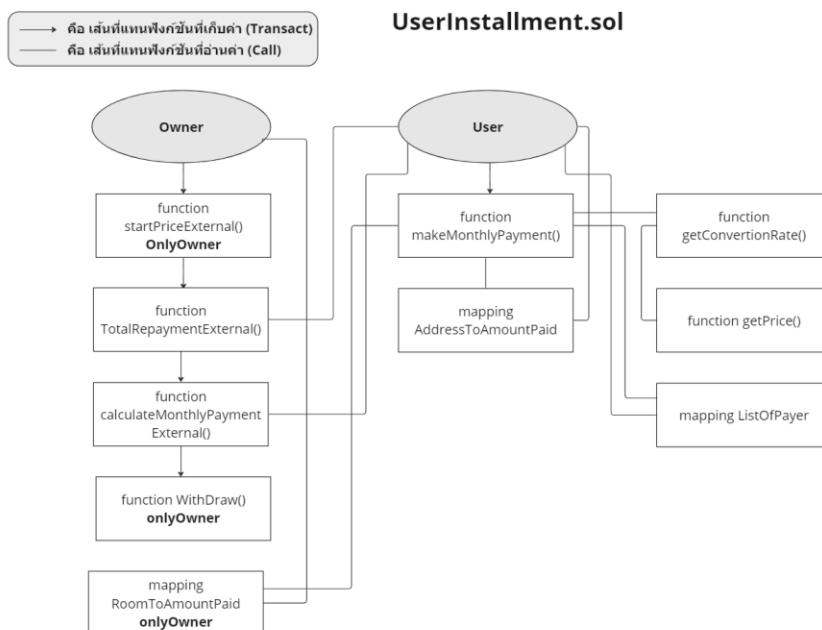
UserInstallment.sol

Function	คำอธิบาย	เจ้าของ	ผู้ใช้/ผู้ผ่อน
StartPrice()	เพิ่มค่าราคาห้อง, ราคากារน์, ระยะเวลา		
calculteMonthlyPaymentExternal()	คำนวณเงินค่าห้องรายเดือน		
TotalRepaymentExternal()	คำนวณยอดรวมการชำระคืน		
makeMonthlyPayment()	ชำระเงินค่าห้องรายเดือน		
WithDraw()	ถอนเงิน		
getRoomPaymentInfo()	แสดงข้อมูลการชำระเงินของห้องคอนโด		

การออกแบบโครงสร้าง/สถาปัตยกรรมระบบผ่อนชำระ



OnlyOwner หมายถึงผู้ขาย หรือ ผู้ที่ Deploy Smart Contract



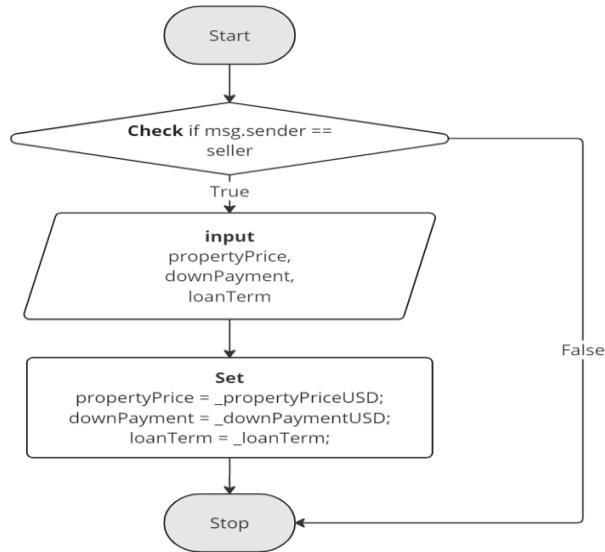
OnlyOwner หมายถึงผู้ขาย หรือ ผู้ที่ Deploy Smart Contract

การออกแบบขั้นตอนการทำงานของทุกฟังก์ชันของระบบผ่อนชำระ

installment.sol

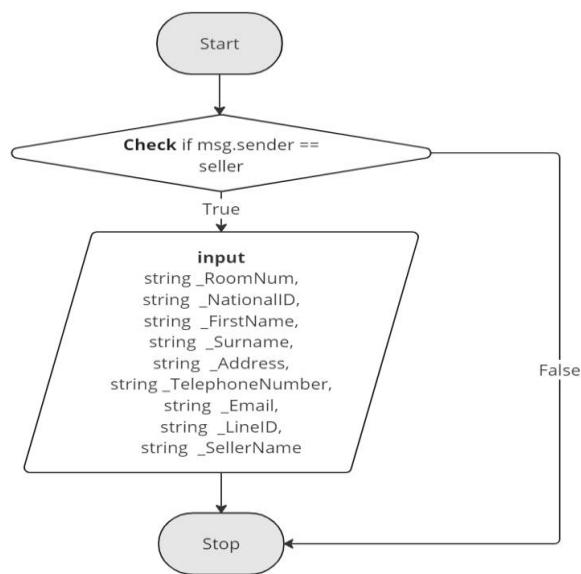
function StartPrice()

เพิ่มค่าราคาห้อง, ราคากลาง, ระยะเวลา



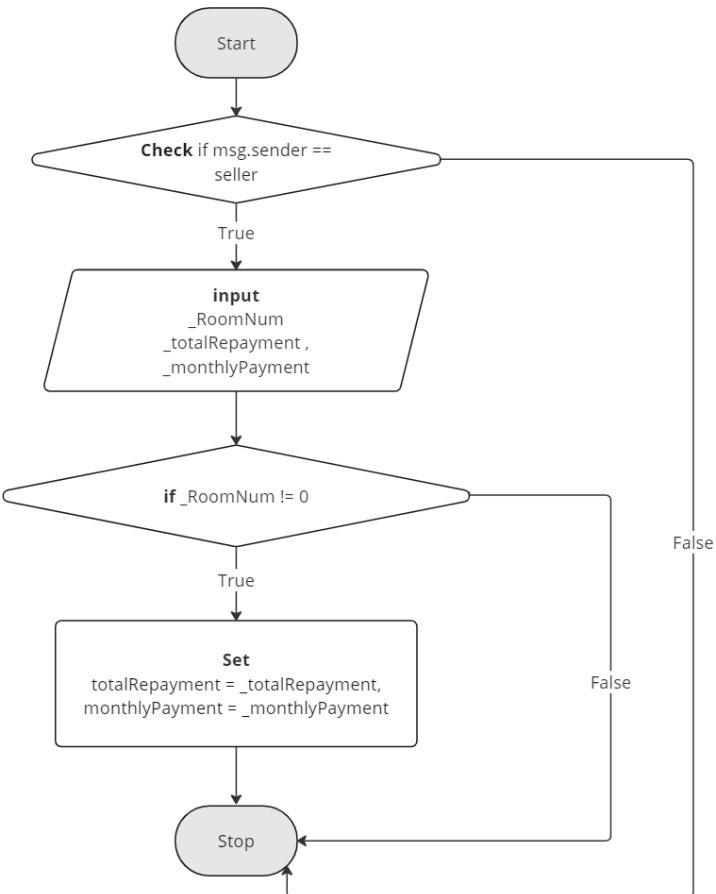
function createInfoUser()

สร้างข้อมูลผู้ซื้อ



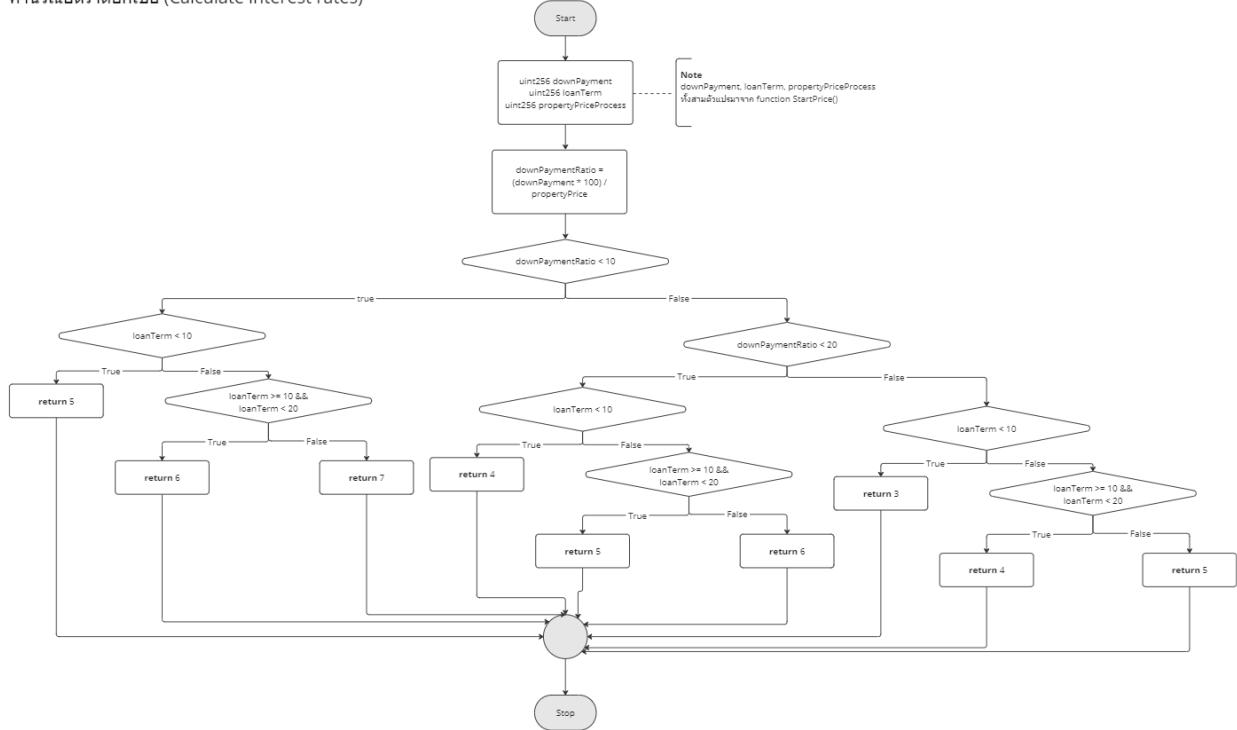
function addPrice()

เพิ่มราคาห้องทั้งหมดรวมดอกเบี้ย และ ราคางวดรายเดือน



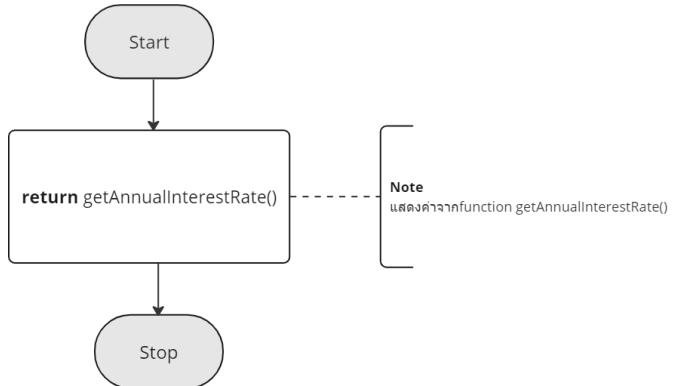
miro

function getAnnualInterestRate()
คำนวณอัตราดอกเบี้ย (Calculate interest rates)



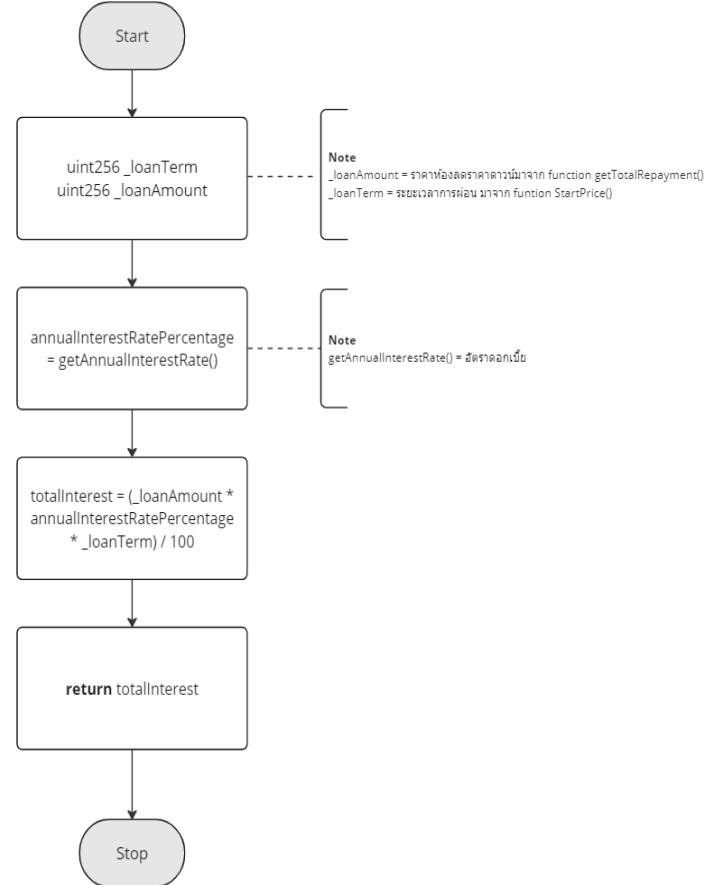
miro

function displayAnnualInterestRate()
แสดงอัตราดอกเบี้ย



miro

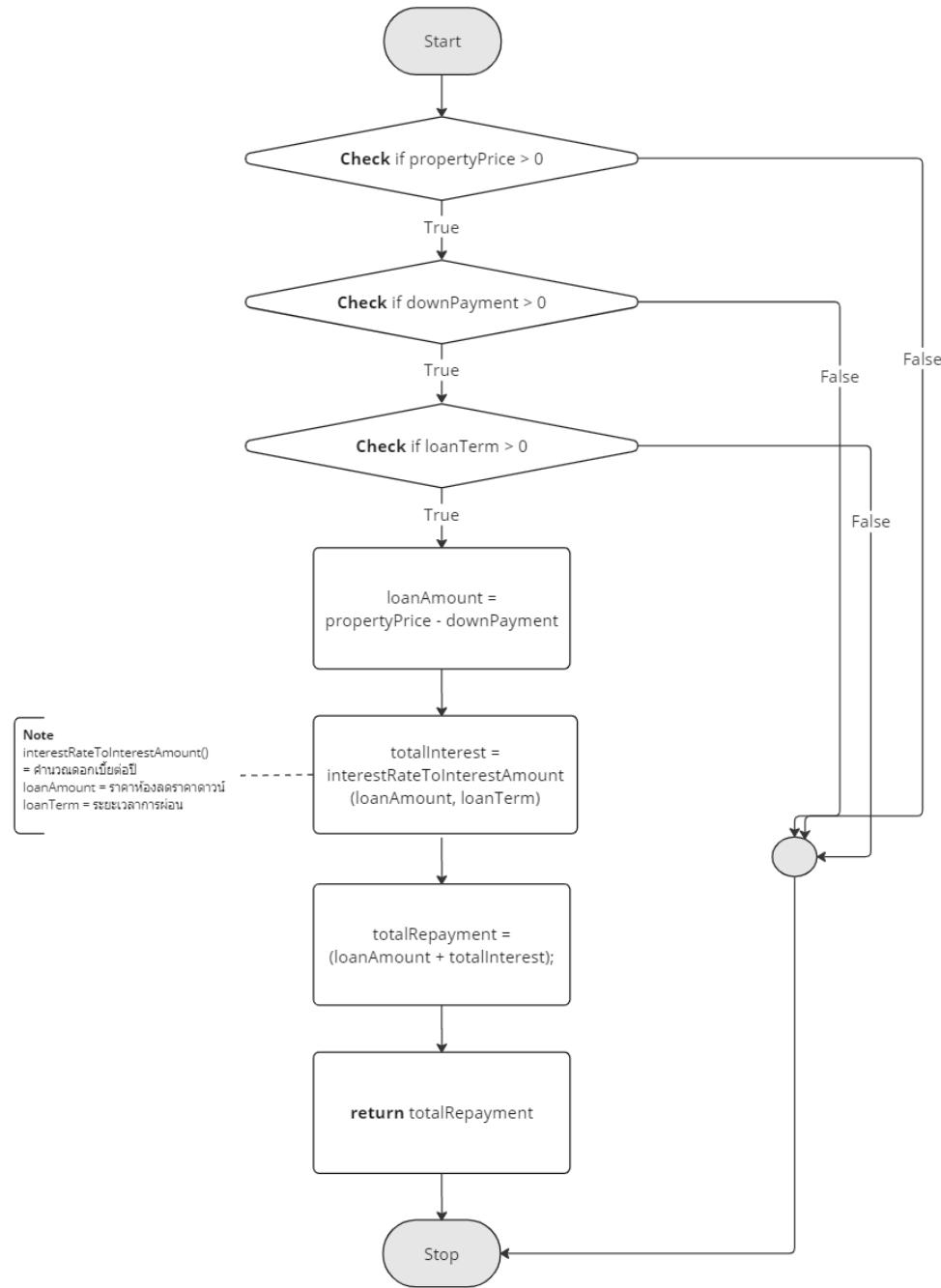
function interestRateToInterestAmount()
คำนวณดอกเบี้ยต่อปี (Calculate annual interest)



miro

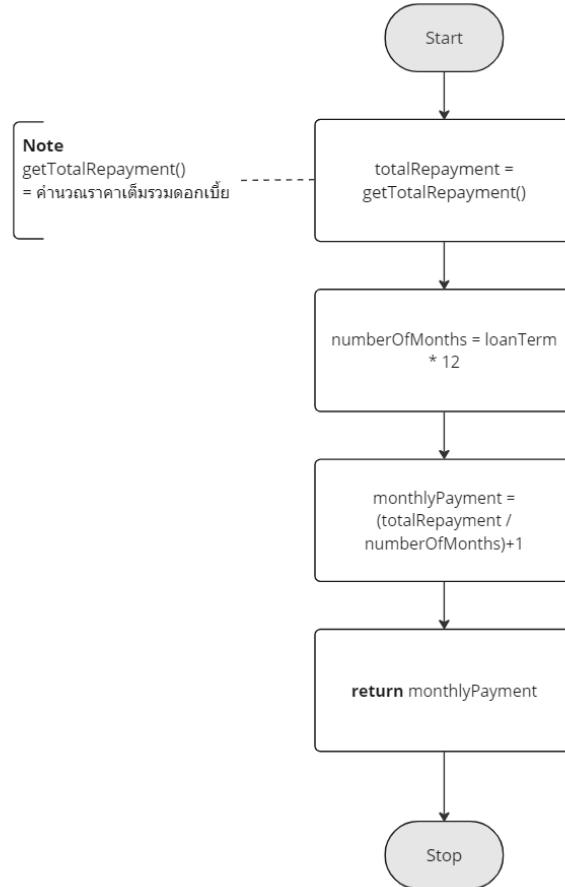
function getTotalRepayment()

ราคาเต็มรวมดอกเบี้ย

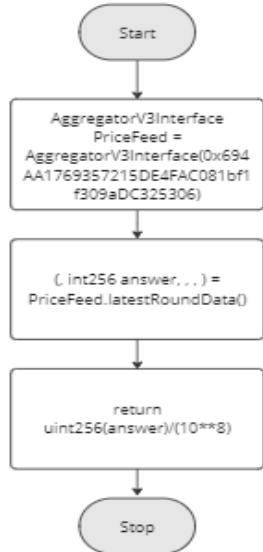


function calculateMonthlyPayment()

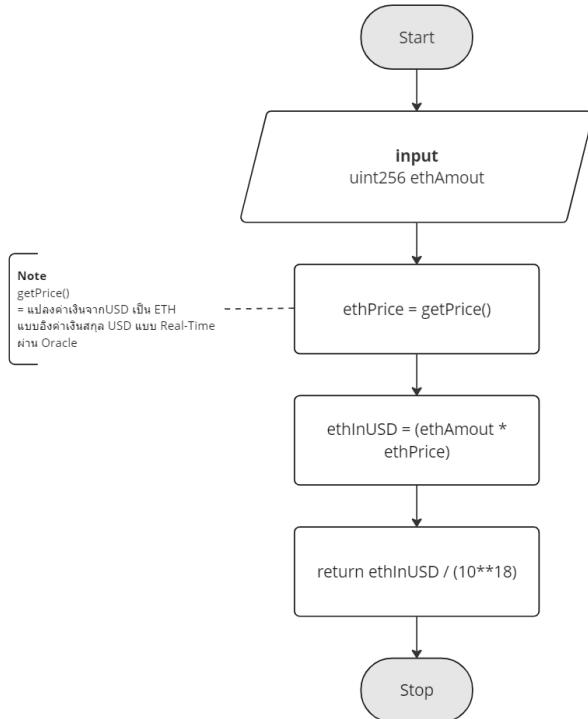
คำนวณค่างวดรายเดือนรวมดอกเบี้ย (Calculate monthly)



```
function getPrice()  
แปลงค่าเงิน USD กับ ETH ผ่าน Oracle
```

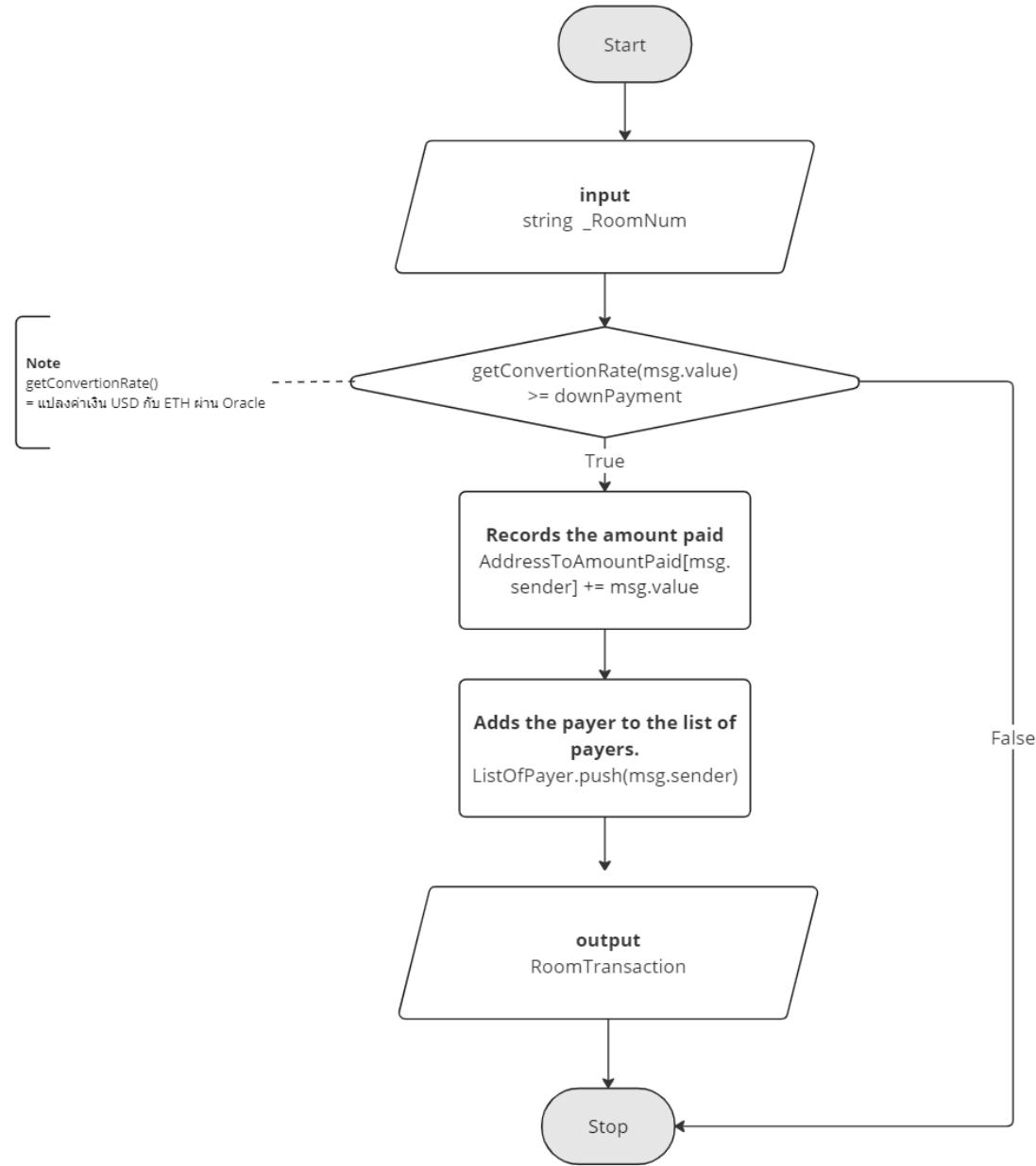


```
function getConversionRate()  
แปลงค่าเงิน USD กับ ETH ผ่าน Oracle
```



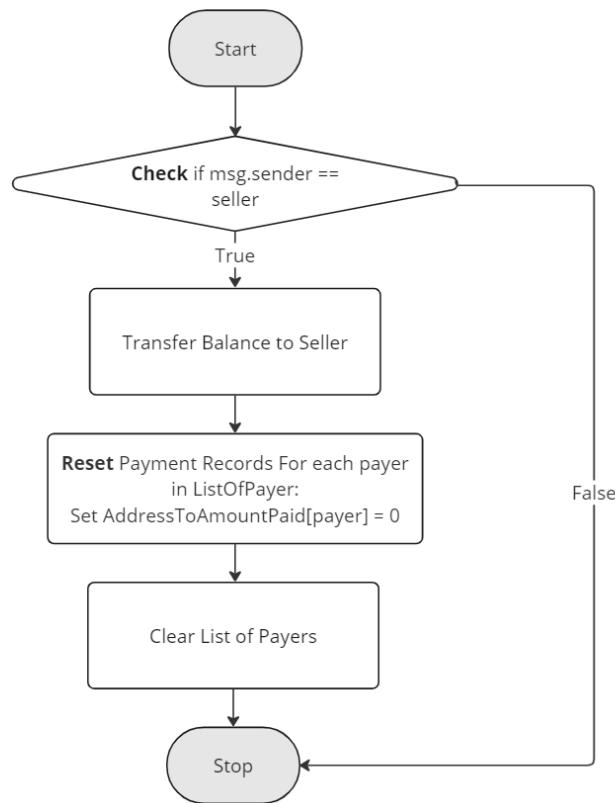
function makeDownPayment()

ชำระเงินค่าห้อง



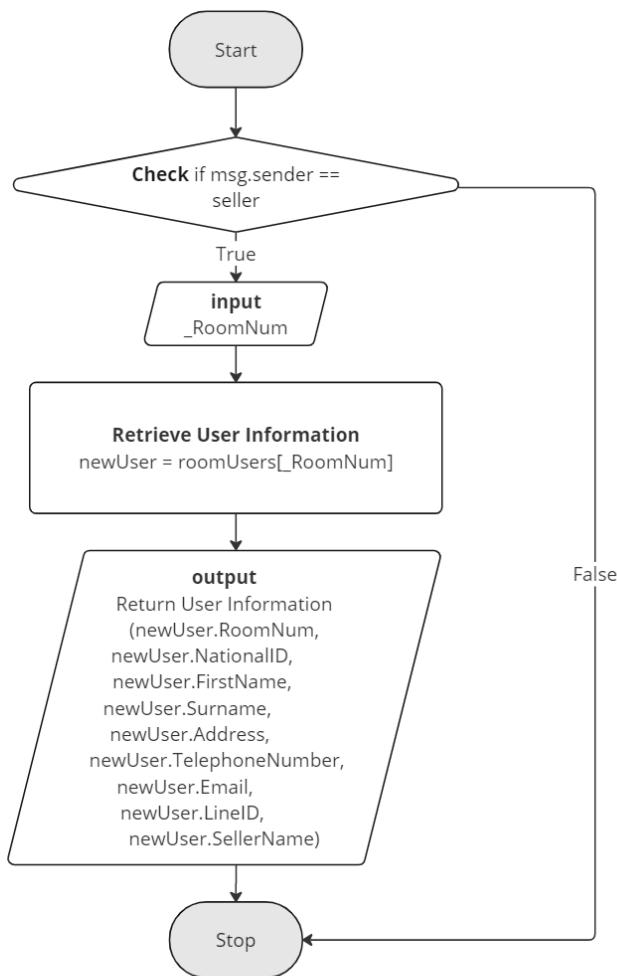
function WithDraw()

ถอนเงิน

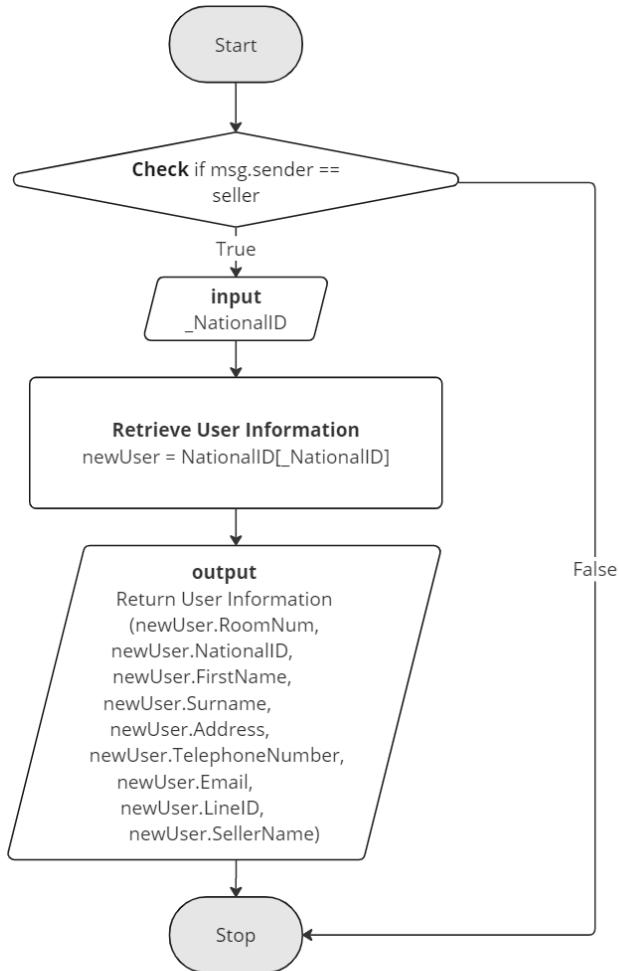


```
function searchByRoomNum()
```

ค้นหาข้อมูลผู้ซื้อโดยเลขที่ห้อง

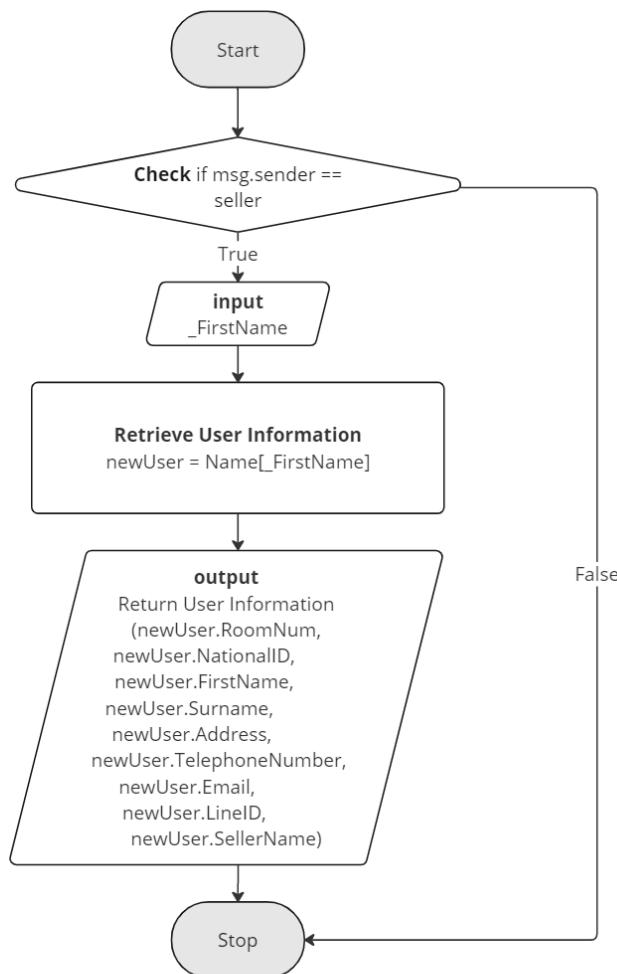


function searchByNationalID()
ค้นหาข้อมูลผู้ซื้อโดยเลขบัตรประชาชน



```
function searchByFirstName()
```

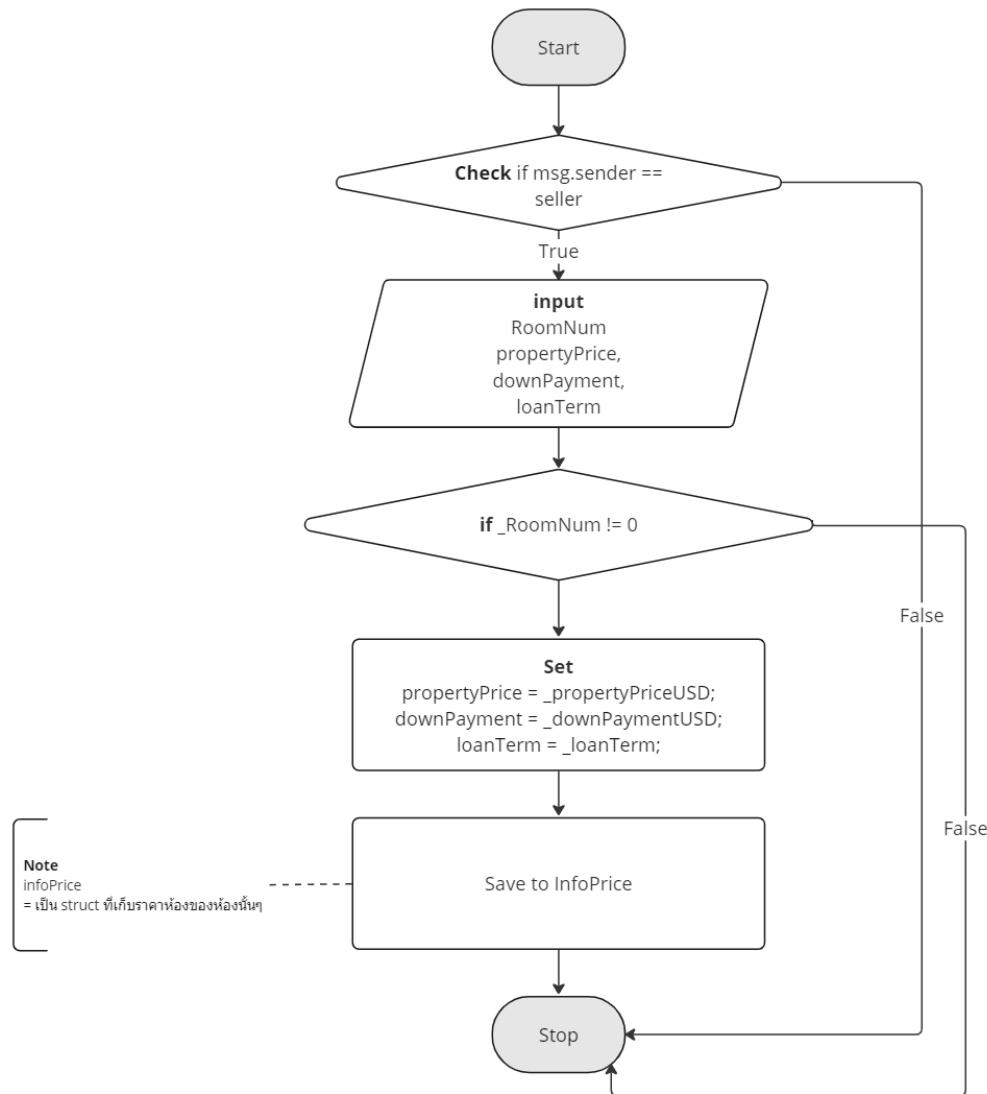
គណនាថាដំឡើងអ្នកដែលបានចូលរួម



Userinstallment.sol

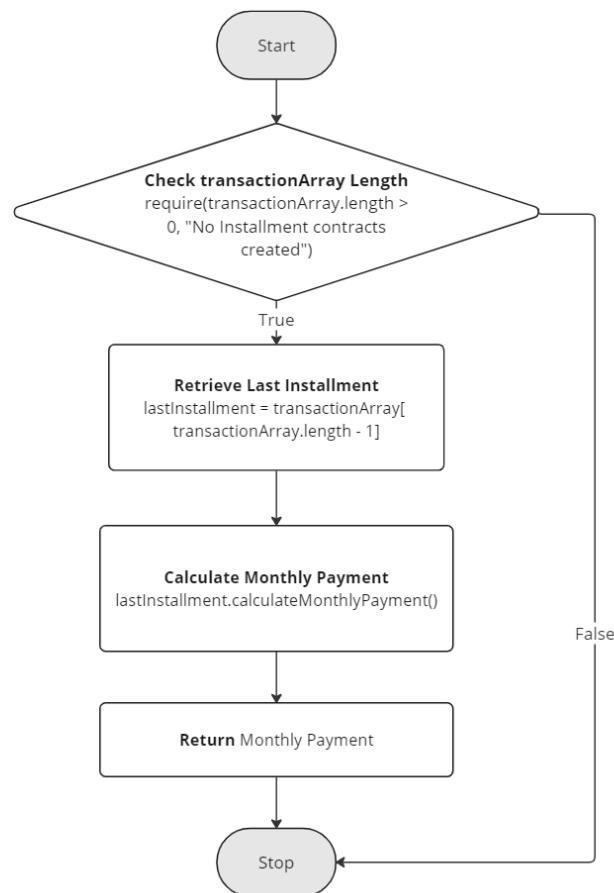
function StartPrice()

เพิ่มค่าราคาห้อง, ราคากลาง, ระยะเวลา



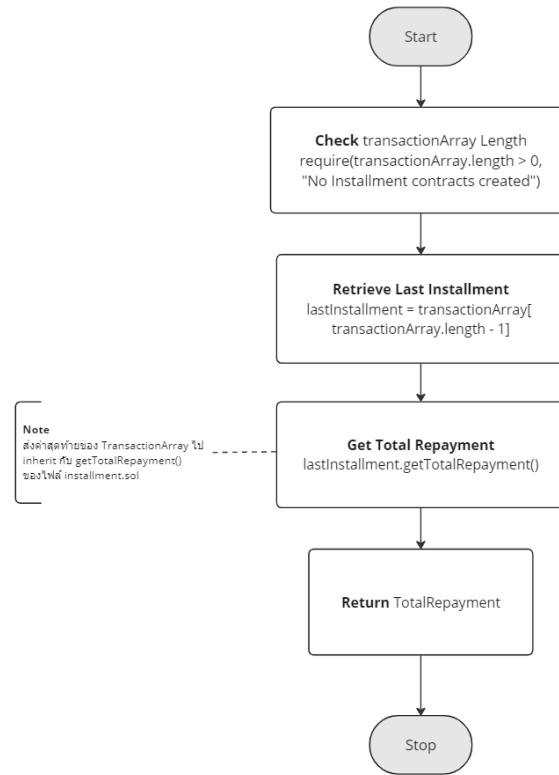
```
function calculateMonthlyPaymentExternal()
```

คำนวณเงินค่าห้องรายเดือน



miro

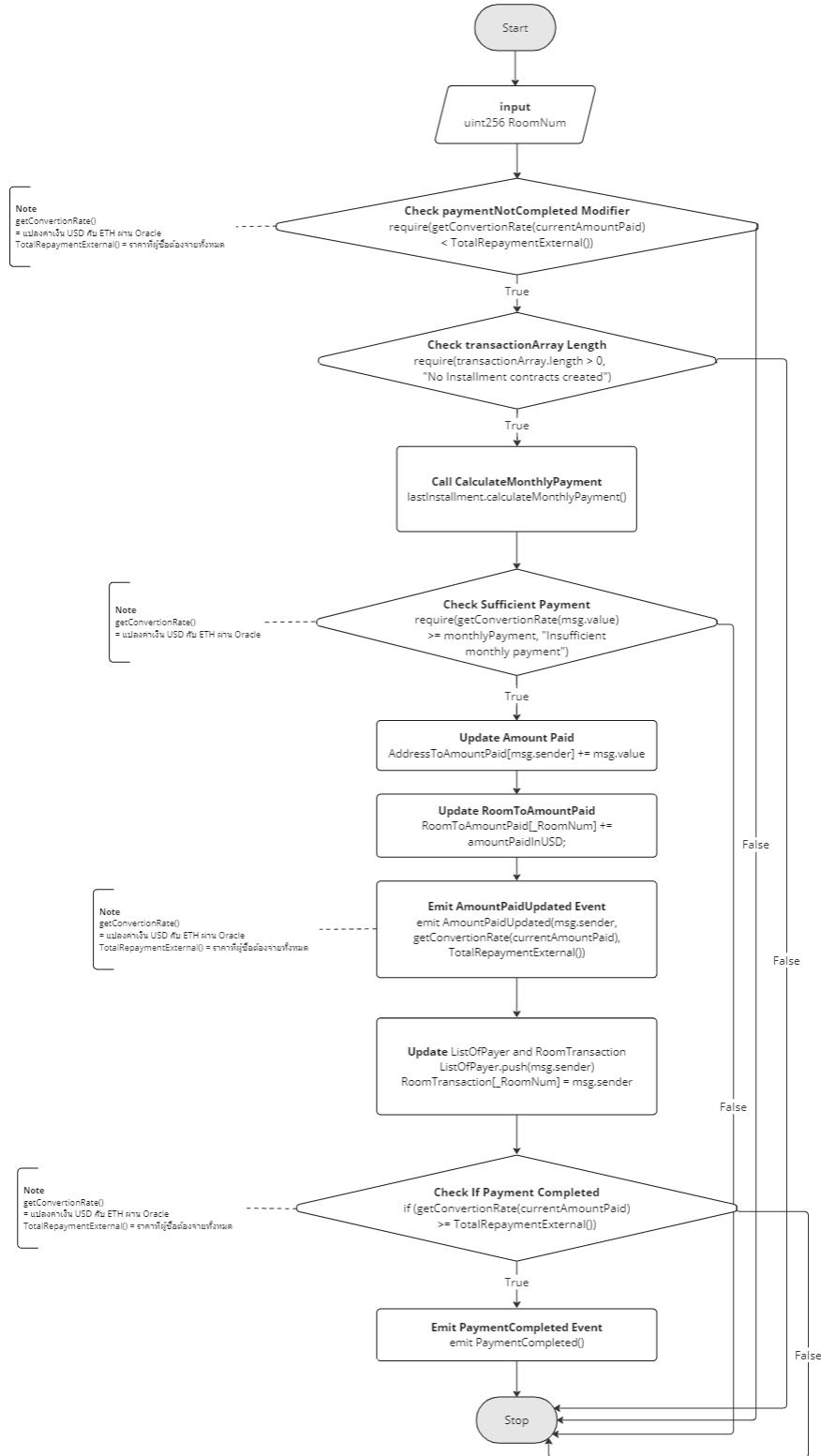
function TotalRepaymentExternal()
គ្រាប់រាយកម្មការចារេចគីន (TotalRepayment)



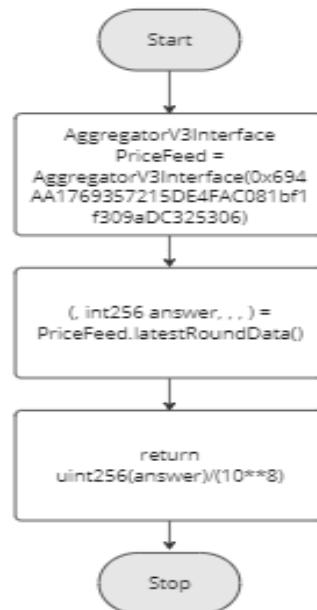
miro

function makeMonthlyPayment()

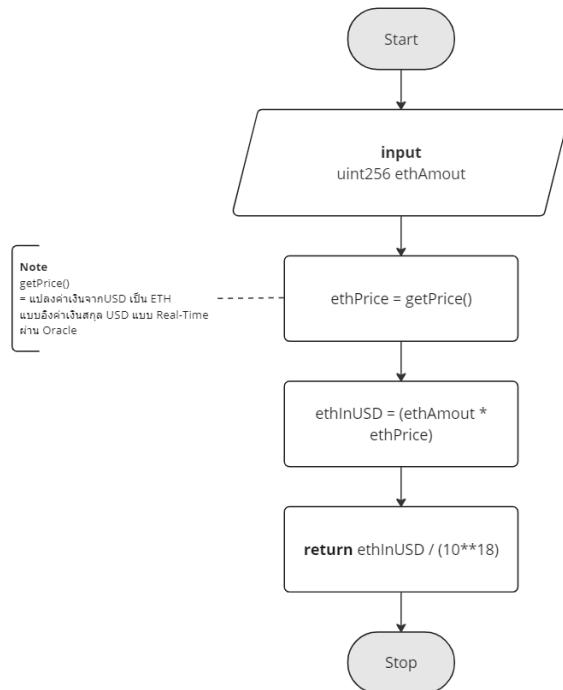
นำร่องคำท่องรายเดือน



```
function getPrice()  
แปลงค่าเงิน USD กับ ETH ผ่าน Oracle
```

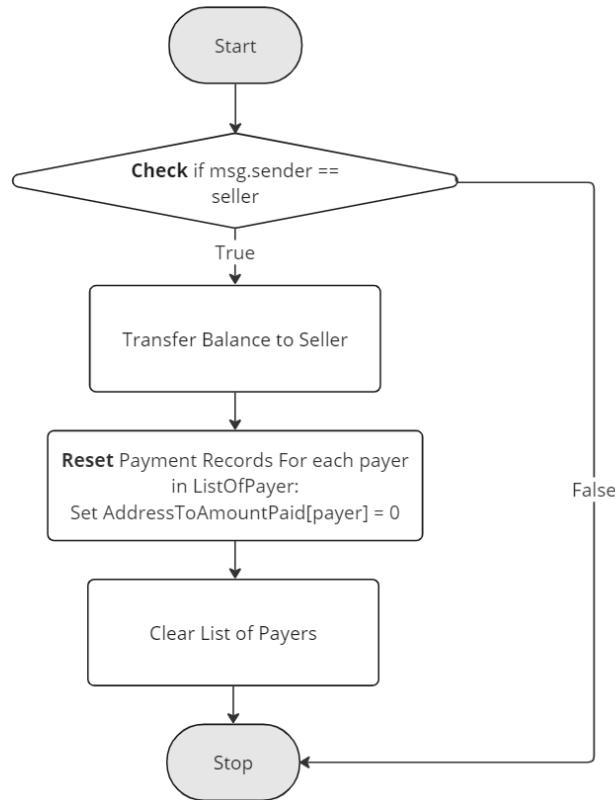


```
function getConversionRate()  
แปลงค่าเงิน USD กับ ETH ผ่าน Oracle
```



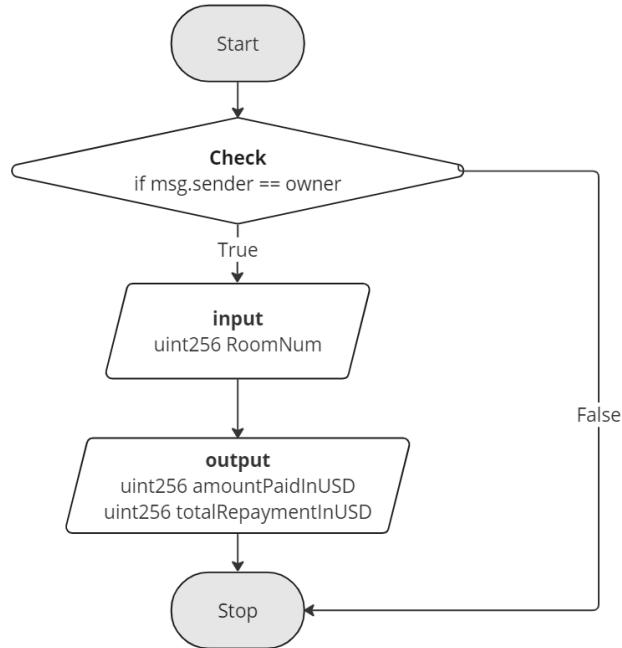
function WithDraw()

ถอนเงิน



```
function getRoomPaymentInfo()
```

แสดงข้อมูลการชำระเงินของห้องคอนโด



miro

การทดสอบ และ การประเมินระบบผ่อนชำระ

การควบคุมการเข้าถึง (Access control)

Installment.sol

ในสัญญา้มีข้อจำกัดในสิทธิ์ เพื่อความปลอดภัยของสัญญาและข้อมูลของผู้ใช้ โดยจะมีฟังก์ชันที่ผู้ใช้ หรือ ผู้ผ่อนคงจะไม่สามารถใช้ได้ มีฟังก์ชัน ดังนี้

- StartPrice()
- createInfoUser()
- searchByRoomNum()
- searchByNationalID()
- searchByFirstName()
- addPrice()
- WithDraw()

เมื่อได้ทดสอบให้ผู้ใช้ หรือ ผู้ผ่อนคงโด ใช้งานฟังก์ชันข้างต้น จะเป็นผลดังนี้

```
call to Installment.searchByNationalID
call to Installment.searchByNationalID errored: Error occurred: execution reverted: Only Seller.
execution reverted: Only Seller
```

```
167     return uint256(answer)/(10**18);
168     //1 ETH = 3670 USD
169 }
170
171 function getConversionRate(uint256 ethPrice) payable {
172     uint256 ethInUSD = (ethPrice * 3670);
173     return ethInUSD / (10**18);
174     // ETH > USD
175 }
176
177 function WithDraw() payable {
178     //transfer() transfer ด้วยการทำ transaction
179     require(msg.sender == seller);
180     address payable sender = payable(msg.sender);
181     sender.transfer(address(this).balance); ❷
182     for(uint256 PayerIndex = 0; PayerIndex < ListOfPayer.length; PayerIndex++){ //SetZero
```

ดังนั้น จากภาพหมายถึงสัญญาเป็นไปได้ถูกต้องในเรื่องของสิทธิ์เข้าถึง modifier()

UserInstallment.sol

สัญญาจะเป็นสัญญาเพื่อการจ่ายค่าผ่อนคงโดยรายเดือน จึงทำให้ยังคงต้องใช้ modifier() เพื่อควบคุม การเข้าถึงฟังก์ชันอยู่ซึ่งในสัญญานี้จะมีเพียงสองฟังก์ชันคือ function startPriceExternal() และ getRoomPaymentInfo() เพราะฟังก์ชัน startPriceExternal() จะเกี่ยวกับการกรอกราคาห้อง, เงินดาวน์, ระยะเวลาการผ่อน และ ฟังก์ชัน getRoomPaymentInfo() เกี่ยวกับการแสดงผลข้อมูลการชำระเงินรายเดือนของผู้ซื้อ โดยระบุแค่เลขห้อง ซึ่งเหตุผลนี้เองจึงทำให้ฟังก์ชัน startPriceExternal และ getRoomPaymentInfo() ต้องควบคุมสิทธิ์ให้เจ้าของท่านนั้น ซึ่งสัญญานี้ทำได้ถูกต้อง

```
[vm] from: 0xAb8...35cb2 to: UserInstallment.startPriceExternal(uint256,uint256,uint256) 0x609...384Fb value: 0 wei data: 0x5c8...00064 logs: 0 hash: 0x165...9a2a4
transact to UserInstallment.startPriceExternal errored: Error occurred: revert.

revert
The transaction has been reverted to the initial state.
Reason provided by the contract: "Only Owner".
You may want to cautiously increase the gas limit if the transaction went out of gas.
```

ตรวจสอบเงื่อนไข (Check conditions)

Installment.sol

ในสัญญาฉบับนี้มีการใช้ require() เพื่อเป็นการตรวจสอบเงื่อนไขก่อนที่จะเริ่มการทำงานของฟังก์ชัน โดยฟังก์ชันที่มีการใช้ require() มีดังนี้

- getAnnualInterestRate()

วางแผนน้อยกว่า 10%, ระยะเวลาผ่อนน้อยกว่า 10 ปี = อัตราดอกเบี้ย 5%

วางแผนน้อยกว่า 10%, ระยะเวลาผ่อนมากกว่าหรือเท่ากับ 10 ปี และน้อยกว่า 20 ปี = อัตราดอกเบี้ย 6%

วางแผนน้อยกว่า 10%, ระยะเวลามากกว่าหรือเท่ากับ 20 ปี = อัตราดอกเบี้ย 7%

วางแผนน้อยกว่า 20%, ระยะเวลาผ่อนน้อยกว่า 10 ปี = อัตราดอกเบี้ย 4%

วางแผนน้อยกว่า 20%, ระยะเวลาผ่อนมากกว่าหรือเท่ากับ 10 ปี และน้อยกว่า 20 ปี = อัตราดอกเบี้ย 5%

วางแผนน้อยกว่า 20%, ระยะเวลามากกว่าหรือเท่ากับ 20 ปี = อัตราดอกเบี้ย 6%

วางแผนมากกว่า 20%, ระยะเวลาผ่อนน้อยกว่า 10 ปี = อัตราดอกเบี้ย 3%

วางแผนมากกว่า 20%, ระยะเวลาผ่อนมากกว่าหรือเท่ากับ 10 ปี และน้อยกว่า 20 ปี = อัตราดอกเบี้ย 4%

วางแผนมากกว่า 20%, ระยะเวลามากกว่าหรือเท่ากับ 20 ปี = อัตราดอกเบี้ย 5%

- createInfoUser()

require(bytes(roomUsers[_RoomNum].RoomNum).length == 0, "There's already a buyer.");

= ตรวจสอบการซ้ำของห้อง ถ้ามีผู้ซื้ออู่แล้วจะแจ้งเตือนว่า "There's already a buyer."

- addPrice()

require(bytes(roomUsers[_RoomNum].RoomNum).length != 0, "Room not found");

= ตรวจสอบว่ามีห้องหรือไม่

- makeDownPayment()

require(getConversionRate(msg.value) >= downPayment, "Insufficient down payment");

= ตรวจสอบว่าเงินที่จะจ่ายเพียงพอต่อราคากำหนดรึเปล่า

- WithDraw()

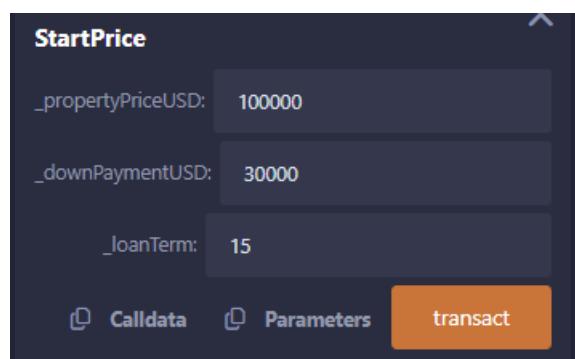
```
require(msg.sender == seller);
```

= ตรวจสอบว่าคนถอนเงินใช้เจ้าของสัญญาหรือไม่

ตัวอย่างการ Deploy เงื่อนไข Function getAnnualInterestRate()

```
if (downPaymentRatio < 10) {
    if (loanTerm < 10) {
        return 5;
    } else if (loanTerm >= 10 && loanTerm < 20) {
        return 6;
    } else {
        return 7;
    }
} else if (downPaymentRatio < 20) {
    if (loanTerm < 10) {
        return 4;
    } else if (loanTerm >= 10 && loanTerm < 20) {
        return 5;
    } else {
        return 6;
    }
} else {
    if (loanTerm < 10) {
        return 3;
    } else if (loanTerm >= 10 && loanTerm < 20) {
        return 4;
    } else {
        return 5;
    }
}
```

เงื่อนไขการคิดอัตราดอกเบี้ย



ตั้งค่าราคาห้อง, ราคางาน, ระยะเวลาการผ่อนที่ Function StartPrice()

เมื่อได้กำหนดค่าพารามิเตอร์ให้กับตัวแปรที่จะนำไปคำนวณแล้ว ต่อไปให้ตรวจสอบฟังก์ชัน getAnnualInterestRate (คำนวณอัตราดอกเบี้ย) โดยวิธีการคำนวณก่อนเข้าเงื่อนไขก็คือ ต้องคำนวณหาอัตราส่วนการชำระเงินดาวน์ก่อนมีสูตรดังนี้

```
uint256 downPaymentRatio = (downPayment * 100) / propertyPrice;
```

สูตรหาอัตราส่วนการชำระเงินดาวน์

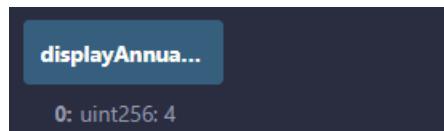
โดยความหมายของตัวแปรคือ

- downPaymentRatio = อัตราส่วนการชำระเงินดาวน์
- downpayment = เงินดาวน์ที่ผู้ซื้อได้จ่ายไป
- propertyPrice = ราคาห้อง

เมื่อคำนวณได้แล้วจะได้อัตราส่วนการชำระเงินดาวน์ 30% ดังนั้นเมื่อตูกาเงื่อนไขการคิดอัตราดอกเบี้ยแล้ว จะเข้าเงื่อนไข

```
    } else {
        if (loanTerm < 10) {
            return 3;
        } else if (loanTerm >= 10 && loanTerm < 20) {
            return 4;
        } else {
            return 5;
        }
    }
```

วงดาวน์มากกว่า 20%, ระยะเวลาผ่อนมากกว่าหรือเท่ากับ 10 ปี และน้อยกว่า 20 ปี



พิมพ์ชันแสดงอัตราดอกเบี้ย

ดังนั้น เงื่อนไขของพิมพ์ชันนี้ทำงานถูกต้องเพราแสดงอัตราดอกเบี้ยได้เป็น 4 หรือ 4%

ตัวอย่างการ Deploy เงื่อนไข Function createInfoUser()

ทดสอบการเพิ่มข้อมูลของผู้ซื้อไปที่ห้อง ครั้งที่ 1

createInfoUser

_RoomNum:	123
_NationalID:	๗๗๔
_FirstName:	ใจดี

ทดสอบการเพิ่มข้อมูลของผู้ซื้อไปที่ห้อง ครั้งที่ 2

createInfoUser

_RoomNum:	123
_NationalID:	สมชาย
_FirstName:	กอล์ฟ

จะเห็นว่าเรากำลังทดสอบให้ห้องมีการใช้ช้ากัน ถ้าเงื่อนไขทำงานได้ถูกต้อง ระบบจะไม่นำข้อมูลการทดสอบครั้งที่สองเข้าไปในฐานข้อมูล และจะขึ้นแจ้งเตือนเจ้าของว่า “There's already a buyer.” ดังภาพต่อไปนี้

```
[vm] from: 0xB3...eddC4 to: Installment.createInfoUser(string,string,string,string,string,string)
hash: 0x82e...85e79
transact to Installment.createInfoUser errored: Error occurred: revert.

revert
The transaction has been reverted to the initial state.
Reason provided by the contract: "There's already a buyer.".
You may want to cautiously increase the gas limit if the transaction went out of gas.
```

การแจ้งเตือนเมื่อมีการซื้อห้อง

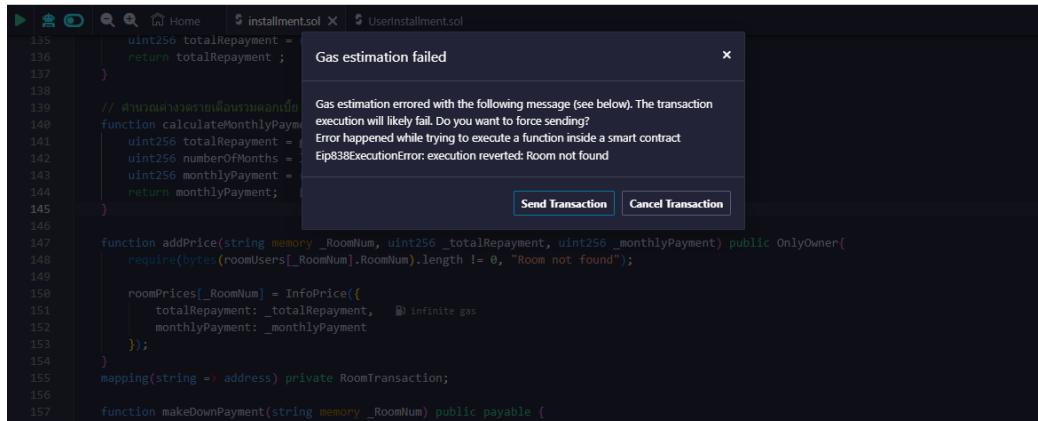
searchByRoo...

123	▼
0: string: 123	
1: string: ๗๗๔	
2: string: ใจดี	

ค้นหาห้องแล้วขึ้นชื่อผู้ซื้อที่ทดสอบครั้งแรก

ดังนั้น เมื่อมีข้อความ “There's already a buyer.” และระบบไม่ถูกบันทึกเข้าไป ก็จะหมายความว่าเงื่อนไขนี้ทำงานได้อย่างถูกต้อง

ตัวอย่างการ Deploy เงื่อนไข Function addPrice()



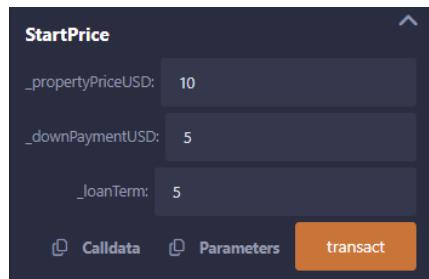
```
135     uint256 totalRepayment =  
136     return totalRepayment;  
137   }  
138  
139   // ฟังก์ชันเพิ่มจำนวนเดือนชำระ  
140   function calculateMonthlyPayment()  
141   uint256 totalRepayment =  
142   uint256 numberOfMonths =  
143   uint256 monthlyPayment =  
144   return monthlyPayment;  
145 }  
146  
147 function addPrice(string memory _RoomNum, uint256 _totalRepayment, uint256 _monthlyPayment) public OnlyOwner{  
148   require(bytes(roomUsers[_RoomNum]).length != 0, "Room not found");  
149  
150   roomPrices[_RoomNum] = InfoPrice({  
151     totalRepayment: _totalRepayment, infinite gas  
152     monthlyPayment: _monthlyPayment  
153   });  
154 }  
155 mapping(string => address) private RoomTransaction;  
156  
157 function makeDownPayment(string memory _RoomNum) public payable {
```

Function addPrice() กรณีที่หาห้องไม่เจอ

นี่คือการ Deploy แบบเงื่อนไขที่กรณีห้องไม่เจอหรือยังไม่ได้เพิ่มห้องจาก function createInfoUser() ดังนั้นเงื่อนไขนี้จึงถูก

ตัวอย่างการ Deploy เงื่อนไข Function makeDownPayment()

ในฟังก์ชันนี้มีเงื่อนไขการตรวจสอบว่าเงินที่จะจ่ายเพียงพอต่อราคากារ์ดหรือไม่ ดังรูป



StartPrice

_propertyPriceUSD: 10

_downPaymentUSD: 5

_loanTerm: 5

Calldata Parameters transact

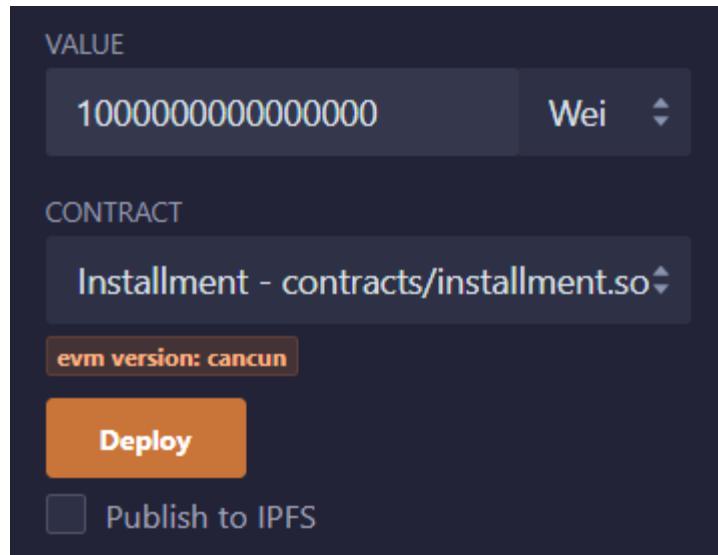
กำหนดราคา

และแน่นอนว่าราคากារ์ดใน transaction นี้คือ 5 USD ถ้าหากเราต้องการจะตรวจสอบเงื่อนไข เราจะเริ่มที่หากจ่ายเงินไม่ครบจะเกิดเช่นนี้

Wei	<input type="text" value="1000000000000000000"/>
Gwei	<input type="text" value="1000000"/>
Ether	<input type="text" value="0.001"/> ▾
Total Price (3800.79 \$ Per Ether)	\$ 3.80

แปลงค่าเงินจาก USD ไป Wei เพื่อนำมาจ่าย

ในรูปภาพคือเว็บไซต์ที่แปลงค่าเงิน โดยค่าเงินนี้เป็นของวันที่ 1 มิถุนายน 2567 เท่านั้น จากภาพจะเห็นว่าเราแลกเปลี่ยนเงินแค่ 3.80 USD หรือ 1000000000000000 Wei ซึ่งไม่พอจ่าย เราจะนำค่านี้มาจ่ายเพื่อทดสอบ



จ่าย 3.80 USD แต่ราคาดาวน์คือ 5 USD

```
7   uint256 downPayment;
8   mapping(address => uint256) downPayments;
9   mapping(address => uint256) totalPayments;
10  address payable[] pendingRefunds;
11  address payable[] pendingRefundPayers;
12  address payable[] pendingRefundPayers;
13  constructor() public {
14      downPayment = 1000000000000000000;
15  }
16  modifier onlyOwner() {
17      require(msg.sender == owner);
18      _;
19  }
20  }
21
22  struct InfoUser {
23      string RoomNum;
```

Gas estimation failed

Gas estimation errored with the following message (see below). The transaction execution will likely fail. Do you want to force sending?

Error happened while trying to execute a function inside a smart contract

Eip838ExecutionError: execution reverted: Insufficient down payment

Send Transaction **Cancel Transaction**

ข้อความแจ้งเตือนเมื่อเงินไม่ที่จ่ายไม่ถึงราคาการ์ด
ดังนั้นเงื่อนไขนี้ถูก และต่อไปจะแสดงถึงเมื่อเราจ่ายเงินตามกำหนดของราคาการ์ด

Wei	<input type="text" value="1400000000000000000"/>
Gwei	<input type="text" value="1400000"/>
Ether	<input type="text" value="0.0014"/>

Total Price \$ 5.32
(3800.79 \$ Per Ether)

แปลงค่าเงินจาก USD ไป Wei เพื่อนำมาจ่าย

เมื่อเรารายได้จำนวนถูกต้องต่อราคากำหนดไว้จะแสดงดังนี้



หน้า MetaMask ของผู้ซื้อ

จะสังเกตได้ว่ารายละเอียดในหน้า MetaMask นี้มีแสดงจำนวนสกุลเงิน USD ให้ แปลว่าเงื่อนไขนี้ทำงานได้อย่างถูกต้อง

ตัวอย่างการ Deploy เงื่อนไข Function WithDraw()

ในฟังก์ชันนี้มีความสำคัญเป็นอย่างมากและต้องการความปลอดภัยมากที่สุด
“เจ้าของ” เป็นคนที่ใช้ฟังก์ชันนี้เท่านั้น หากเป็นผู้ซื้อมาใช้จะแสดงข้อความดังนี้

โดยเราตั้งเงื่อนไขให้

```
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
```

```
function WithDraw() payable public OnlyOwner{ // infinite gas
    //transfer() tranfer ของจากที่นำไปที่นี่
    require(msg.sender == seller);
```

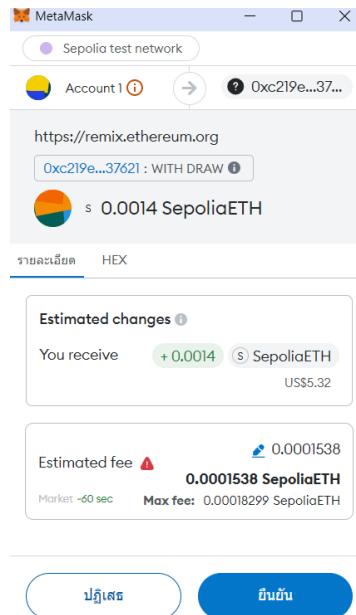
แจ้งเตือนของ Transaction จาก Alert

```
transact to Installment.WithDraw errored: Error occurred: execution reverted: Only Seller.
```

```
execution reverted: Only Seller
```

แจ้งเตือนของ Transaction จาก Console

ดังรูปข้างบน แปลว่าเงื่อนไขเป็นไปตามที่กำหนดไว้ แต่เมื่อถ้าเจ้าของเป็นคนถอนเงินจะขึ้น MetaMask และเงินจะเข้ากระเป๋าของเจ้าของตามรูป



กระเป่าเงินของเจ้าของ Smart Contract

UserInstallment.sol

ตรวจสอบการชำระเงินค่าผ่อนคงได้

โดยในฟังก์ชันชำระเงินค่าผ่อนคงได้ จะมีเงื่อนไขว่า หากผู้ผ่อนคงได้ จ่ายเงินครบจำนวน ตามราคาเต็มรวมดอกเบี้ยแล้ว ผู้ใช้จะไม่สามารถชำระเงินได้อีก และจะแจ้งเตือนผู้ผ่อนคงได้ว่า "You have paid in full."

ทดลองชำระเงินครั้งที่ 1

```
input          0xbf4...00000  ⓘ
decoded input  {
    "string _RoomNum": "123"
} ⓘ
decoded output - ⓘ
logs          [
    {
        "from": "0x933e4c77aefef88b87506d2b94a19b64e5338d088",
        "topic": "0x532822bb2152c7a6ac9c09884db26b9762abfb1cfb6bfcf8faf967dc20f61b6d",
        "event": "AmountPaidUpdated",
        "args": {
            "0": "0xcd2D3e7cE3773086C84bFEE412e22bBaCB2CD18C",
            "1": "3",
            "2": "5",
            "user": "0xcd2D3e7cE3773086C84bFEE412e22bBaCB2CD18C",
            "balance": "3",
            "totalRepayment": "5"
        }
    }
] ⓘ ⓘ
value         1000000000000000 wei ⓘ
```

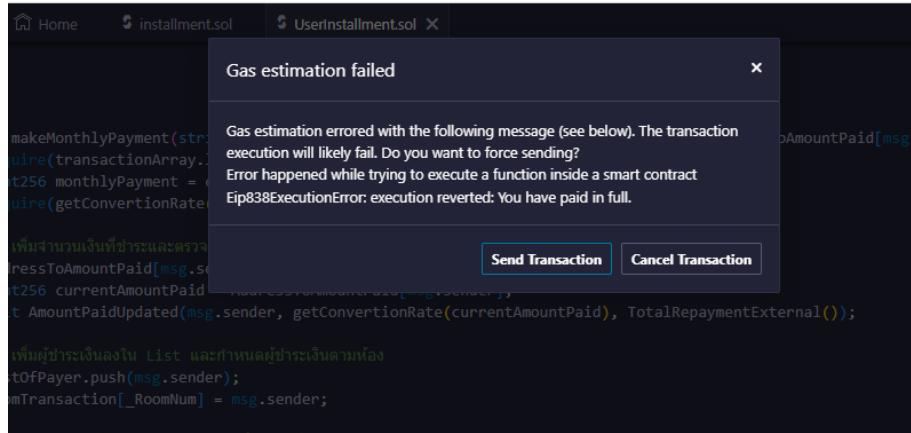
ชำระไป 3 USD จากราคาห้องรวมดอกเบี้ย 5 USD

ทดลองชำระเงินครั้งที่ 2

```
transaction cost      128633 gas ⓘ
input          0xbf4...00000  ⓘ
decoded input  {
    "string _RoomNum": "123"
} ⓘ
decoded output - ⓘ
logs          [
    {
        "from": "0x933e4c77aefef88b87506d2b94a19b64e5338d088",
        "topic": "0x532822bb2152c7a6ac9c09884db26b9762abfb1cfb6bfcf8faf967dc20f61b6d",
        "event": "AmountPaidUpdated",
        "args": {
            "0": "0xcd2D3e7cE3773086C84bFEE412e22bBaCB2CD18C",
            "1": "6",
            "2": "5",
            "user": "0xcd2D3e7cE3773086C84bFEE412e22bBaCB2CD18C",
            "balance": "6",
            "totalRepayment": "5"
        }
    },
    {
        "from": "0x933e4c77aefef88b87506d2b94a19b64e5338d088",
        "topic": "0x8ab0b3sea051fcf14722ab7d12aa288024affb7448031097d85723a4deae5Fc",
        "event": "PaymentCompleted",
        "args": {}
    }
] ⓘ ⓘ
value         700000000000000 wei ⓘ
```

ชำระไป 3 USD + คงเหลือ 3 USD = 6 USD จากราคาห้องรวมดอกเบี้ย 5 USD

ทดลองทำระเงินครั้งที่ 3



เมื่อชำระไปแล้วมากกว่าหรือเท่ากับราคากอนโดยรวมดอกเบี้ย จะไม่สามารถโอนได้และจะแจ้งเตือนผู้ผ่อนคอกโดยว่า "You have paid in full." ดังนั้น เงื่อนไขนี้ถูกต้อง

พังก์ชันการเช่าคอนโด

1. พังก์ชันบันทึกข้อมูลเช่าคอนโด

พังก์ชันนี้ใช้ในการรับเงินค่าเช่าและเงินประกันจากผู้เช่า และบันทึกข้อมูลผู้เช่าและเวลาเริ่มต้นการเช่า

2. พังก์ชันเช็คอิน

เมื่อถึงเวลาเช่า พังก์ชันนี้ใช้ในการตรวจสอบว่าผู้เช่าได้เข้าพักในคอนโดแล้วหรือยัง

3. พังก์ชันตรวจสอบสถานะ

พังก์ชันนี้ใช้ในการตรวจสอบสถานะปัจจุบันของการเช่า เช่น ว่าคอนโดถูกเช่าแล้วหรือยัง หรือว่าผู้เช่าได้เช็คอินแล้วหรือยัง

4. พังก์ชันการชำระค่าเช่าเดือนแรกและเงินประกัน

จะเป็นการจ่ายเงินค่าเช่าเดือนแรกบางกันเงินประกัน

5. พังก์ชันการชำระค่าเช่ารายเดือน

จะเป็นการจ่ายเงินเดือนต่อไปโดย ไม่ต้องจ่ายเงินประกันเพิ่ม

6. พังก์ชันการถอนเงิน

พังก์ชันนี้ใช้ในการถอนเงินทั้งหมดออกจากสัญญาโดยเจ้าของ

7. พังก์ชันแสดงยอดที่ต้องชำระทั้งหมด

เป็นการแสดงยอดเงินทั้งหมดที่ต้องจ่ายเพื่อให้ผู้เช่าเติมเงินมาพอตีกับค่าเช่าทั้งหมด

8. พังก์ชันคืนเงินประกันให้ผู้เช่า

เมื่อมีการเช่าสิ้นสุด ต้องมีพังก์ชันสำหรับการคืนเงินประกันให้แก่ผู้เช่า

9. พังก์ชันตรวจสอบความถูกต้องของการชำระเงิน

หากมีการใช้Walletในการชำระเงิน อาจต้องมีพังก์ชันเพื่อตรวจสอบความถูกต้องของการชำระเงินและบันทึกการทำธุกรรมในสมาร์ทคอนโดแทร็กต์

สิทธิ์การเข้าถึงระบบเช่า

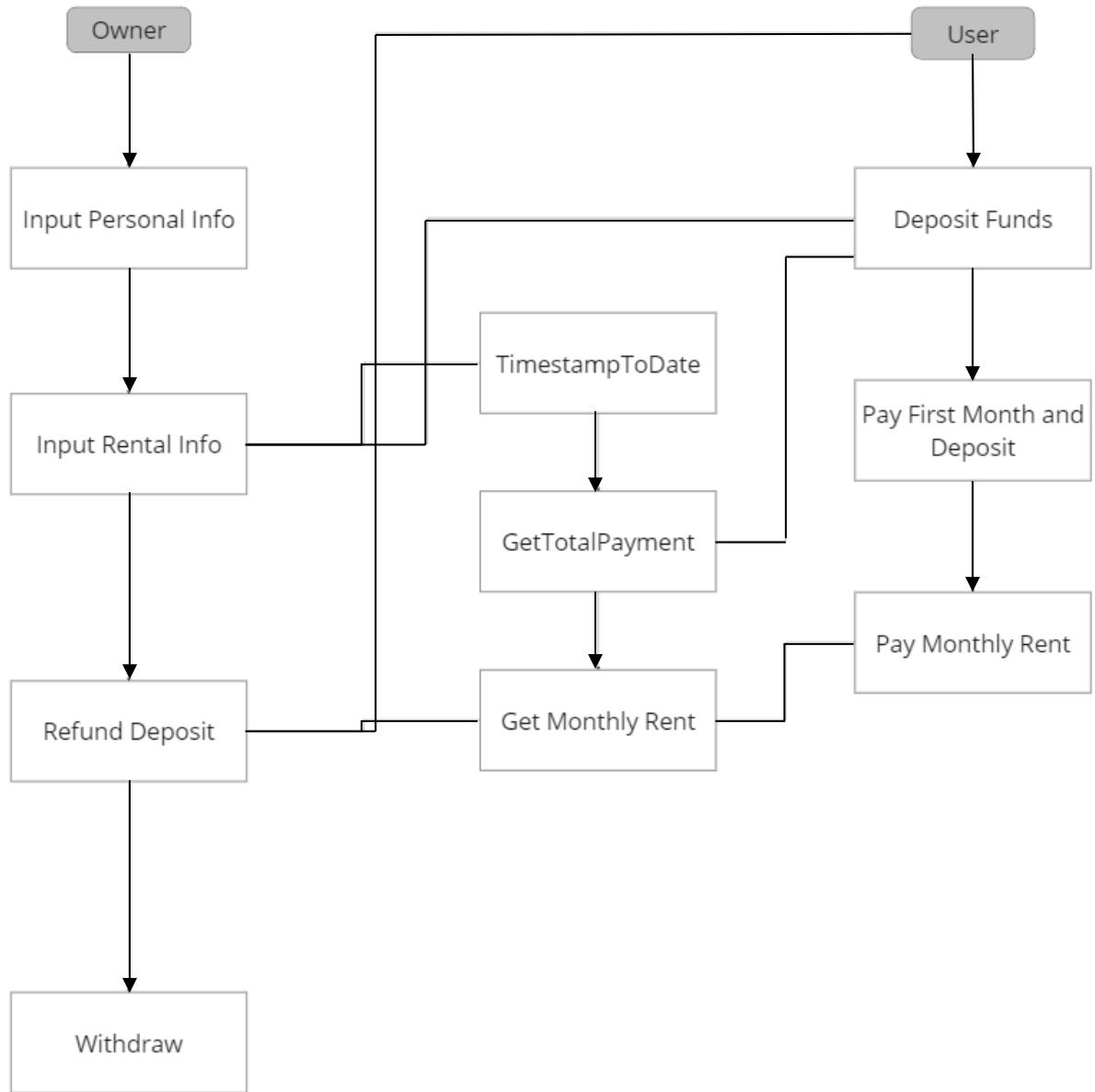
Rent.sol

Function	คำอธิบาย	เจ้าของ	ผู้ใช้/ผู้เช่า
depositFunds()	ฝากเงิน		
getMonthlyRent()	คืนค่าเช่าและจำนวนเดือนที่เหลือ		
payMonthlyRent()	ชำระค่าเช่าเดือนถัดไป		
inputPersonalInfo()	บันทึกข้อมูลส่วนตัวของผู้เช่า		
inputRentalInfo()	บันทึกข้อมูลการเช่าห้อง		
payFirstMonthAndDeposit()	ชำระค่าเช่าเดือนแรกและเงินมัดจำ		
refundDeposit()	คืนเงินประกันแก่ผู้เช่า		
WithDraw()	ถอนเงิน		
Uint2str(uint_i)	แปลง uint เป็น string		
timestampToDate()	แปลง timestamp เป็นรูปแบบ วัน-เดือน-ปี		
getTotalPayment()	แสดงจำนวนเงินที่ต้องชำระทั้งหมด		

สถาปัตยกรรมของระบบเช่า

→ เส้นที่แทนฟังก์ชันที่เก็บค่า(Transact)

— เส้นที่แทนฟังก์ชันที่อ่านค่า(Call)

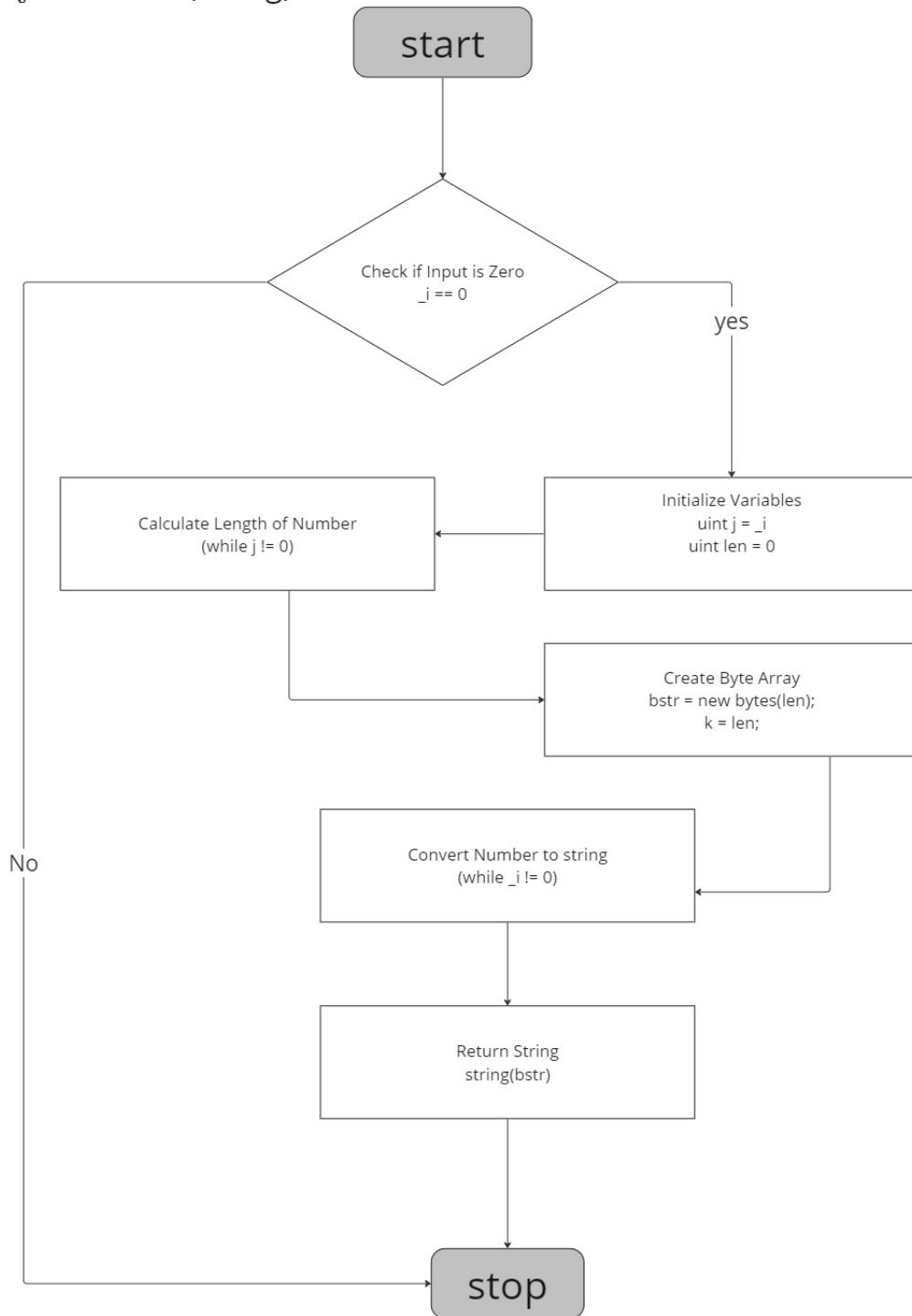


การออกแบบขั้นตอนการทำงานของระบบเช่า

function uint2str(uint _i)

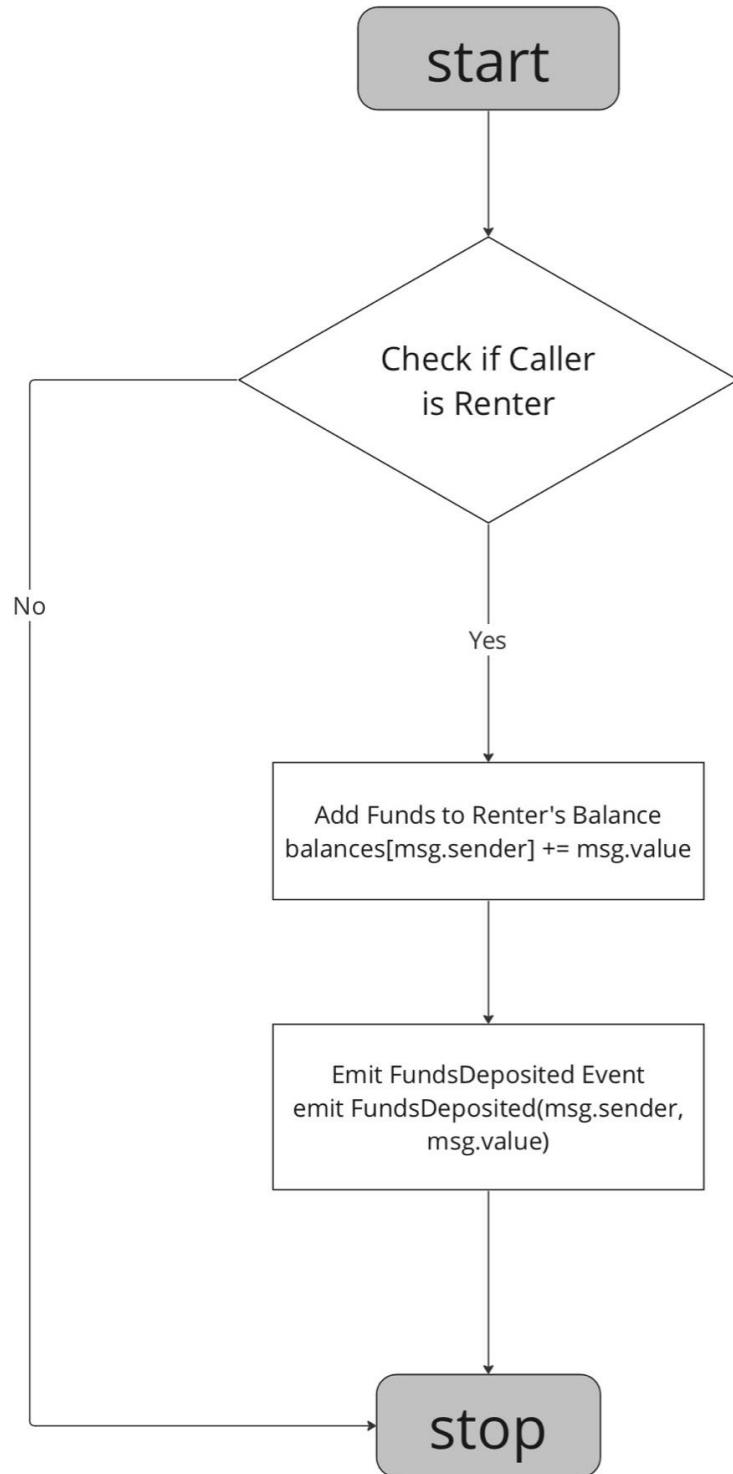
แปลงจำนวนเต็มบวก (uint)

เป็นรูปแบบสตริง (string)

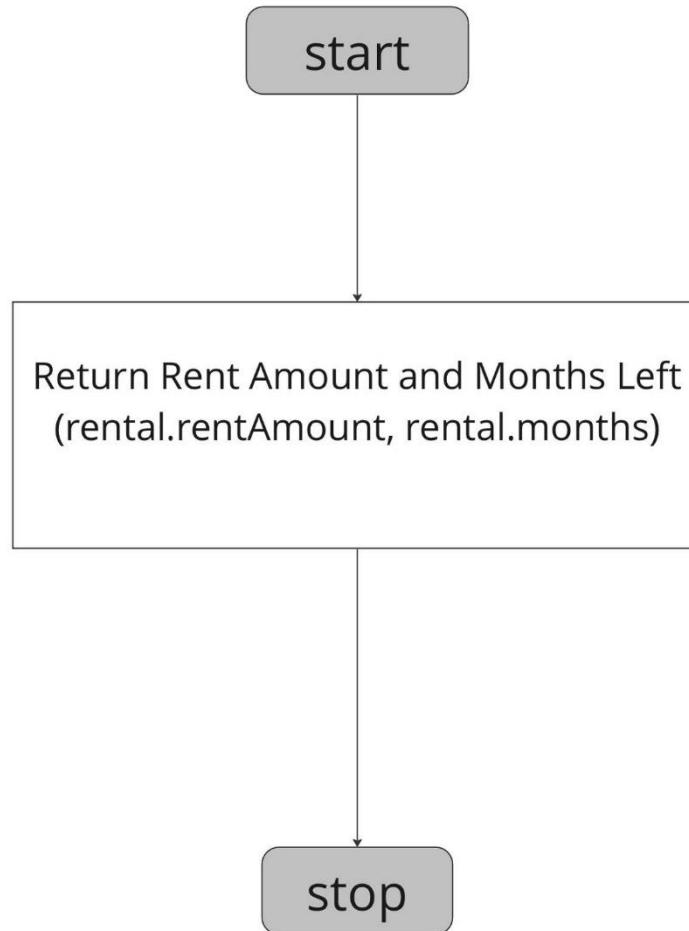


function depositFunds

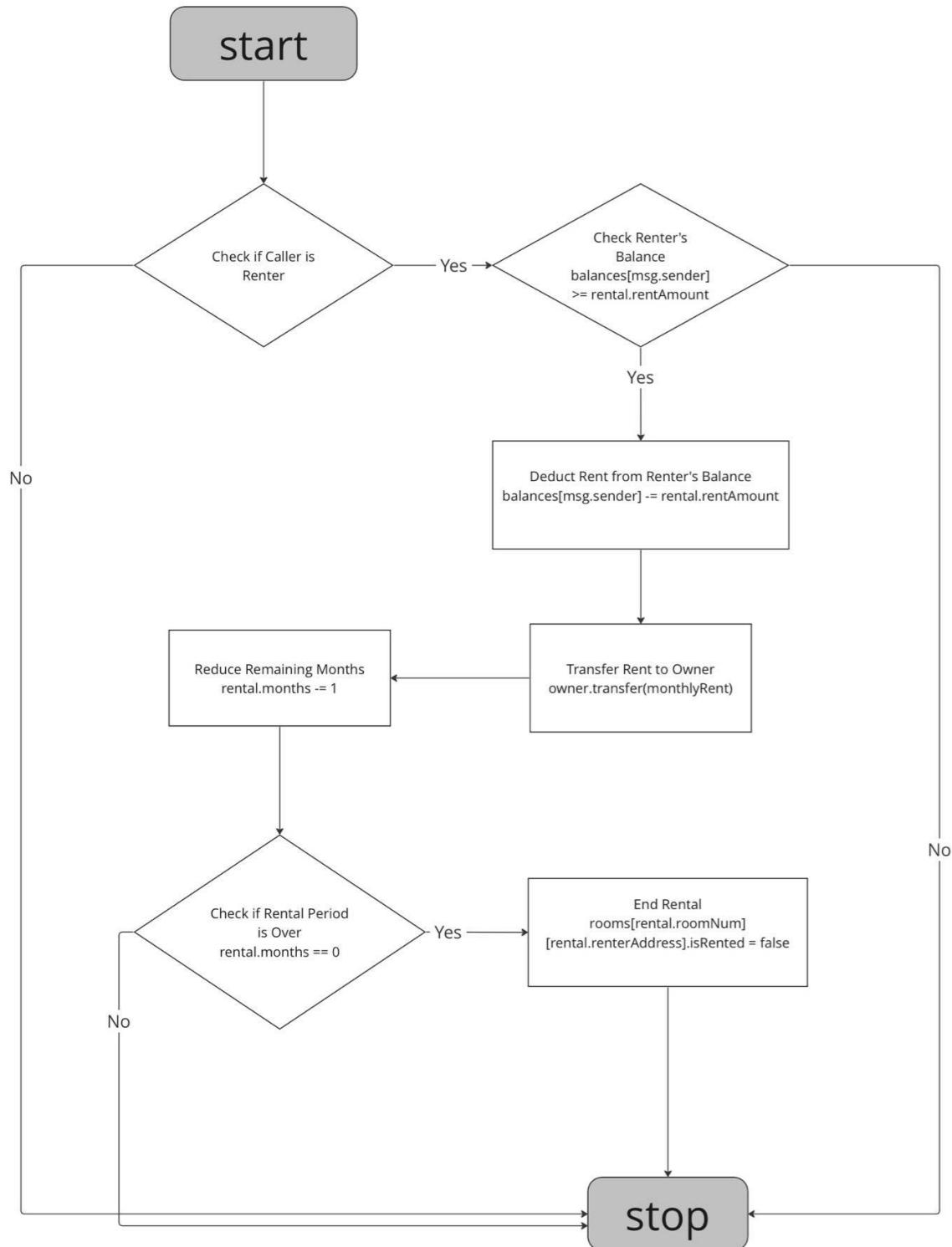
การฝากเงิน



function getMonthlyRent
การคืนค่าค่าเช่าเดือนและจำนวน
เดือนที่เหลือ

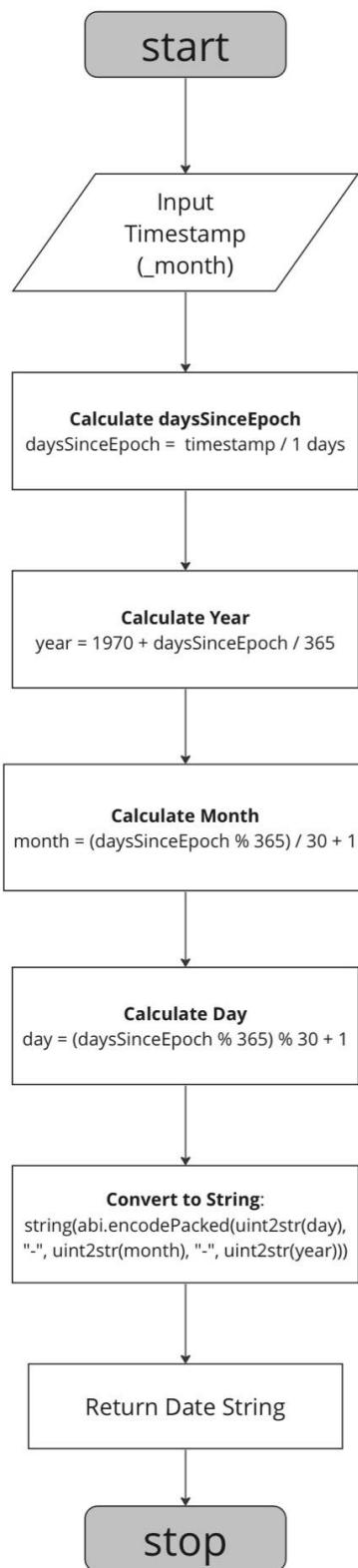


function payMonthlyRent
ชำระค่าเช่าเดือนต่อไป

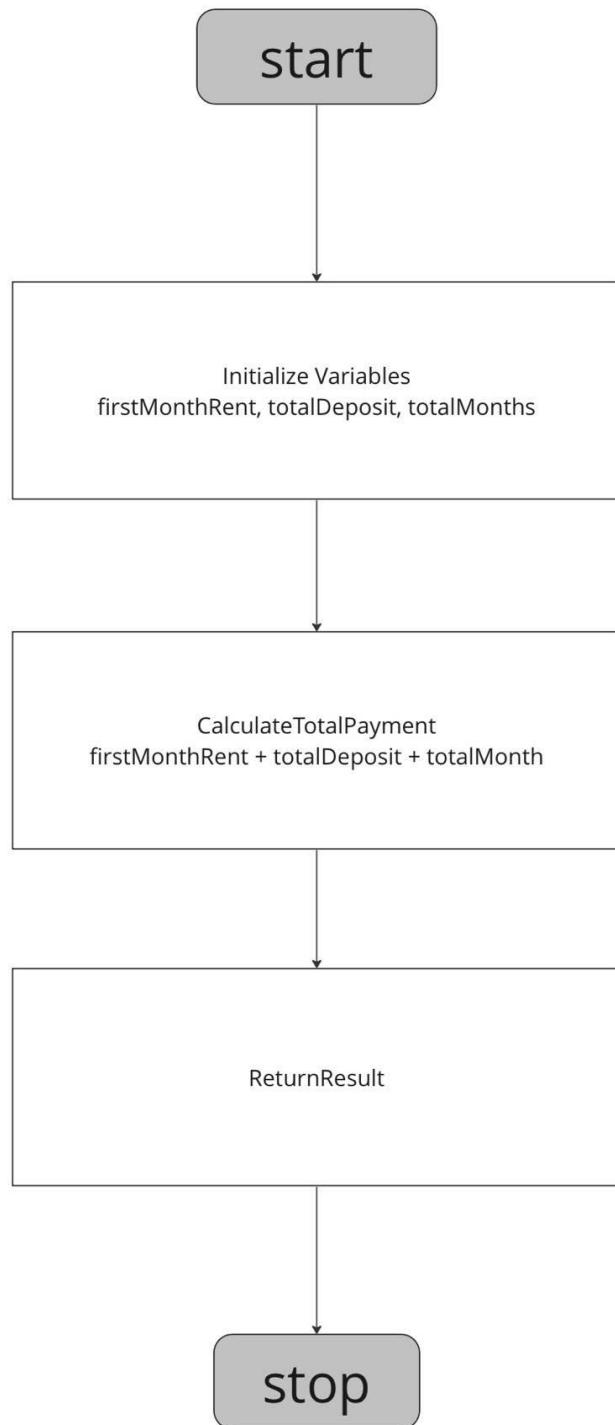


```
function timestampToDate
```

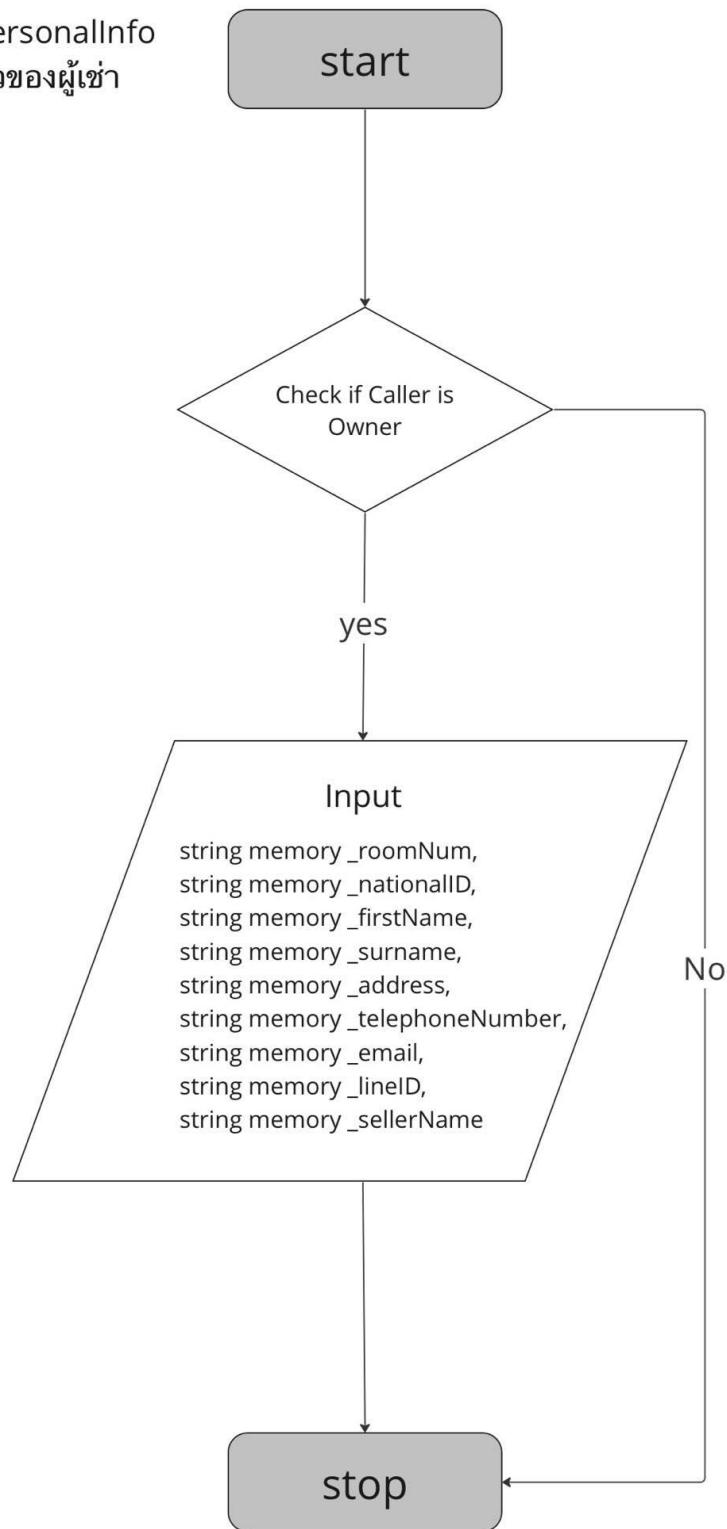
แปลง timestamp
ให้เป็นรูปแบบวันที่
(วัน-เดือน-ปี)



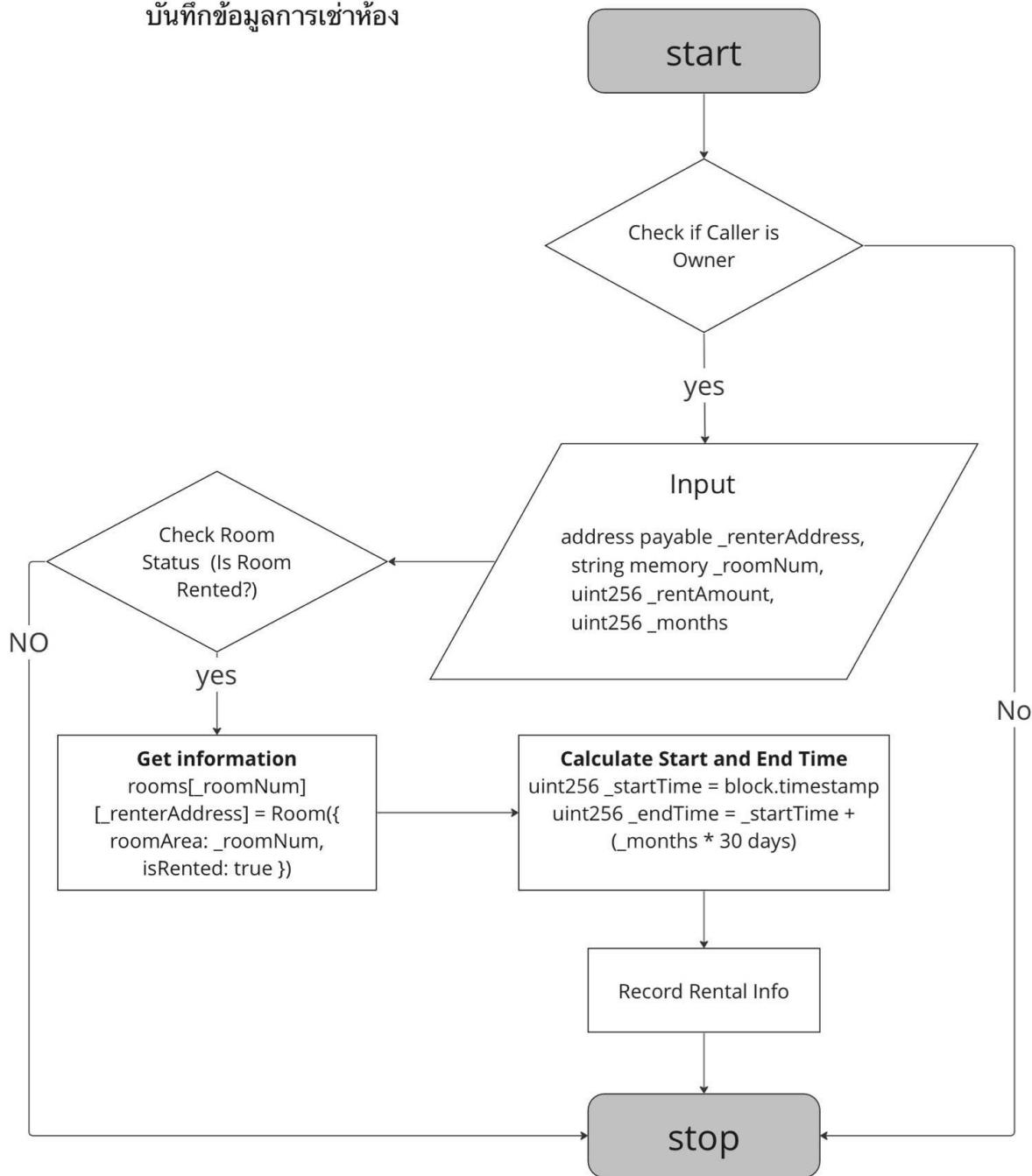
function getTotalPayment
แสดงค่าจำนวนเงินที่ต้องชำระทั้งหมด



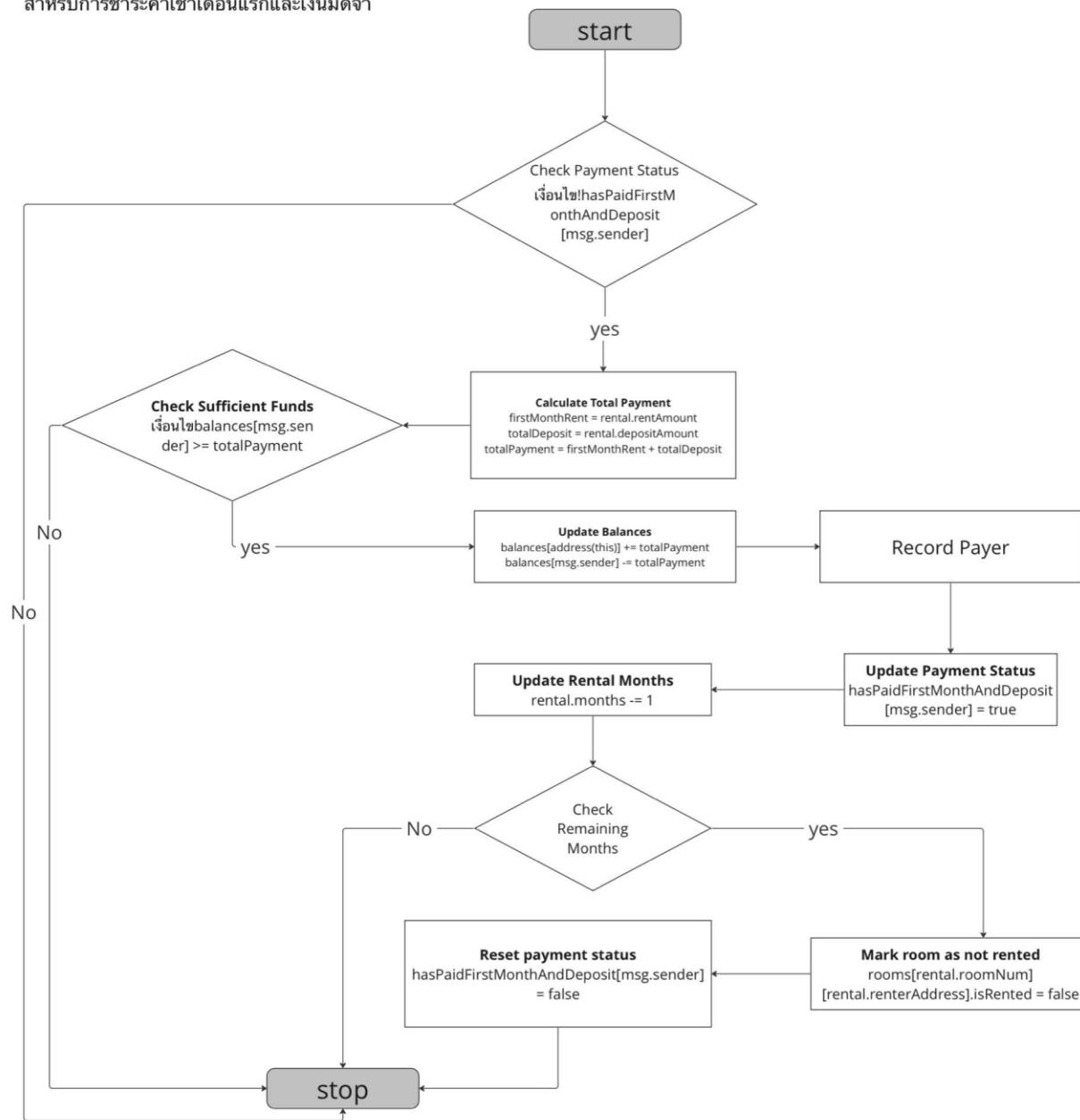
function inputPersonalInfo
บันทึกข้อมูลส่วนตัวของผู้เช่า



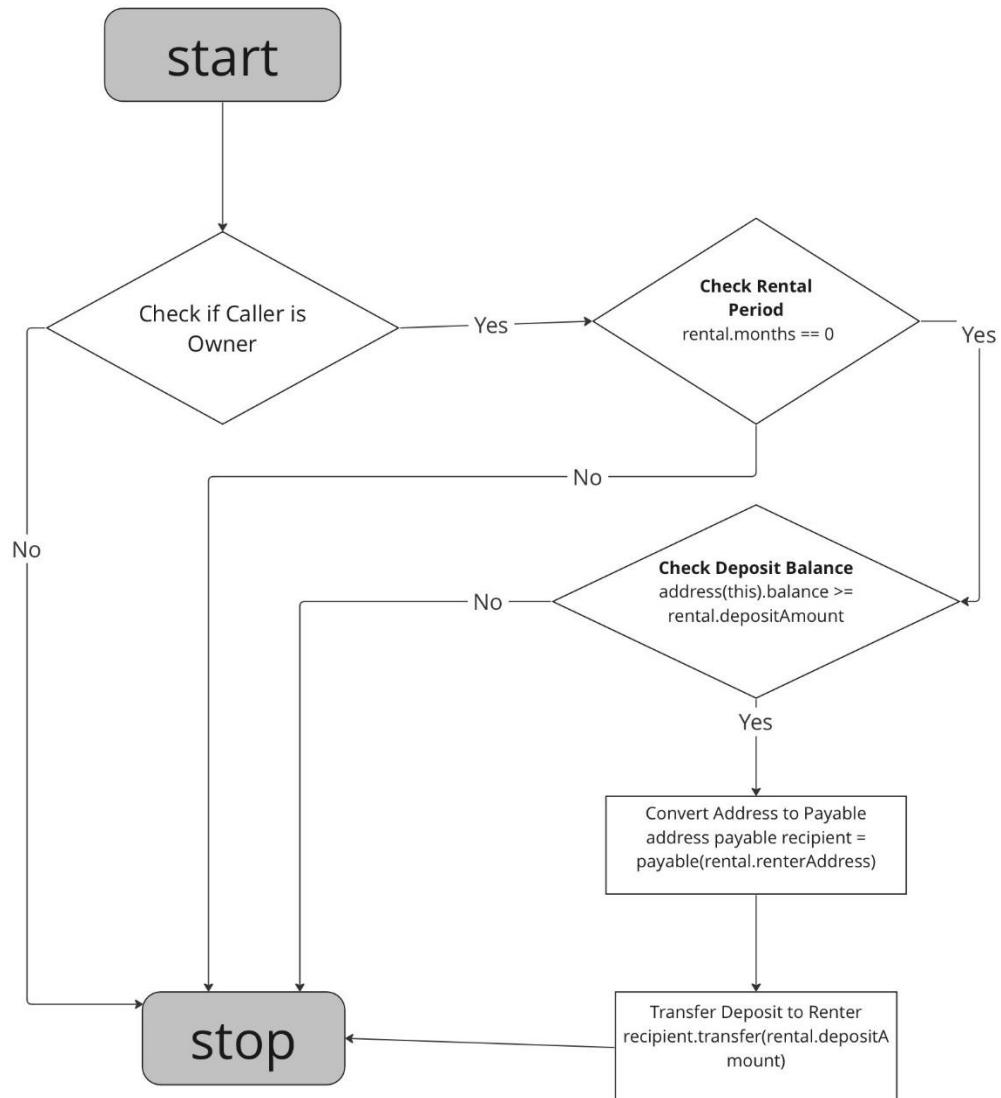
function inputRentalInfo
บันทึกข้อมูลการเช่าห้อง



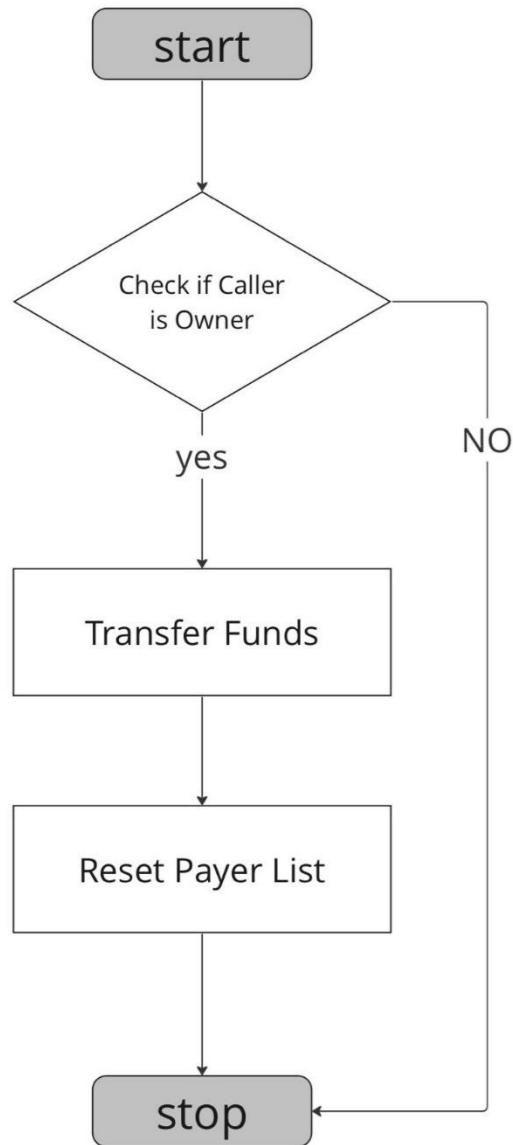
function payFirstMonthAndDeposit
สำหรับการชำระค่าเช่าเดือนแรกและเงินมัดจำ



function refundDeposit
ฟังก์ชันคืนเงินประกันให้ผู้เช่า



function Withdraw
การถอนเงิน โดยเจ้าของ



การทดสอบของระบบเช่า

Rent.sol

ในสัญญาี้จะมีการเข็คในการใช้สิทธิ์ เพื่อความปลอดภัยของสัญญาและข้อมูลของเจ้าของคอนโด โดยจะมีฟังก์ชันที่เจ้าของคอนโดใช้ได้ ผู้เช่าคอนโดจะไม่สามารถใช้ได้ มีฟังก์ชัน ดังนี้

-inputPersonalInfo()

-inputRentalInfo ()

-refundDeposit ()

-Withdraw()

ผลของการ Deploy เมื่อผู้เช่าใช้ฟังก์ชันของเจ้าของ

```
string memory _sellerName
) public onlyOwner {
renter = Renter( // บันทึกข้อมูลลูกค้าของผู้เช่าลงใน struct Renter
roomNum: _roomNum,
nationalID: _nationalID,
firstName: _firstName,
surname: _surname,
```

0 Listen on all transactions Filter with transaction hash or address

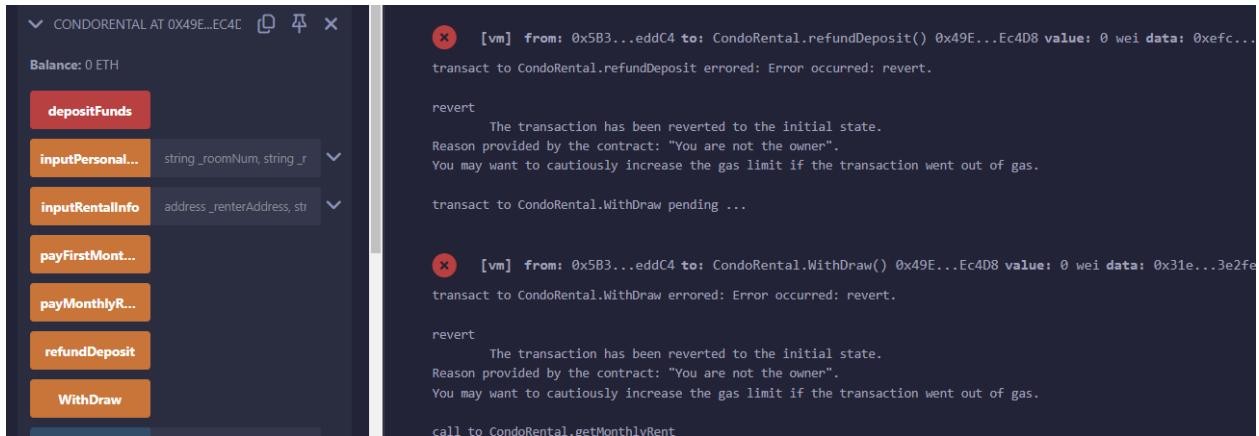
[vm] from: 0xAB8...35cb2 to: CondoRental.(constructor) value: 0 wei data: 0x608...80033 logs: 0 hash: 0x451...32fc

[vm] from: 0x5B3...eddc4 to: CondoRental.inputPersonalInfo(string,string,string,string,string,string,string,string) 0x49E...Ec4D8 value: 0 wei data: 0xa89...00000 logs: 0 hash: 0xd3d3...19d74

revert The transaction has been reverted to the initial state.
Reason provided by the contract: "You are not the owner".
You may want to cautiously increase the gas limit if the transaction went out of gas.

[vm] from: 0x5B3...eddc4 to: CondoRental.inputRentalInfo(address,uint256,uint256) 0x49E...Ec4D8 value: 0 wei data: 0x3a3...00000 logs: 0 hash: 0x8a8...456f1

revert The transaction has been reverted to the initial state.
Reason provided by the contract: "You are not the owner".
You may want to cautiously increase the gas limit if the transaction went out of gas.



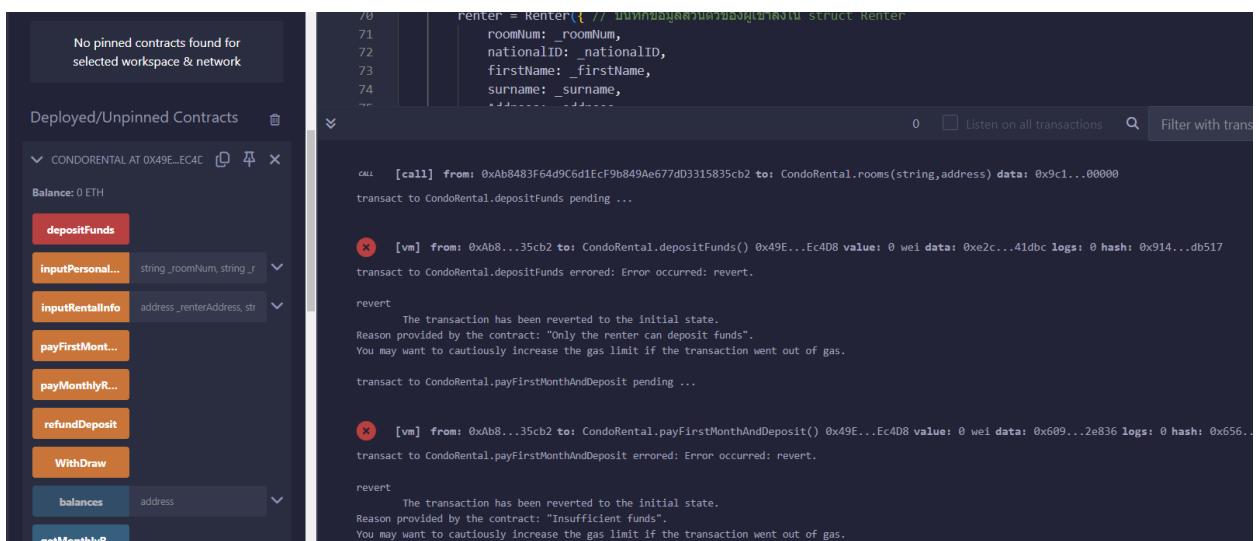
จากรูปภาพจะเห็นได้ว่าถ้าไม่ใช่เจ้าของไม่สามารถบันทึกข้อมูลการเช่าได้ ทำให้สัญญาเป็นไปได้ยากต้องในเรื่องของสิทธิ์เข้าถึง modifier()

และกำหนดสิทธิ์ให้แค่ผู้เช่าใช้ได้ เจ้าของไม่สามารถใช้ได้ มีพังก์ชัน ดังนี้

-depositFunds()

-payMonthlyRent()

-payFirstMonthAndDeposit()



จากรูปจะเห็นได้ว่ามีเช็คว่าไม่ใช่ผู้เช่าไม่สามารถเติมเงินเข้าและจ่ายค่าเช่าได้ และมีการเช็คยอดเงินว่าพอจ่ายหรือเปล่า

ส่วนนี้เป็นฟังก์ชันการเช็คว่ายอดที่ผู้เช่าห้องต้องจ่ายเท่าไรและจ่ายกี่เดือน โดยเจ้าของและผู้เช่าสามารถดูได้

-getMonthlyRent()

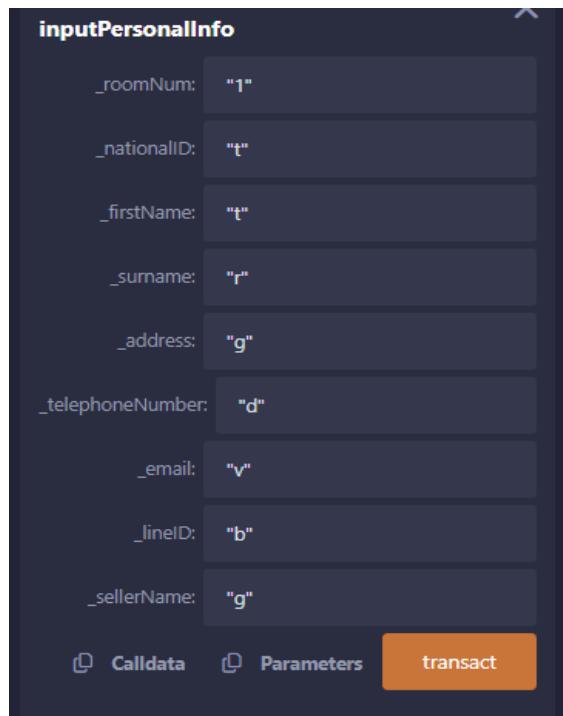
-getTotalPayment()

getMonthlyR...	getMont...
0: uint256: rentAmount 2	
1: uint256: monthsLeft 5	
getTotalPaym...	
0: uint256: 11	

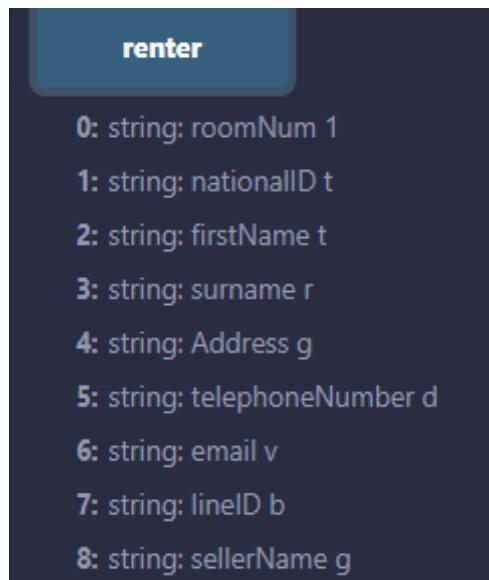
ทั้งสองจะแสดงก็ต่อเมื่อมีการบันทึกการเช่าแล้วเท่านั้น ถ้าหากยังไม่มีการบันทึกจะขึ้นเป็น uint 0

การประเมินของระบบการเช่า

inputPersonalInfo() เป็นการบันทึกข้อมูลผู้เช่า

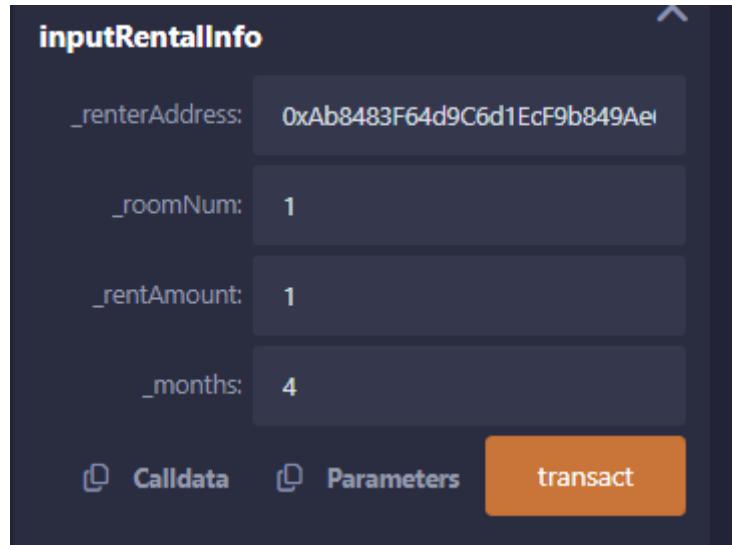


(ข้อมูลที่สมมุติขึ้นมา ดังรูป ข้อมูลจะถูกบันทึกตามตัวแปรต่างๆ)



ผลจากการ transact จะเห็นได้ว่าแสดงข้อมูลผู้เช่าตามที่บันทึกถูกต้อง

`inputRentalInfo()` เป็นการบันทึกข้อมูลห้องที่จะเช่าโดยจะมีการนำ `_rentAmount` และ `_months` ไปคำนวณต่อเพื่อหาเวลาที่สิ้นสุดสัญญา ค่าเช่าเดือนแรกพร้อมค่าประกันและค่าเช่าทั้งหมดและคืนเงินประกัน ตั้งราคาประกันอยู่ที่ 2 เท่าของราคาห้อง (`depositAmount: 2 * _rentAmount`)



ตัวอย่างโค้ดที่จะนำ `rentAmount` และ `_months` ไปใช้

```
uint256 firstMonthRent = rental.rentAmount;
uint256 totalDeposit = rental.depositAmount;
```

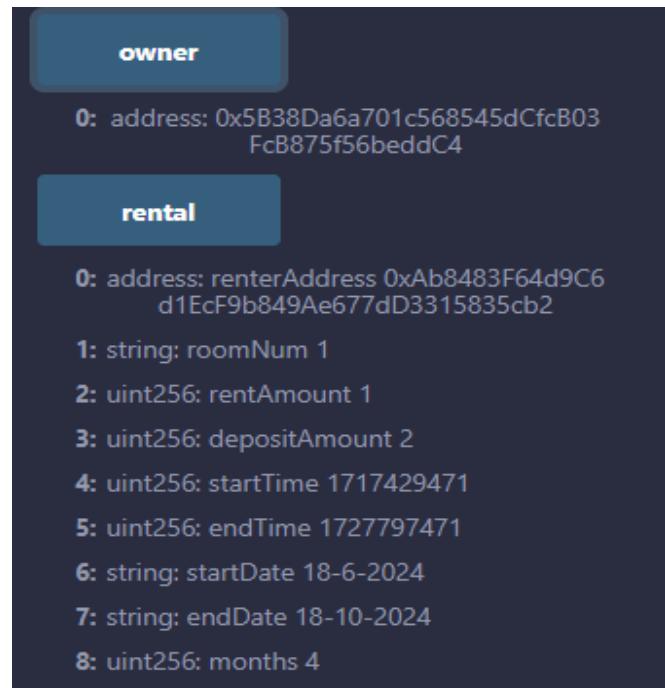
`firstMonthRent` ให้มีค่าเท่ากับค่าเช่าในเดือนแรกที่ถูกกำหนดไว้ในสัญญา

```
// ลดจำนวนเดือนที่เหลือเมื่อชำระค่าเช่าเดือนแรก
rental.months -= 1;

// หากยอดคงชำระหมด ให้สิ้นสุดการเช่า ฿ infinite gas
if (rental.months == 0) {
    rooms[rental.roomNum][rental.renterAddress].isRented = false;
    // เมื่อสัญญาหมดอายุ ตั้งค่า hasPaidFirstMonthAndDeposit เป็น false
    hasPaidFirstMonthAndDeposit[msg.sender] = false;
}
```

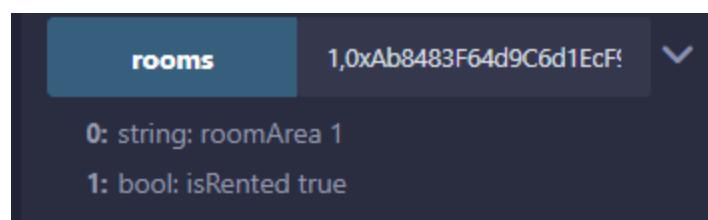
ลดจำนวนเดือนที่เหลือจากการจ่ายค่าชำระ

ผลลัพธ์เมื่อคัดบันทึกจะเห็นได้ว่ามีการคำนวณเวลาการเช่าและการแปลงเป็นวัน-เดือน-ปี ด้วย



พังก์ชันการแสดงการเช็คห้องว่าถูกเช่าหรือยัง โดยจะให้ใส่เป็นหมายเลขห้องและaddressของผู้เช่า

ถ้ามีการเช่าห้องนั้นไปแล้ว isRented จะบอกว่า true ถ้ายังไม่การเช่าหรือสิ้นสุดสัญญาจะขึ้นว่า false

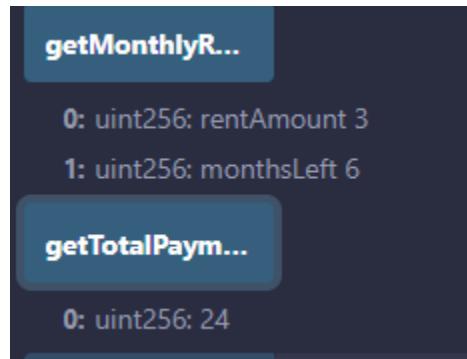


ตัวอย่างโค้ดจะเช็คจาก months ว่าเท่ากับ 0 หรือไม่ ถ้าเท่ากับ 0 จะออกมาเป็นfalse คือห้องว่าง

```
// หากยอดค้างชำระหมด ให้สิ้นสุดการเช่า
if (rental.months == 0) {
    rooms[rental.roomNum][rental.renterAddress].isRented = false;
}
```

`getTotalPayment()` จะเป็นการช่วยให้ผู้เช่าฝากเงินให้พอดีกับค่าเช่า ตัวอย่างตามรูปคือทั้งหมด 7 ETH

`getMonthlyRent()` จะเป็นการบอกระยะเวลาการจ่ายเงินตามที่เช่า เช่น ตัวอย่างในรูปจะแสดงราคา 1 ETH ต่อเดือน ระยะเวลา 4 เดือน

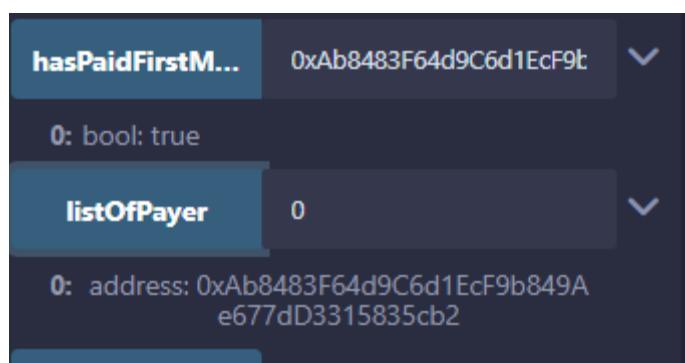


ตัวอย่างโค้ดคำนวณค่าเช่าทั้งหมดจากค่าเช่าคูณระยะเวลาเช่าบวกเงินประกัน

```
//ใช้ในการคืนค่าจำนวนเงินที่ต้องชำระทั้งหมด
function getTotalPayment() public view returns (uint256) {
    uint256 firstMonthRent = rental.rentAmount * rental.months;
    uint256 totalDeposit = rental.depositAmount;

    return firstMonthRent + totalDeposit;
}
```

พังก์ชันเช็คการจ่ายเงินค่าเช่าเดือนแรกพร้อมกับเงินประกัน เช็คเพื่อยืนยันการเช่าว่าเช่าจริงหรือไม่



ตัวอย่างโค้ด `hasPaidFirstMonthAndDeposit` จะเช็คจากการจ่ายเงินของผู้เช่าถ้ามีการจ่ายเงินแล้วจะบอกว่า `true`

```

// ตรวจสอบว่ามีเงินเพียงพอสำหรับชำระค่าเช่าเดือนแรกและเงินมัดจำ
require(balances[msg.sender] >= totalPayment, "Insufficient funds");

// เพิ่มเงินค่าเช่าและเงินมัดจำเข้าบัญชีของสัญญา
balances[address(this)] += totalPayment;

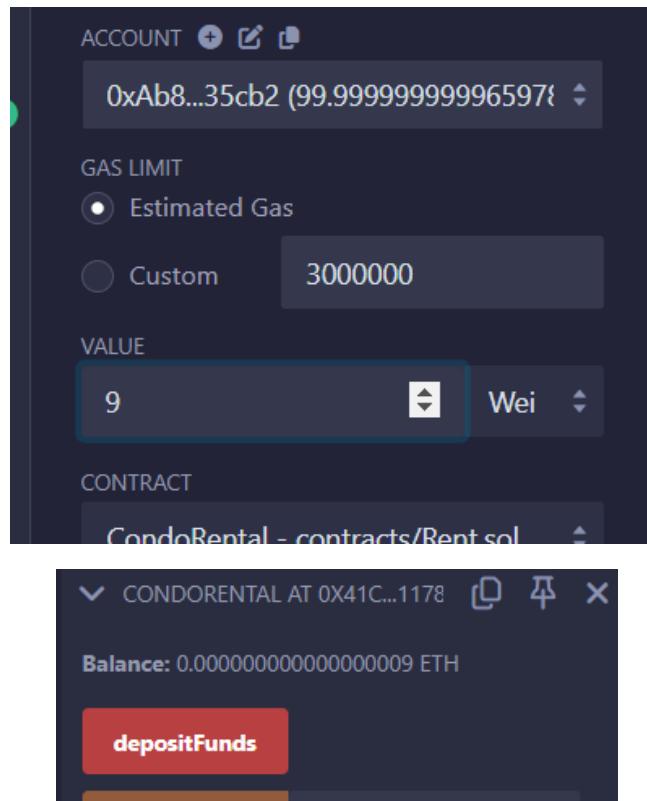
// ลดยอดเงินในบัญชีผู้เช่า
balances[msg.sender] -= totalPayment;

// เก็บรายชื่อผู้ชำระค่าเช่าและมัดจำ
listOfPayer.push(msg.sender);

hasPaidFirstMonthAndDeposit[msg.sender] = true;

```

พงก์ชันการฝากเงินจะให้เพียงผู้เช่าเท่านั้นที่สามารถฝากเงินได้ถ้าไม่ใช่ผู้เช่าจะเกิด error และส่งข้อความบอกว่า contract: "Only the renter can deposit funds".

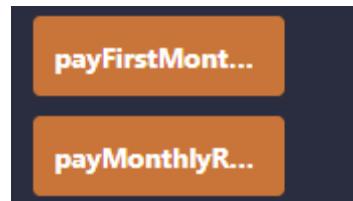


เมื่อเติมเงินเข้าจะมียอดเงินแสดงด้านบนตามรูป

ตัวอย่างโค้ดจะเช็คว่าใช้ผู้เช่าใหม่ เพิ่มจำนวน ETH ที่ถูกฝากเข้ามาในบัญชี ส่งเหตุการณ์ FundsDeposited ซึ่งระบุว่ามีการฝากเงิน

```
/ใช้ในการฝากเงินสำหรับการชำระค่าเช่าห้อง
function depositFunds() public payable {
    require(msg.sender == rental.renterAddress, "Only the
    balances[msg.sender] += msg.value;
    emit FundsDeposited(msg.sender, msg.value);
}
```

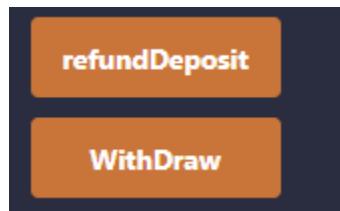
ฟังก์ชันการจ่ายค่าเช่าเดือนแรกกับประกันและเดือนต่อไป ผู้เช่าสามารถได้แค่คนเดียว



ตัวอย่างโค้ดจ่ายค่าเช่าเดือนแรกกับประกันจะมีการเช็คว่าจ่ายเดือนแรกยังจะไม่สามารถจ่ายเดือนต่อไปได้ จะคำนวณค่าเช่าได้จ่ายโค้ดบรรทัด 142-144 มีการเช็คยอดเงินว่าวางเดือนหรือไม่ในบรรทัดที่ 147

```
137     event FundsDeposited(address indexed renter, uint256 amount),
138     // ฟังก์ชัน payFirstMonthAndDeposit ใช้สำหรับการชำระค่าเช่าเดือนแรกและเงินมัดจำ
139     function payFirstMonthAndDeposit() public {
140         require(!hasPaidFirstMonthAndDeposit[msg.sender], "First month rent and depo
141
142         uint256 firstMonthRent = rental.rentAmount;
143         uint256 totalDeposit = rental.depositAmount;
144         uint256 totalPayment = firstMonthRent + totalDeposit;
145
146         // ตรวจสอบว่ามีเงินเพียงพอสำหรับชำระค่าเช่าเดือนแรกและเงินมัดจำ
147         require(balances[msg.sender] >= totalPayment, "Insufficient funds");
148
149         // เพิ่มเงินค่าเช่าและเงินมัดจำเข้าบัญชีของสัญญา
150         balances[address(this)] += totalPayment;
151
152         // ลดยอดเงินในบัญชีผู้เช่า
153         balances[msg.sender] -= totalPayment;
154
155         // เก็บรายชื่อผู้ชำระค่าเช่าและมัดจำ
156         listOfPayer.push(msg.sender);
157
158         hasPaidFirstMonthAndDeposit[msg.sender] = true;
159
160         // ลดจำนวนเดือนที่เหลือเมื่อชำระค่าเช่าเดือนแรก
161         rental.months -= 1;
162
163         // หากยอดค้างชำระหมด ให้ล็อกสูตรการเช่า
164         if (rental.months == 0) {
165             rooms[rental.roomNum][rental.renterAddress].isRented = false;
166             // เมื่อล็อกหมดอยู่ ล็อกค่า hasPaidFirstMonthAndDeposit เป็น false
```

ฟังก์ชันการคืนเงินประกันและฟังก์ชันถอนเงิน



ตัวอย่างโค้ดคืนเงินประกันใช้ได้แค่เจ้าของคอนโด จะเช็คเดือนว่ามีค่าเป็น0ใหม่(สิ้นสุดสัญญา) มีการตรวจสอบว่า การเรียกคืนค่าประกันใหม่คำสั่งนี้ใช้ในการแปลง address ของผู้เช่า (ผู้รับเงินค่าประกัน) จาก address ธรรมดามาเป็น address ที่สามารถรับเงินได้ (address payable) เพื่อให้สามารถโอนเงินได้

```
170
171     //คืนค่าประกัน
172     function refundDeposit() public onlyOwner {
173         require(rental.months == 0, "Rental period is not over yet");
174
175         // ตรวจสอบว่ามีเงินค่าประกันที่เรียกคืนหรือไม่
176         require(address(this).balance >= rental.depositAmount, "Insufficient deposit balance");
177
178         // แปลง address เป็น address payable ก่อนที่จะทำการโอนเงิน
179         address payable recipient = payable(rental.renterAddress);
180
181         // ถอนเงินค่าประกันให้ผู้เช่า
182         recipient.transfer(rental.depositAmount);
183     }
```

ตัวอย่างโค้ดถอนเงินเช็คว่าใช่เจ้าของใหม่ สามารถถอนเงินได้แค่ที่อยู่ในสัญญาเท่านั้น

โดยจำนวนเงินที่โอนคือยอดเงินทั้งหมดที่อยู่ในสัญญา (ดึงจาก address(this).balance)

```
//ใช้ในการถอนเงินทั้งหมดออกจากสัญญาโดยเจ้าของ
function WithDraw() public onlyOwner {
    require(msg.sender == owner);
    address payable sender = payable(msg.sender);
    sender.transfer(address(this).balance);
    listOfPayer = new address [](0);
}
```

พังก์ชันในการซื้อคอนโด

1.พังก์ชันกรอกข้อมูลส่วนตัว

ให้ผู้ใช้กรอก ชื่อ นามสกุล เลขบัตรประจำตัวประชาชน ที่อยู่ เบอร์โทรศัพท์

2.พังก์ชันการจอง

กรอกเลขห้อง address ผู้ซื้อ กรอกราคาการจอง

3.พังก์ชันยืนยันการจอง

ให้ผู้ใช้กรอกตามจำนวนเงินที่ตกลงกันไว้

4.พังก์ชันการซื้อ

ให้ผู้ใช้กรอกเงินตามราคาห้อง เพื่อจ่าย

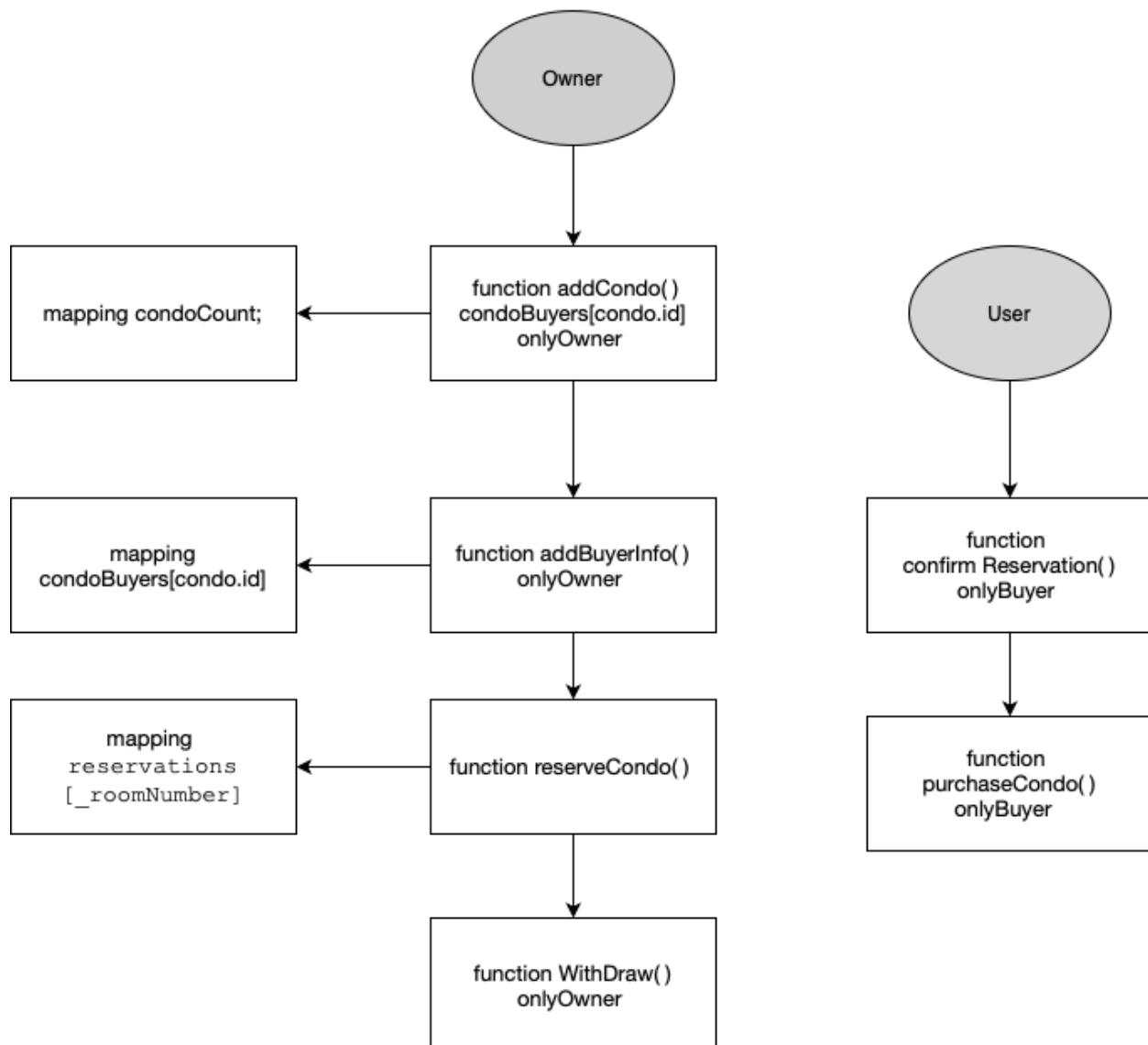
สิทธิ์การเข้าถึงระบบซื้อขาย

Buycondo.sol

Function	คำอธิบาย	เจ้าของ	ผู้ใช้/ผู้ซื้อ
addCondo()	เพิ่มคอนโด		
addBuyerInfo()	เพิ่มข้อมูลผู้ซื้อ		
reserveCondo()	เพิ่มข้อมูลการจอง		
purchaseCondo()	ซื้อคอนโด		
confirmReservation()	ยืนยันการจอง		
WithDraw()	ถอนเงิน		

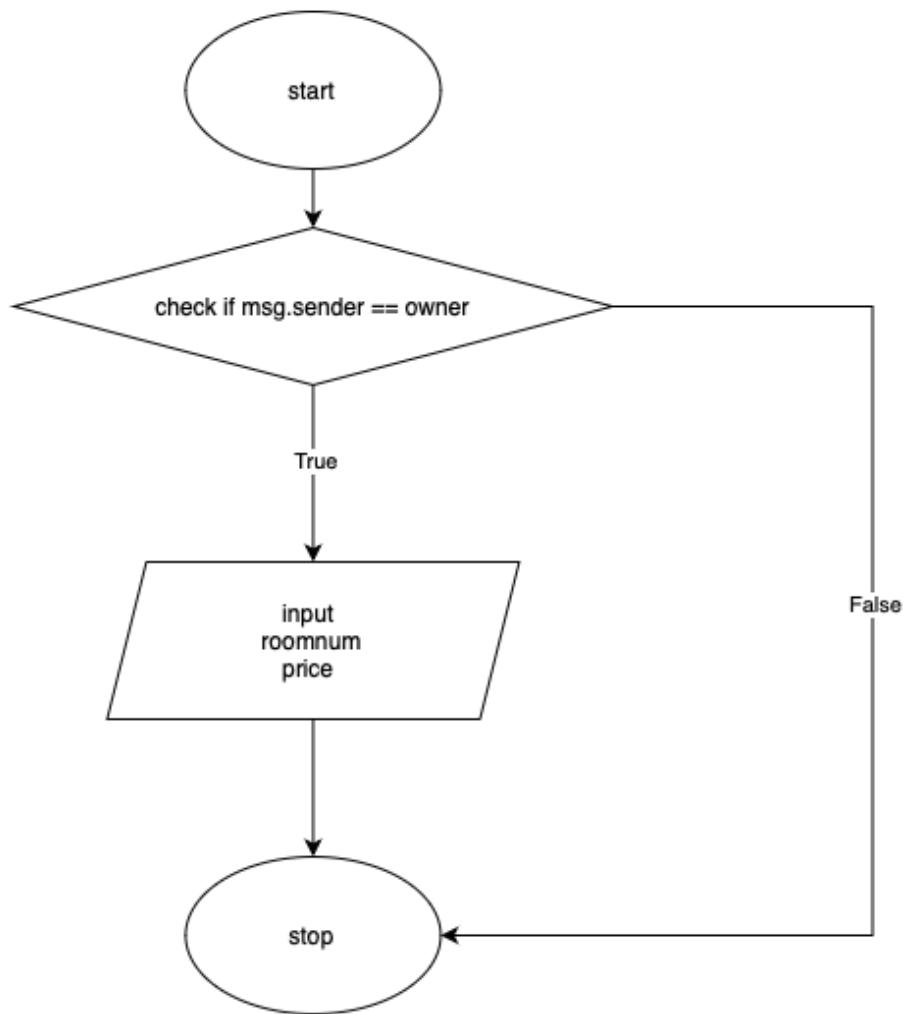
ສາປັຕຍກຣມຂອງຮະບບຈື້ອຂາຍ

Buy condo

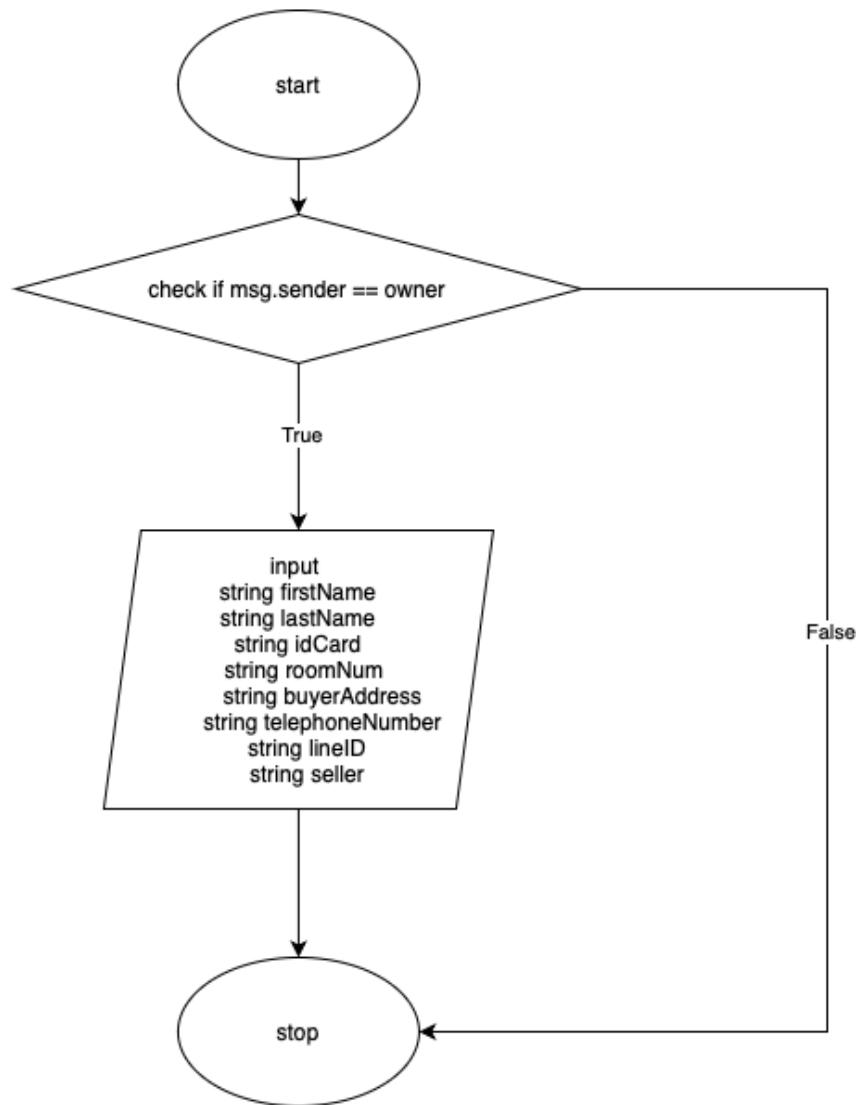


การออกแบบขั้นตอนการทำงานของระบบซื้อขาย

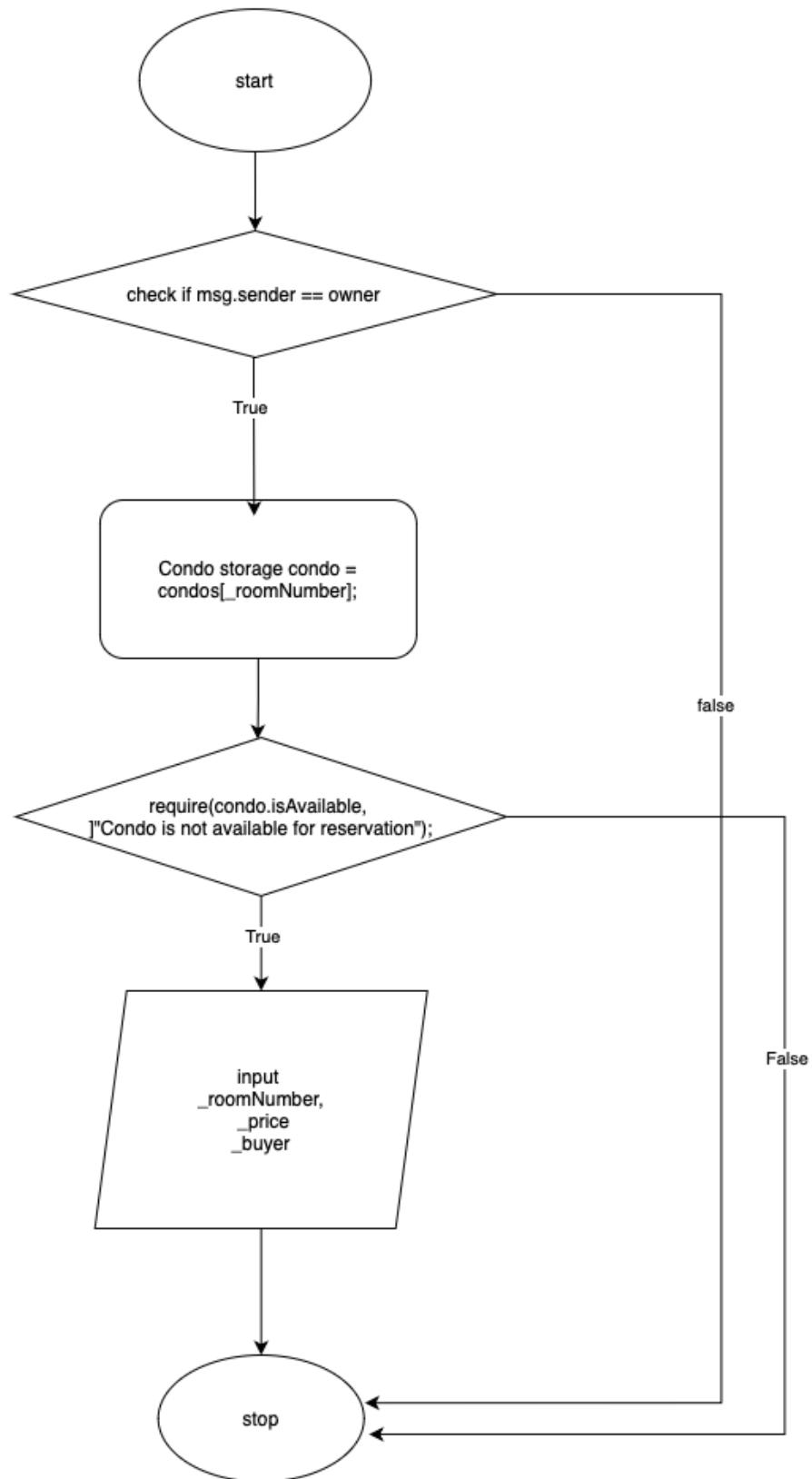
function addCondo
เพิ่มคอนโด

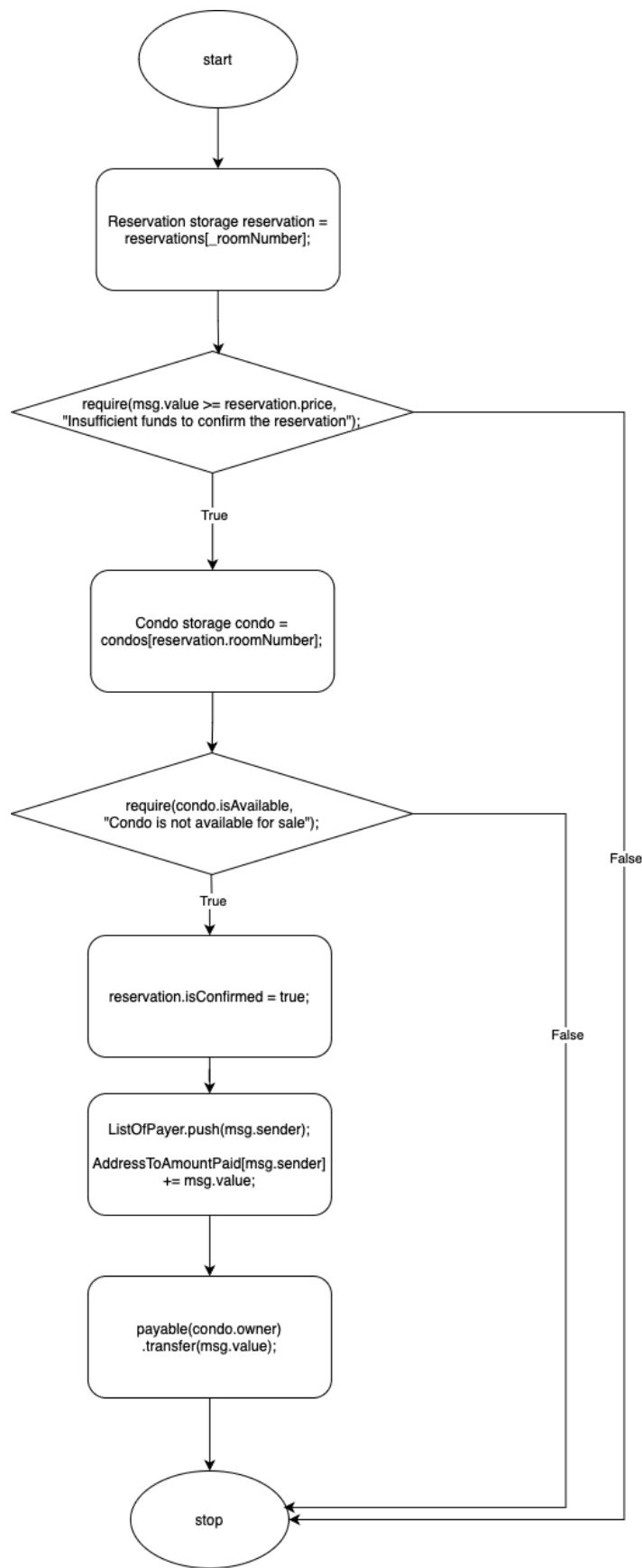


```
function addBuyerInfo  
เพิ่มข้อมูลคนซื้อ
```

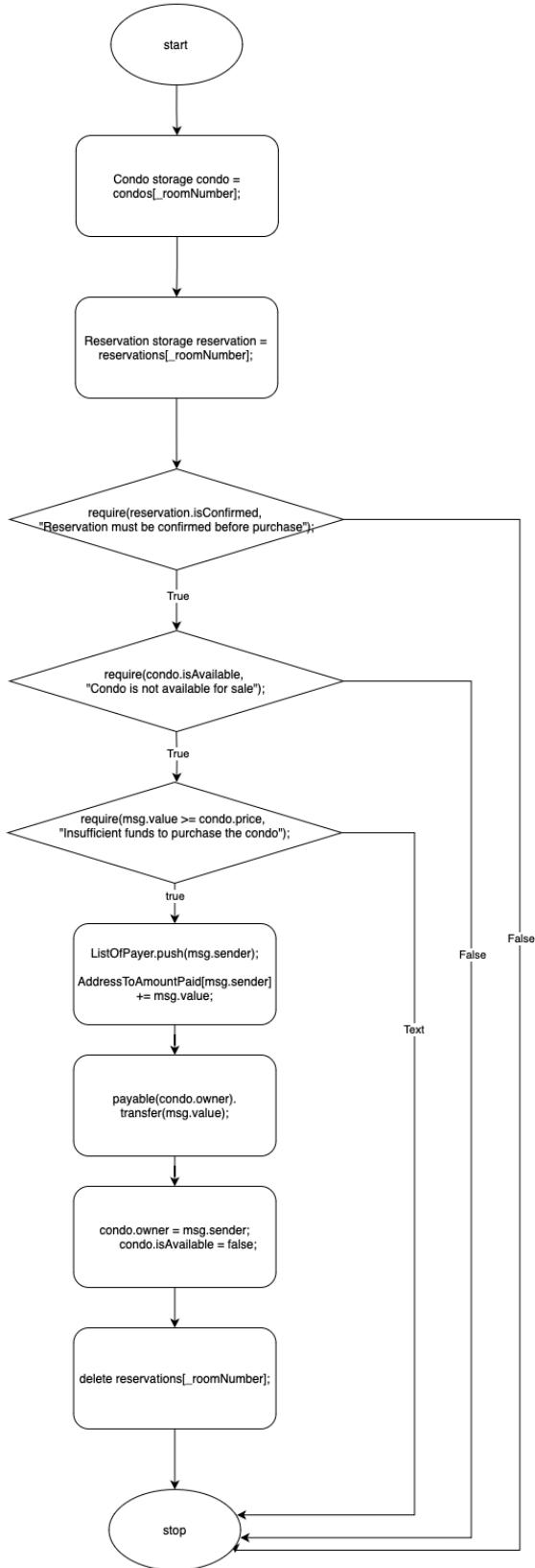


```
function reserveCondo  
เพิ่มชื่อ Mukarrajong
```





```
function purchaseCondo  
    การซื้อคอนโด
```



การทดสอบของระบบซื้อขายและการประเมิน

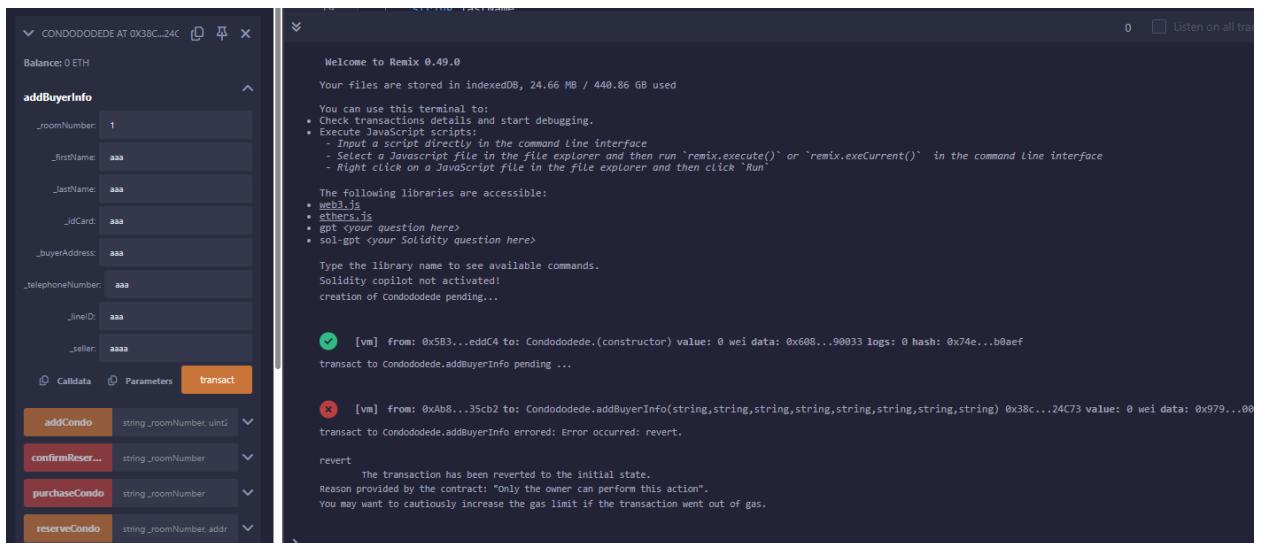
การควบคุมการเข้าถึง

Buycondo.sol

ในสัญญาเรียกใช้คำสั่งในสิทธิ์ เพื่อความปลอดภัยของสัญญาและข้อมูลของผู้ใช้ โดยจะมีฟังก์ชันที่ผู้ใช้ หรือ ผู้ซื้อคอนโดจะไม่สามารถใช้ได้เมื่อฟังก์ชันดังนี้

- addcondo()
- addBuyerInfo()
- reserveCondo()
- WithDraw()

ผลของการ Deploy เมื่อผู้ช่วยใช้ฟังก์ชันของเจ้าของ

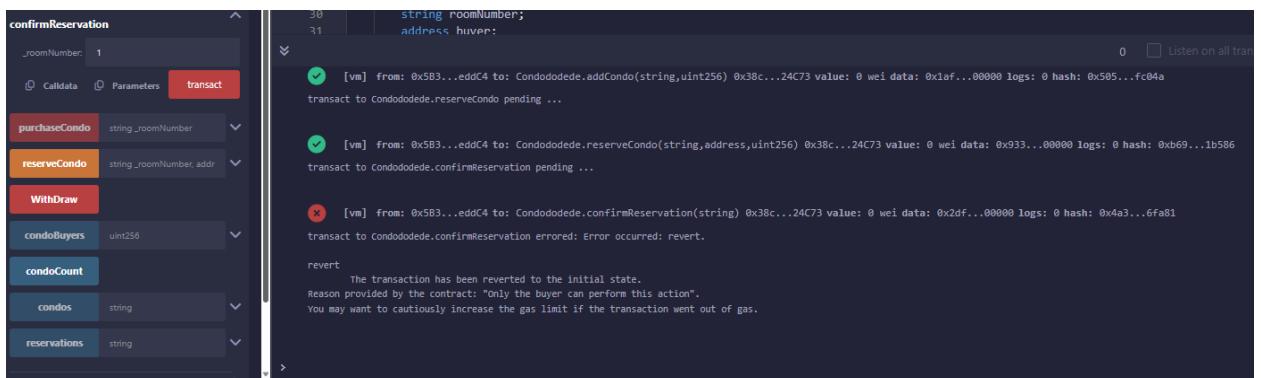


จากรูปภาพจะเห็นได้ว่าถ้าไม่ใช้ เจ้าของไม่สามารถบันทึกข้อมูลการซื้อได้ ทำให้สัญญาเป็นไปได้ยากต้องในเรื่องของ สิทธิ์เข้าถึง modifier()

และกำหนดสิทธิ์ให้ เจ้าของไม่สามารถใช้ได้มีฟังก์ชัน ดังนี้

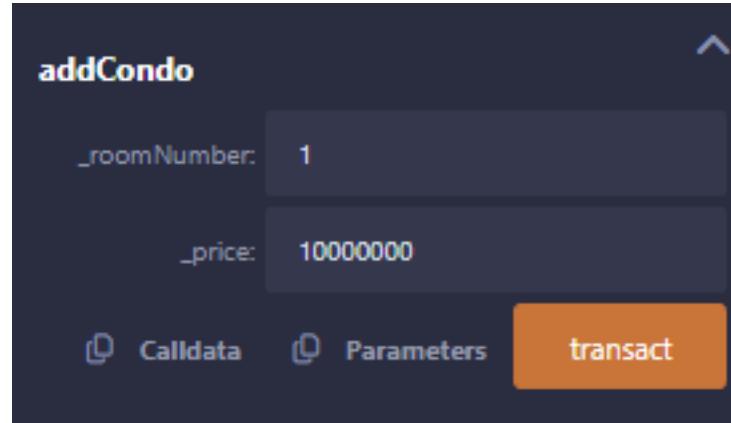
-confirmReservation()

-purchaseCondo()



จากตัวอย่างการทำงานจะมีการเช็คว่าใช้ผู้ช่วยไหมหากว่าไม่ใช้ผู้ช่วยแสดงข้อความ ดังกล่าว

อันดับแรกของการเริ่มสัญญาคือการ addCondo เข้าไปก่อนกรอก เลขห้อง กับ ราคา โดยเจ้าของจะเป็นคนตั้ง



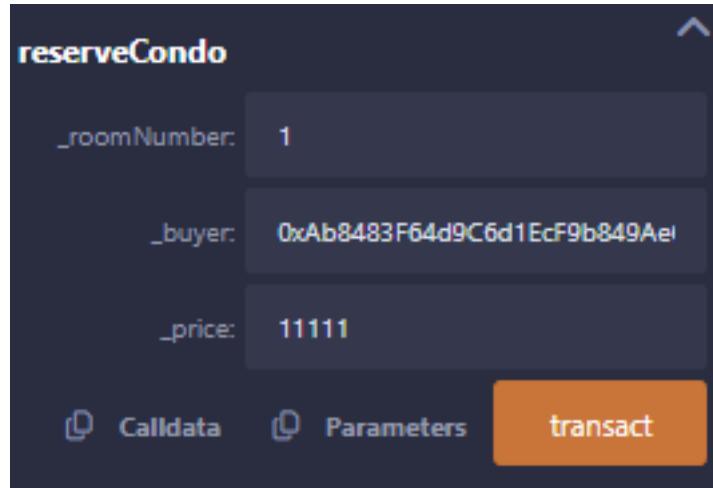
หากเพิ่มเสร็จแล้วจากนั้นลองกดปุ่ม condoCount จะแสดงผลตามตัวอย่าง



และจะเห็นว่าห้องเป็นของใคร ต้องใช้ฟังก์ชัน condos ให้กรอก เลขห้องและกด call จะแสดงผลตามตัวอย่าง ในตัวอย่างนี้จะยังเป็นเจ้าของที่ยังเป็นเจ้าของห้องอยู่ และจะแสดงผลว่าห้องสามารถซื้อได้ และนี่คือตัวอย่างหากไม่ใช่เจ้าของที่เป็นเจ้าของห้อง ตัวห้องจะแสดงผลว่าไม่สามารถซื้อได้

0: uint256: id 1	1
1: string: roomNumber 1	
2: address: owner 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2	
3: uint256: price 100000000	
4: bool: isAvailable false	

reserveCondo ในส่วนฟังก์ชันตัวนี้เจ้าของจะเป็นเพิ่มข้อมูลการจอง โดยจะต้องเพิ่มเลขห้องที่เคยสร้างไว้แล้วใน addCondo กรอก adress ของคนซื้อ และกรอกเงินค่าจอง และ หากเพิ่มเลขห้องไม่ที่ไม่มีอยู่ในระบบจะแสดงตามนี้



revert
The transaction has been reverted to the initial state.
Reason provided by the contract: "Condo is not available for reservation".
You may want to cautiously increase the gas limit if the transaction went out of gas.

สามารถเช็คสถานะการจองของคนซื้อได้ ในส่วนของ isConfirmed หมายถึงผู้ซื้อได้จ่ายเงินค่าจองแล้วหรือยัง หากยังจะแสดงผลตามด้วยร่างและถ้าหากจ่ายแล้วจะแสดงผลตามรูปทางขวา

reservations

: 1

Calldata Parameters call

0: string: roomNumber 1
1: address: buyer 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
2: uint256: price 11111
3: bool: isConfirmed false

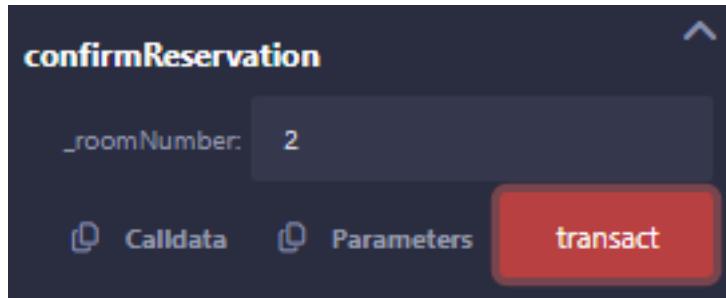
reservations

: 1

Calldata Parameters call

0: string: roomNumber 1
1: address: buyer 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
2: uint256: price 11111
3: bool: isConfirmed true

confirmReservation ដើម្បីចែករាយលេខលេខទីនៃការរំសៀវភៅនៃការបង់បានស៊ូវុយ។ តាមរូបរាងខាង



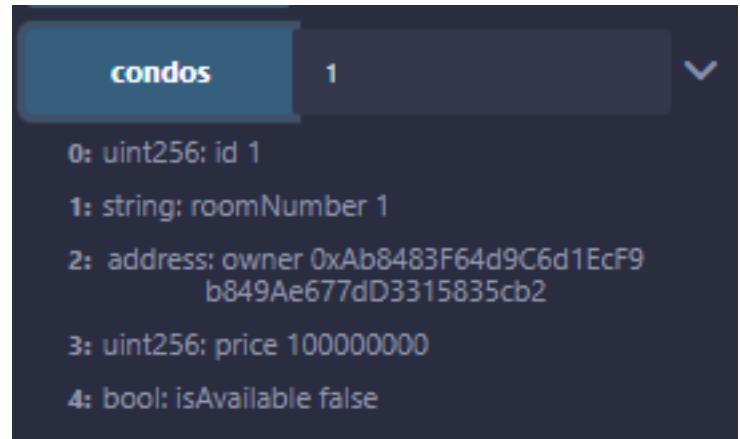
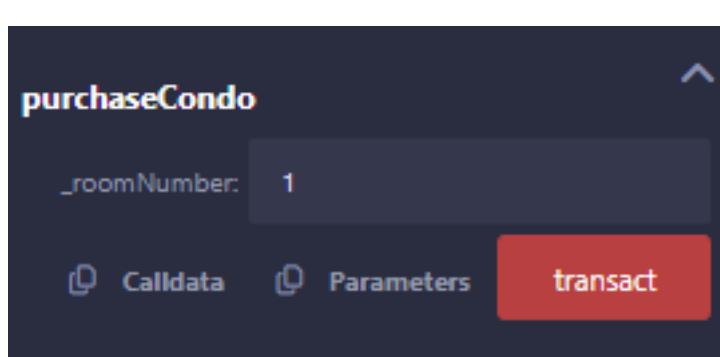
revert

The transaction has been reverted to the initial state.

Reason provided by the contract: "Only the buyer can perform this action".

You may want to cautiously increase the gas limit if the transaction went out of gas.

purchaseCondo ដើម្បីចែករាយលេខលេខទីនៃការបង់បាននិងតម្លៃរាយទីនៃការបង់បាន។ ក្នុងការបង់បាន អ្នកដឹងថា តួនាទីលើលក្ខណៈនៃលក្ខណៈទាំងអស់នៅក្នុងការបង់បាន និងតួនាទីលើលក្ខណៈទាំងអស់នៅក្នុងការបង់បាន។



หากผู้ใช้งานไม่ได้ยืนยันการจองก่อนจะไม่สามารถทำการได้

```
revert
```

The transaction has been reverted to the initial state.

Reason provided by the contract: "Reservation must be confirmed before purchase".

You may want to cautiously increase the gas limit if the transaction went out of gas.

หลังจากข้อแล้วสัญญาจะลบข้อมูลการจองออก

The screenshot shows a blockchain transaction interface. At the top, there is a dark header with the word "reservations" in blue. Below the header, there is a light gray input field containing the number "1". To the right of this field are two buttons: "Calldata" and "Parameters", both with small icons. A large blue button labeled "call" is positioned next to them. Below these buttons, there is a list of four items, each with a small icon and a label: "0: string: roomNumber", "1: address: buyer 0x00000000000000000000000000000000", "2: uint256: price 0", and "3: bool: isConfirmed false". The entire interface has a dark background with light-colored text and buttons.

ตรวจสอบเงื่อนไข (Check conditions)

Buycondo.sol ในสัญญาฉบับนี้มีการใช้ require() เพื่อเป็นการตรวจสอบเงื่อนไขก่อนที่จะเริ่มการทำงานของฟังก์ชัน โดย ฟังก์ชันที่มีการใช้ require() มีดังนี้

```
function reserveCondo(string memory _roomNumber, address _buyer, uint _price)
public onlyOwner {
```

```
    require(condo.isAvailable, "Condo is not available for reservation");
โดยจะตรวจสอบว่าห้องนี้สามารถจองได้ไหม
```

```
function confirmReservation(string memory _roomNumber) public payable
onlyBuyer(_roomNumber) {
```

```
    require(msg.value >= reservation.price, "Insufficient funds to confirm the
reservation");
```

require จะตรวจสอบว่าเงินที่ผู้ใช้กรอกมา มีค่าเท่ากับค่าจองที่กำหนดไว้ไหม และ

```
    require(condo.isAvailable, "Condo is not available for sale");
require จะตรวจสอบว่าคอนโดยังคงว่างอยู่ ถ้าไม่ว่าง (ถูกจองหรือขายไปแล้ว) ฟังก์ชันจะล้มเหลวและ
แสดงข้อความว่า "Condo is not available for sale" เพื่อป้องกันการยืนยันการจองสำหรับคอนโดที่ไม่
พร้อมขาย
```

```
function purchaseCondo(string memory _roomNumber) public payable
onlyBuyer(_roomNumber)
```

```
    require(reservation.isConfirmed, "Reservation must be confirmed before purchase");
ตรวจสอบว่าการจองได้รับการยืนยันแล้ว ถ้าไม่ฟังก์ชันจะล้มเหลวและแสดงข้อความว่า "Reservation
must be confirmed before purchase"
```

```
require(condo.isAvailable, "Condo is not available for sale");
```

ตรวจสอบว่าคอนโดว่างและพร้อมขาย ถ้าไม่ฟังก์ชันจะล้มเหลวและแสดงข้อความว่า "Condo is not available for sale"

```
require(msg.value >= condo.price, "Insufficient funds to purchase the condo");
```

ตรวจสอบว่าจำนวนเงินที่ส่งมาพอเพียงสำหรับการซื้อคอนโด ถ้าไม่ฟังก์ชันจะล้มเหลวและแสดงข้อความว่า "Insufficient funds to purchase the condo"

```
function Withdraw
```

```
require(msg.sender == seller); =
```

ตรวจสอบว่าคนถอนเงินใช้เจ้าของสัญญาหรือไม่

ฐานข้อมูล

ระบบผ่อนชำระ

Installment.sol

ไฟล์นี้จะเน้นการเก็บข้อมูลของผู้ซื้อ และการจ่ายเงินดาวน์ ดังนั้นจะมีการเก็บข้อมูลดังนี้

1. การเก็บข้อมูลของผู้ซื้อ

createInfoUser()	
string	RoomNum
string	NationalID
string	FirstName
string	Surname
string	Address
string	PhoneNumber
string	Email
string	LineID
string	SellerName

2. การเก็บข้อมูลราคาของห้อง

InfoPrice	
string	RoomNum
uint256	propertyPriceUSD
uint256	downPaymentUSD
uint256	loanTerm
uint256	totalRepayment
uint256	monthlyPayment

3. เป็น Array ที่เก็บ Address ของผู้ที่จ่ายเงิน

ListOfPayer	
address	InfoUser

4. เก็บ Address ของผู้โอนเงิน และ จำนวนเงินที่ผู้ซื้อห้องนั่นจ่ายไป

AddressToAmountPaid	
Address	msg.sender
uint256	msg.value

UserInstallment.sol

1. เก็บข้อมูลเลขห้อง, จำนวนเงินที่ผู้ซื้อห้องนั้นจ่ายไป, ราคารห้องรวมดอกเบี้ยที่ผู้ซื้อต้องจ่ายให้ครบ

RoomToAmountPaid	
String	RoomNum
uint256	amountPaid
uint256	totalRepayment

2. เก็บ Address ของผู้โอนเงิน และ จำนวนเงินที่ผู้ซื้อห้องนั่นจ่ายไป

AddressToAmountPaid	
Address	msg.sender
uint256	msg.value

ระบบเช่าคอนโด

1. เก็บข้อมูลของลูกค้า

inputPersonalInfo	
string	roomNum
string	nationalID
string	firstName
string	surname
string	Address
string	telephoneNumber
string	email
string	lineID
string	sellerName

2. เก็บข้อมูลการเช่าห้องและคำนวณระยะเวลาการเช่าห้อง

inputRentalInfo	
address	renterAddress
string	roomNum
uint256	rentAmount
uint256	depositAmount
uint256	startTime
uint256	endTime
string	startDate
string	endDate
uint256	months
address	renterAddress

3. เก็บข้อมูลห้อง

Room	
String	roomArea
Bool	isRented

4. เก็บข้อมูลเวลาเพื่อมาแปลงเป็นวันที่

timestampToDate	
Uint256	timestamp

5. เก็บค่าเช่าเดือนแรกพร้อมประกัน

payFirstMonthAndDeposit	
Uint256	firstMonthRent
Uint256	totalDeposit
Uint256	totalPayment

6. เก็บการชำระค่าเช่าเดือนต่อไป

payMonthlyRent	
Uint256	monthlyRent

7. เก็บข้อมูลการคืนประกัน

getMonthlyRent	
Uint256	rentAmount
Uint256	monthsLeft

8. เก็บจำนวนค่าเช่าทั้งหมด

Uint256	firstMonthRent
Uint256	totalDeposit
Uint256	totalMonths

ระบบซื้อคอนโด

1. เก็บข้อมูลผู้ซื้อ

addBuyerInfo()	
string	RoomNumber
string	FirstName
string	LastName
string	idCard
string	Address
string	telephoneNumber
string	LineID
string	seller
string	RoomNumber

2.เก็บข้อมูลห้อง

addCondo	
string	roomNumber
uint256	price

3.เก็บของมูลการจอง

reserveCondo	
string	string
address	address
uint256	uint256

สรุป

โครงการนี้มีวัตถุประสงค์เพื่อพัฒนาระบบจัดการธุกรรมที่เกี่ยวข้องกับการผ่อนค่าโอน โดยมีการเชื่อมต่อระบบของผู้ให้กู้และผู้รับชำระเงิน ตลอดจนผู้จัดการห้องแม่ข่าย ผ่าน Oracle ที่เชื่อมต่อไปยัง Blockchain ที่มีประสิทธิภาพ ดังนี้

- ระบบผ่อนค่าโอน (Installment System)

ระบบนี้ถูกออกแบบมาเพื่อช่วยให้ผู้ใช้สามารถทำการผ่อนชำระค่าค่าโอนได้โดยมีการบันทึกจำนวนเงินที่ชำระแล้วและยอดรวมที่ต้องชำระทั้งหมด การชำระเงินจะใช้สกุลเงิน ETH ซึ่งจะถูกแปลงเป็น USD ผ่านการใช้งาน Oracle จาก Chainlink เพื่อให้เป็นไปตามข้อกำหนดของ Consensus

- ระบบเช่าค่าโอน (Rental System)

ระบบเช่าค่าโอนมีการจัดการการเช่าค่าโอนโดยอย่างละเอียด ทั้งนี้รวมถึงการบันทึกข้อมูลการเช่า การชำระค่าเช่า และการจัดการสัญญาเช่า ผู้เช่าจะทำการชำระเงินในสกุลเงิน ETH ซึ่งจะแปลงเป็น USD ผ่าน Oracle เช่นเดียวกับระบบผ่อนค่าโอน

- ระบบซื้อขายค่าโอน (Trading System)

ระบบซื้อขายค่าโอนได้รับการออกแบบมาเพื่ออำนวยความสะดวกในการซื้อและขายค่าโอนโดย มีการบันทึกธุกรรมการซื้อขาย การชำระเงิน การชำระเงินจะใช้สกุลเงิน ETH และแปลงเป็น USD ผ่าน Oracle เพื่อรักษาความเป็นมาตรฐานและความโปร่งใสของระบบ

การใช้ Oracle ในระบบค่าเงิน

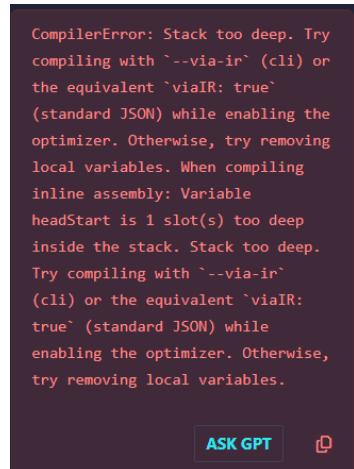
เพื่อรักษาความโปร่งใสและความเป็นมาตรฐานของการทำธุกรรม ระบบทุกรอบจะใช้ Oracle จาก Chainlink ในการแปลงค่าเงินจาก ETH เป็น USD สิ่งนี้จะช่วยให้การคำนวณยอดเงินที่ต้องชำระและยอดเงินที่ได้รับเป็นไปอย่างถูกต้องและเป็นไปตามสภาพตลาดปัจจุบัน

บทสรุป

โครงการนี้จัดตั้งขึ้นเพื่อพัฒนาระบบจัดการธุกรรมค่าโอนโดยมีประสิทธิภาพและโปร่งใส โดยแยกแต่ละระบบออกจากกันอย่างชัดเจนและใช้งาน Oracle ในการแปลงค่าเงิน เพื่อรักษาความเป็นมาตรฐานของการทำธุกรรมในสภาพแวดล้อมแบบ Blockchain

อุปสรรคในการทำโครงการ

ในการทำงานมีอุปสรรคในด้านของภาษาที่ใช้เขียน Smart Contract โดยมีปัญหาที่ เมื่อตั้งตัวแปรมากกว่า 12 ตัว จะขึ้นดังภาพ



Error

โดยแก้ไขปัญหาโดยการเพิ่มฟังก์ชันการรับค่า แต่มีข้อเสียคือ เสียค่าธรรมเนียมโดยใช้เหตุ และยังทำให้ช้าลงต่อการพัฒนาโครงการ

อ้างอิง

การแปลงค่าเงินETH, Wei, USD จาก <https://eth-converter.com/>

Suitability Evaluation Framework จาก Evaluating Suitability of Applying Blockchain by Lo et al.

Oracle การแปลงค่าเงิน

<https://github.com/smartcontractkit/chainlink/blob/develop/contracts/src/v0.8/shared/interfaces/AggregatorV3Interface.sol>