Selecting Test Data Amidst Non-standard Compliance

Theerasak Maneeneim
2023-09-09
[Tester’s Day 2023 - Everything You Can Test]

The billing app bugs were found on production.

TESTER'S DAY '2023

Rounding

Programmer didn't recognize about default mode of midpoint-rounding. It cause a bug in production.

Json date string

To cut date string without realize datetime standard

Payment QR

Error found on production with unknown cause.

Theerasak Maneeneim

1st times software tester

theerasak.com

Recap - Test data selecting technique

Challenge 1

Requirement :

We will give a discount per bill in different rate.

If spending less than 1,000THB, discount will not be applied.
If spending between 1,000-3,000THB, we give 5% discount.
If spending between 3,000-5,000THB, we give 10% discount.
If spending over 5,000THB, we will give 15% discount.

(Note: use number with 2 decimal digits)

Recap - Test data selecting technique

TESTER'S DAY 2023

Challenge 1

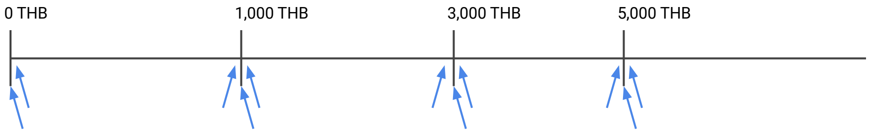
Requirement :

We will give a discount per bill in different rate.

If spending less than 1,000THB, discount will not be applied.
If spending between 1,000-3,000THB, we give 5% discount.
If spending between 3,000-5,000THB, we give 10% discount.
If spending over 5,000THB, we will give 15% discount.

(Note: use number with 2 decimal digits)

Boundary Value Analysis :



Test Data (for getDiscountRate) :

No	Spending Amount	Discount rate
1	0.00 THB	0.00
2	0.01 THB	0.00
3	999.99 THB	0.00
4	1000.00 THB	0.05
5	1000.01 THB	0.05
6	2999.99 THB	0.05
7	3000.00 THB	0.10
8	3000.01 THB	0.10
9	4999.99 THB	0.10
10	5000.00 THB	0.15
11	5000.01 THB	0.15

Selecting Test Data Amidst Non-standard Compliance

Rounding

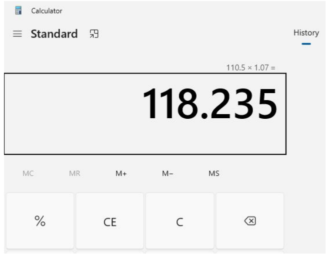
Requirement :

To follow TFRS, the program must calculate VAT 7% from subtotal on invoice and rounding it to 2 digits. After that, add it to Total amount and rounding it again to 2 digits.

(Note: if fragment over or equal than half, round up. Ex. VAT amount 7.735 will be rounded to 7.74)

What’s test data should be use?

No	Subtotal Amount	VAT amount	Total amount (included VAT 7%)
1	0.00 THB	0.00 THB	0.00 THB
2	0.01 THB	0.00 THB	0.01 THB
3	100.00 THB	7.00 THB	107.00 THB
4	110.5 THB	7.74 THB	118.24 THB



110.5 × 1.07 =

118.235

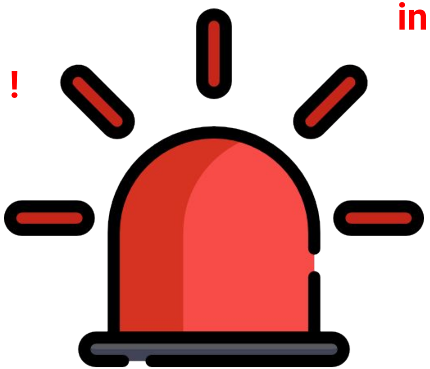
Rounding

Requirement :

To follow TFRS, the program must calculate VAT 7% from subtotal on invoice and rounding it to 2 digits. After that, add it to Total amount and rounding it again to 2 digits.

(Note: if fragment over or equal than half, round up. Ex. VAT amount 7.735 will be rounded to 7.74)

No	Subtotal Amount	VAT amount	Total amount (included VAT 7%)
1	0.00 THB	0.00 THB	0.00 THB
2	0.01 THB	0.00 THB	0.01 THB
3	100.00 THB	7.00 THB	107.00 THB
4	110.5 THB	7.74 THB	118.24 THB



BUG found in production !

BUG found in production !

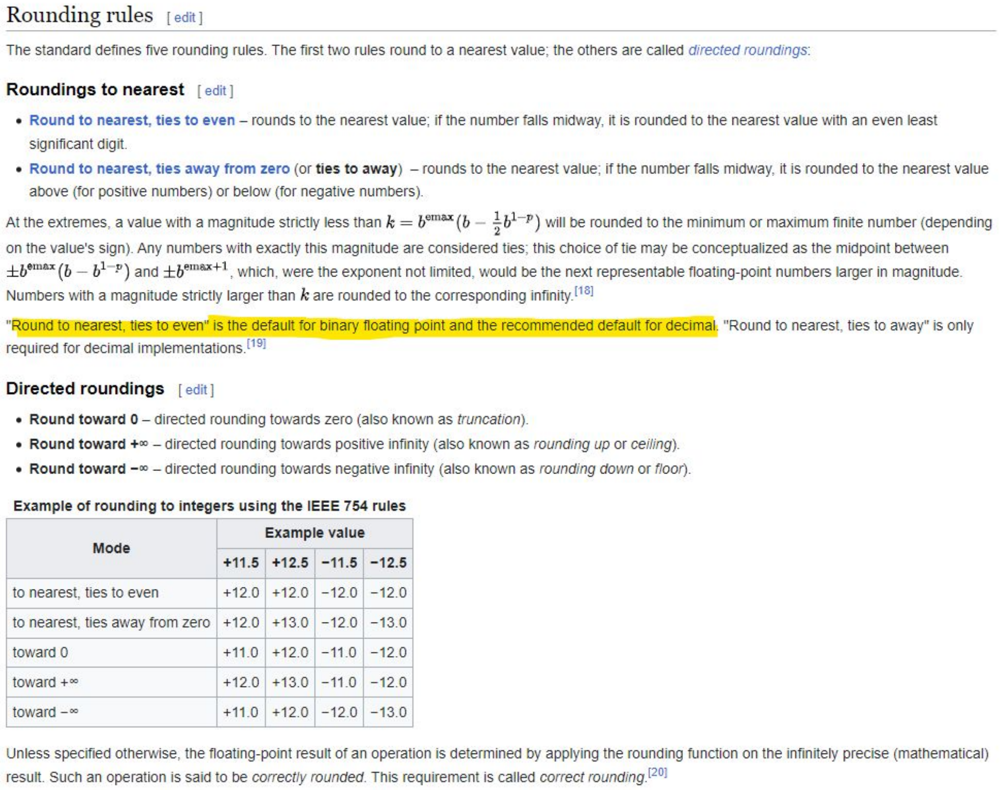
BUG found in production !

BUG found in production !

BUG found in production !

Bug found if subtotal amount = 101.5 THB

Rounding



Rounding rules [edit]

The standard defines five rounding rules. The first two rules round to a nearest value; the others are called *directed roundings*:

Roundings to nearest [edit]

- **Round to nearest, ties to even** – rounds to the nearest value; if the number falls midway, it is rounded to the nearest value with an even least significant digit.
- **Round to nearest, ties away from zero** (or **ties to away**) – rounds to the nearest value; if the number falls midway, it is rounded to the nearest value above (for positive numbers) or below (for negative numbers).

At the extremes, a value with a magnitude strictly less than $k = b^{\text{emax}}(b - \frac{1}{2}b^{1-p})$ will be rounded to the minimum or maximum finite number (depending on the value's sign). Any numbers with exactly this magnitude are considered ties; this choice of tie may be conceptualized as the midpoint between $\pm b^{\text{emax}}(b - b^{1-p})$ and $\pm b^{\text{emax}+1}$, which, were the exponent not limited, would be the next representable floating-point numbers larger in magnitude. Numbers with a magnitude strictly larger than k are rounded to the corresponding infinity.[18]

"Round to nearest, ties to even" is the default for binary floating point and the recommended default for decimal. "Round to nearest, ties to away" is only required for decimal implementations.[19]

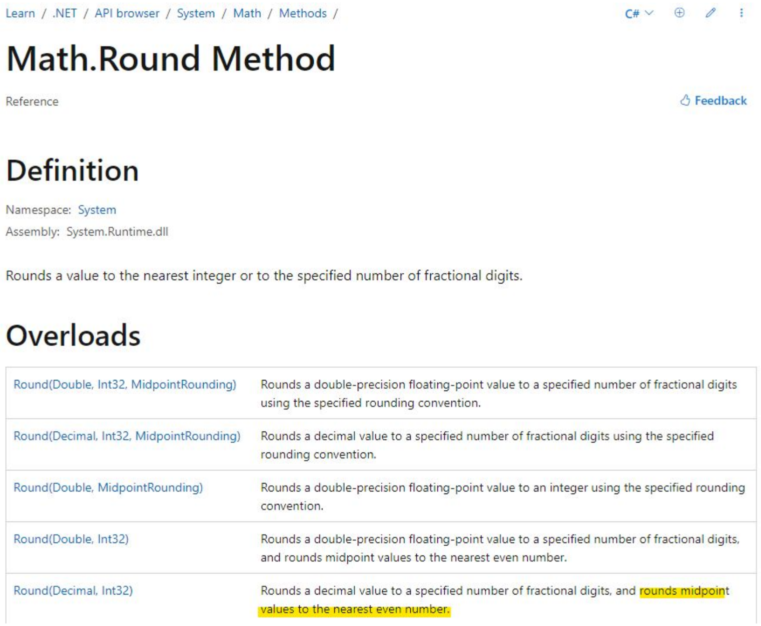
Directed roundings [edit]

- **Round toward 0** – directed rounding towards zero (also known as *truncation*).
- **Round toward +∞** – directed rounding towards positive infinity (also known as *rounding up* or *ceiling*).
- **Round toward −∞** – directed rounding towards negative infinity (also known as *rounding down* or *floor*).

Example of rounding to integers using the IEEE 754 rules

Mode	Example value			
	+11.5	**+12.5**	**−11.5**	**−12.5**
to nearest, ties to even	+12.0	+12.0	−12.0	−12.0
to nearest, ties away from zero	+12.0	+13.0	−12.0	−13.0
toward 0	+11.0	+12.0	−11.0	−12.0
toward +∞	+12.0	+13.0	−11.0	−12.0
toward −∞	+11.0	+12.0	−12.0	−13.0

Unless specified otherwise, the floating-point result of an operation is determined by applying the rounding function on the infinitely precise (mathematical) result. Such an operation is said to be *correctly rounded*. This requirement is called *correct rounding*.[20]



Learn / .NET / API browser / System / Math / Methods /

C#

Math.Round Method

Reference

Feedback

Definition

Namespace: System

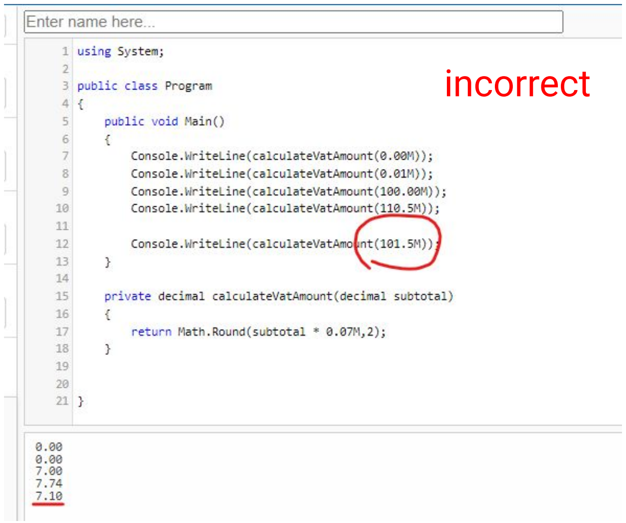
Assembly: System.Runtime.dll

Rounds a value to the nearest integer or to the specified number of fractional digits.

Overloads

Round(Double, Int32, MidpointRounding)	Rounds a double-precision floating-point value to a specified number of fractional digits using the specified rounding convention.
Round(Decimal, Int32, MidpointRounding)	Rounds a decimal value to a specified number of fractional digits using the specified rounding convention.
Round(Double, MidpointRounding)	Rounds a double-precision floating-point value to an integer using the specified rounding convention.
Round(Double, Int32)	Rounds a double-precision floating-point value to a specified number of fractional digits, and rounds midpoint values to the nearest even number.
Round(Decimal, Int32)	Rounds a decimal value to a specified number of fractional digits, and rounds midpoint values to the nearest even number.

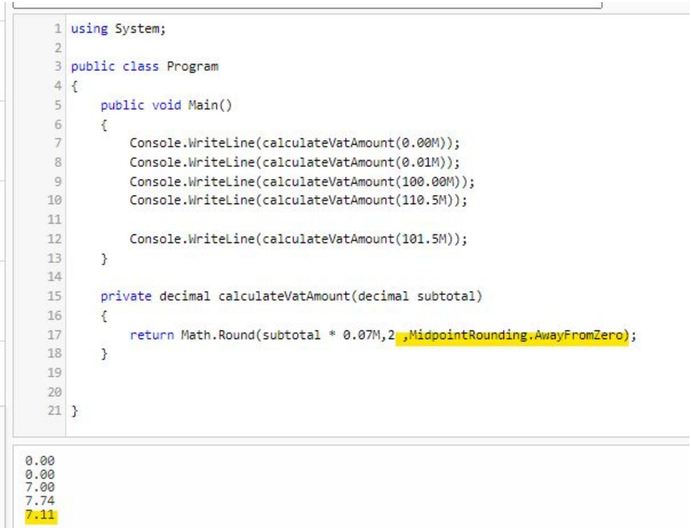
Rounding



incorrect

```
using System;

public class Program
{
    public void Main()
    {
        Console.WriteLine(calculateVatAmount(0.00M));
        Console.WriteLine(calculateVatAmount(0.01M));
        Console.WriteLine(calculateVatAmount(100.00M));
        Console.WriteLine(calculateVatAmount(110.5M));

        Console.WriteLine(calculateVatAmount(101.5M));
    }

    private decimal calculateVatAmount(decimal subtotal)
    {
        return Math.Round(subtotal * 0.07M,2);
    }



}
```

```
0.00
0.00
7.00
7.74
7.10
```



```
using System;

public class Program
{
    public void Main()
    {
        Console.WriteLine(calculateVatAmount(0.00M));
        Console.WriteLine(calculateVatAmount(0.01M));
        Console.WriteLine(calculateVatAmount(100.00M));
        Console.WriteLine(calculateVatAmount(110.5M));

        Console.WriteLine(calculateVatAmount(101.5M));
    }

    private decimal calculateVatAmount(decimal subtotal)
    {
        return Math.Round(subtotal * 0.07M,2 ,MidpointRounding.AwayFromZero);
    }



}
```

```
0.00
0.00
7.00
7.74
7.11
```

Lesson Learn : There is a standard which might be ignored in code. If tester or programmer doesn't realize that there is different output partition, we cannot detect unexpected result although we using the test technique.

JSON date string

Requirement :
To search invoice, user want to see all invoices which have invoice date between selected from and to date.

Technical requirement :
Frontend must prepare searching criteria in JSON and submit to backend via REST url : xxxxxx
At the backend, it must **parse JSON payload** and pass it as parameters to database's store procedure

Note : Timezone Asia/Bangkok (GMT+7)

What's test data should be use?

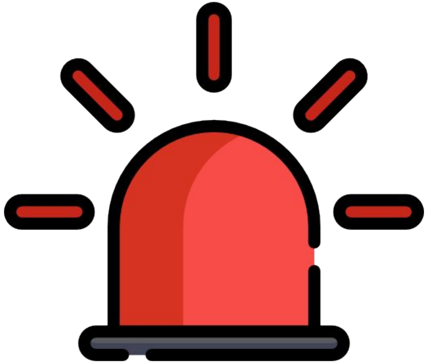
```
{
        "from_date":"2023-09-01T00:00:00.000+07:00",
        "to_date"     :"2023-09-02T00:00:00.000+07:00",
        ....
}
```

JSON date string

Requirement :
To search invoice, user want to see all invoices which have invoice date between selected from and to date.

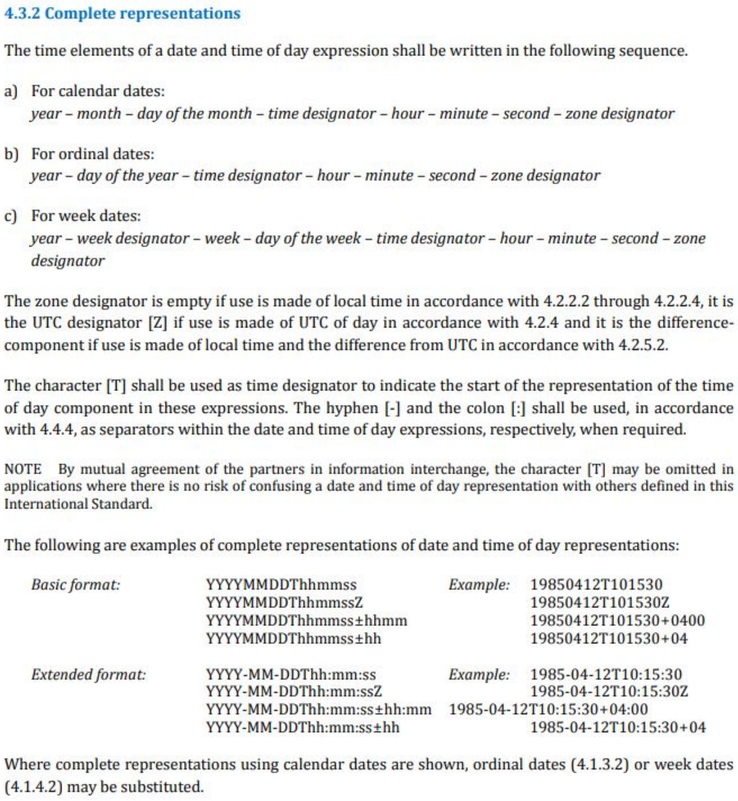
Technical requirement :
Frontend must prepare searching criteria in JSON and submit to backend via REST url : xxxxxx
At the backend, it must **parse JSON payload** and pass it as parameters to database's store procedure

No	Json date	Date
1	"2023-09-01T00:00:00.000+07:00"	Date(2023, 9, 1)
2	"2023-09-01T06:59:59.000+07:00"	Date(2023, 9, 1)
3	"2023-09-01T07:00:00.000+07:00"	Date(2023, 9, 1)
4	"2023-09-01T07:00:01.000+07:00"	Date(2023, 9, 1)
5	"2023-09-01T23:59:59.000+07:00"	Date(2023, 9, 1)
6	??	??
7	??	??
8	??	??



Bug found in production if search data in the early morning.

JSON date string



4.3.2 Complete representations

The time elements of a date and time of day expression shall be written in the following sequence.

a) For calendar dates:
year – month – day of the month – time designator – hour – minute – second – zone designator

b) For ordinal dates:
year – day of the year – time designator – hour – minute – second – zone designator

c) For week dates:
year – week designator – week – day of the week – time designator – hour – minute – second – zone designator

The zone designator is empty if use is made of local time in accordance with 4.2.2.2 through 4.2.2.4, it is the UTC designator [Z] if use is made of UTC of day in accordance with 4.2.4 and it is the difference-component if use is made of local time and the difference from UTC in accordance with 4.2.5.2.

The character [T] shall be used as time designator to indicate the start of the representation of the time of day component in these expressions. The hyphen [-] and the colon [:] shall be used, in accordance with 4.4.4, as separators within the date and time of day expressions, respectively, when required.

NOTE By mutual agreement of the partners in information interchange, the character [T] may be omitted in applications where there is no risk of confusing a date and time of day representation with others defined in this International Standard.

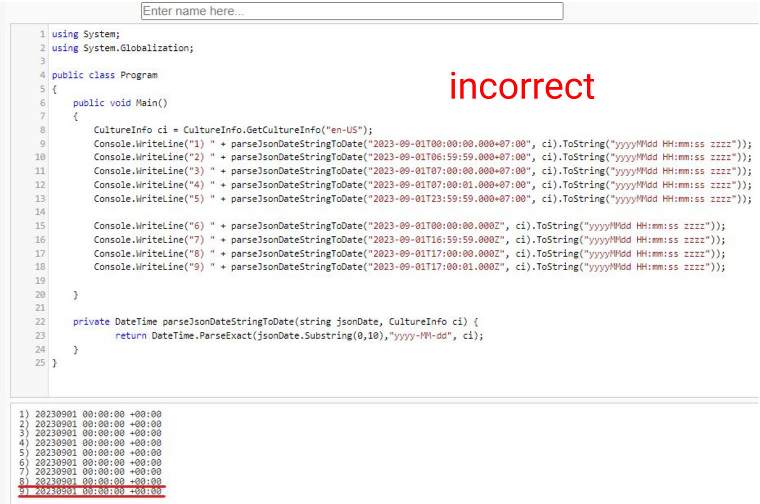
The following are examples of complete representations of date and time of day representations:

Basic format:	YYYYMMDDThhmmss	*Example:*	19850412T101530
	YYYYMMDDThhmmssZ		19850412T101530Z
	YYYYMMDDThhmmss±hhmm		19850412T101530+0400
	YYYYMMDDThhmmss±hh		19850412T101530+04
Extended format:	YYYY-MM-DDThh:mm:ss	*Example:*	1985-04-12T10:15:30
	YYYY-MM-DDThh:mm:ssZ		1985-04-12T10:15:30Z
	YYYY-MM-DDThh:mm:ss±hh:mm	1985-04-12T10:15:30+04:00	
	YYYY-MM-DDThh:mm:ss±hh		1985-04-12T10:15:30+04

Where complete representations using calendar dates are shown, ordinal dates (4.1.3.2) or week dates (4.1.4.2) may be substituted.

No	Json date	Date (with GMT+7)
1	"2023-09-01T00:00:00.000+07:00"	Date(2023, 9, 1)
2	"2023-09-01T06:59:59.000+07:00"	Date(2023, 9, 1)
3	"2023-09-01T07:00:00.000+07:00"	Date(2023, 9, 1)
4	"2023-09-01T07:00:01.000+07:00"	Date(2023, 9, 1)
5	"2023-09-01T23:59:59.000+07:00"	Date(2023, 9, 1)
6	"2023-09-01T00:00:00.000Z"	Date(2023, 9, 1)
7	"2023-09-01T16:59:59.000Z"	Date(2023, 9, 1)
8	"2023-09-01T17:00:00.000Z"	Date(2023, 9, 2)
9	"2023-09-01T17:00:01.000Z"	Date(2023, 9, 2)

JSON date string

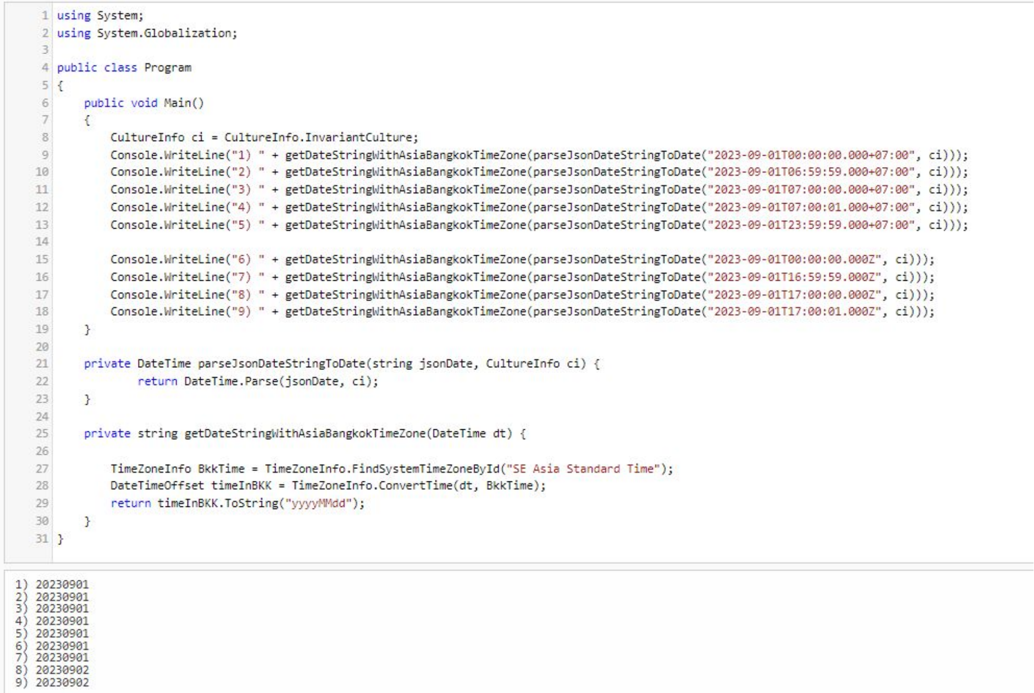


Enter name here...

incorrect

```
using System;
using System.Globalization;

public class Program
{
    public void Main()
    {
        CultureInfo ci = CultureInfo.GetCultureInfo("en-US");
        Console.WriteLine("1) " + parseJsonDateStringToDate("2023-09-01T00:00:00.000+07:00", ci).ToString("yyyyMMdd HH:mm:ss zzzz"));
        Console.WriteLine("2) " + parseJsonDateStringToDate("2023-09-01T06:59:59.000+07:00", ci).ToString("yyyyMMdd HH:mm:ss zzzz"));
        Console.WriteLine("3) " + parseJsonDateStringToDate("2023-09-01T07:00:00.000+07:00", ci).ToString("yyyyMMdd HH:mm:ss zzzz"));
        Console.WriteLine("4) " + parseJsonDateStringToDate("2023-09-01T07:00:01.000+07:00", ci).ToString("yyyyMMdd HH:mm:ss zzzz"));
        Console.WriteLine("5) " + parseJsonDateStringToDate("2023-09-01T23:59:59.000+07:00", ci).ToString("yyyyMMdd HH:mm:ss zzzz"));

        Console.WriteLine("6) " + parseJsonDateStringToDate("2023-09-01T00:00:00.000Z", ci).ToString("yyyyMMdd HH:mm:ss zzzz"));
        Console.WriteLine("7) " + parseJsonDateStringToDate("2023-09-01T16:59:59.000Z", ci).ToString("yyyyMMdd HH:mm:ss zzzz"));
        Console.WriteLine("8) " + parseJsonDateStringToDate("2023-09-01T17:00:00.000Z", ci).ToString("yyyyMMdd HH:mm:ss zzzz"));
        Console.WriteLine("9) " + parseJsonDateStringToDate("2023-09-01T17:00:01.000Z", ci).ToString("yyyyMMdd HH:mm:ss zzzz"));

    }

    private DateTime parseJsonDateStringToDate(string jsonDate, CultureInfo ci) {
            return DateTime.ParseExact(jsonDate.Substring(0,10),"yyyy-MM-dd", ci);
    }
}
```

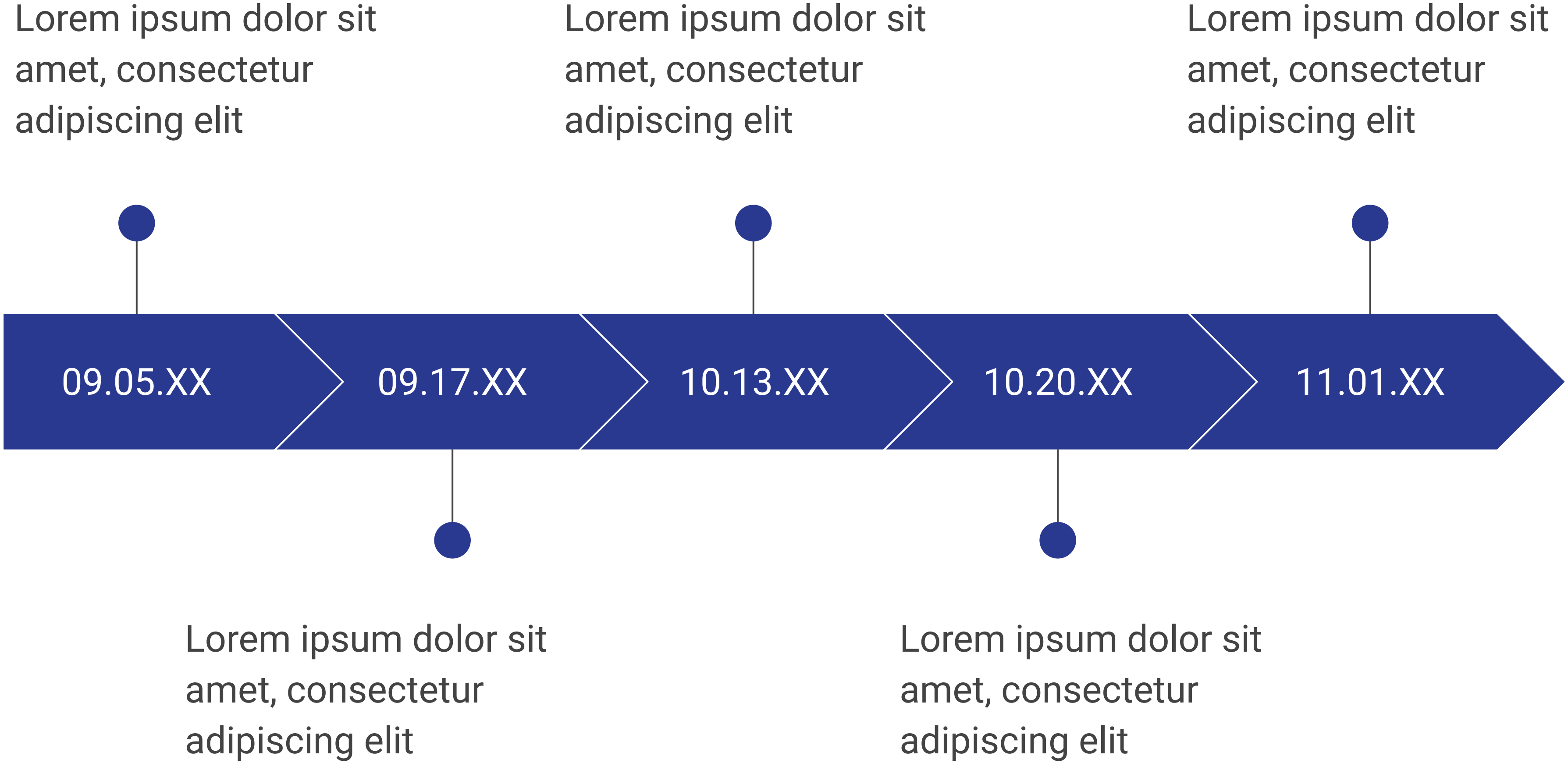
```
1) 20230901 00:00:00 +00:00
2) 20230901 00:00:00 +00:00
3) 20230901 00:00:00 +00:00
4) 20230901 00:00:00 +00:00
5) 20230901 00:00:00 +00:00
6) 20230901 00:00:00 +00:00
7) 20230901 00:00:00 +00:00
8) 20230901 00:00:00 +00:00
9) 20230901 00:00:00 +00:00
```



```
using System;
using System.Globalization;

public class Program
{
    public void Main()
    {
        CultureInfo ci = CultureInfo.InvariantCulture;
        Console.WriteLine("1) " + getDateStringWithAsiaBangkokTimeZone(parseJsonDateStringToDate("2023-09-01T00:00:00.000+07:00", ci)));
        Console.WriteLine("2) " + getDateStringWithAsiaBangkokTimeZone(parseJsonDateStringToDate("2023-09-01T06:59:59.000+07:00", ci)));
        Console.WriteLine("3) " + getDateStringWithAsiaBangkokTimeZone(parseJsonDateStringToDate("2023-09-01T07:00:00.000+07:00", ci)));
        Console.WriteLine("4) " + getDateStringWithAsiaBangkokTimeZone(parseJsonDateStringToDate("2023-09-01T07:00:01.000+07:00", ci)));
        Console.WriteLine("5) " + getDateStringWithAsiaBangkokTimeZone(parseJsonDateStringToDate("2023-09-01T23:59:59.000+07:00", ci)));

        Console.WriteLine("6) " + getDateStringWithAsiaBangkokTimeZone(parseJsonDateStringToDate("2023-09-01T00:00:00.000Z", ci)));
        Console.WriteLine("7) " + getDateStringWithAsiaBangkokTimeZone(parseJsonDateStringToDate("2023-09-01T16:59:59.000Z", ci)));
        Console.WriteLine("8) " + getDateStringWithAsiaBangkokTimeZone(parseJsonDateStringToDate("2023-09-01T17:00:00.000Z", ci)));
        Console.WriteLine("9) " + getDateStringWithAsiaBangkokTimeZone(parseJsonDateStringToDate("2023-09-01T17:00:01.000Z", ci)));
    }

    private DateTime parseJsonDateStringToDate(string jsonDate, CultureInfo ci) {
        return DateTime.Parse(jsonDate, ci);
    }

    private string getDateStringWithAsiaBangkokTimeZone(DateTime dt) {

        TimeZoneInfo BkkTime = TimeZoneInfo.FindSystemTimeZoneById("SE Asia Standard Time");
        DateTimeOffset timeInBKK = TimeZoneInfo.ConvertTime(dt, BkkTime);
        return timeInBKK.ToString("yyyyMMdd");
    }
}
```

```
1) 20230901
2) 20230901
3) 20230901
4) 20230901
5) 20230901
6) 20230901
7) 20230901
8) 20230902
9) 20230902
```

Lesson Learn : There is a standard to parse date string format. Use JSON serialize/deserialize library or parsing date string and converting with standard aware.



09.05.XX

Lorem ipsum dolor sit amet, consectetur adipiscing elit

09.17.XX

Lorem ipsum dolor sit amet, consectetur adipiscing elit

10.13.XX

Lorem ipsum dolor sit amet, consectetur adipiscing elit

10.20.XX

Lorem ipsum dolor sit amet, consectetur adipiscing elit

11.01.XX

Lorem ipsum dolor sit amet, consectetur adipiscing elit

Payment QR

Requirement :

To make an payment by user, billing app must create Payment QR which able to work with Banking mobile application and customer able to use the QR to make payment by their mobile.

Technical Require :

Program must calculate due amount by Store Procecure and create data into the format of banking app.

What’s test data should be use?

```
{
        "QR Data":"xxxxxxxxxxxxxxxxxxxxxxxxx",
}
```

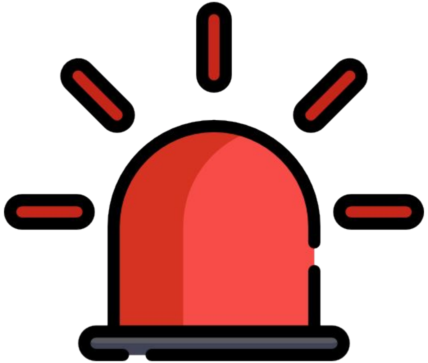
Payment QR

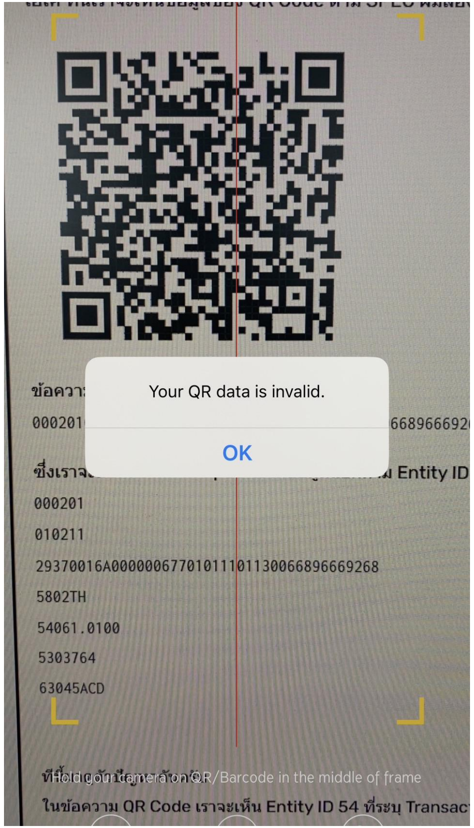
Requirement :
To make an payment by user, billing app must create Payment QR which able to work with Banking mobile application and customer able to use the QR to make payment by their mobile.

Technical Require :

Program must calculate due amount by Store Procecure and create data into the format of banking app.

Bug found in production.
We can make payment to all Bank except KMA.





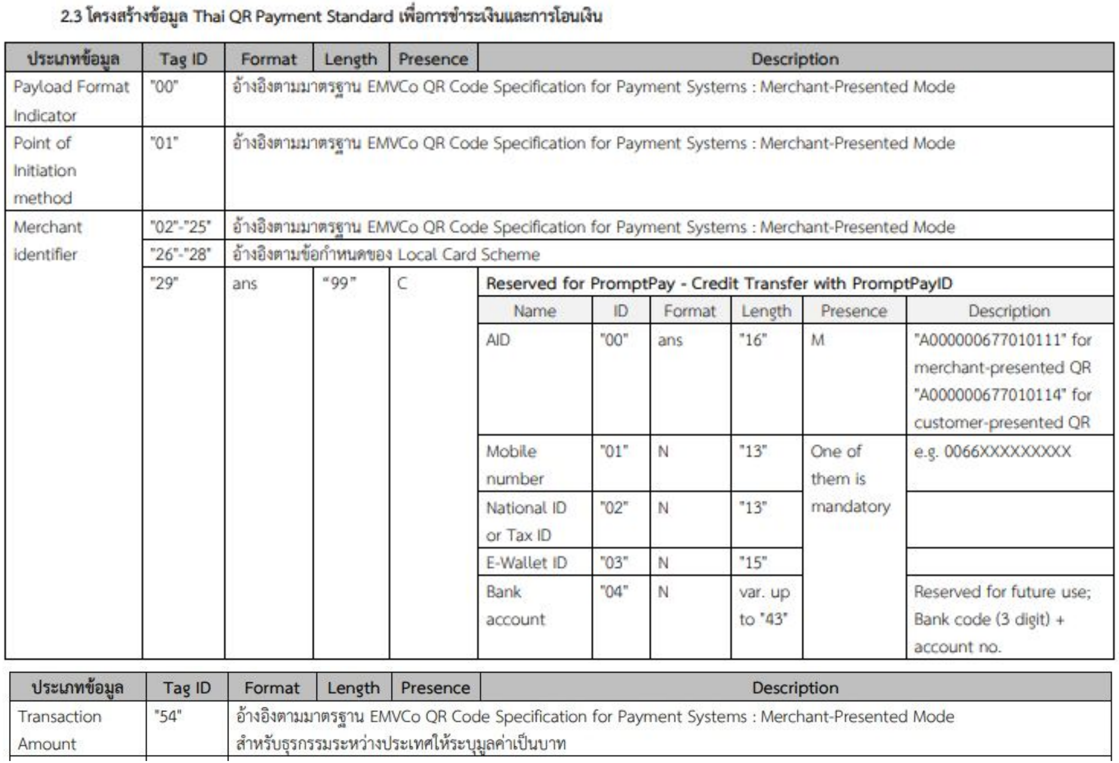
Your QR data is invalid.

ข้อควา
000201 668966692
ซึ่งเราจ Entity ID
000201
010211
29370016A000000677010111011300668966669268
5802TH
54061.0100
5303764
63045ACD

Hold your camera on QR/Barcode in the middle of frame

ในข้อความ QR Code เราจะเห็น Entity ID 54 ที่ระบุ Transac

Payment QR



2.3 โครงสร้างข้อมูล Thai QR Payment Standard เพื่อการชำระเงินและการโอนเงิน

<table>
<tr><th>ประเภทข้อมูล</th><th>Tag ID</th><th>Format</th><th>Length</th><th>Presence</th><th>Description</th></tr>
<tr><td>Payload Format Indicator</td><td>"00"</td><td colspan="4">อ้างอิงตามมาตรฐาน EMVCo QR Code Specification for Payment Systems : Merchant-Presented Mode</td></tr>
<tr><td>Point of Initiation method</td><td>"01"</td><td colspan="4">อ้างอิงตามมาตรฐาน EMVCo QR Code Specification for Payment Systems : Merchant-Presented Mode</td></tr>
<tr><td rowspan="3">Merchant identifier</td><td>"02"-"25"</td><td colspan="4">อ้างอิงตามมาตรฐาน EMVCo QR Code Specification for Payment Systems : Merchant-Presented Mode</td></tr>
<tr><td>"26"-"28"</td><td colspan="4">อ้างอิงตามข้อกำหนดของ Local Card Scheme</td></tr>
<tr><td>"29"</td><td>ans</td><td>"99"</td><td>C</td><td><b>Reserved for PromptPay - Credit Transfer with PromptPayID</b>
<table>
<tr><th>Name</th><th>ID</th><th>Format</th><th>Length</th><th>Presence</th><th>Description</th></tr>
<tr><td>AID</td><td>"00"</td><td>ans</td><td>"16"</td><td>M</td><td>"A000000677010111" for merchant-presented QR "A000000677010114" for customer-presented QR</td></tr>
<tr><td>Mobile number</td><td>"01"</td><td>N</td><td>"13"</td><td rowspan="4">One of them is mandatory</td><td>e.g. 0066XXXXXXXXX</td></tr>
<tr><td>National ID or Tax ID</td><td>"02"</td><td>N</td><td>"13"</td><td></td></tr>
<tr><td>E-Wallet ID</td><td>"03"</td><td>N</td><td>"15"</td><td></td></tr>
<tr><td>Bank account</td><td>"04"</td><td>N</td><td>var. up to "43"</td><td>Reserved for future use; Bank code (3 digit) + account no.</td></tr>
</table></td></tr>
</table>

ประเภทข้อมูล	Tag ID	Format	Length	Presence	Description
Transaction Amount	"54"	อ้างอิงตามมาตรฐาน EMVCo QR Code Specification for Payment Systems : Merchant-Presented Mode สำหรับธุรกรรมระหว่างประเทศให้ระบุมูลค่าเป็นบาท			

Payment QR

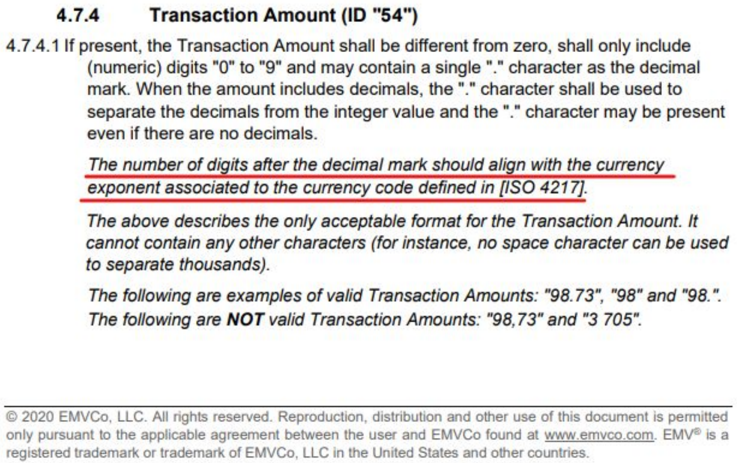
QRData :

000201**01**0211**29**370016A000000677010111011300668966692685
5802TH**54**061.0100**53**03764**63**045ACD

Explain

000201
010211
29370016A0000006770101110113006689666926 8
5802TH
54061.0100 ←- 54 (Transaction amount has 6 length) = 1.0100
5303764 ←- 53 (Currency number) = 764 (Thai bath)
63045ACD

Payment QR



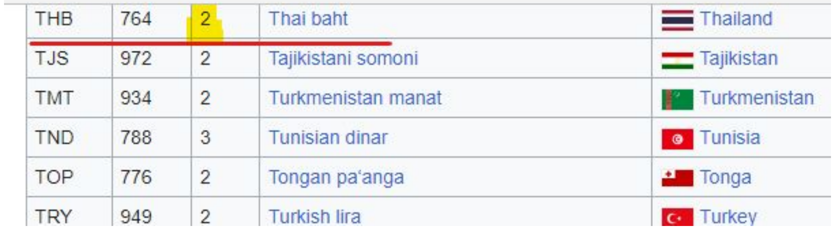
4.7.4 Transaction Amount (ID "54")

4.7.4.1 If present, the Transaction Amount shall be different from zero, shall only include (numeric) digits "0" to "9" and may contain a single "." character as the decimal mark. When the amount includes decimals, the "." character shall be used to separate the decimals from the integer value and the "." character may be present even if there are no decimals.

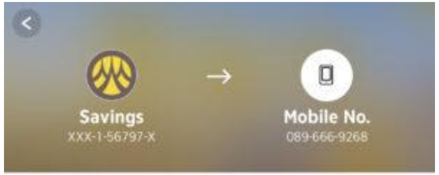
The number of digits after the decimal mark should align with the currency exponent associated to the currency code defined in [ISO 4217].

The above describes the only acceptable format for the Transaction Amount. It cannot contain any other characters (for instance, no space character can be used to separate thousands).

*The following are examples of valid Transaction Amounts: "98.73", "98" and "98.". The following are **NOT** valid Transaction Amounts: "98,73" and "3 705".*



THB	764	2	Thai baht	Thailand
TJS	972	2	Tajikistani somoni	Tajikistan
TMT	934	2	Turkmenistan manat	Turkmenistan
TND	788	3	Tunisian dinar	Tunisia
TOP	776	2	Tongan pa'anga	Tonga
TRY	949	2	Turkish lira	Turkey



Savings
XXX-1-56797-X

→

Mobile No.
089-666-9268

00020101021129370016A000000677010111011300 6
6896669268 5802TH54041.025303764 6304B09E

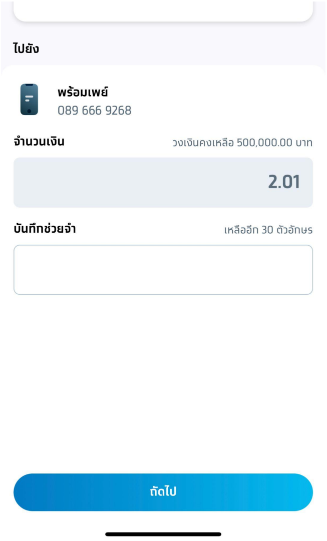
Amount

1.02 THB

Payment QR

What’s behaviour of other bank’s app if…?

```
00020101021129370016A000000677010111011300668
966692685802TH54062.00515303764630453331
```



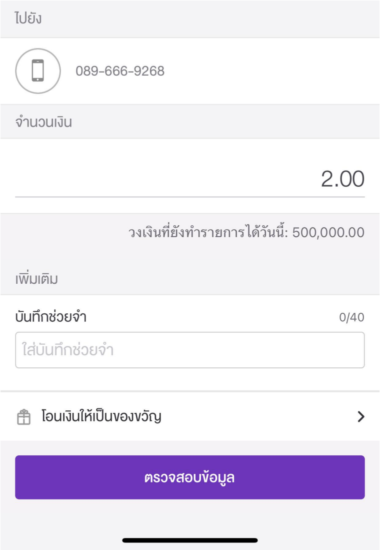
ไปยัง

พร้อมเพย์
089 666 9268

จำนวนเงิน วงเงินคงเหลือ 500,000.00 บาท

2.01

บันทึกช่วยจำ เหลืออีก 30 ตัวอักษร



ไปยัง

089-666-9268

จำนวนเงิน

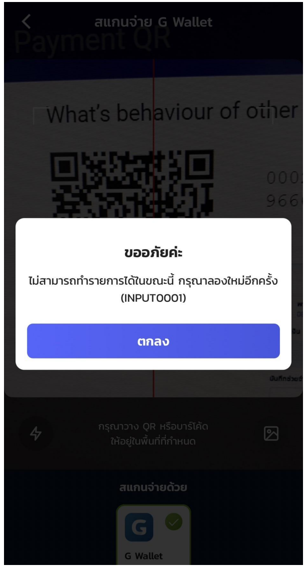
2.00

วงเงินที่ยังทำรายการได้วันนี้: 500,000.00

เพิ่มเติม

บันทึกช่วยจำ 0/40

ใส่บันทึกช่วยจำ



ขออภัยค่ะ

ไม่สามารถทำรายการได้ในขณะนี้ กรุณาลองใหม่อีกครั้ง (INPUT0001)

References

1. IEEE 754 https://en.wikipedia.org/wiki/IEEE_754
2. .net 7.0 Math.Round method Math.Round Method (System) | Microsoft Learn
3. ISO 8601 ISO 8601 - Wikipedia
4. 3 ways to work with dates in JSON JSON date format: 3 ways to work with dates in JSON (jsoneditoronline.org)
5. Thai standard payment https://www.bot.or.th/content/dam/bot/fipcs/documents/FPG/2562/ThaiPDF/25620084.pdf
6. EVMCo QR Code https://www.emvco.com/emv-technologies/qr-codes/
7. theerasak.com theerasak.com

Thank you