



SOFTWARE ENGINEERING

Software Architecture

Dr. Rathachai Chawuthai

Department of Computer Engineering

Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

Agenda

- Software Architecture
- Layered Architecture
- Client-Server Model
- 3-Tier Architecture
- Service-Oriented Architecture (SOA)
- Model-View-Controller (MVC)
- Microservice
- Distributed Systems

Software Architecture

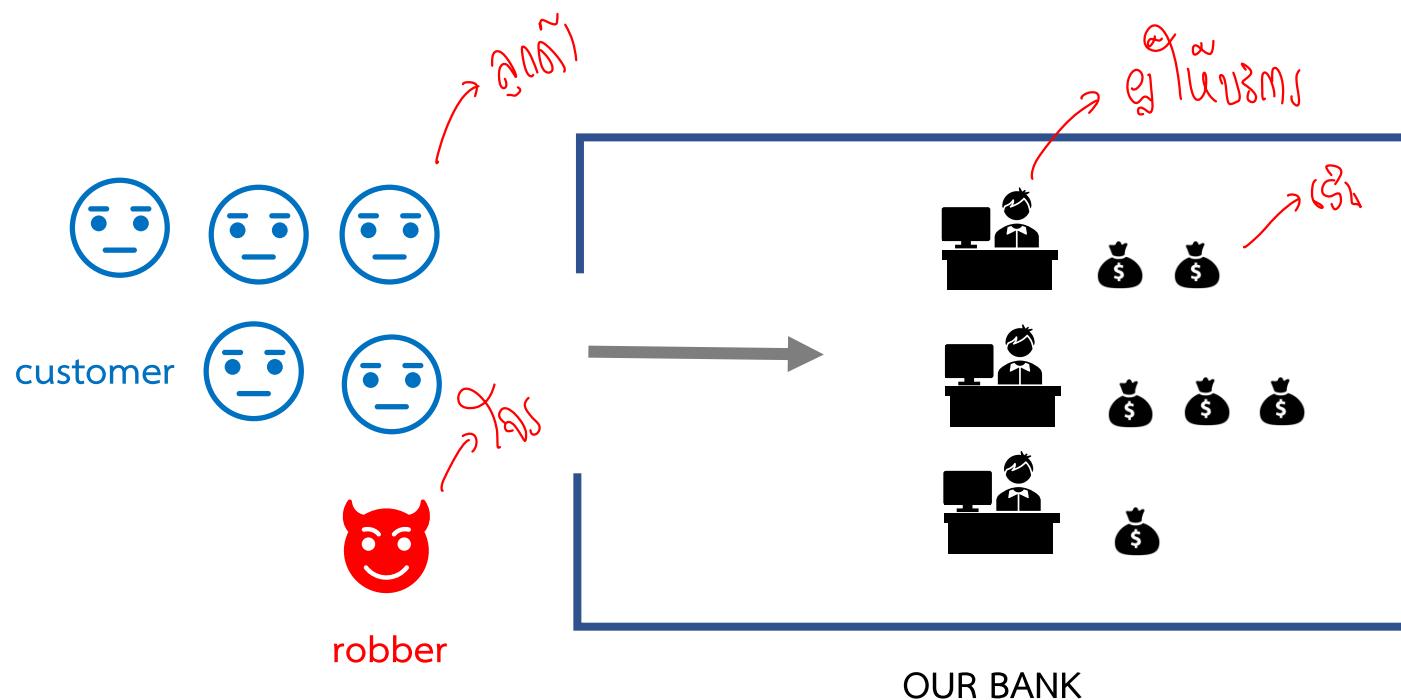
At a Bank

የፌዴራል ደንብ አንድ ትምህር ነው?

ስኔጻዊ-ሙሉ-ይሆኑን ማረጋገጫዎች

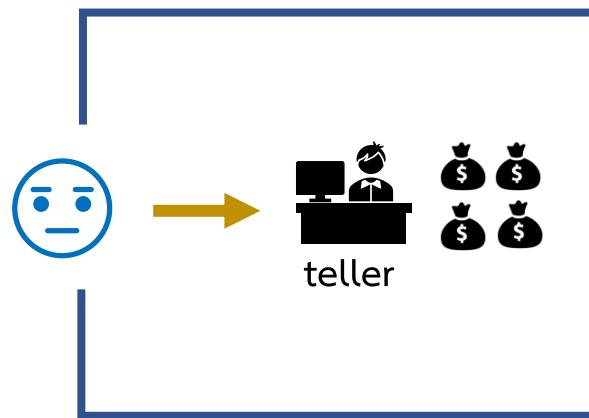


At a Bank



At a Bank

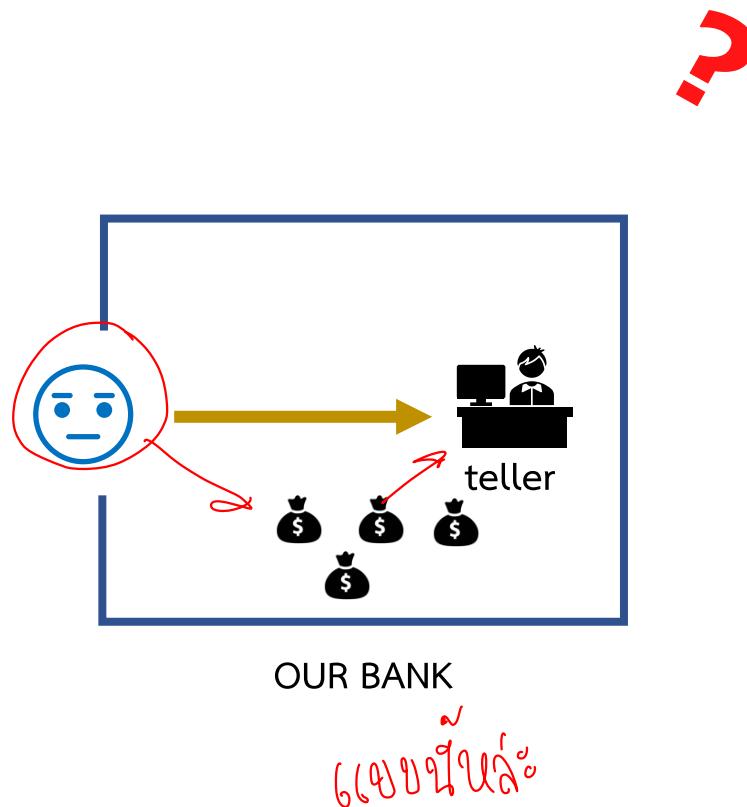
မင်္ဂလာနည်းလုပ်



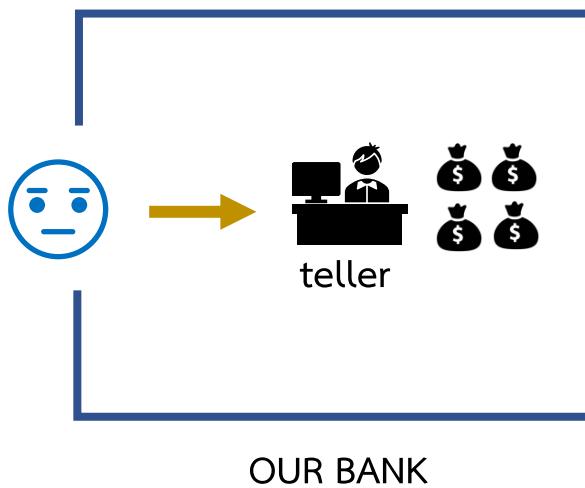
OUR BANK

ဘယ်နဲ့ ဝေယော ?

At a Bank

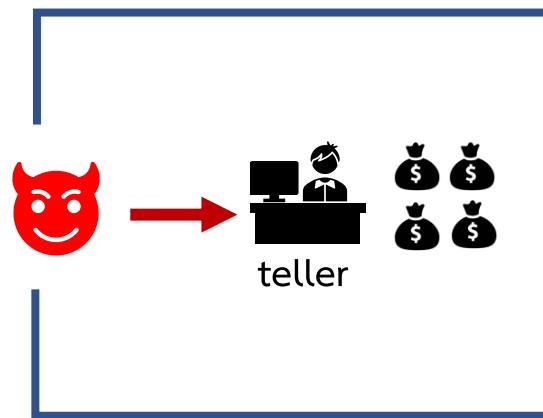


At a Bank



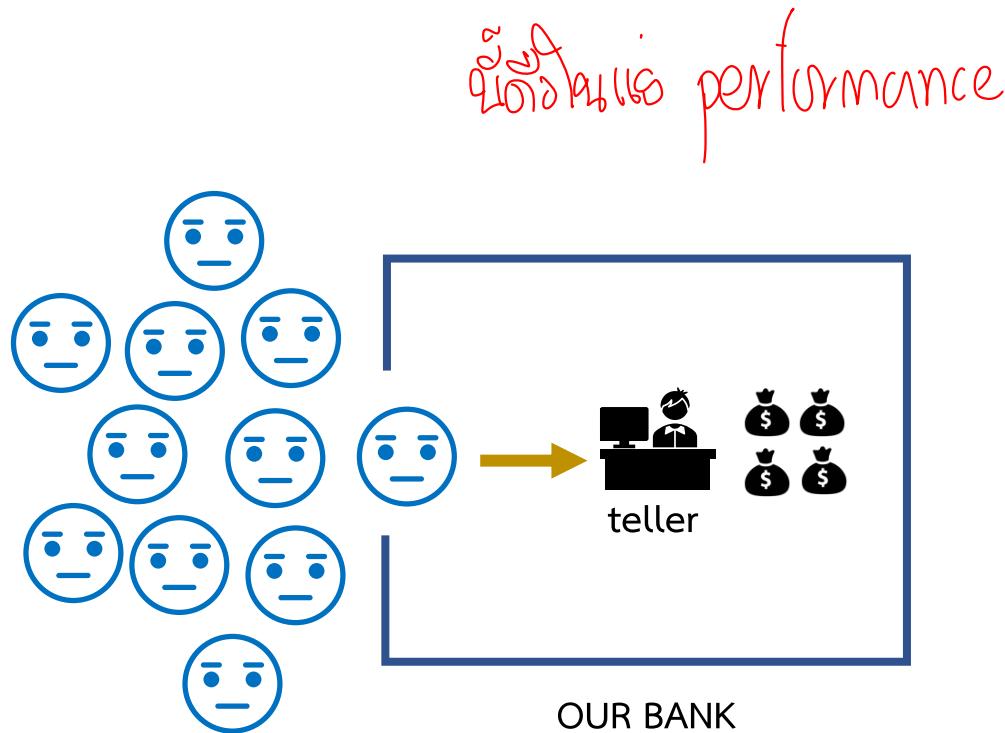
ເບີນທີ່ຈະໄດ້ຮັບຊາຍຈຳກັດ

At a Bank



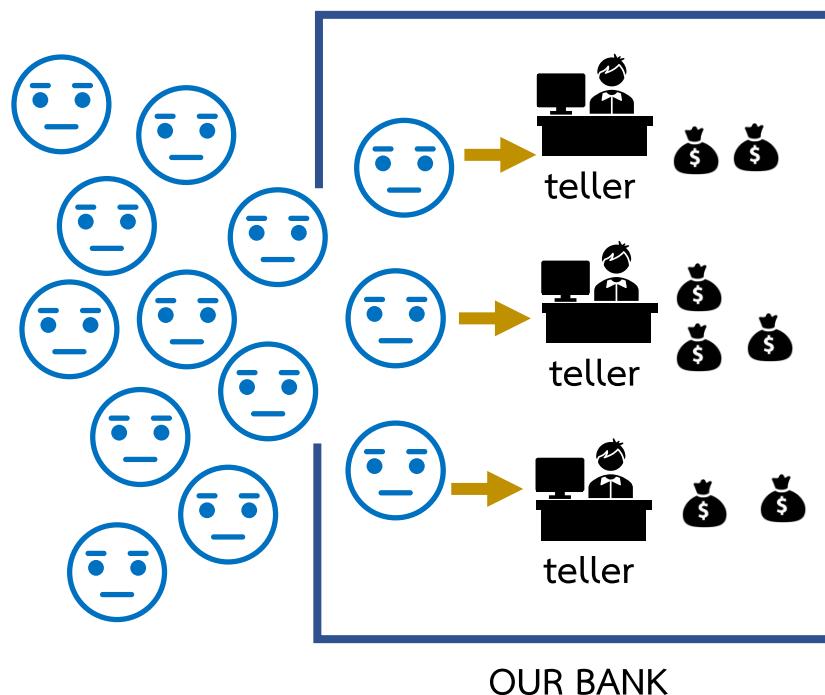
ບໍລິສັດທະນາ non-functional
↳ security → ອຳນັດໃຫຍ່
performance → ອຳນັດເວັບໄວ້

At a Bank



At a Bank

សារមិនអាមេរិកលើចំណែកជាន់ទៅ

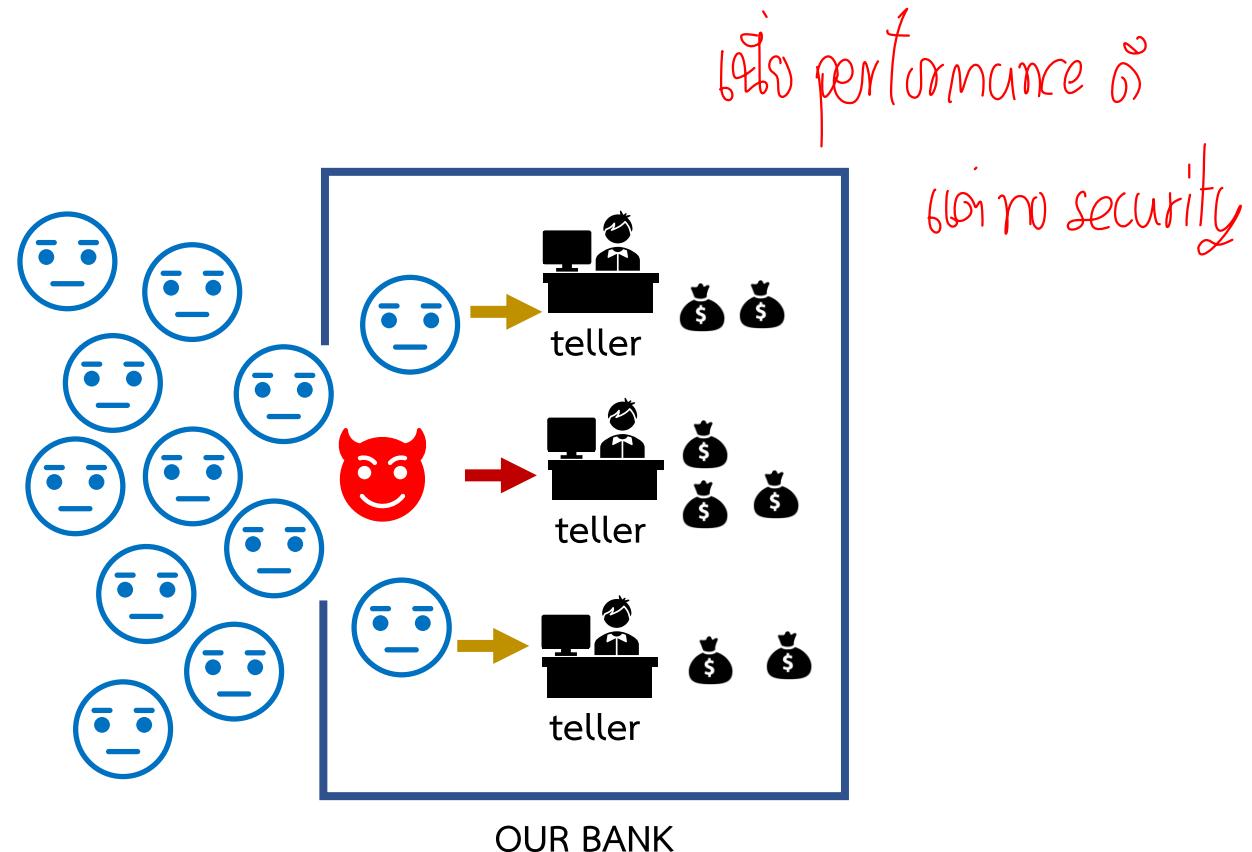


ចិត្តមិនអាមេរិកលើចំណែកជាន់ទៅ

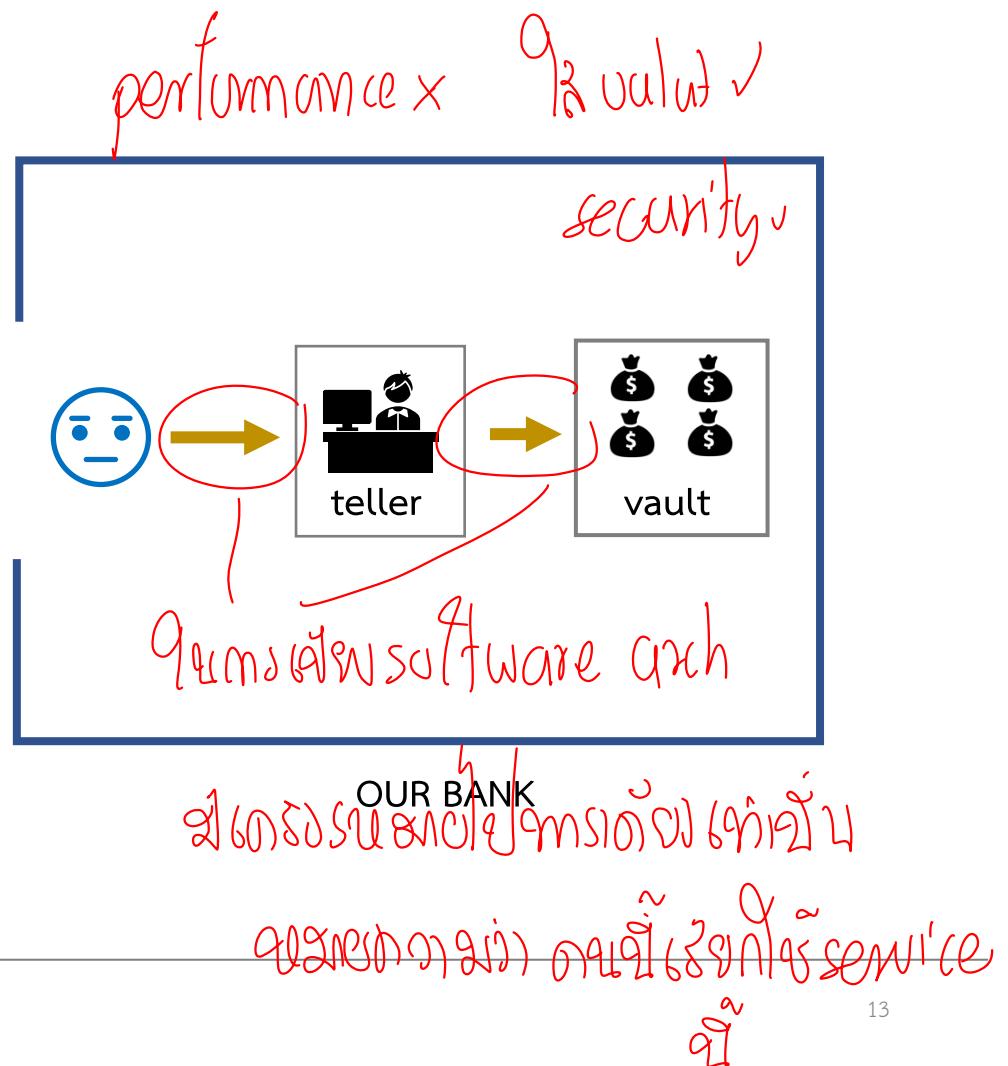
បាន

ចិត្តមិនអាមេរិកលើចំណែកជាន់ទៅ

At a Bank

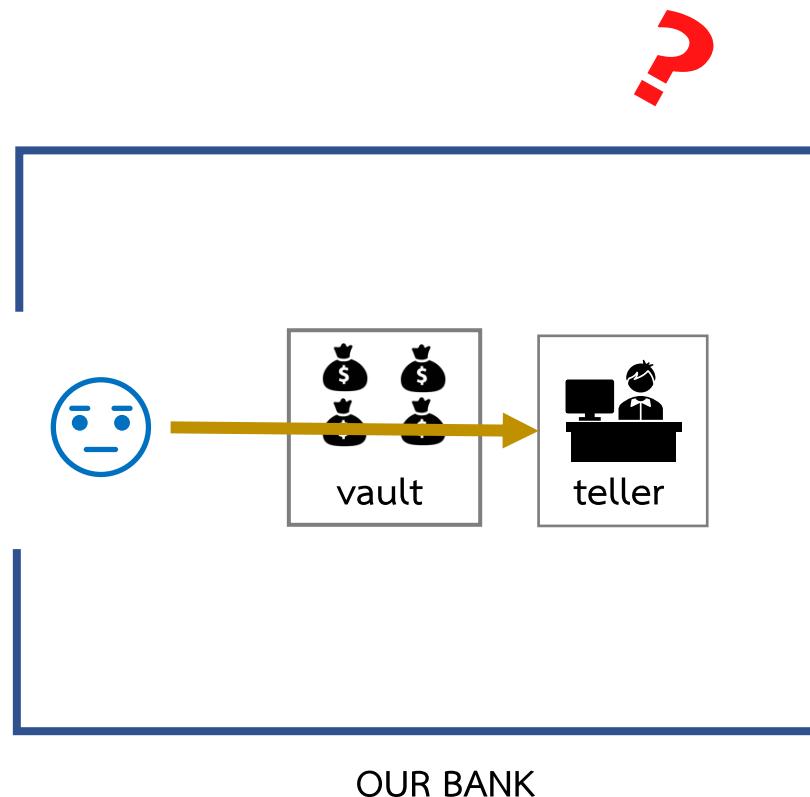


At a Bank

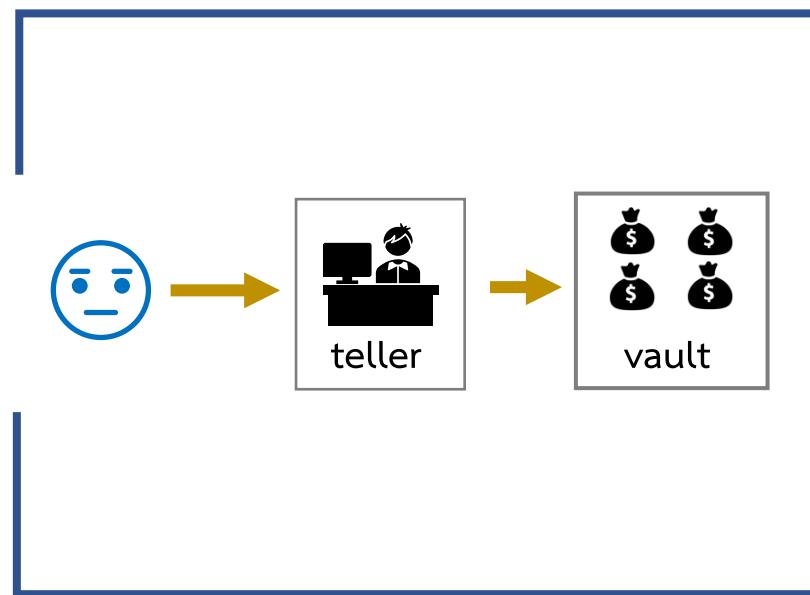


At a Bank

ຕຳແໜ່ງອະນຸມາດ (API requests)

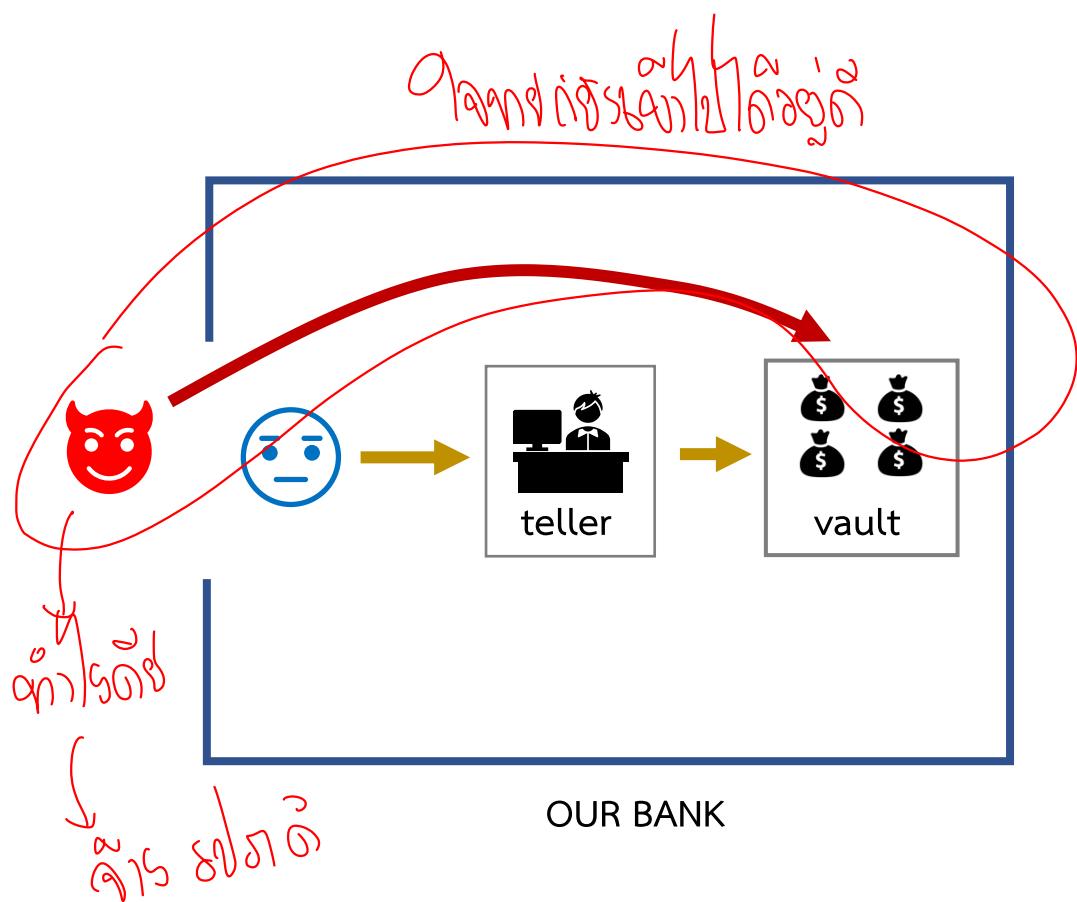


At a Bank

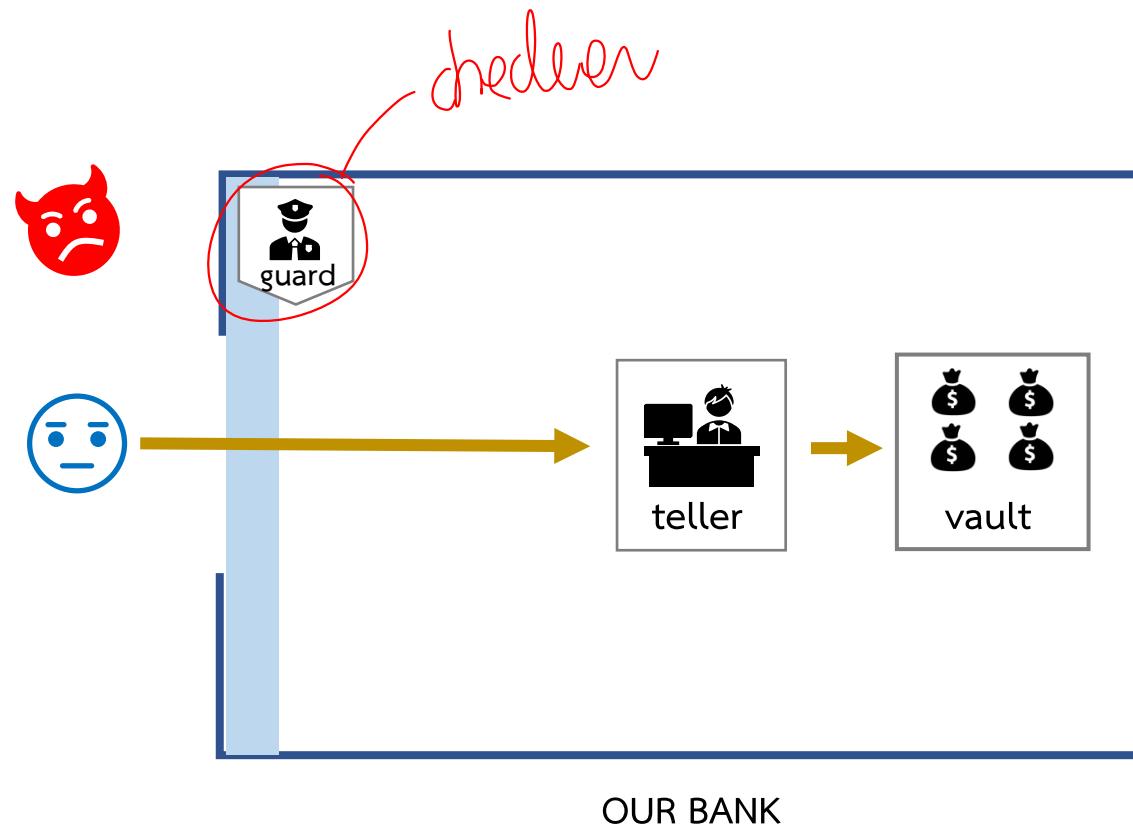


OUR BANK

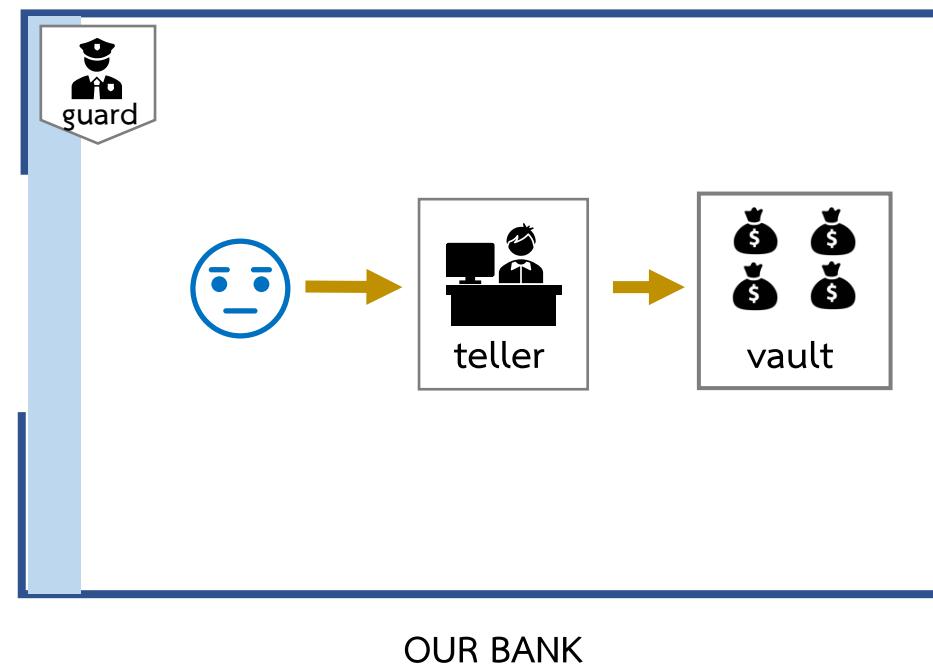
At a Bank



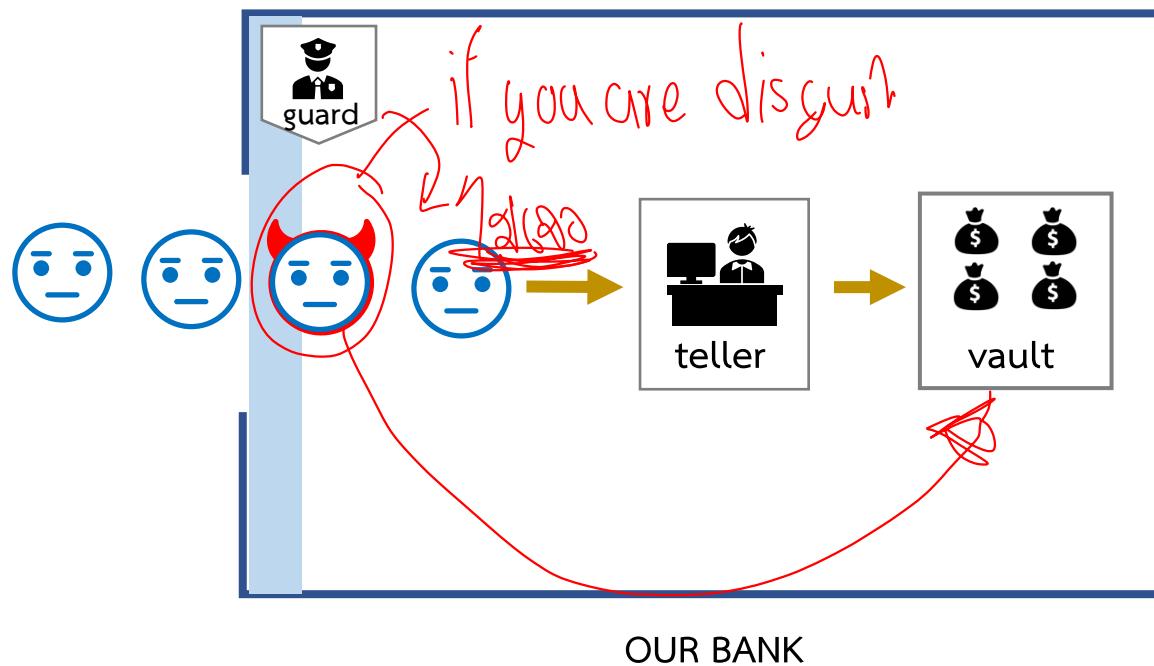
At a Bank



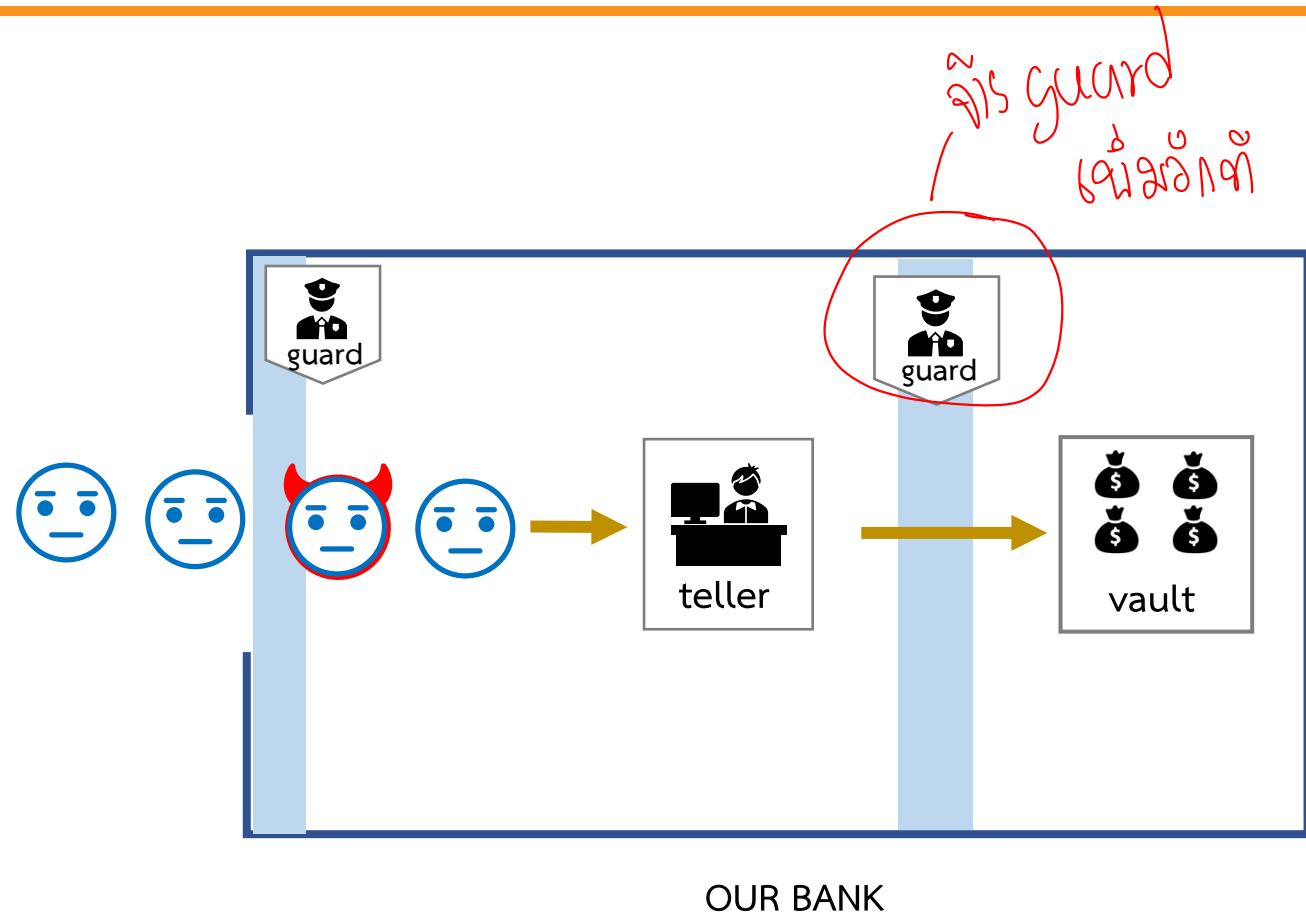
At a Bank



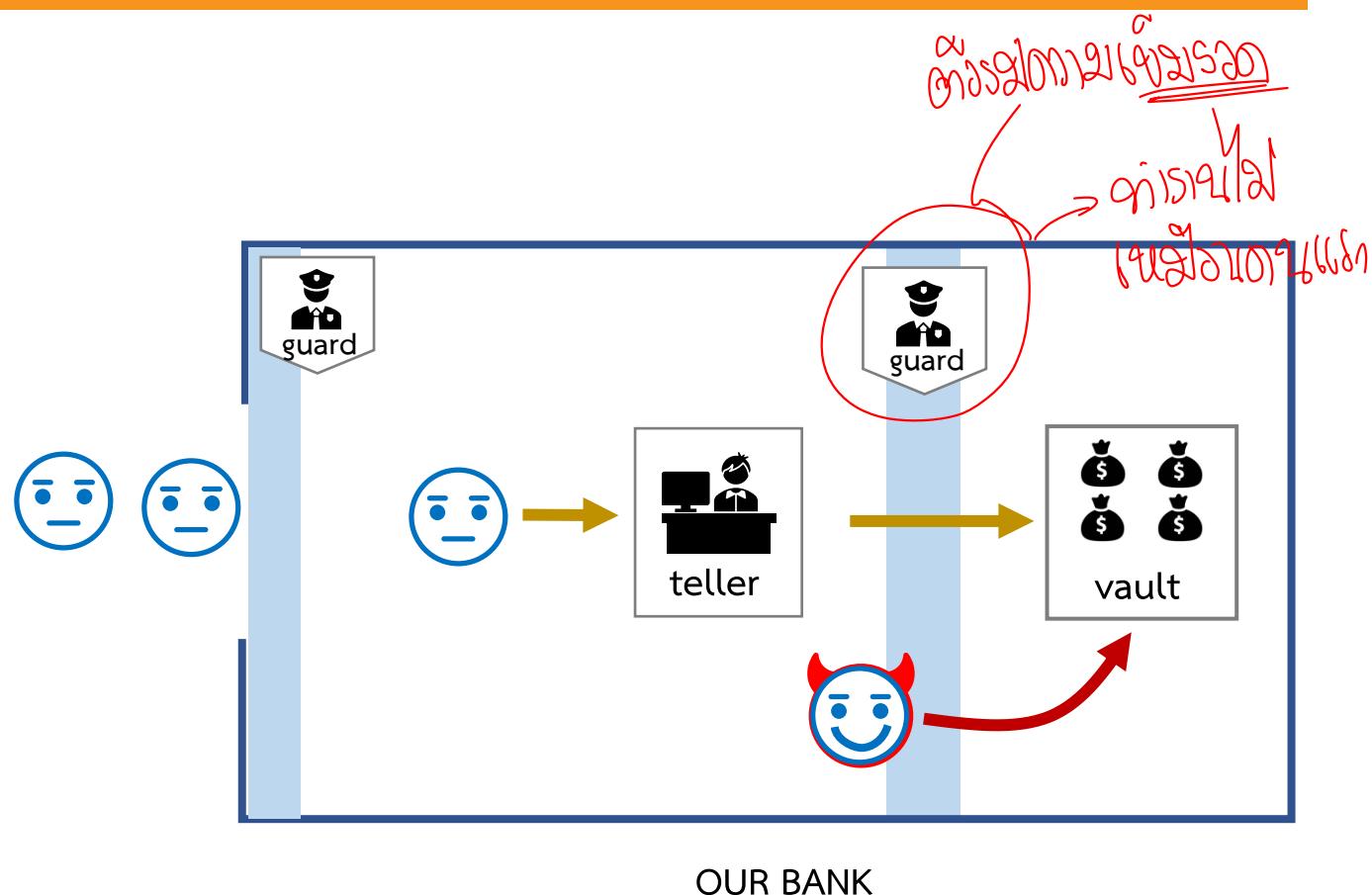
At a Bank



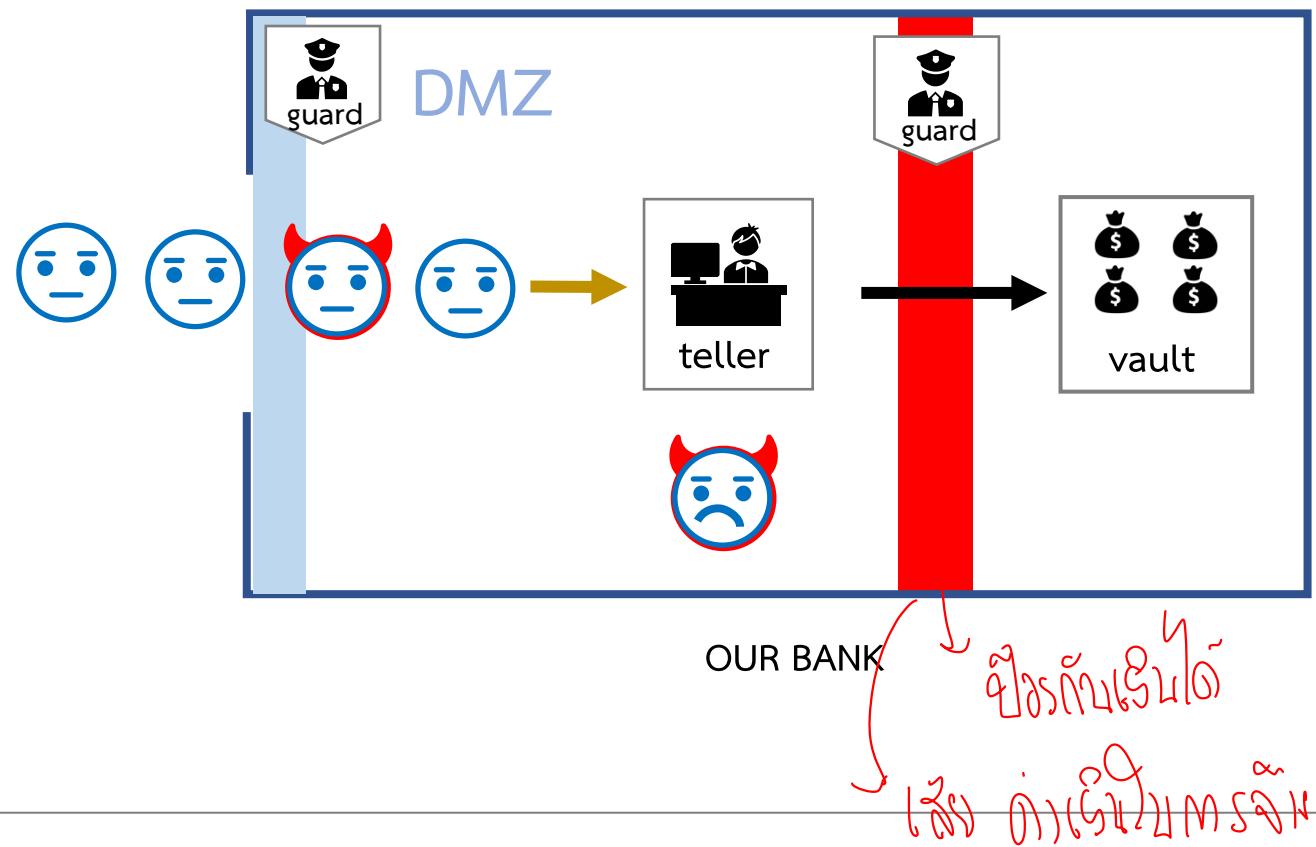
At a Bank



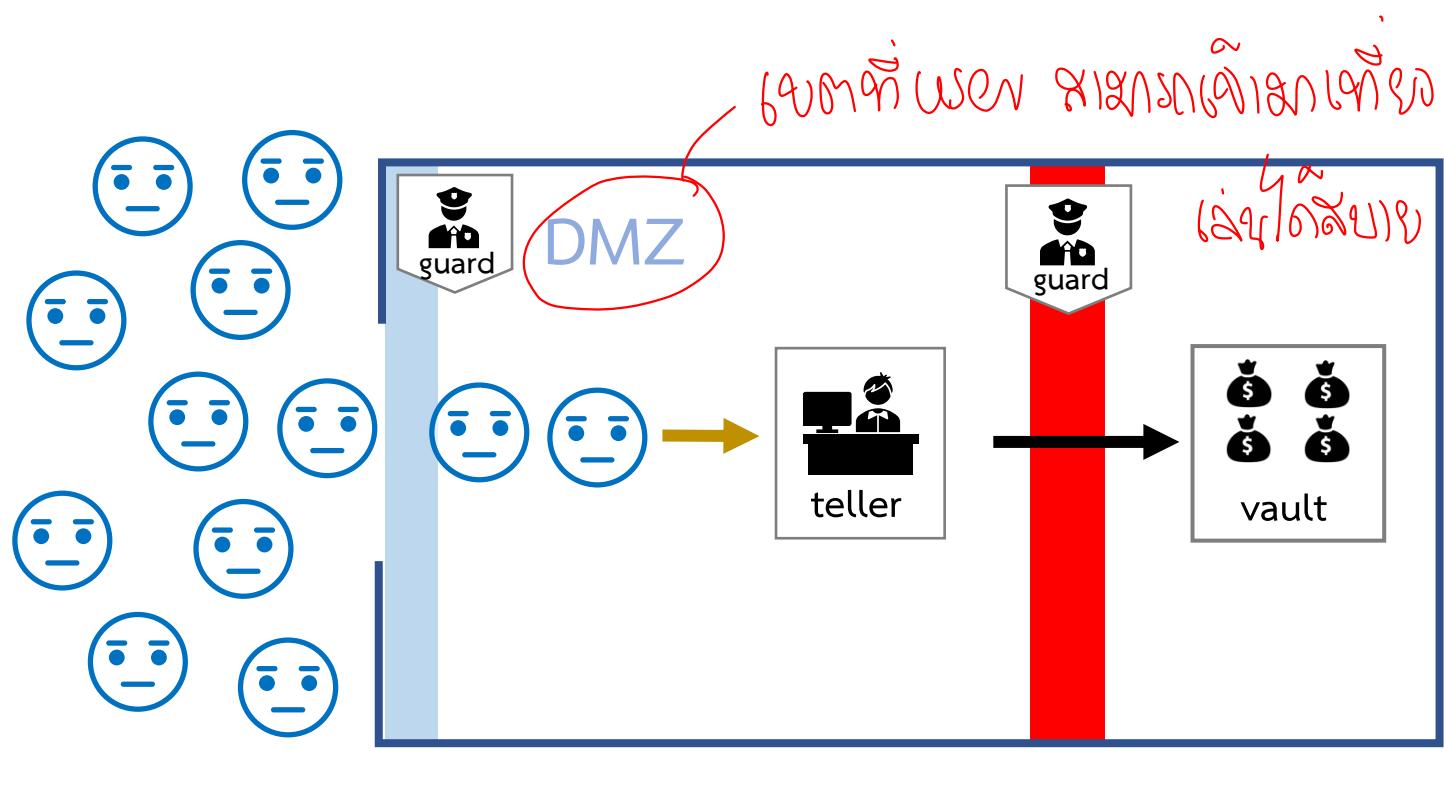
At a Bank



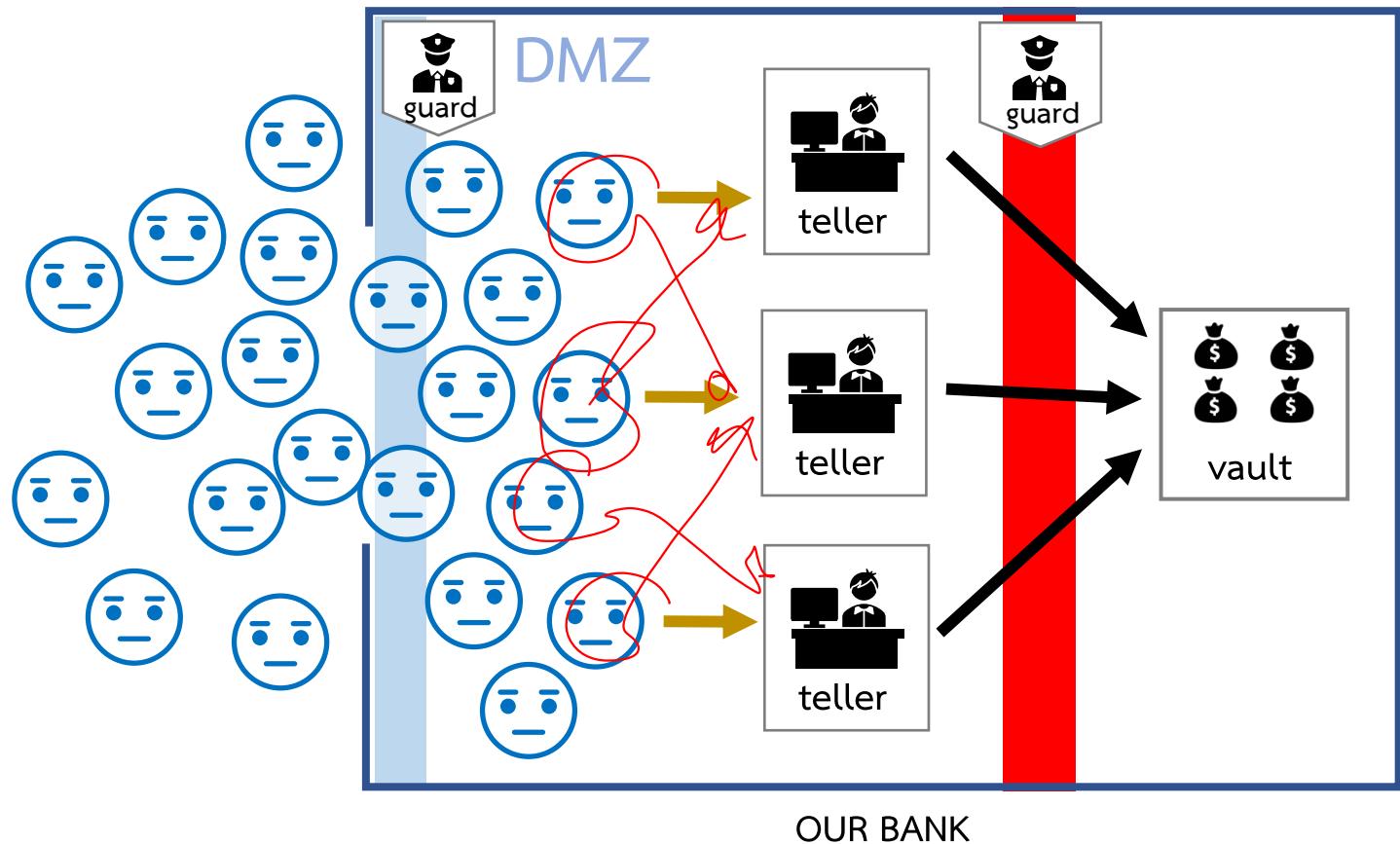
At a Bank



At a Bank

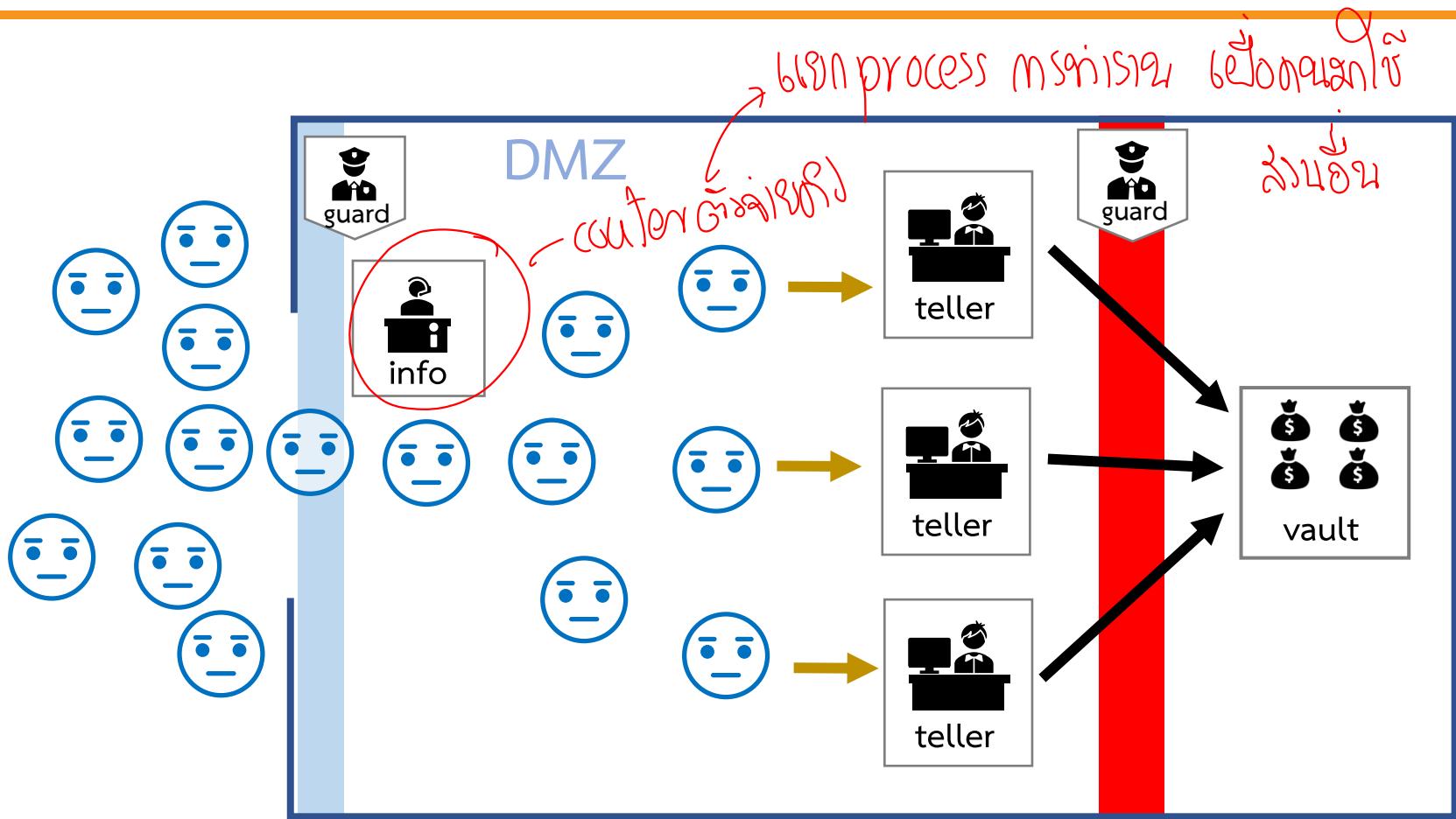


At a Bank

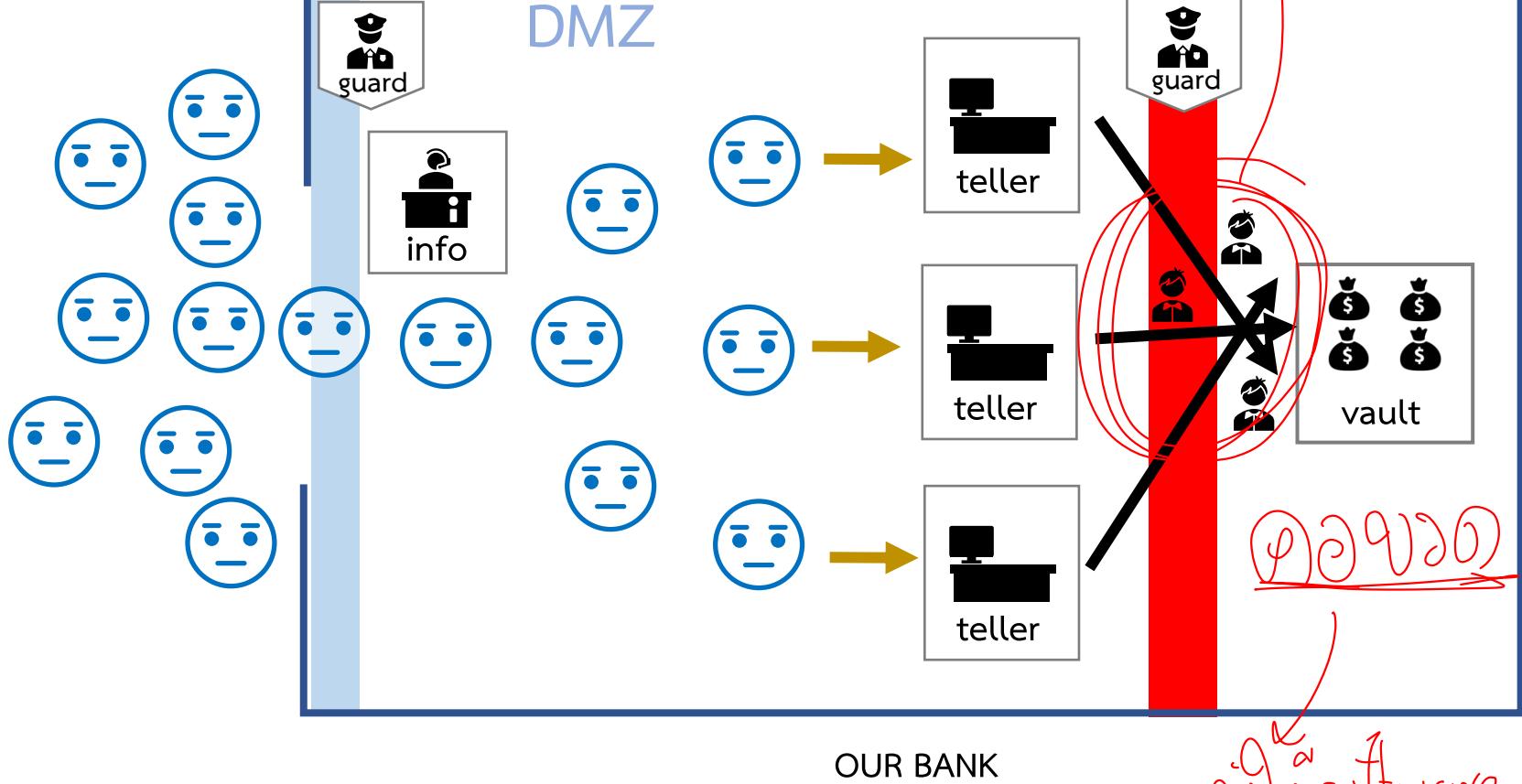


ເມືດຈະ Queue ອັດວ?

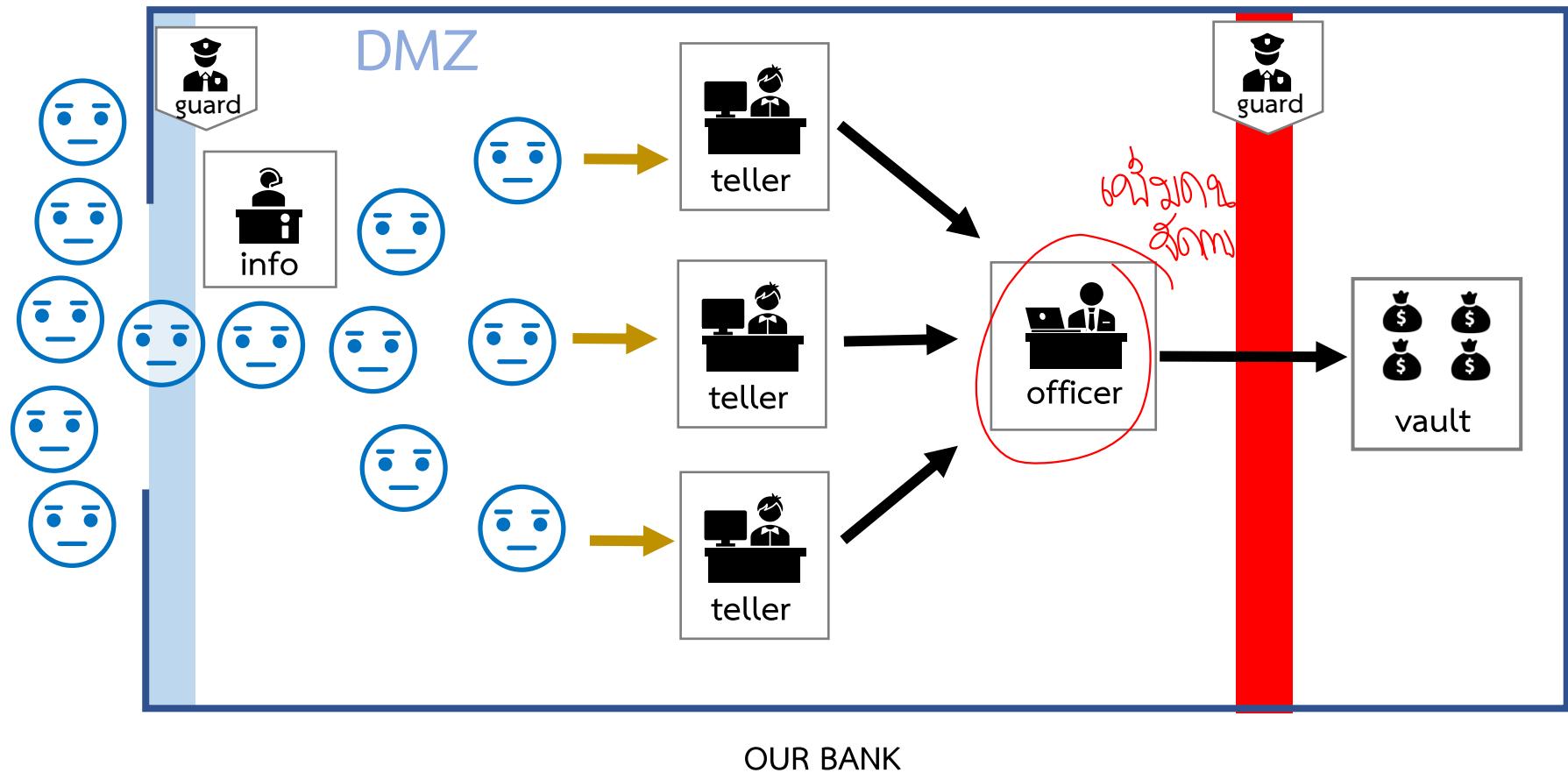
At a Bank



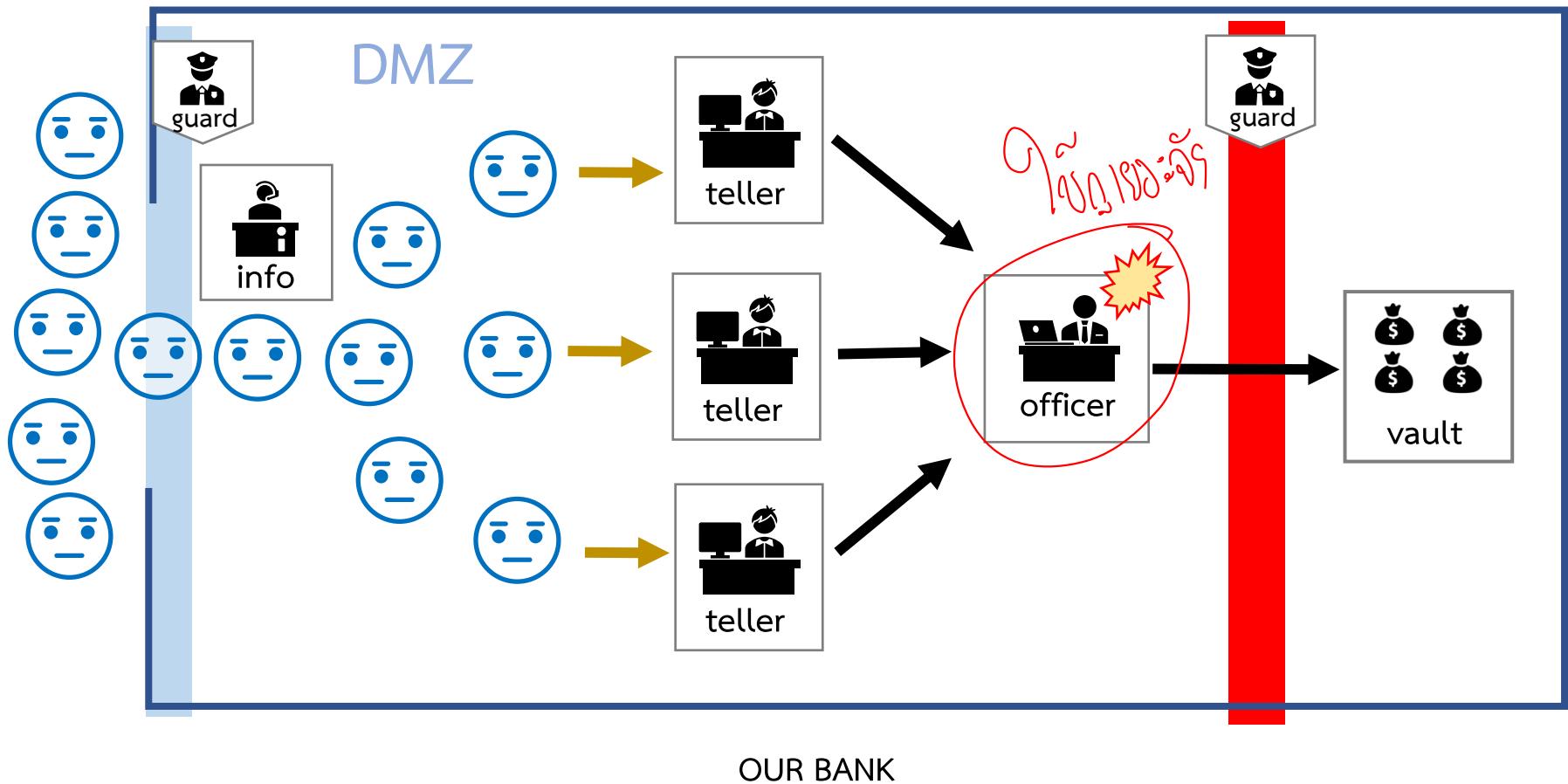
At a Bank



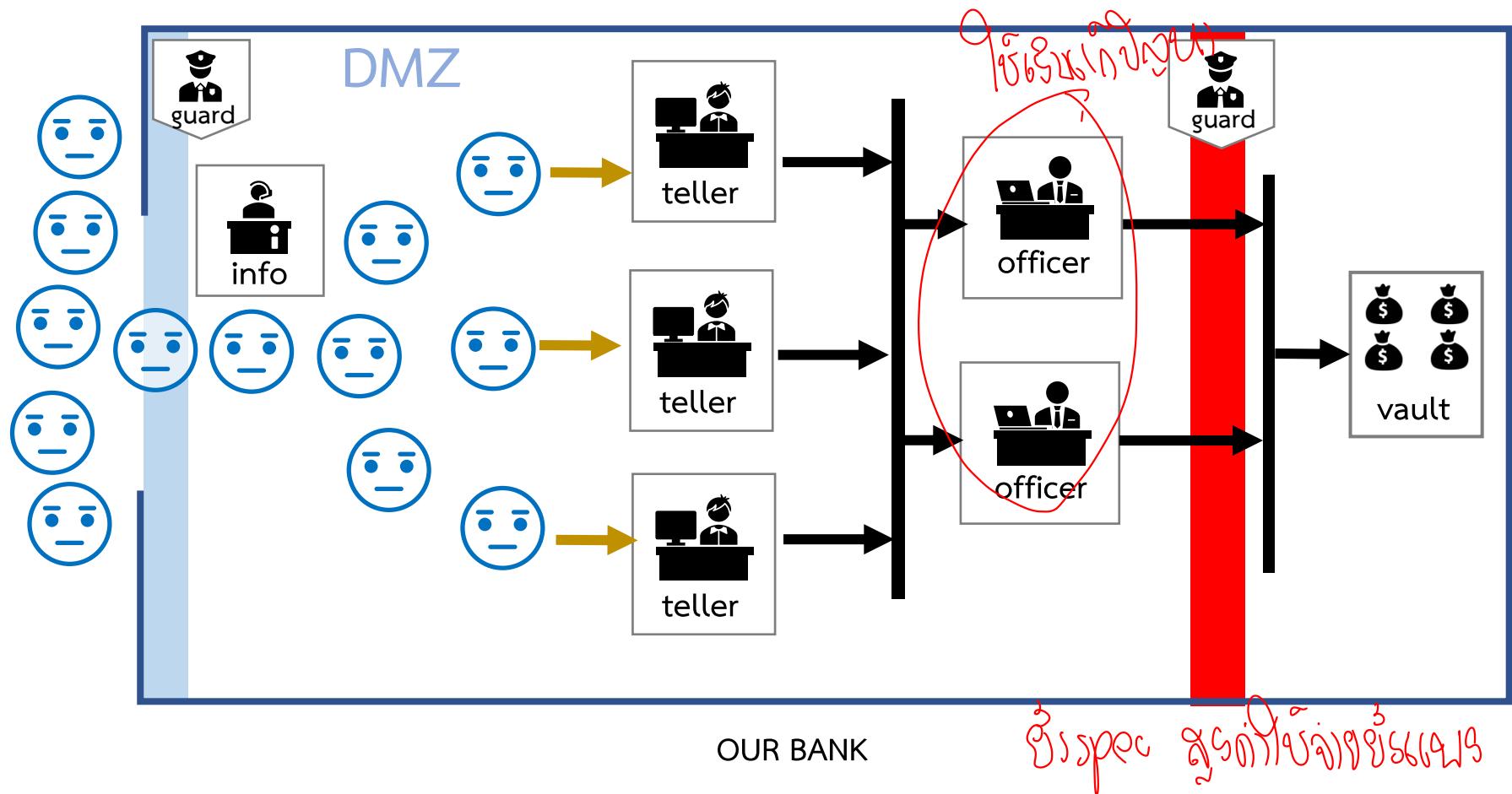
At a Bank



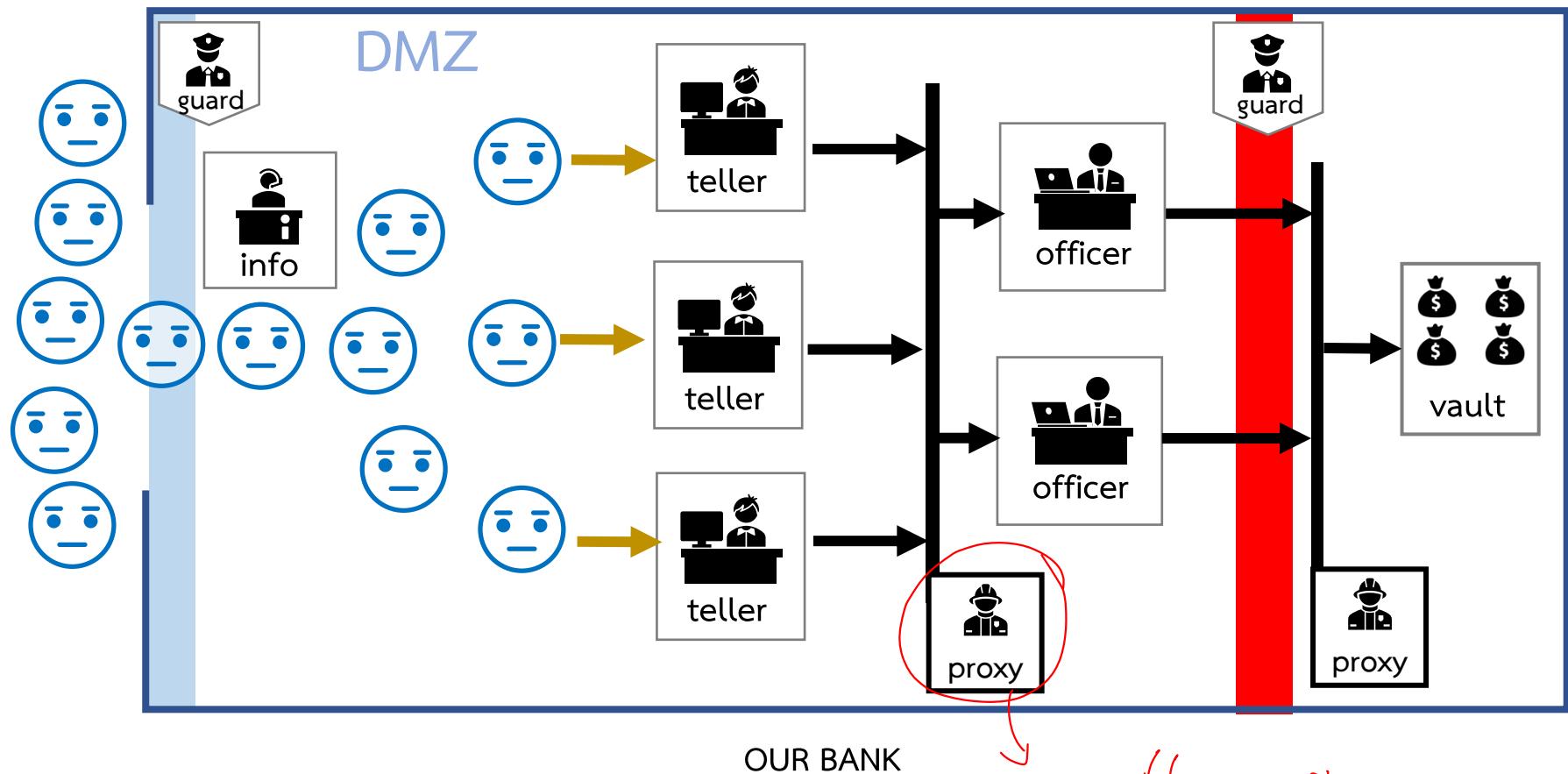
At a Bank



At a Bank

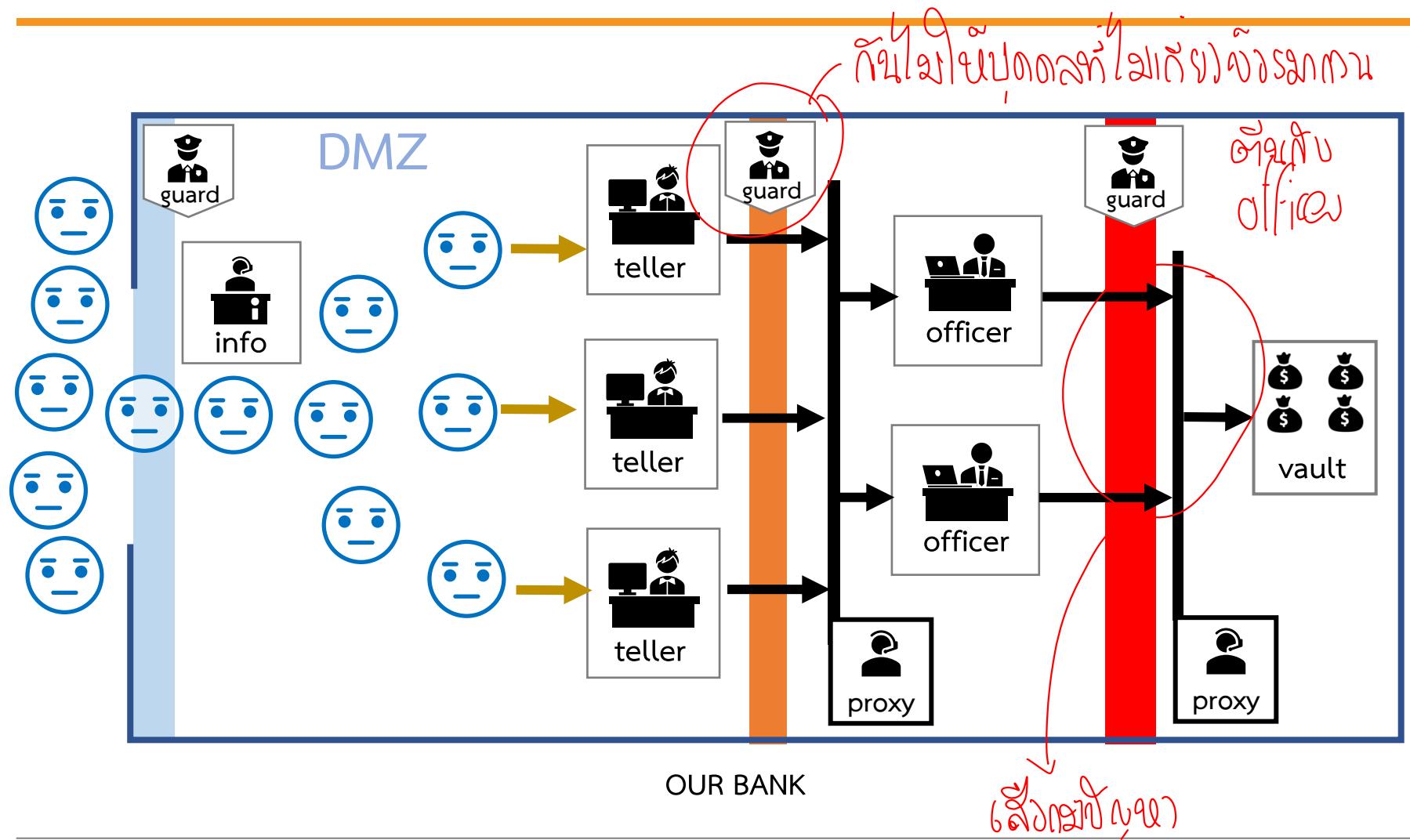


At a Bank

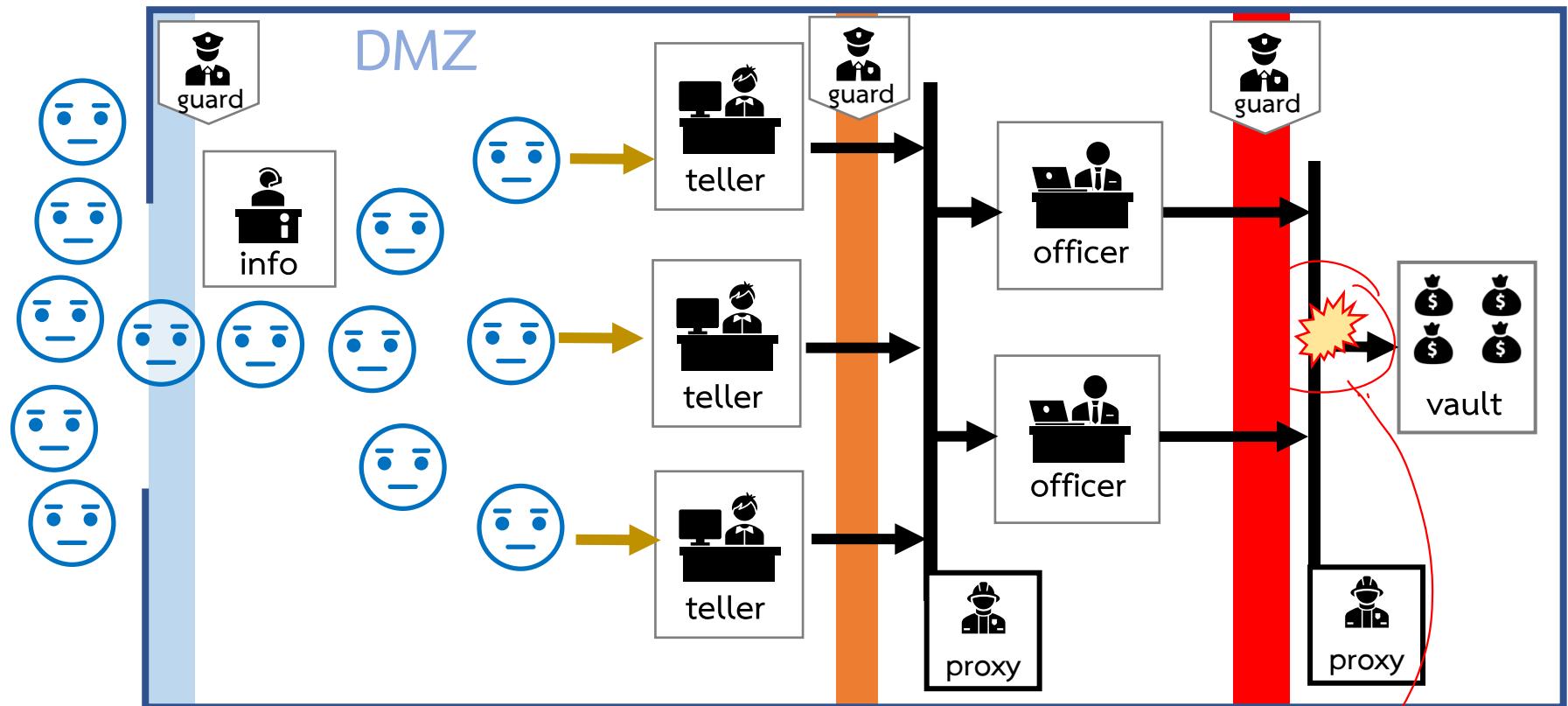


Proxy office នៃរបស់ខ្លួន
teller នឹងចិត្តនទល់ខ្សោយបានផែល) 30

At a Bank



At a Bank



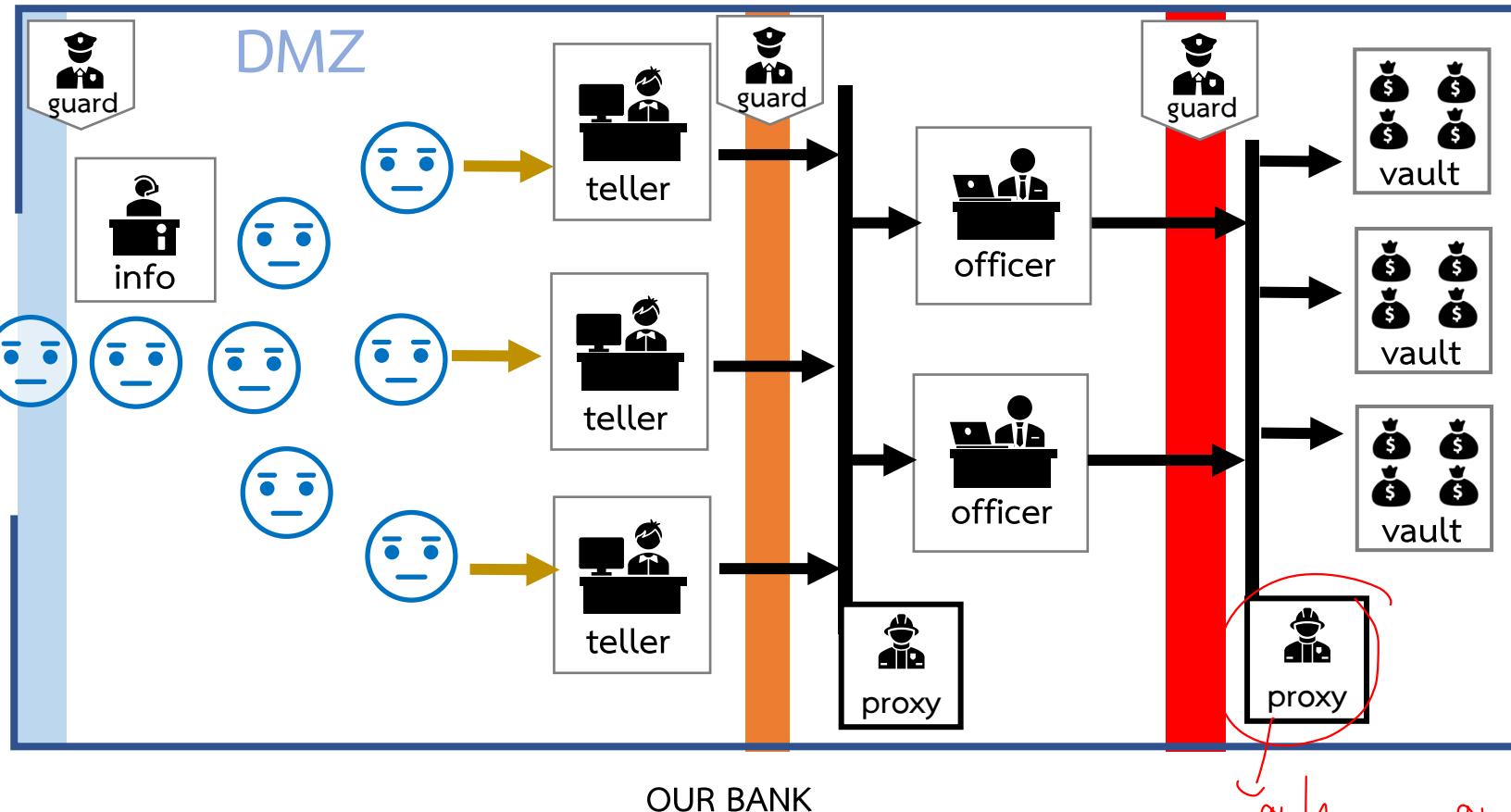
OUR BANK

single point of failure

a single point of failure

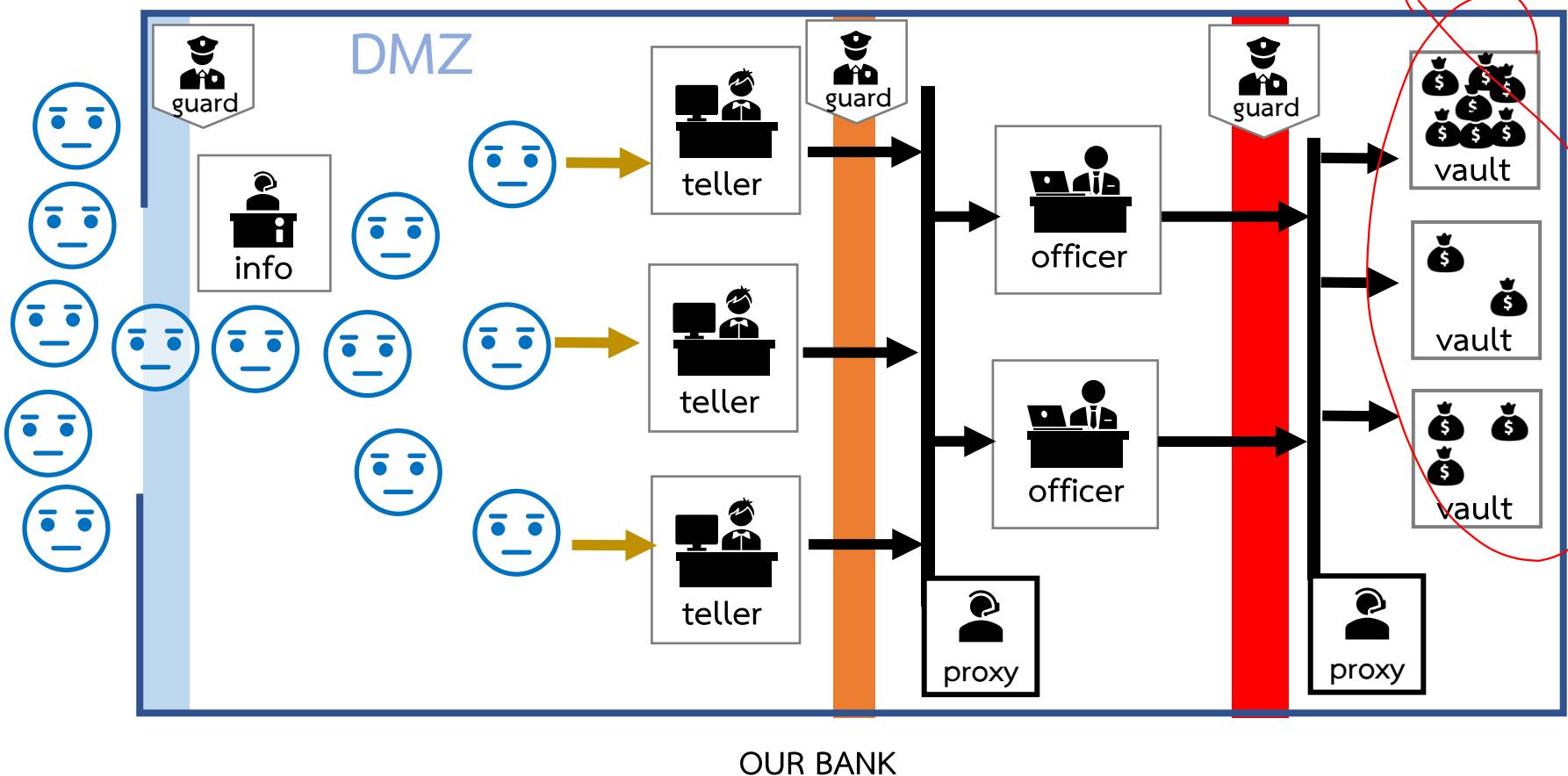
At a Bank

մօօնալիք գործոցներ
աշխատանք

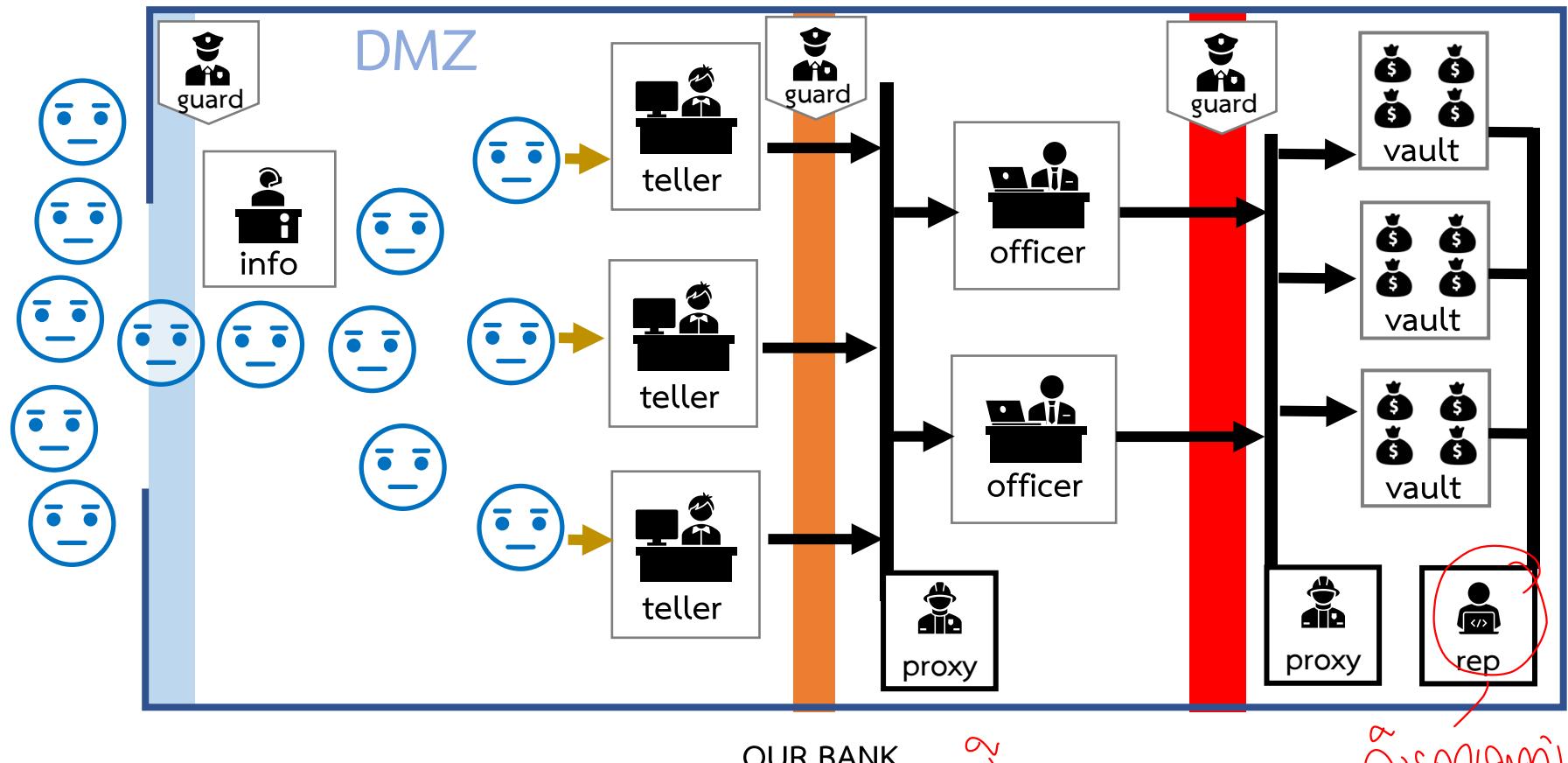


օյլութեալ պարագաներ

At a Bank



At a Bank



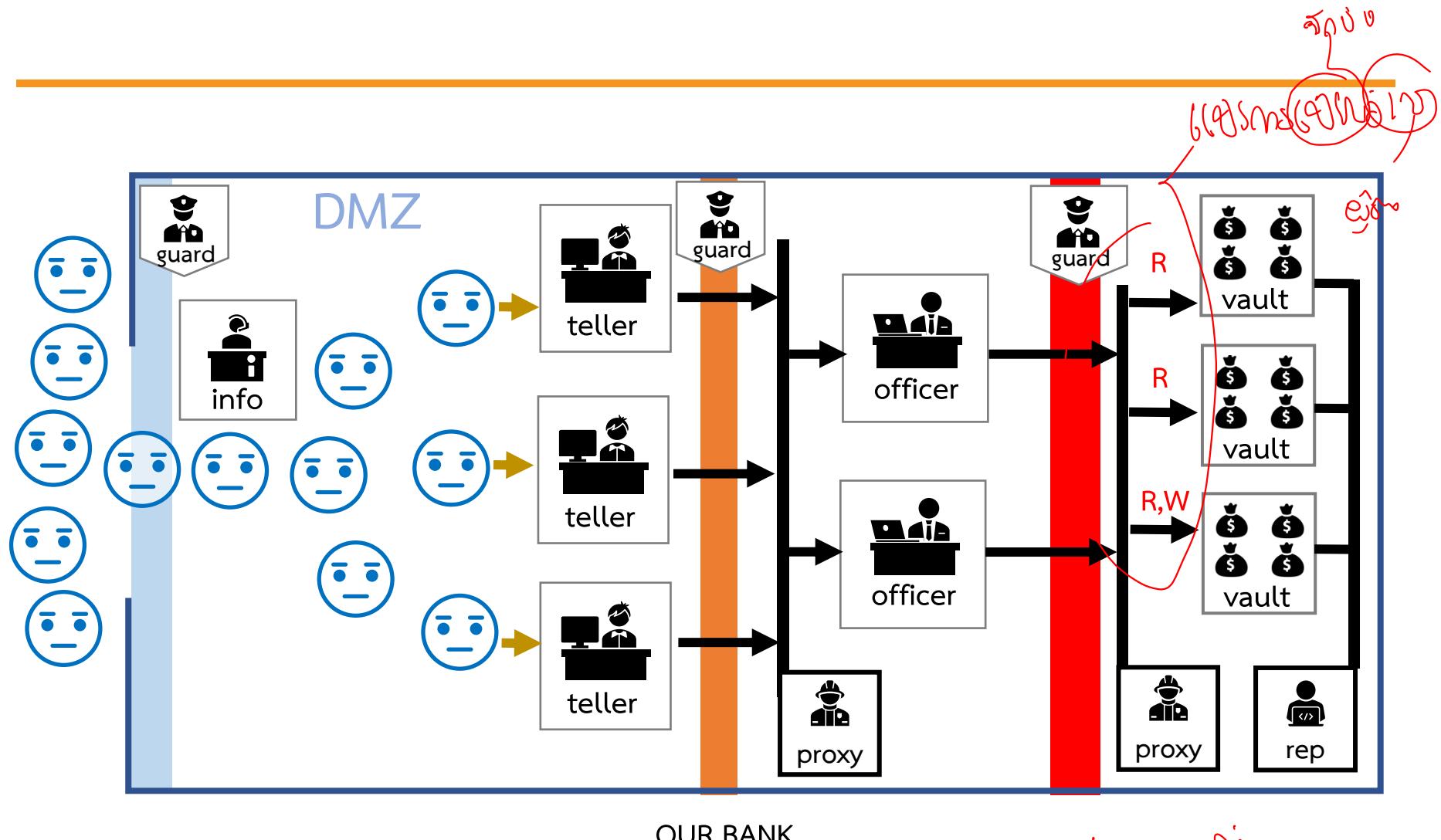
OUR BANK

လုပ်ငန်းများ
နည်းလုပ်ငန်းများ

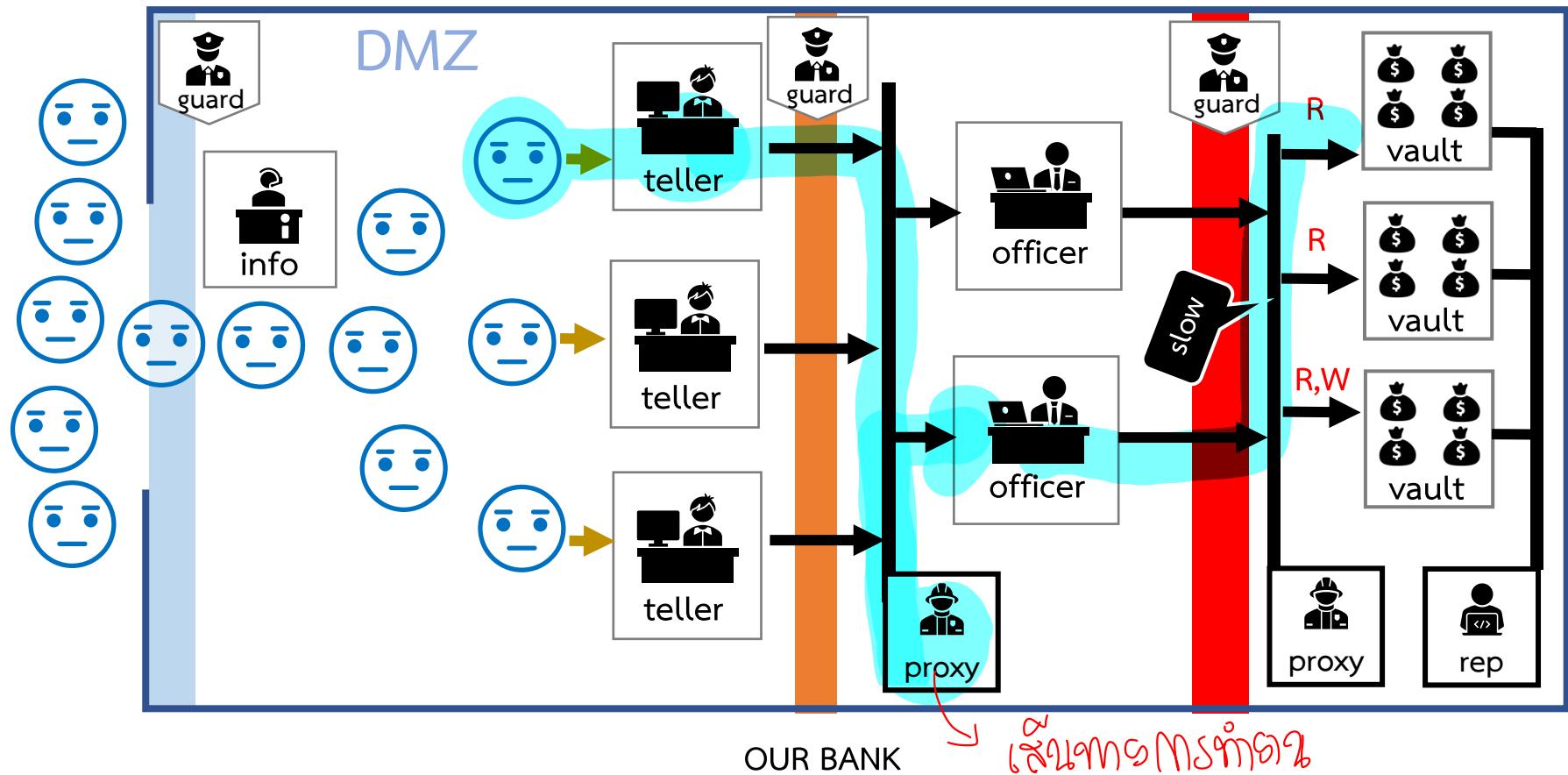
လုပ်ငန်းများ

လုပ်ငန်းများ

At a Bank



At a Bank

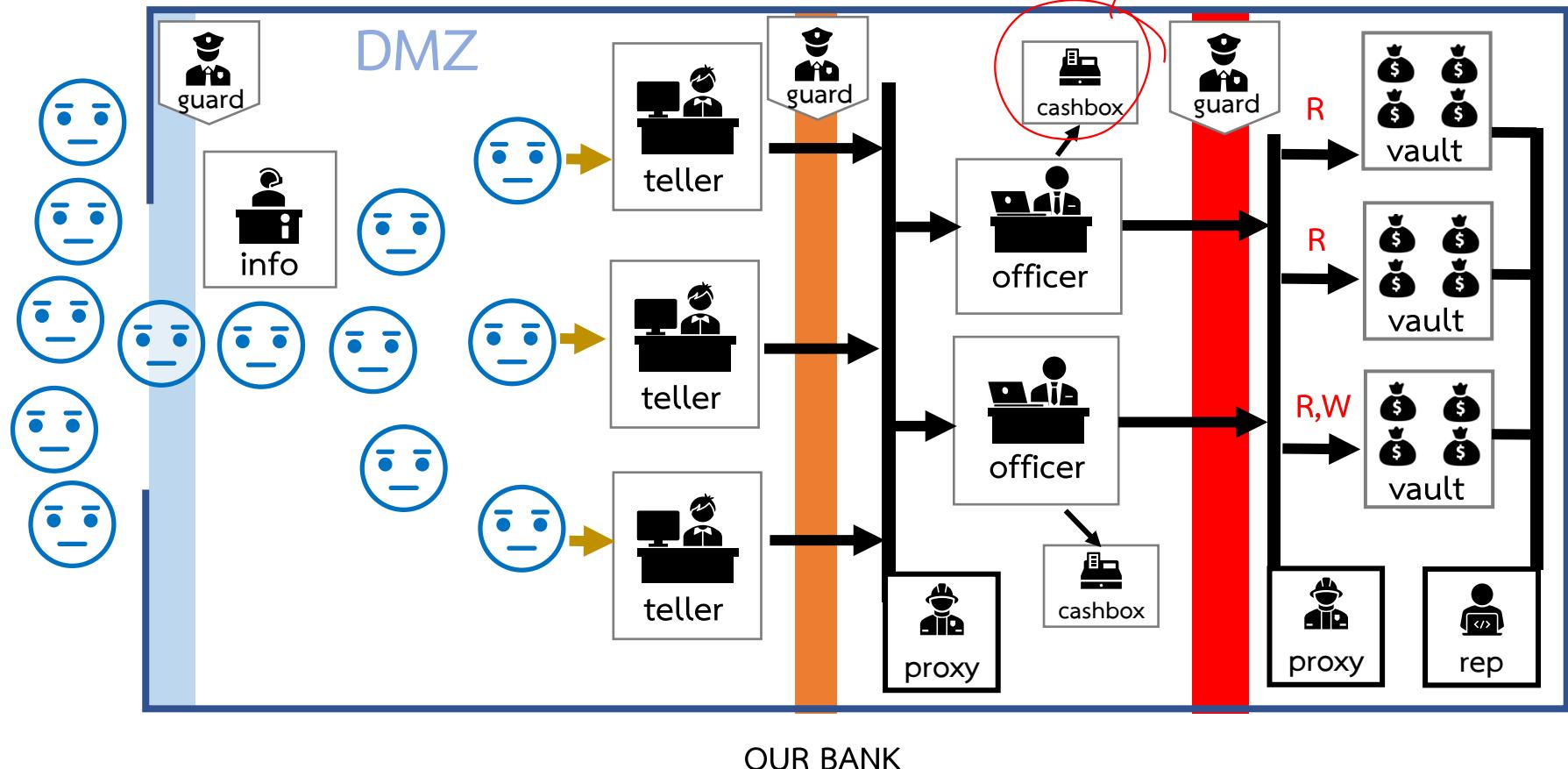


At a Bank

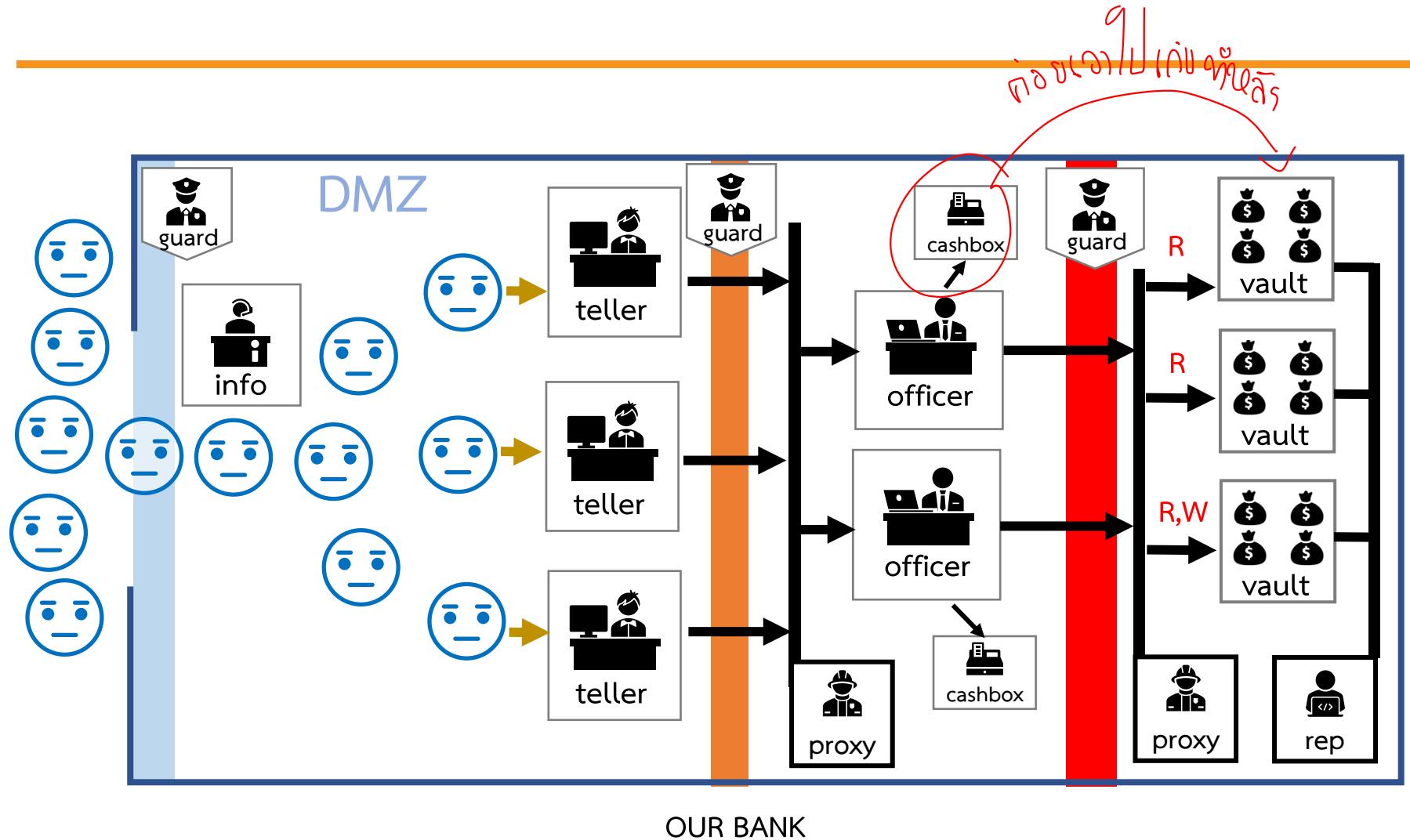
caching

ယေဂတ္ထိနာ ၂ ယု

လီမျှော်ကျင်းကျင်းချို့ခြင်း



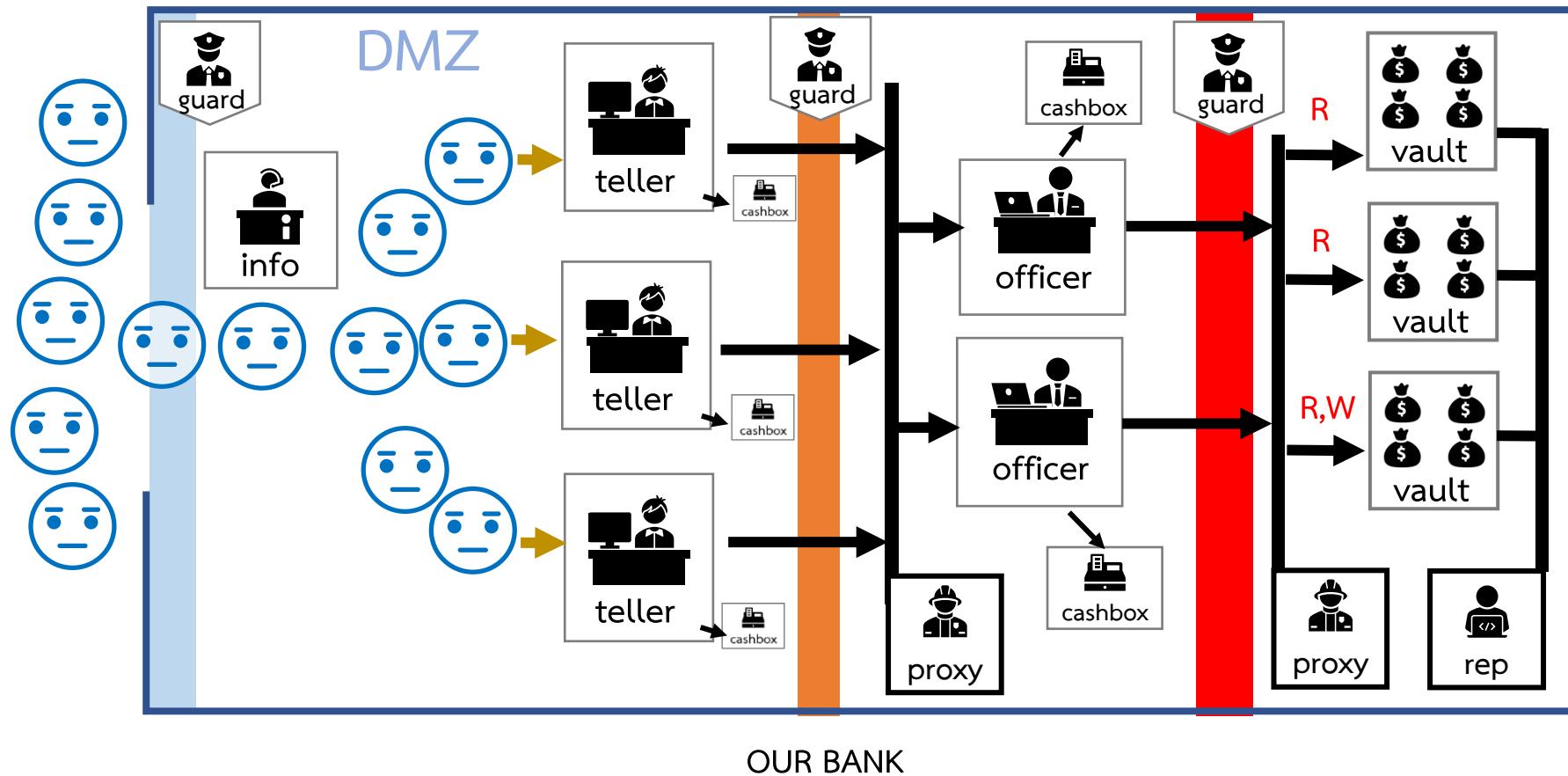
At a Bank



At a Bank

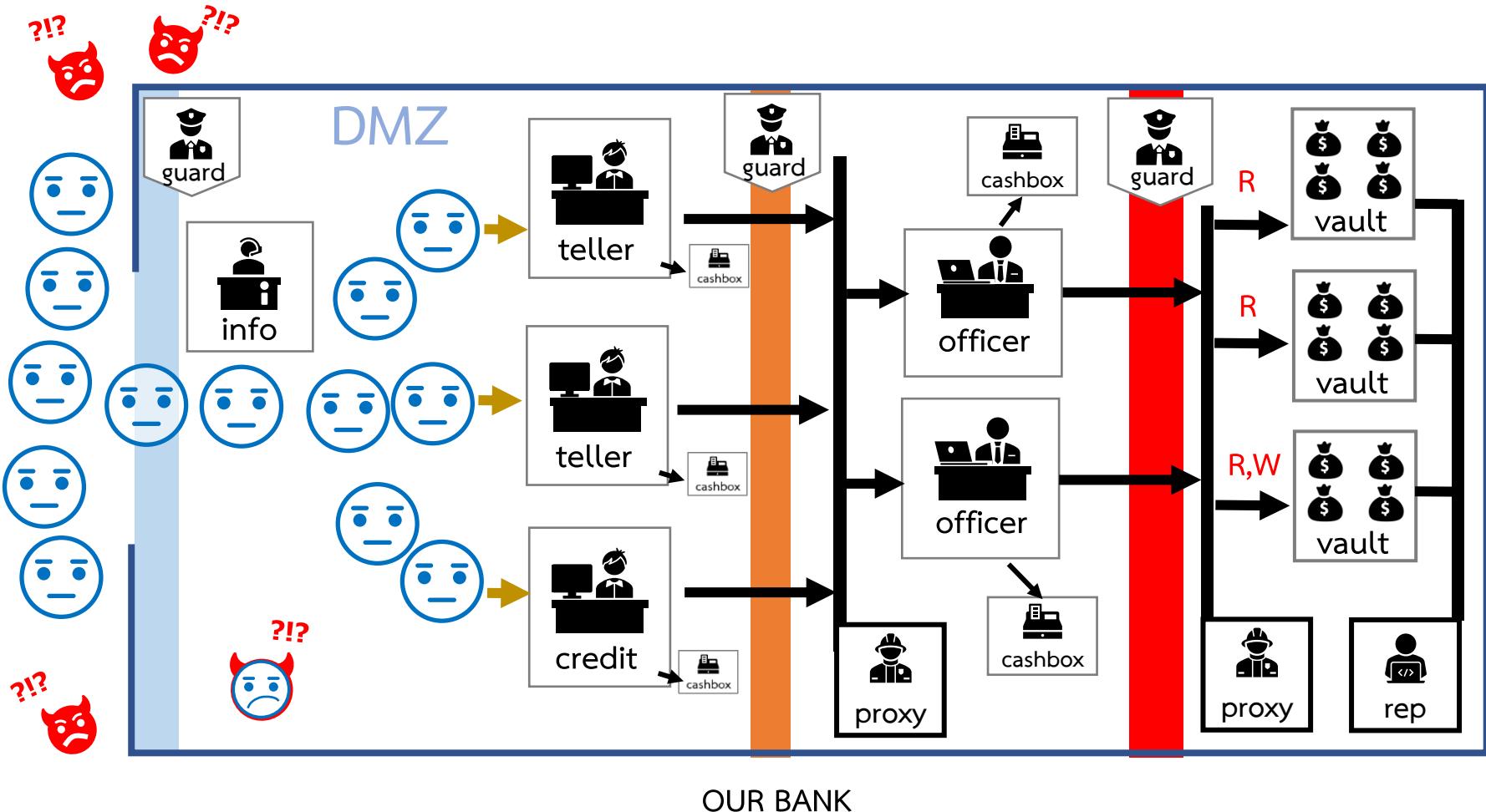
ເມືລາກ່າໄຊແກ້ວມໂຄງການ ໃປ່ງເສົ້າຫຸ້ນທີ່ວັນຕາງຊູ່ກໍຕິກ່ອນ ?

କାନ୍ଦିବିଲୁପ୍ତି



At a Bank

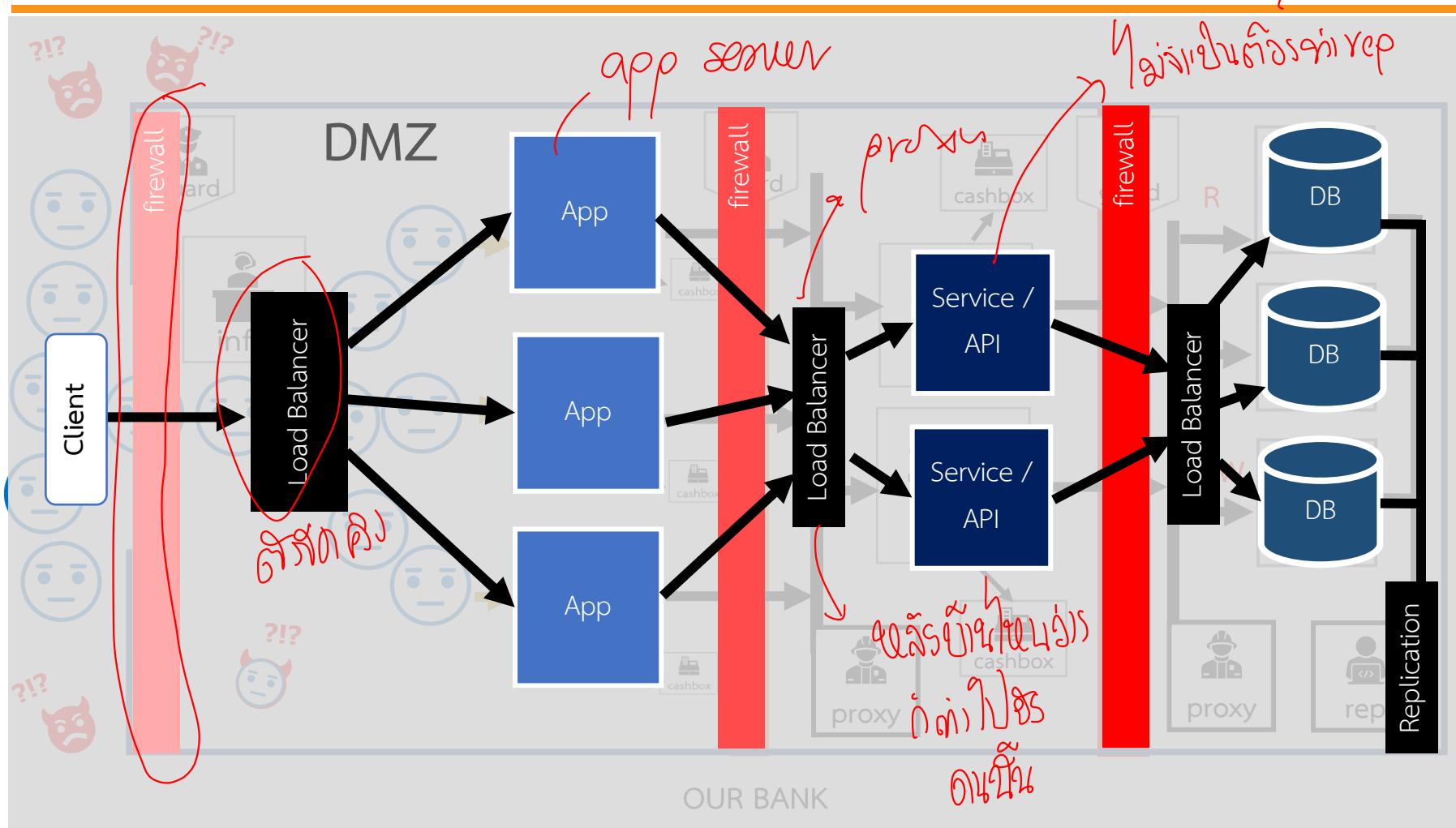
ນີ້ແມ່ນກຳສັງເກດ ສົດຍຸປະກອບໄຈນີ້



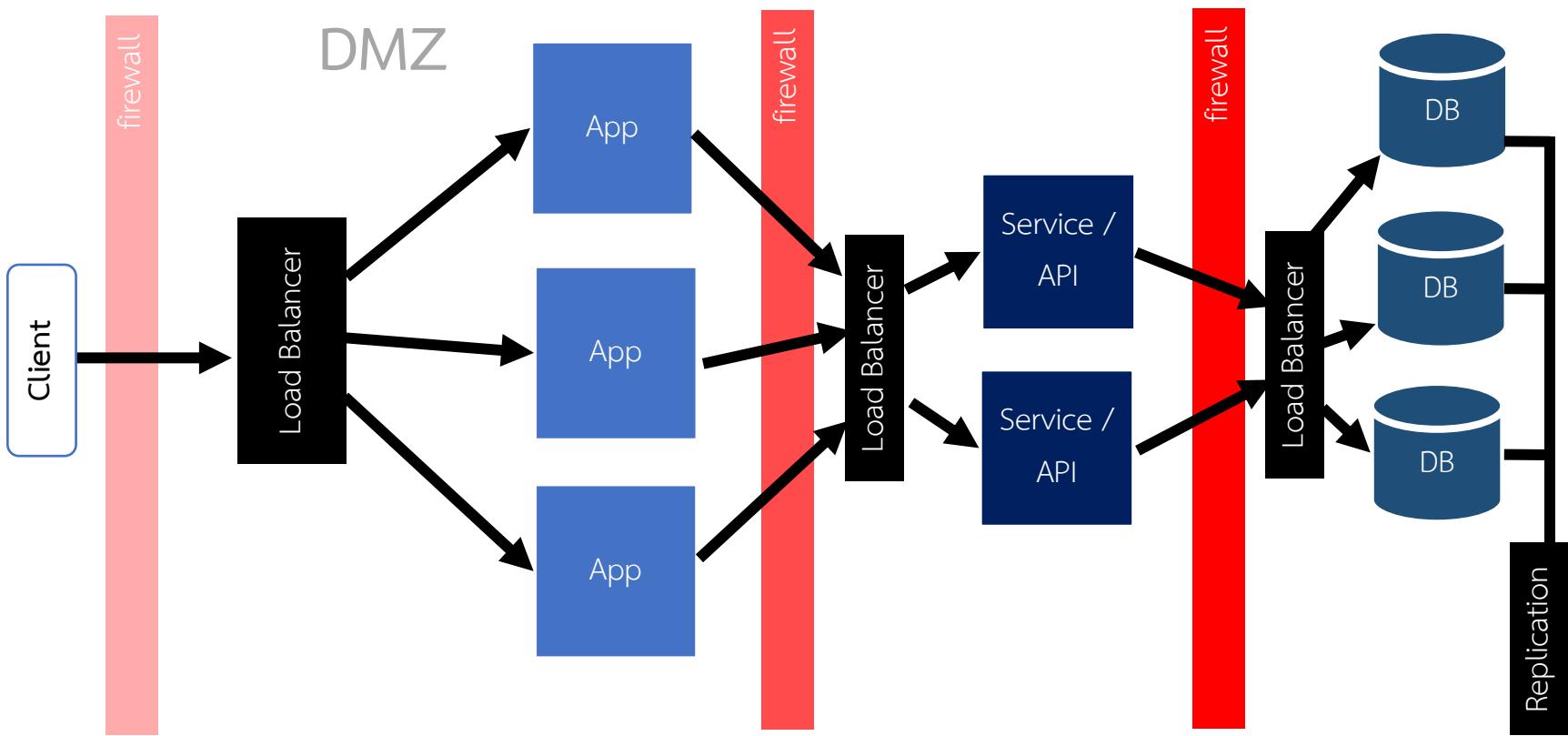
At a Software System

ចំណាំនេះហើយ software នេះ

មានកែងចិត្តណា

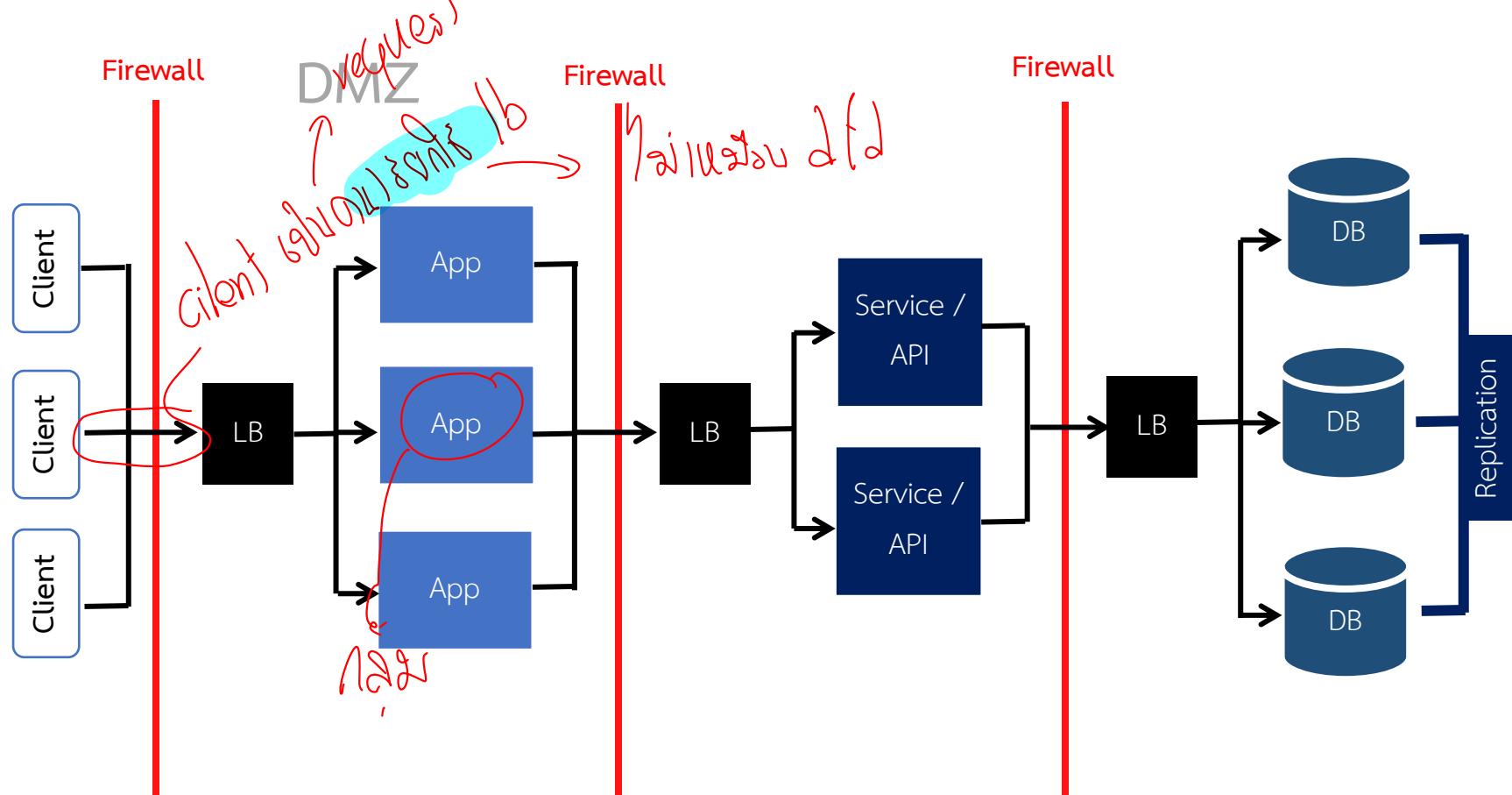


At a Software System



At a Software System

ବାହୀନ ପରିଷଦ



Web service ជាអ្នកទំនាក់ទំនង web api ?

↳ web service

↳ ទេសចរណ៍ website ដែលបានរោចចិត្តថា មិនមែនការពារស៊ីវស, protocol http នៃស៊ីអ្នកនឹង XML, JSON

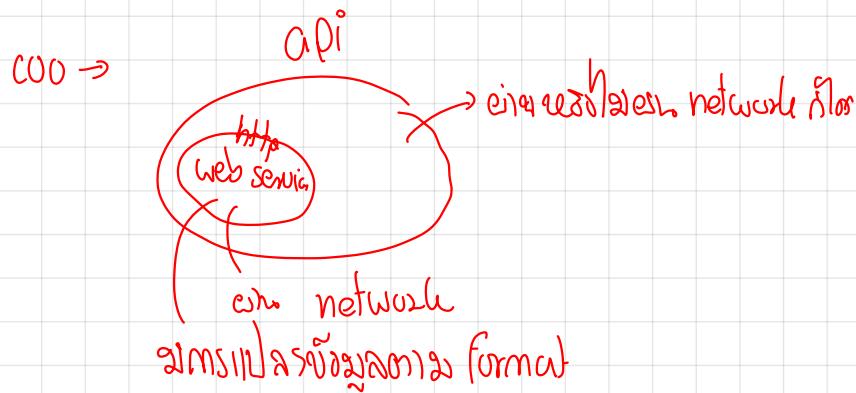
client → web service
[
 មិនមែនការពារស៊ីវស, ទៅទែ, ឬទេ
 មួយប្រភេទ model នូវខ្លួន rest, soap

↳ api

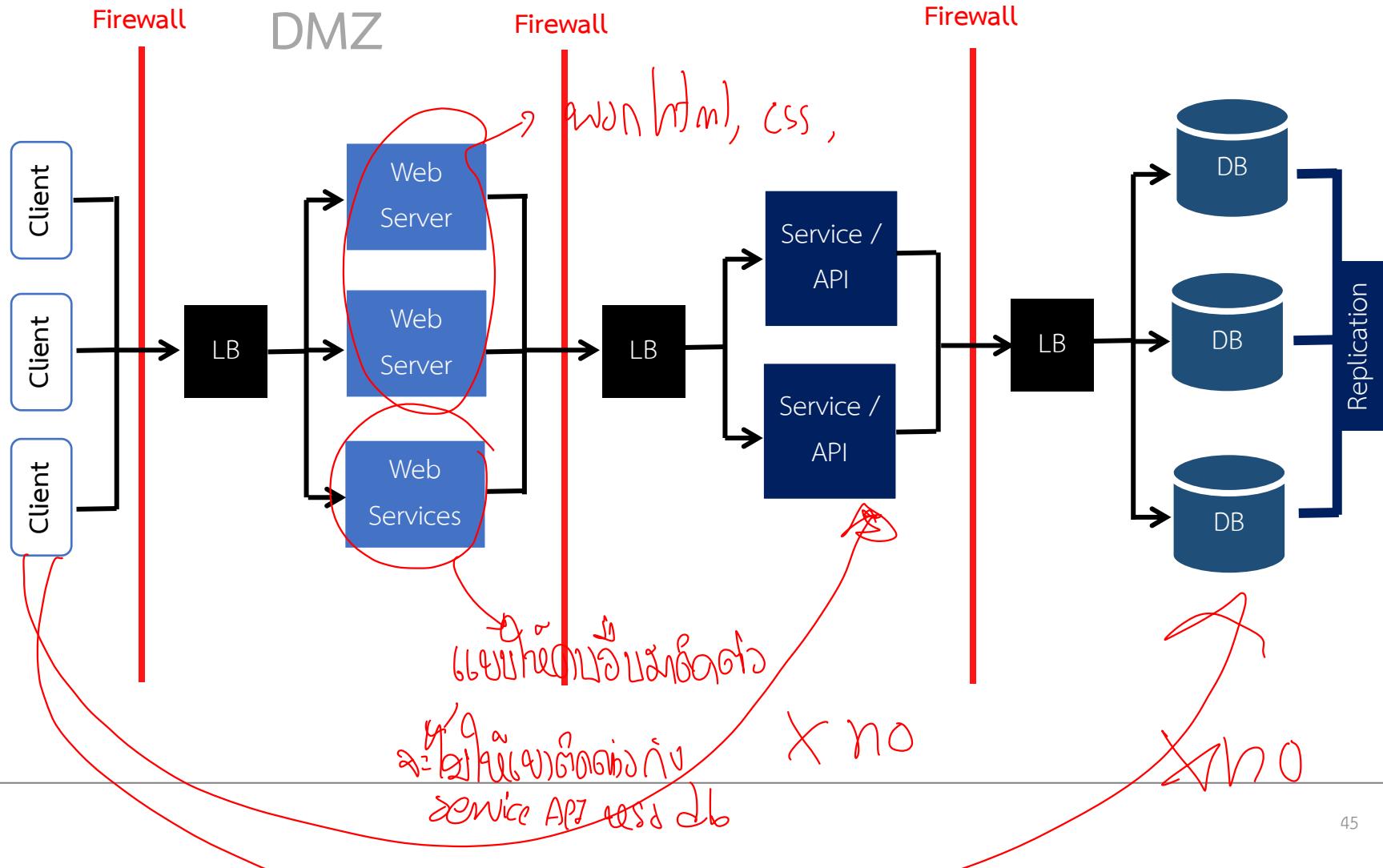
↳ មិនមែនការពារស៊ីវសទេ ប៉ុន្មានការពារស៊ីវសនៃ web service បានរៀបចំឡើងទៅលើ http ex នូវការប្រើប្រាស់មិនមែន

↳ មិនមែនការពារស៊ីវសទេ

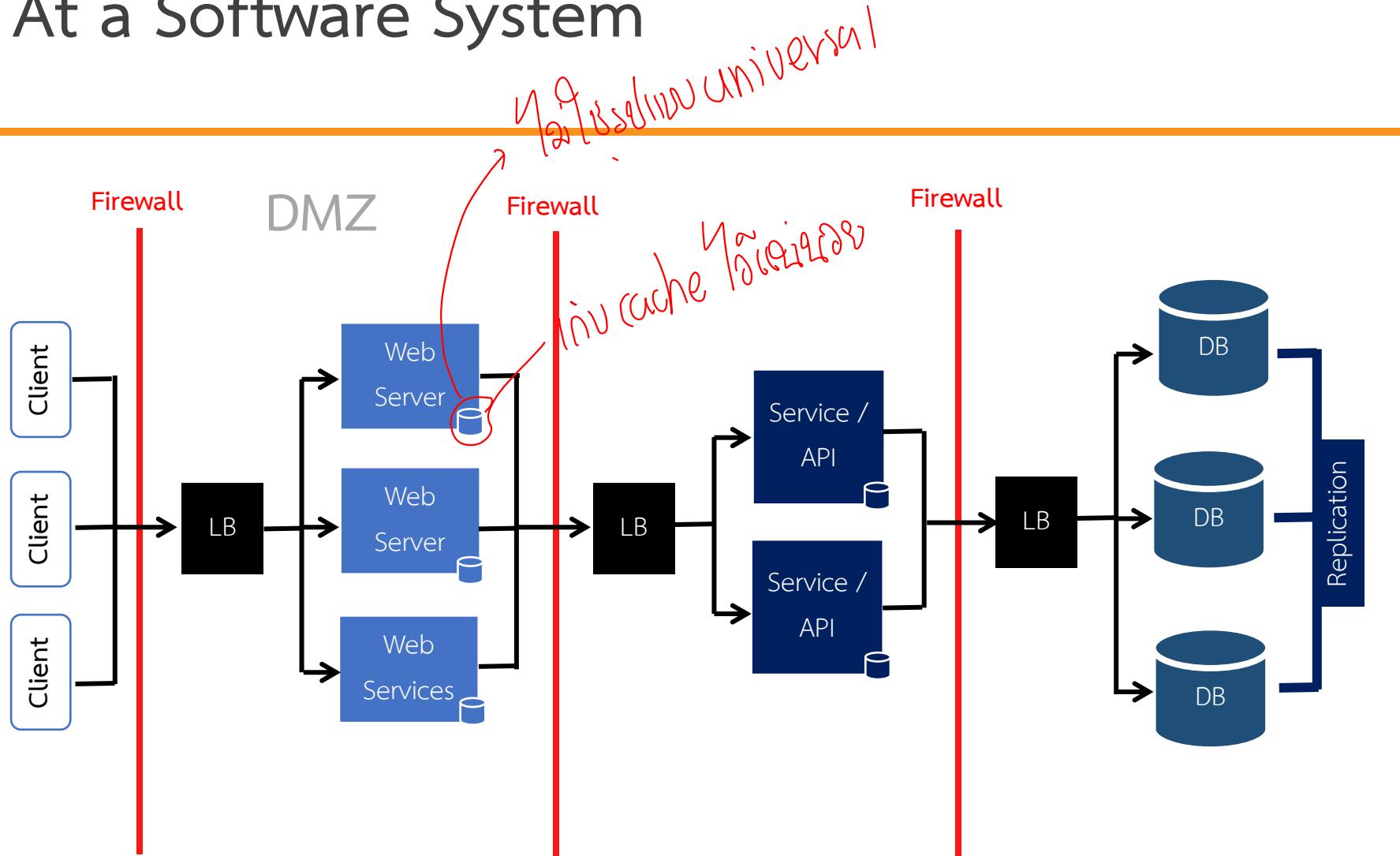
↳ ដើម្បីទទួលិន្ទី api នៅក្នុង website → Web api = web service គឺឯង



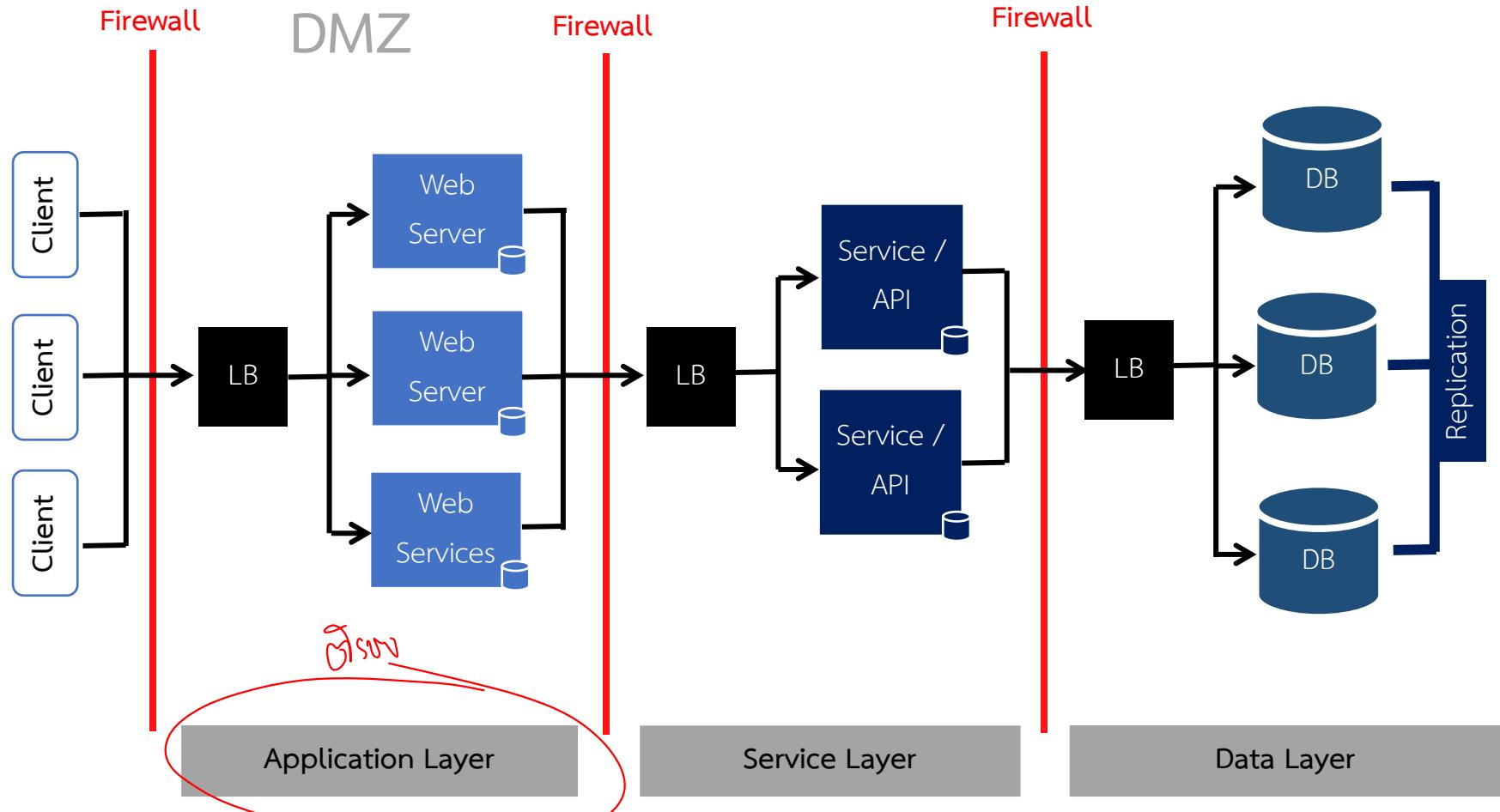
At a Software System



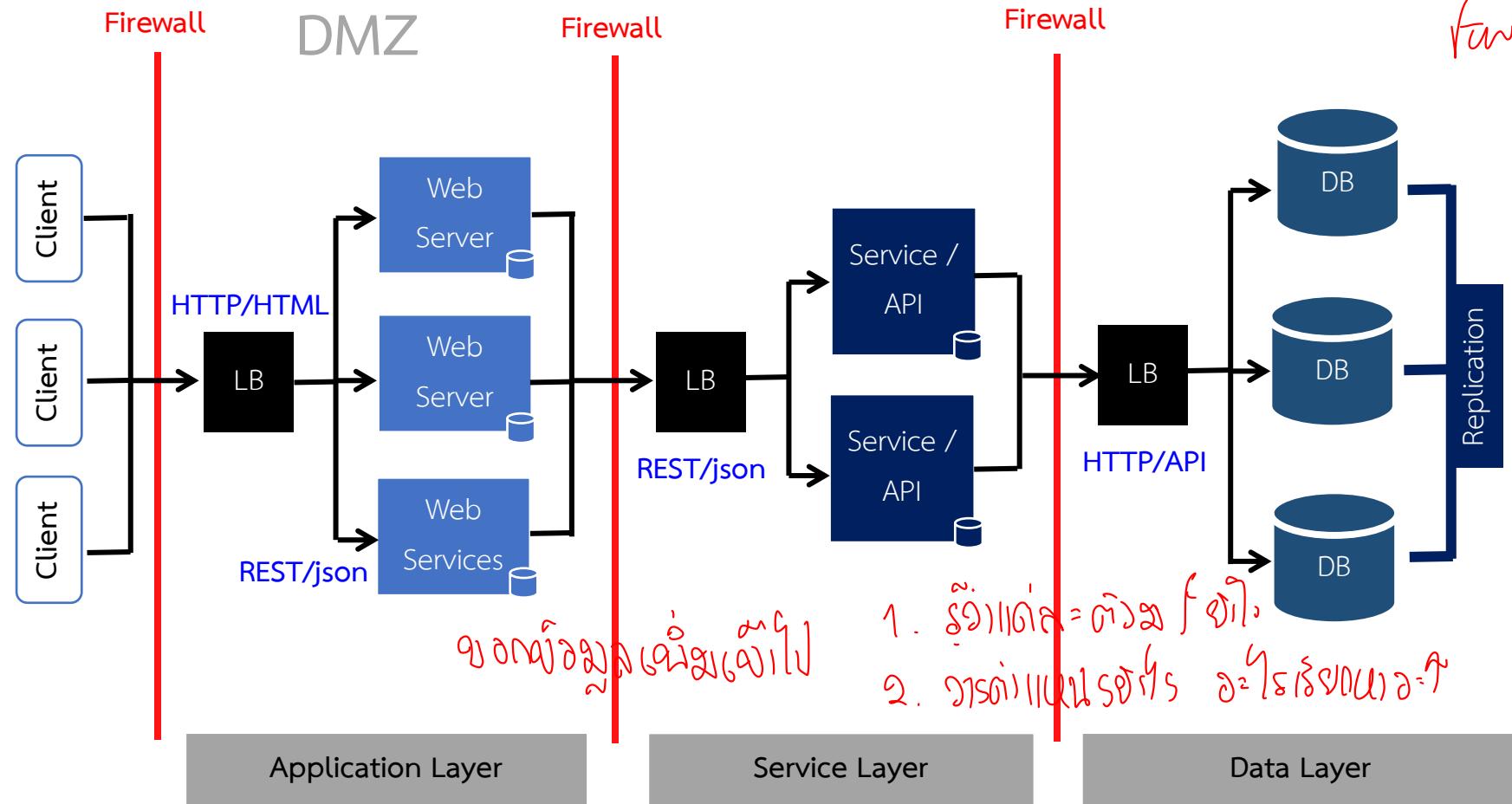
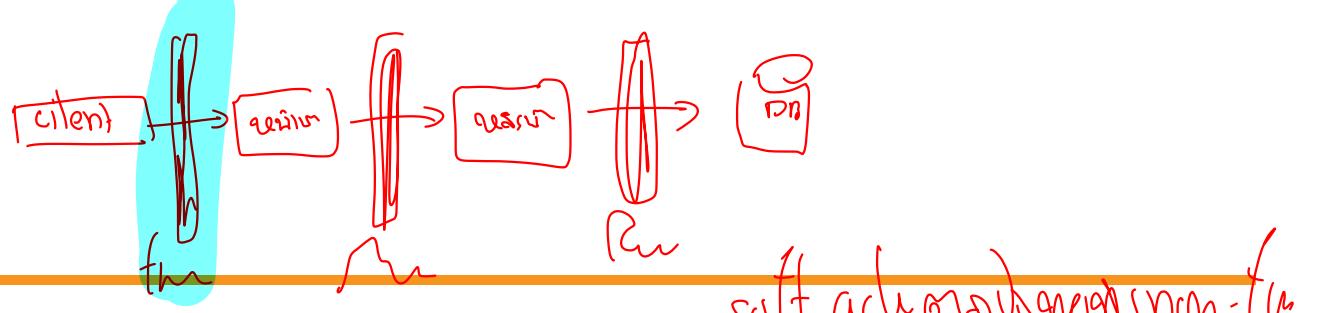
At a Software System



At a Software System



SA : Server



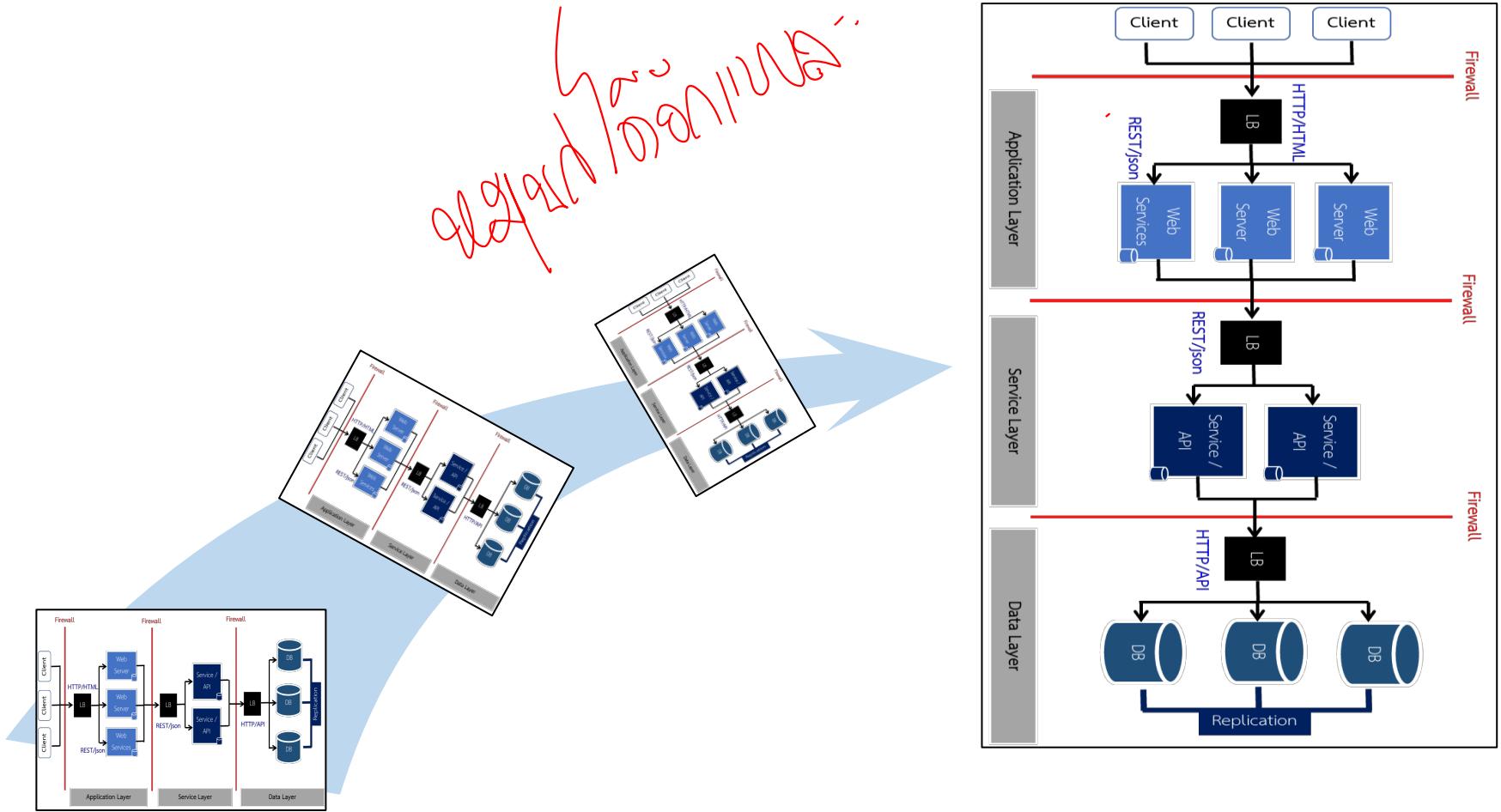
1. សេវាដែលត្រូវការអនុញ្ញាត
2. សេវាដែលត្រូវការអនុញ្ញាត ដែលមិនមែនជាការបញ្ចូនទេ

3. នឹងរួម

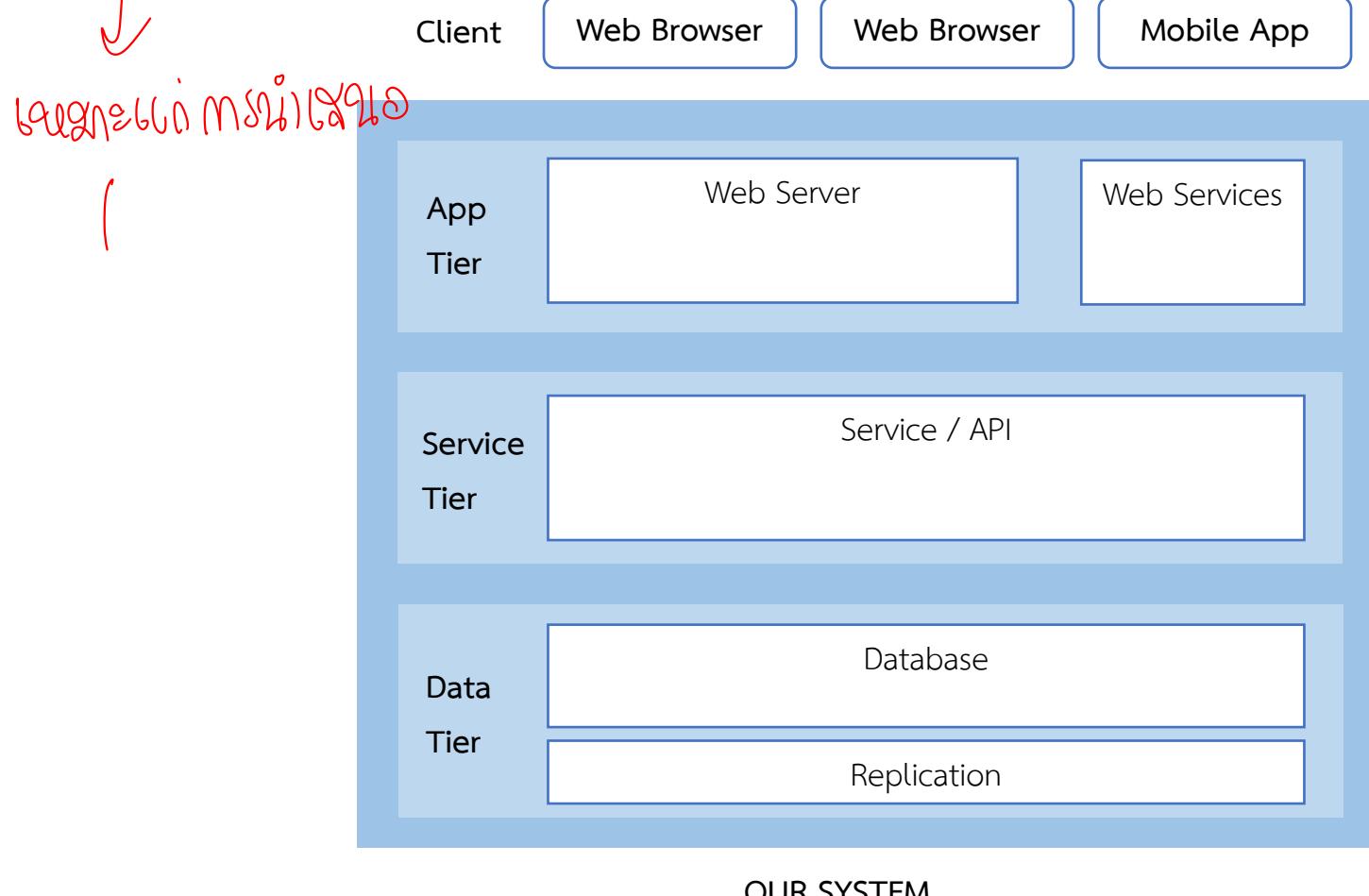
4. ផ្សេងៗពីការបញ្ចូន

និងការអនុញ្ញាត

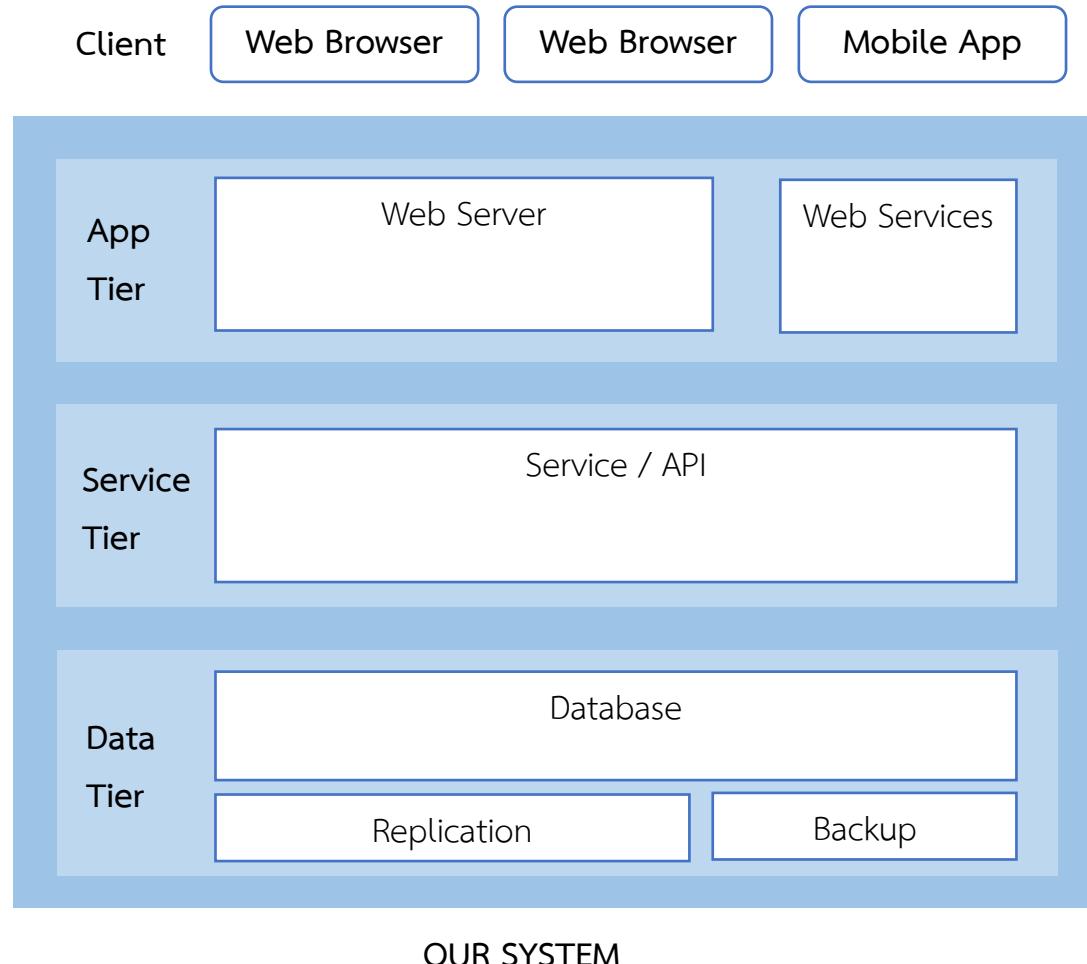
SA



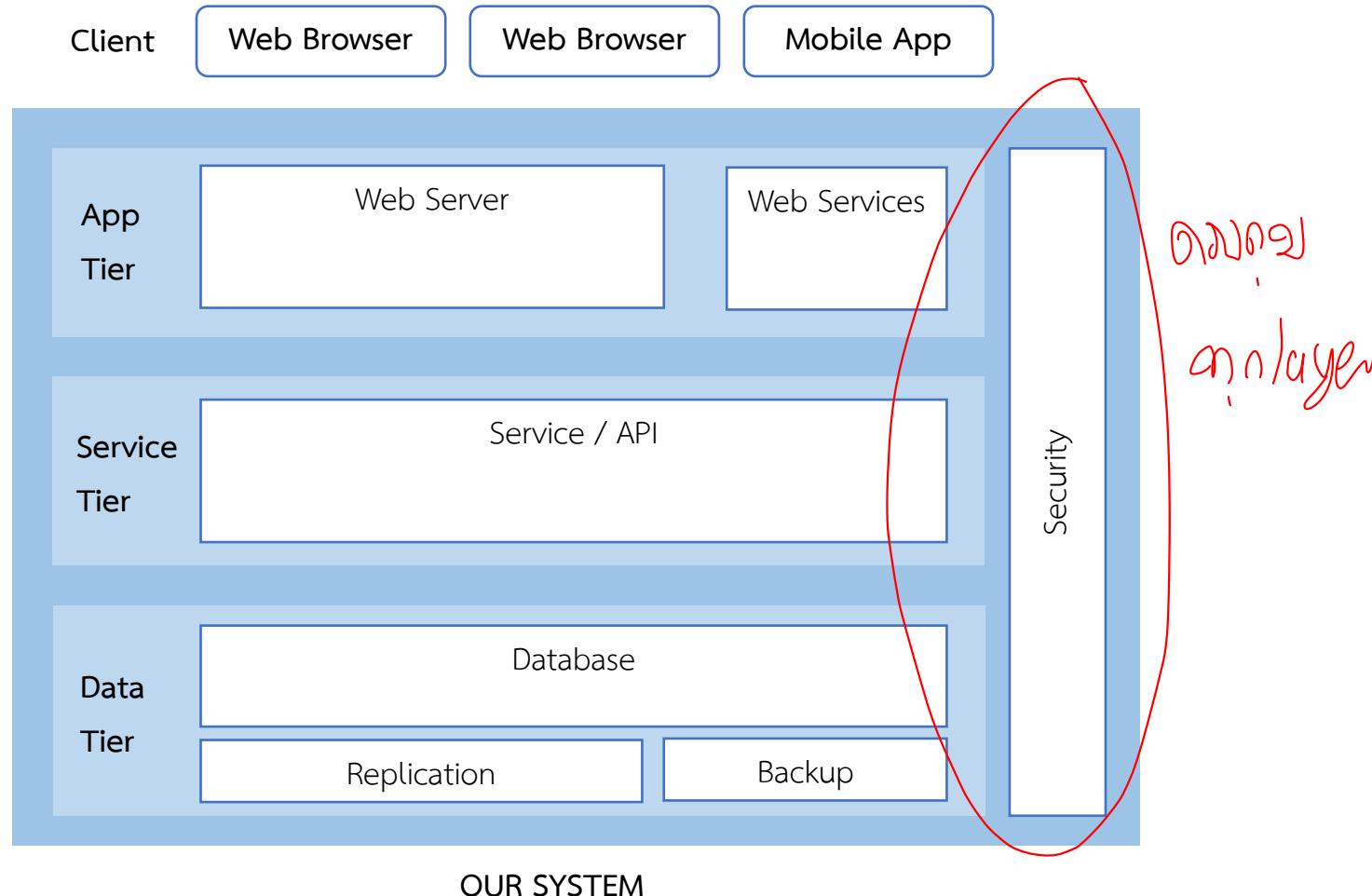
SA : Stack → မြန်တိုက်ခွဲများ



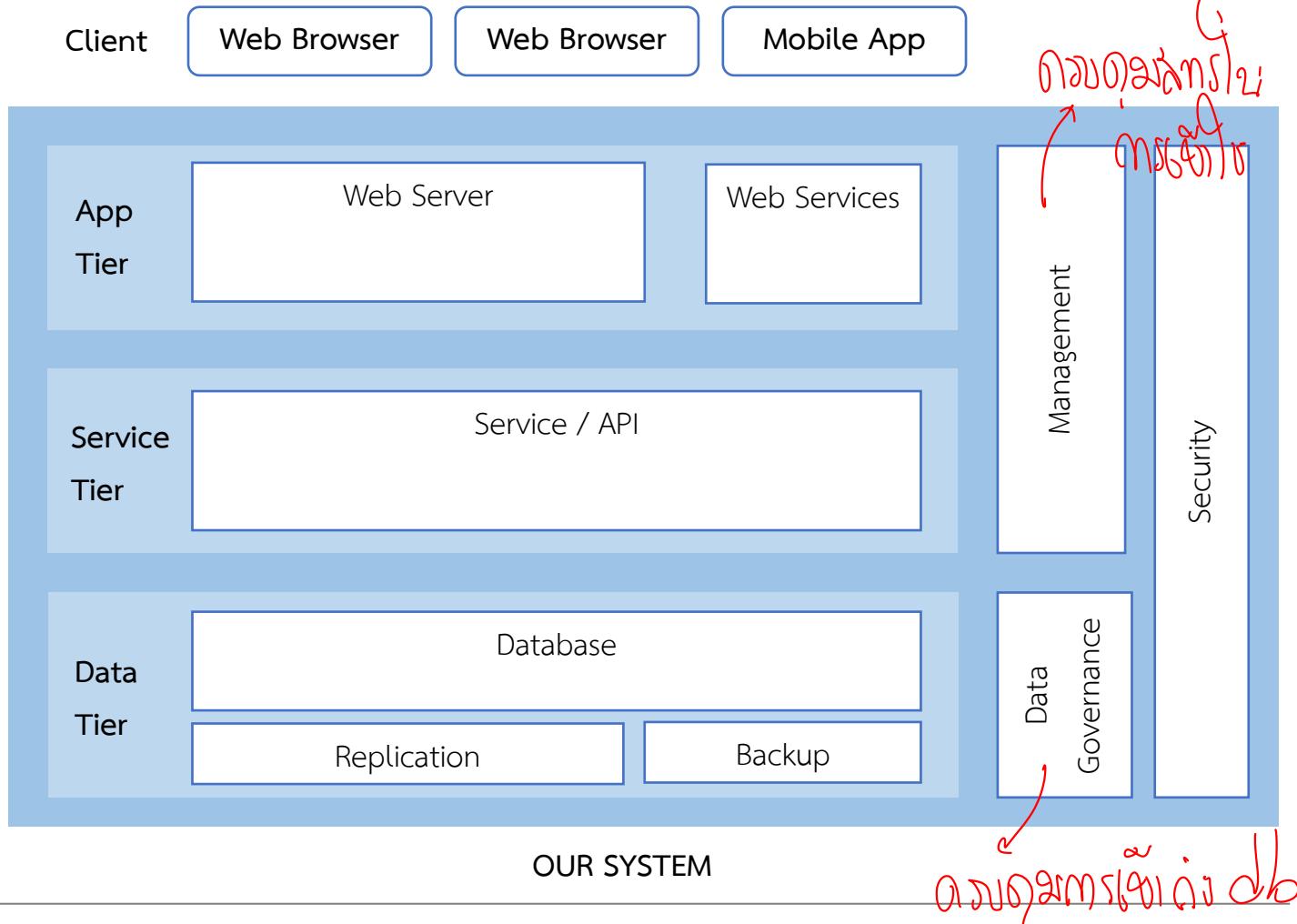
SA : Stack



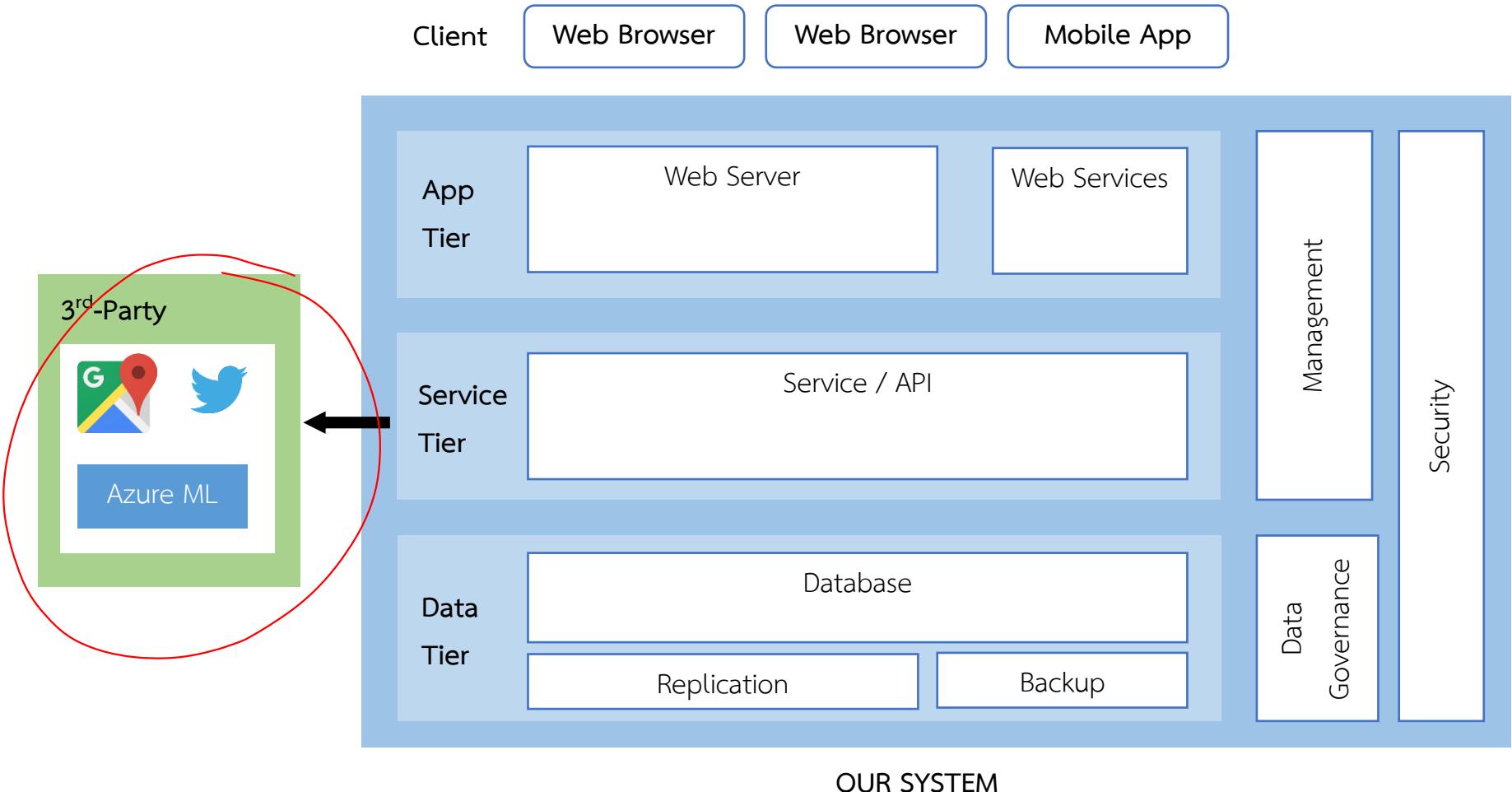
SA : Stack



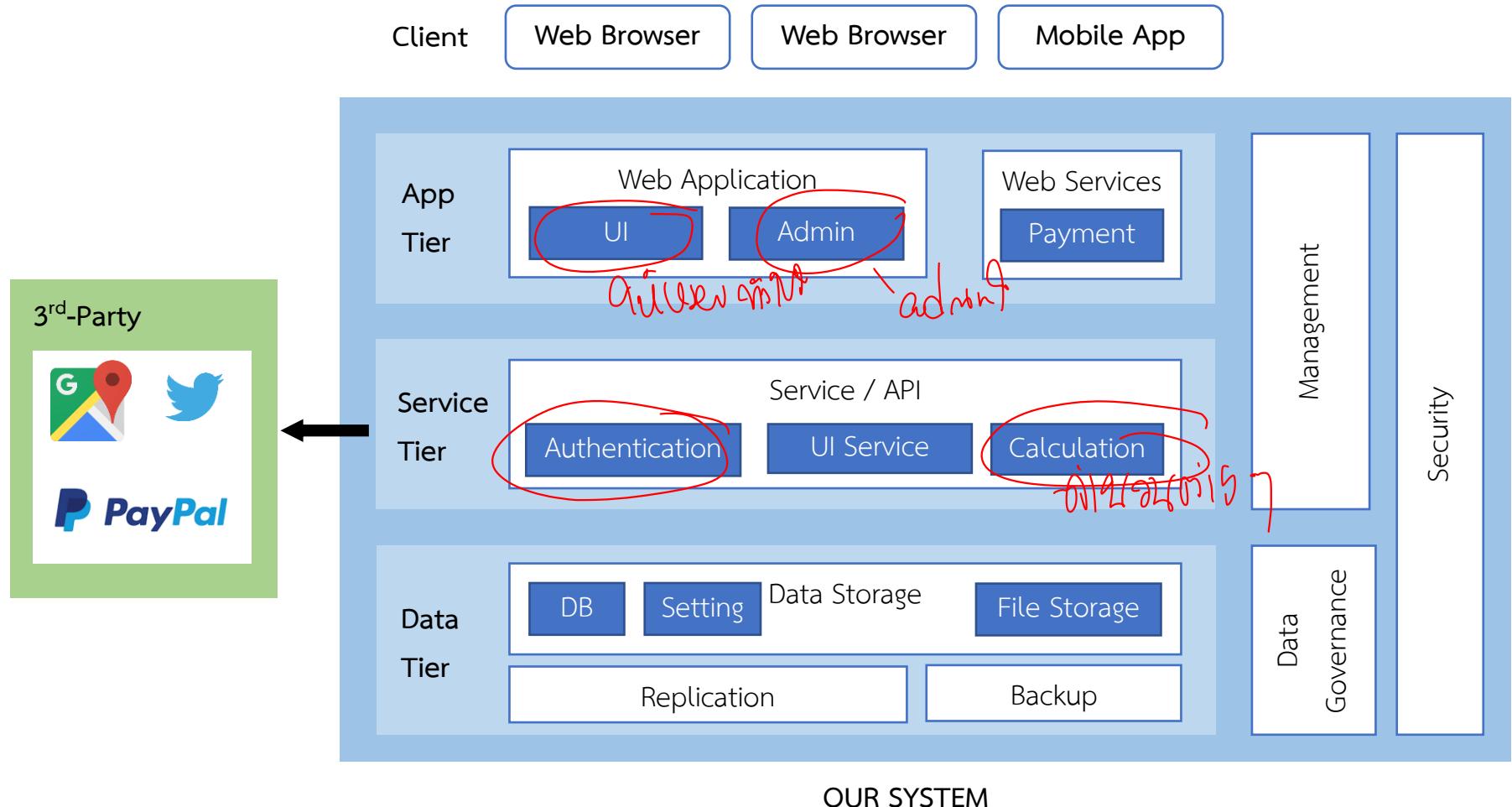
SA : Stack



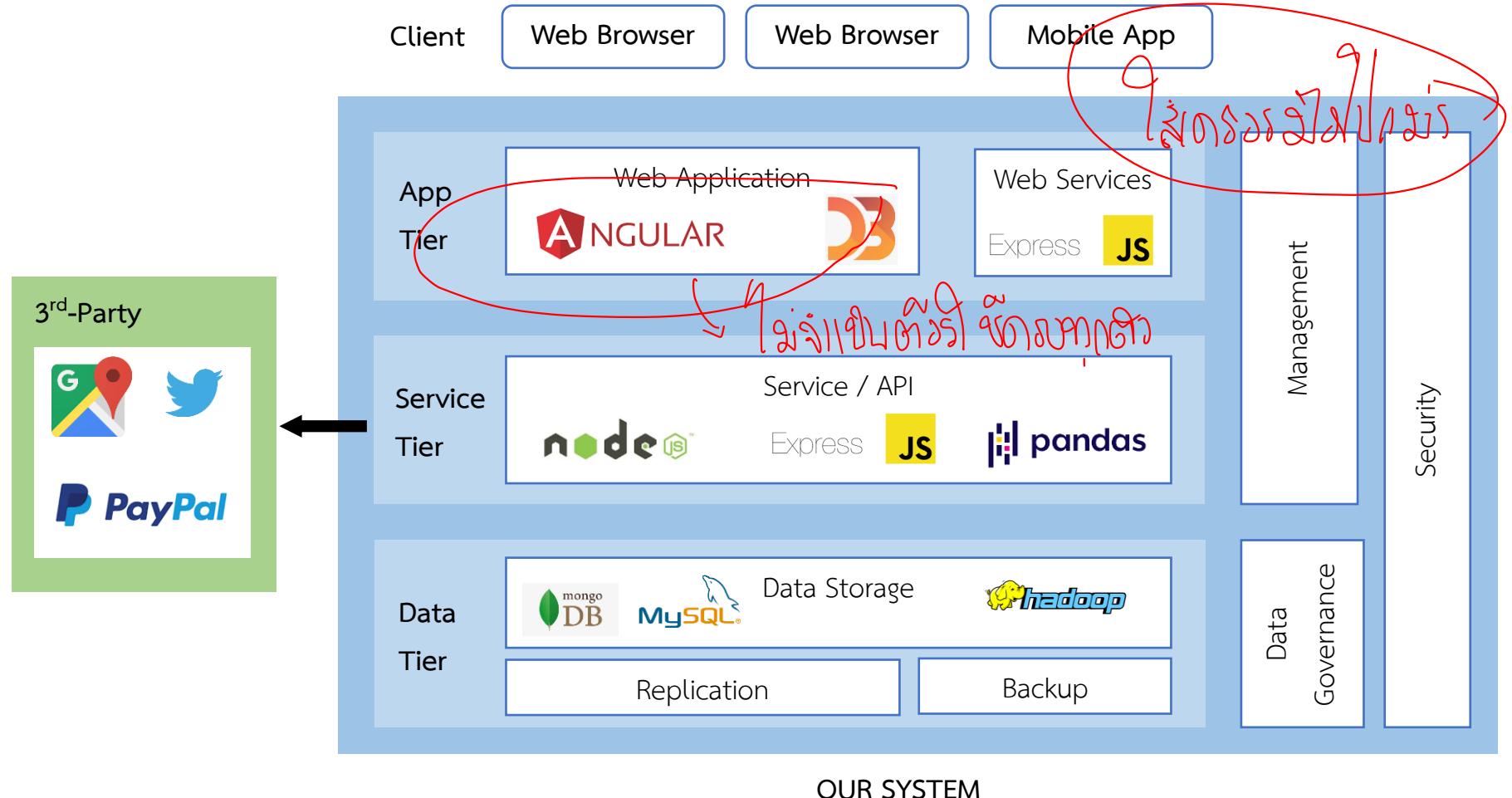
SA : Stack : 3rd-Party



SA : Stack : Components + 3rd-Party



SA : Stack : Tools + 3rd-Party



SA : Stack : Components + Tools

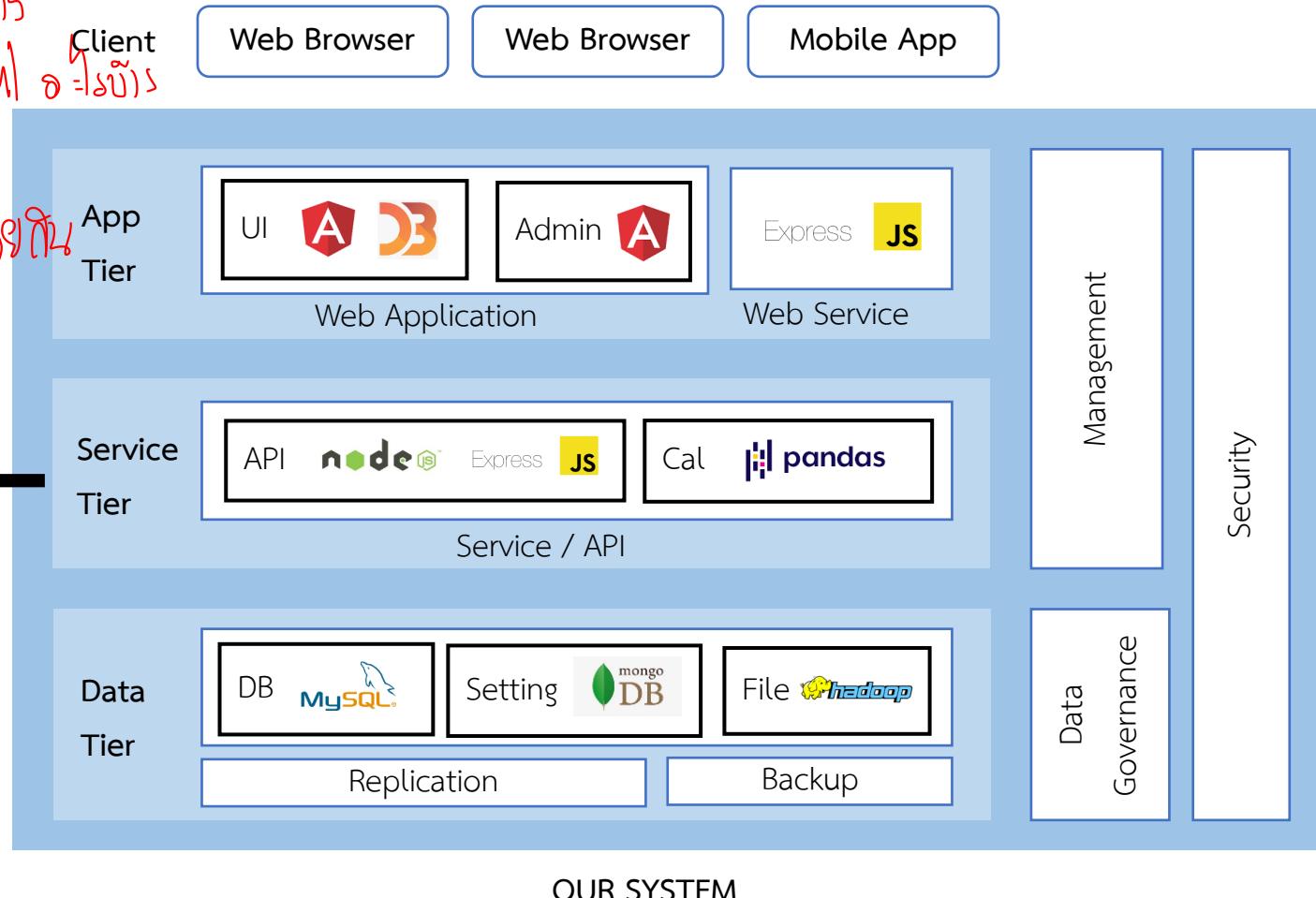
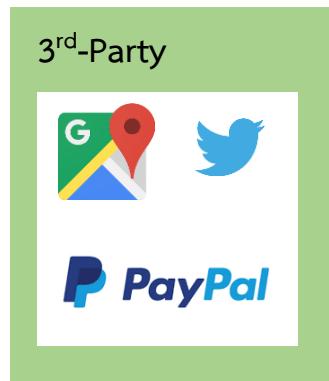
សម្រាប់បង្កើតអនឡាញ

1. Function តាមរយៈការបង្កើត

2. Non-functional តាមរយៈការបង្កើត

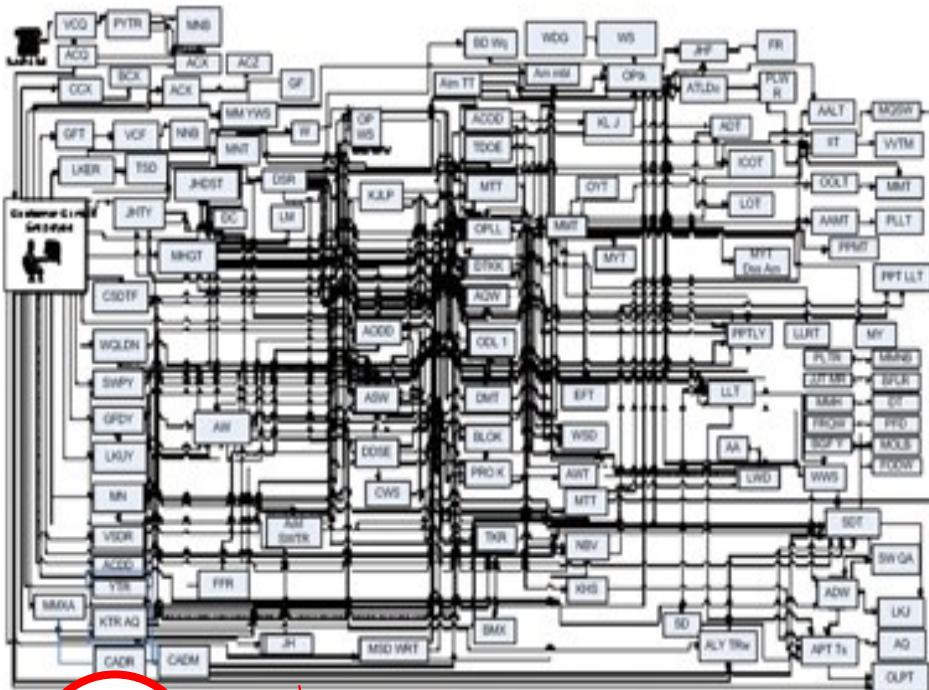
3. ទាន់សំណង់

4. protocol ពីក្រុមហ៊ុន



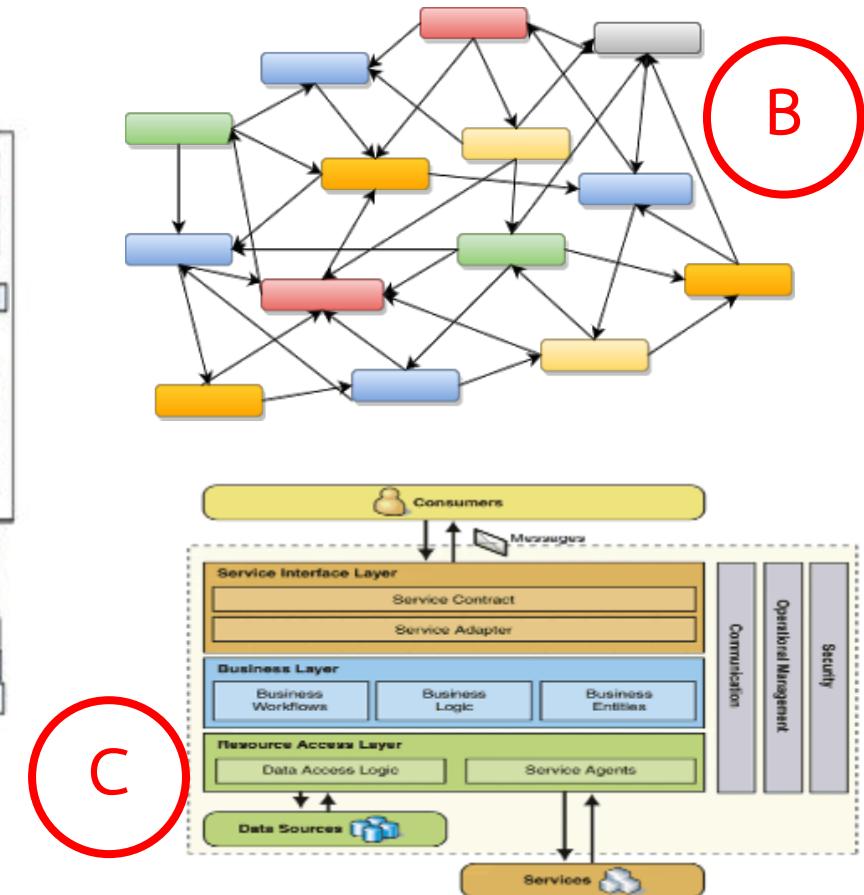
Which one you like most?

27.09.2019 in agenda 96



A

जेहाविजेन्द्र



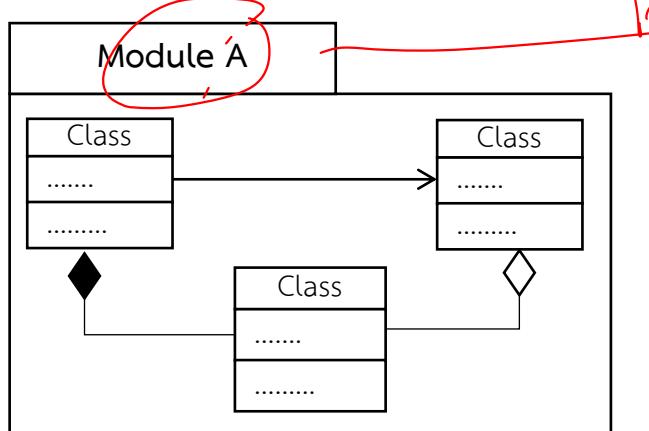
1

- Ref:**

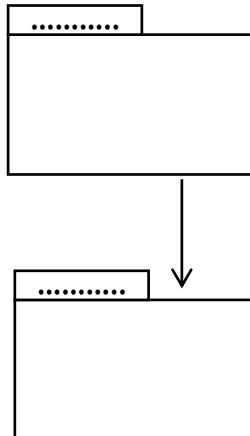
 - (image) <http://www.xclsoft.com/products.html>
 - (image) <https://www.javacodegeeks.com/2017/07/patterns-antipatterns-architecture.html>
 - (image) <https://robboxman.wordpress.com/2011/08/30/web-service-software-factory-beyond-the-service-interface-part-1/>

Software Design vs. Architecture

Software Design	Software Architecture
<p>สนใจลักษณะ class ของบุคคล, สนใจแก่ function หรือ module ของบุคคล</p> <p>สนใจเพียงแค่การออกแบบ Module หรือ Component หนึ่งๆ</p> <p>ดูที่หน้าที่และ function ของ Module หนึ่งๆ หรือ Class หนึ่งๆ</p>	<p>ออกแบบองค์รวมของระบบ และสนใจการประสานงานร่วมกันทั้งระบบ</p> <p>มองระดับ High Level และสามารถออกแบบได้โดยใช้หน่วยเก็บข้อมูลแบบใด</p>

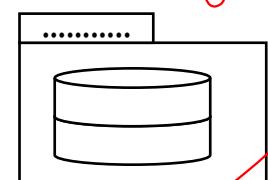


high
level



มองระดับ High Level
และสามารถออกแบบได้โดยใช้หน่วยเก็บข้อมูลแบบใด

จัดรวมต่อสิ่ง
module จัดการเชื่อม
ต่อภายนอก



จัดการเชื่อม กับ Infrastructure

Architectural Design

- สนใจการบริหารจัดการโครงสร้าง และ การจัดกลุ่มของ classes หรือ files ต่างๆ ในระบบซอฟต์แวร์
- การประสานงานระหว่างกลุ่มต่างๆ รวมถึง protocol และ โครงสร้างข้อมูล เพื่อใช้ในการสื่อสารระหว่างกัน
- คิดถึงเรื่อง Reuse เป็นสำคัญ
- เป็นภาพกว้างของกลุ่มต่างๆ ของระบบ
- เป็นโครงสร้างที่จะไม่แก้ไขบ่อย (ต้องทำให้ดีเป็นอันดับแรกของทุก Software Process)
- Software Architecture ที่ดีต้องเรียบง่าย สามารถนำไปสื่อสารให้ผู้อื่นเข้าใจได้ง่าย
- สามารถ maintain แต่ละส่วนได้ง่าย และประเมินความเสี่ยงผลกระทบได้
- มีหลากหลายรูปแบบ ให้เลือกใช้กับงานต่างๆ

ulgarnen's book (ปี 2558) ปรับปรุงครั้งที่ 1 หน้า 10

Architecture and System Characteristics

ស្ថិតិយោន្តិវិនិន័យ non-functional reqcm

• Performance

- Localize critical operations and minimize communications. Use large rather than fine-grain components.

• Security

- Use a layered architecture with critical assets in the inner layers.

• Safety

- Localize safety-critical features in a small number of sub-systems.

• Availability → ការអនុវត្តន៍ការងារនៃបច្ចេកទេស

- Include redundant components and mechanisms for fault tolerance.

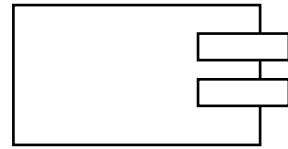
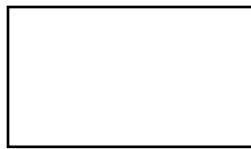
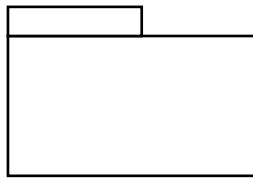
• Maintainability → ការថតដំឡើង Version នូវកម្មវិធីណែនាំ

- Use fine-grain, replaceable components.

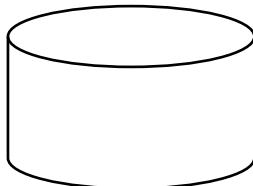
↓ library របៀប

Definitions

ໃຊ້ຫຼັກດັບ ແລ້ວ



Module หรือ Component หรือ ระบบຍ່ອຍ ที่
บรรจູ Classes หรือ Files ต่างๆ ที่ใช໌ทำงาน



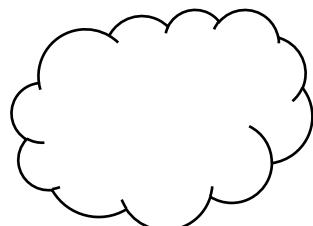
ຮ້ານຂໍ້ມູນ (Database)

ກຳນົດໃຫຍ່ການປະກາດ Class



ເສັ້ນທາງການ Request (ໄມ່ໃໝ່ data flow) ເພື່ອຮັບຂອ້າງຕ່າງໆ

ຂໍາຕາມເຮືອດີຍ * → ນາຍົກແຍ້ນໃນ dataflow



Internet หรือ Cloud

ပရောဂါန ၁

Layered Architecture

Layered Architecture

→ មិនគឺ software ទៅ standalone → គ្រប់គ្រង់
ឬតារាង ឬ ផ្លូវការណ៍នេះ

Presentation

ជីវិតជាអ្នកប្រើប្រាស់
ជីវិតជាអ្នកសំគាល់

User Interface

ឯកសារការងារ

Business Logic

រាយការណ៍នេះ

Middleware

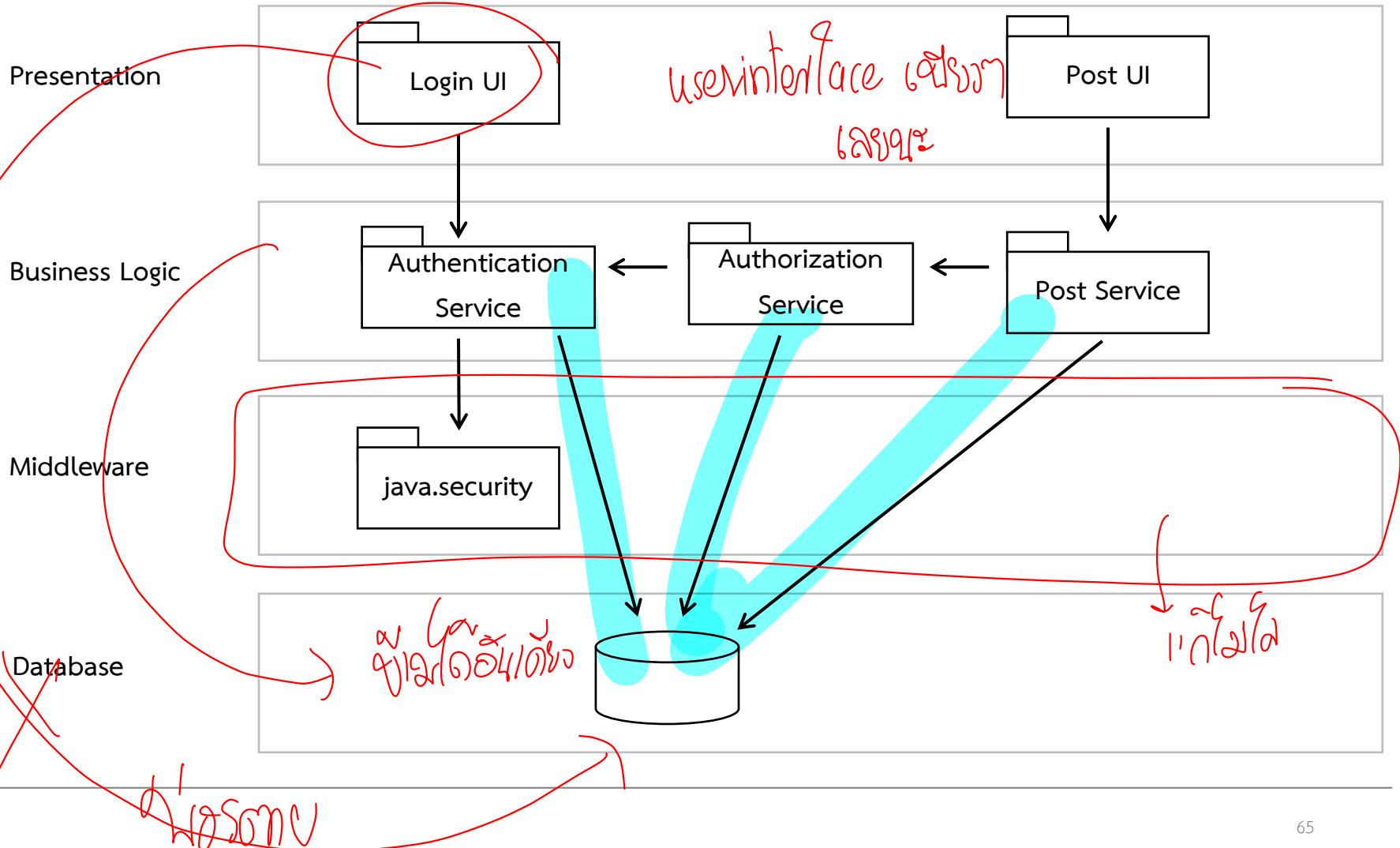
ជីវិតជាអ្នកប្រើប្រាស់
ជីវិតជាអ្នកសំគាល់

Database

កើតឡើង

Layered Architecture

ສູງລົງນະວັດຈິນ=ມາຈຳເປັດຢູ່ທີ່



Layered Architecture

- **รายละเอียด**

- แบ่ง Components ประเภทต่างๆ เป็น layer
- มีการทำงานกันระหว่าง layer ที่ติดกันอย่างชัดเจน

- **ข้อดี**

- เหมาะสำหรับ stand-alone app
- มีการแบ่ง component และ dependency ที่ชัดเจน สามารถเพิ่มเติม component ได้สะดวก

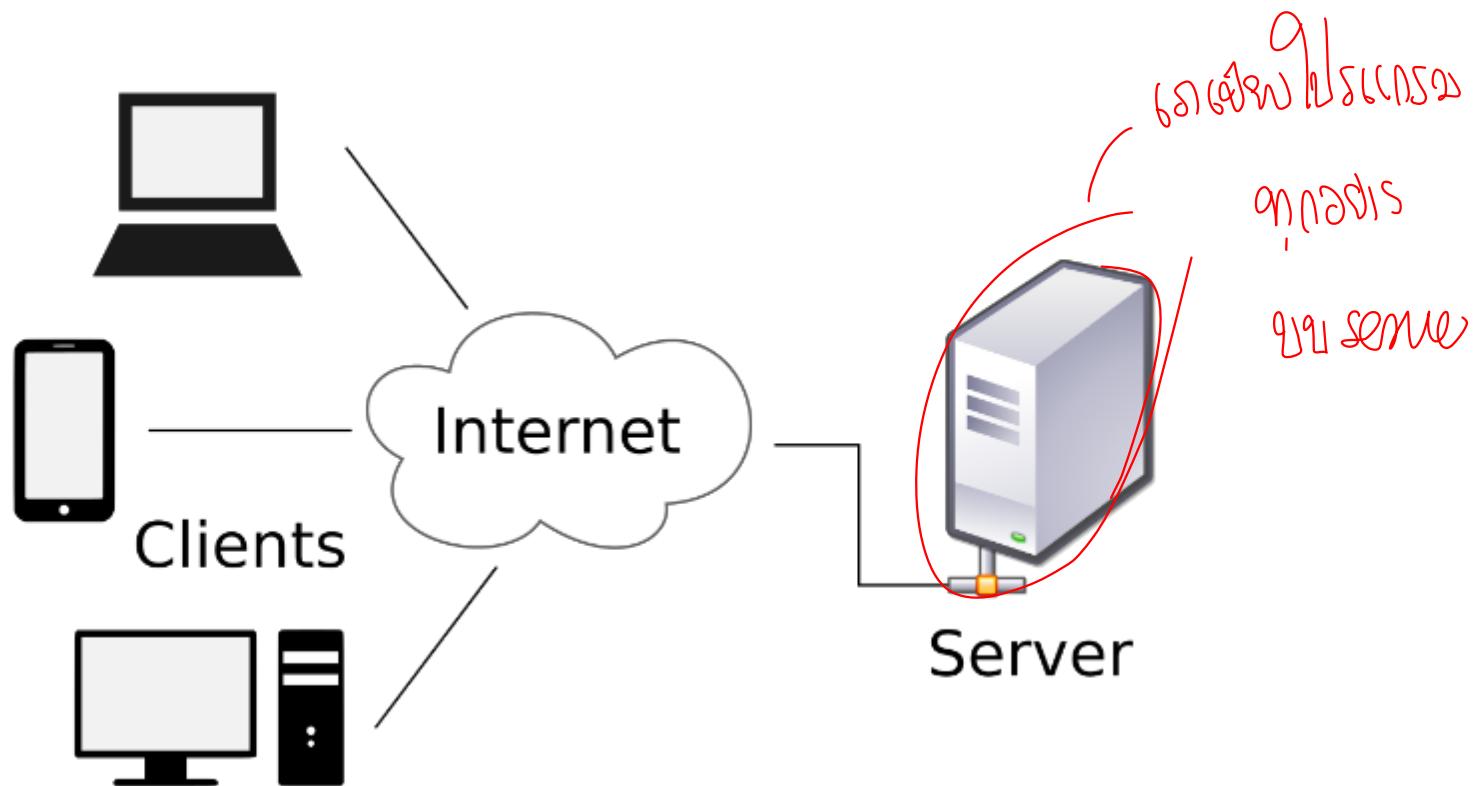
- **ข้อด้อย**

- ในทางปฏิบัติบางครั้งก็มีเหตุการณ์ที่ component จาก layer บนๆ เรียกใช้งานข้าม layer มากไปยัง layer ล่างๆ

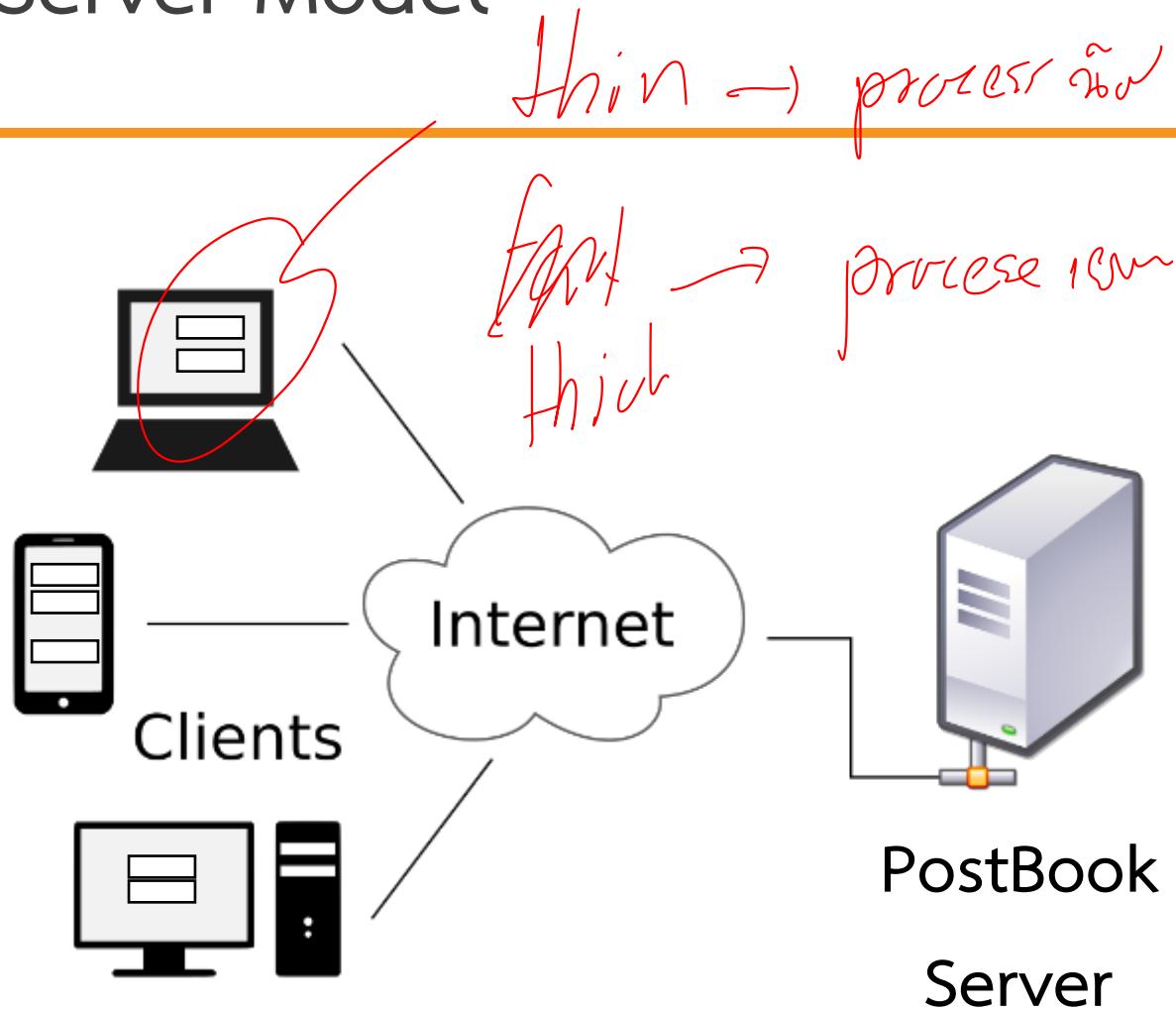


Client-Server Model

Client-Server Model



Client-Server Model



Client-Server Model

- **รายละเอียด**

- มีการแบ่งระบบ Client กับ Server อย่างชัดเจน และเรียกใช้งานข้าม network เช่น Internet หรือ Intranet
- Client จะเป็น application ที่ผู้ใช้ใช้งาน
- Server จะมี Service กลางที่ให้บริการ Client

- **ข้อดี**

- Servers สามารถกระจายข้าม network ได้

- **ข้อด้อย**

- มี server เป็น point of failure คือถ้า sever พัง client ก็ใช้งานไม่ได้

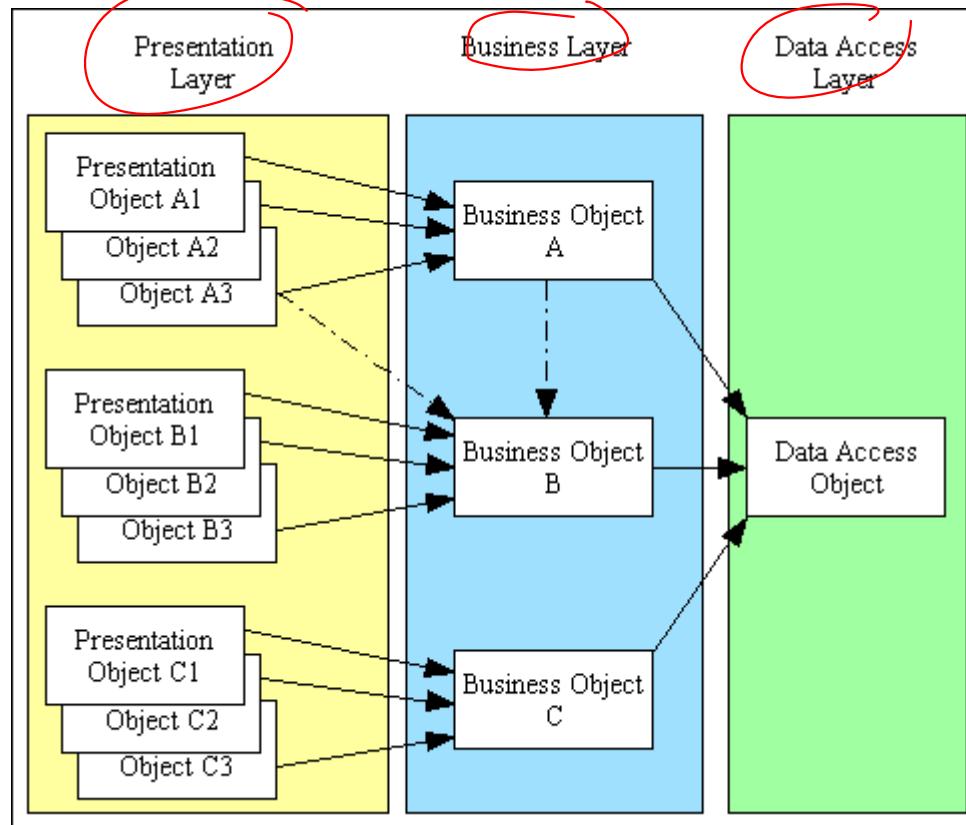
Falz

3-Tier Architecture

ପ୍ରସ୍ତୁତିକାରୀ client server

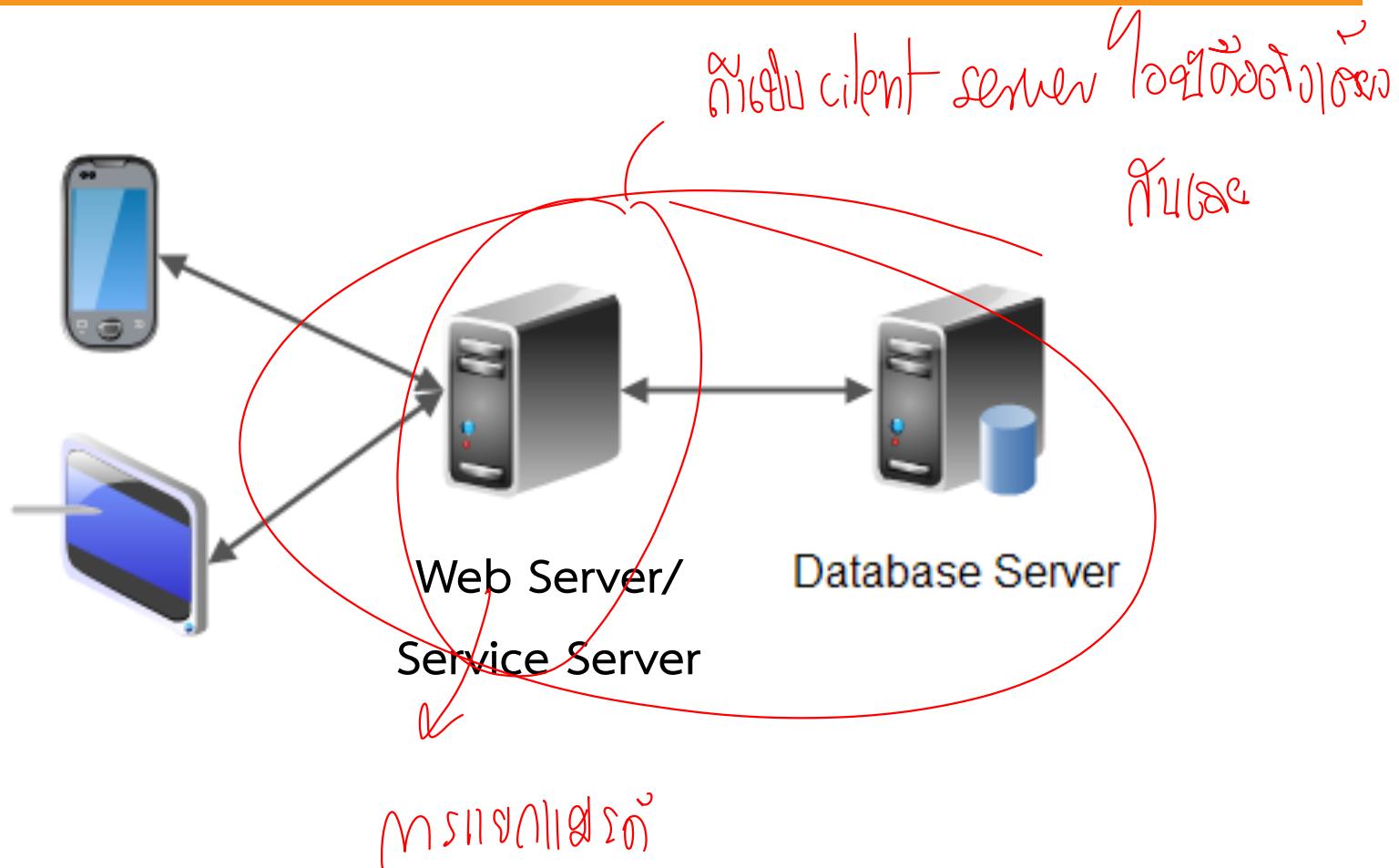
ଫୁଲ୍‌ସ୍କେମ୍ Web

3-Tier Architecture



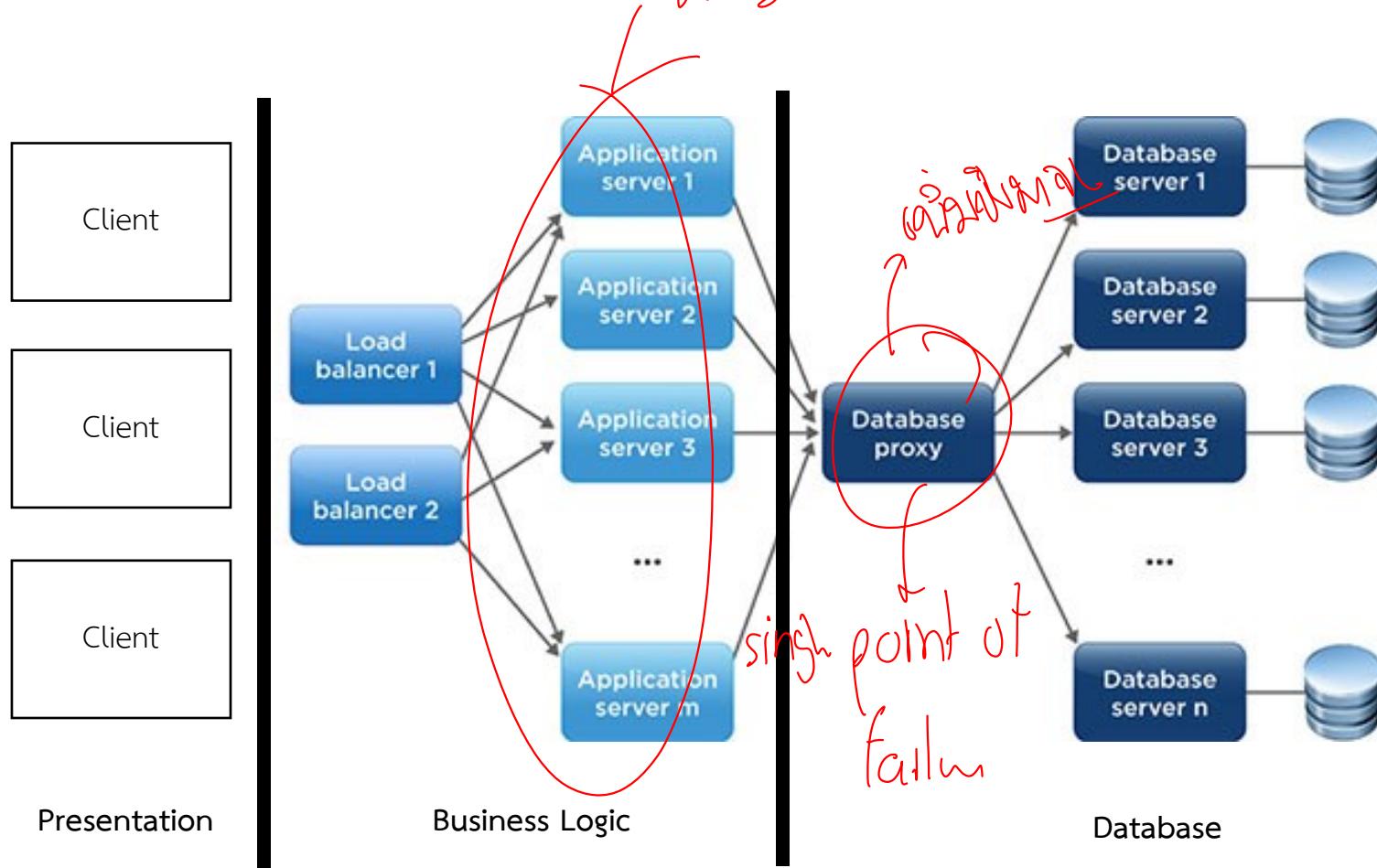
1 tier layer
 Firewall

3-Tier Architecture



Scaled 3-Tier Architecture

Q of logical failover
Q of unavailability



Scaled 3-Tier Architecture

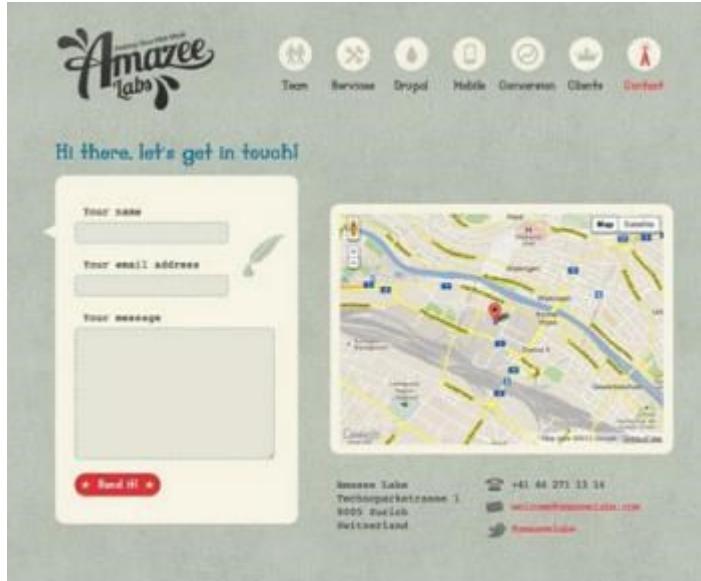
- รายละเอียด
 - มีการแบ่งระบบ Application, Business Logic และ Database อย่างชัดเจน
 - แต่ละส่วนสามารถอยู่บน servers และเรียกใช้งานข้าม network เช่น Internet หรือ Intranet ได้
- ข้อดี
 - เป็นรูปแบบที่ยืดหยุ่นและเป็นที่นิยมใช้สำหรับทำ Web Application
- ข้อด้อย
 - ถ้าออกแบบไม่ดีก็จะมี server เป็น point of failure โดยเฉพาะ Database Server

Service-Oriented Architecture (SOA)

SOA

- SOA เป็นกิจกรรมการให้บริการ (Service) จากฝ่ายผู้ให้บริการ (Service Provider) ไปยังอีกฝ่ายที่เป็นผู้รับบริการ (Consumer) ซึ่งทั้งสองฝ่ายอาจจะอยู่ในองค์กรเดียวกันหรือต่างองค์กรกันก็ได้ แต่ต้องมีการตกลงลักษณะการสื่อสารซึ่งกันและกัน
- ความสำคัญคือการให้ Service (ไม่ใช่ Application)
- ที่ต้องทำแบบนี้สาเหตุหนึ่งคือบาง functions นั้น ผู้พัฒนา application ไม่สามารถทำเองได้ หรือไม่คุ้มที่จะทำ เช่น การแสดงแผนที่ (เลือกใช้ service จาก Google Map)
- ดังนั้น Service Provider จะต้องคิดตั้งต้นว่า จะให้ผู้ใช้ได้ใช้ service เพื่ออะไร (ไม่ใช่ออกแบบหน้าจอให้ และต้องไม่ยึดติดว่าผู้ใช้จะใช้ Programming Language หรือ Platform อะไร)
ผู้ใช้ทางสุขคล่องตัว
- ส่วน Consumer จะนำ service เหล่านั้นไปใช้ในการพัฒนา Application เอง
consumer to service ผู้เช่าบริการ

SOA : Google Map Service



- การพัฒนา Map ด้วยทีมพัฒนาคน外องก์เป็นเรื่องที่ไม่เกินความสามารถ แต่ต้องใช้ effort สูงมาก อาจจะไม่คุ้มที่จะทำ
- ซึ่งถ้าเรียกใช้จาก Google Map ก็เพียงแค่เรียกใช้ code ตามด้านล่าง

```
<div id="map"></div>
<script>
    function initMap() {
        var map = new google.maps.Map(document.getElementById('map'), {
            center: {lat: -34.397, lng: 150.644}, zoom: 8 });
    }
</script>
<script src="https://maps.googleapis.com/maps/api/js?callback=initMap async defer"></script>
```

Service Interface Design

- **Design**
 - Involves thinking about the operations associated with the service and the messages exchanged
 - The number of messages exchanged to complete a service request should normally be minimized.
 - Service state information may have to be included in messages
- **Interface Design Stages**
 - Logical interface design
 - Starts with the service requirements and defines the operation names and parameters associated with the service. Exceptions should also be defined
 - Message design (SOAP)
 - For SOAP-based services, design the structure and organization of the input and output messages. Notations such as the UML are a more abstract representation than XML
 - The logical specification is converted to a WSDL description
 - Interface design (REST)
 - Design how the required operations map onto REST operations and what resources are required.

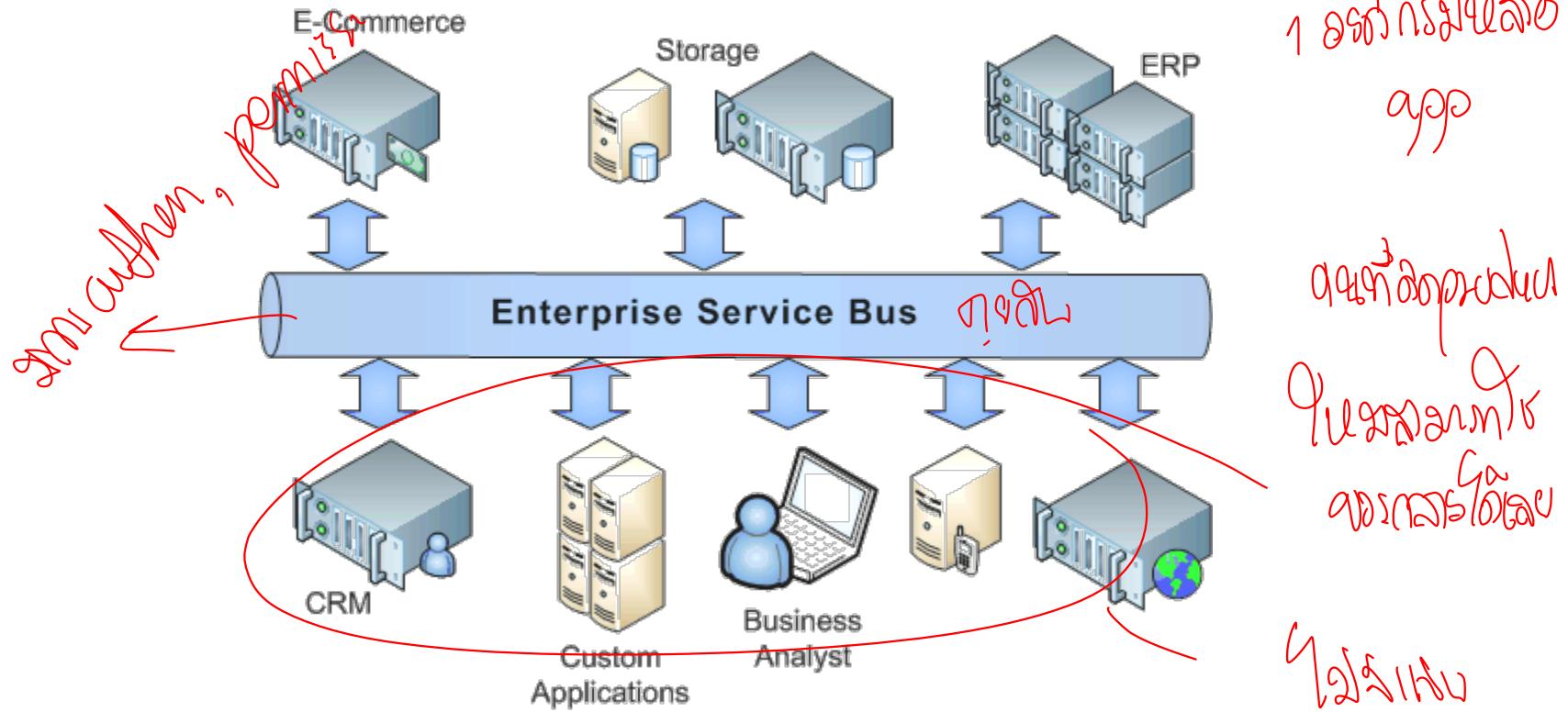
SOA : Google Map Service

- Operation **Distance Matrix**
- Input {
 origins: [{lat: 55.93, lng: -3.118}, 'Greenwich, England'],
 destinations: ['Stockholm, Sweden', {lat: 50.087, lng: 14.421}],
 travelMode: 'DRIVING',
 drivingOptions: {
 departureTime: new Date(Date.now() + N),
 trafficModel: 'optimistic'
 }
}
- Output
 - Duration (e.g. 2 hours 20 mins)
 - Distance (e.g. 100 km)
- Exception
 - Place not found

Enterprise Service Bus

- Service ที่มีการสื่อสารต่างกลางคือ ESB

ໃບ ຢັ້ງຢືນຂອງ



SOA

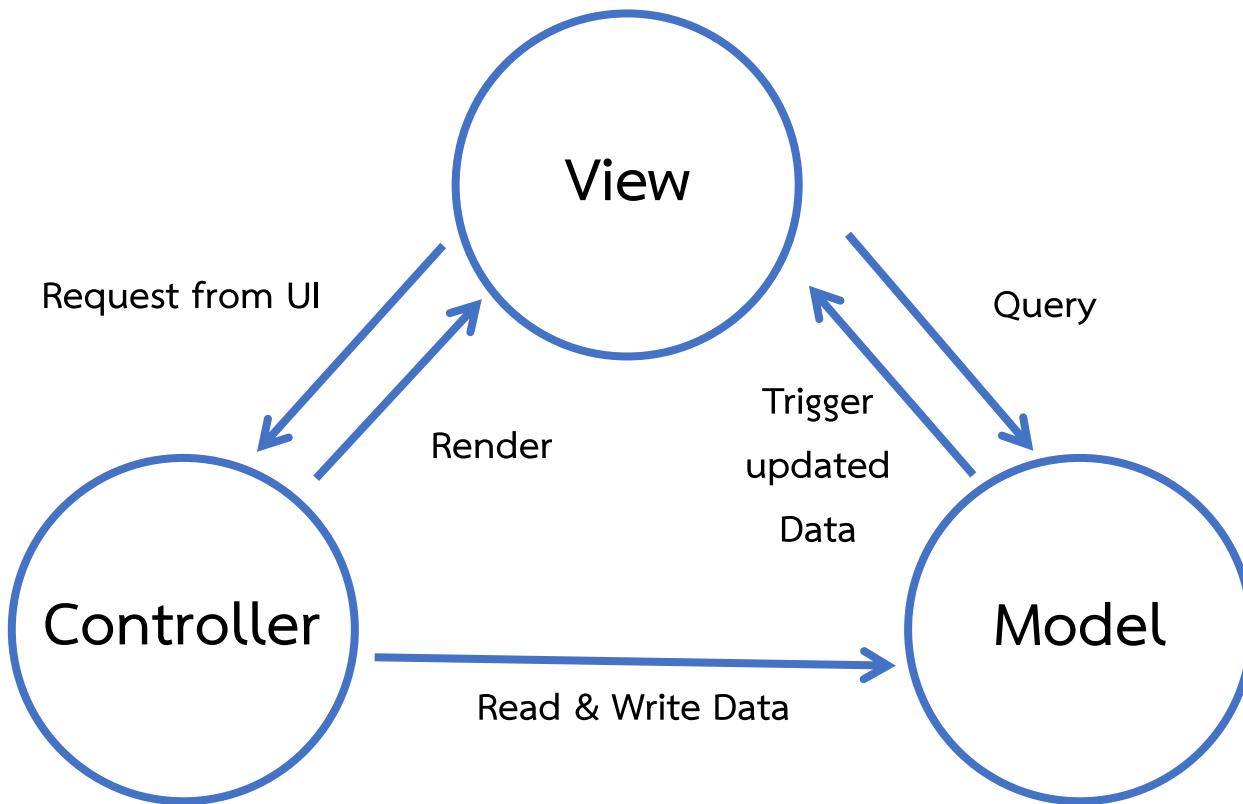
- รายละเอียด
 - แบ่งซอฟต์แวร์เป็น component ที่เป็น service
 - service provider เป็นผู้ให้บริการ ให้ consumer เรียกใช้
 - มี protocol สำหรับสื่อสารกันระหว่าง service และมีการสื่อสารผ่าน ESB
- ข้อดี
 - การใช้งานเป็นอิสระ ไม่ขึ้นอยู่กับ Application หรือ Platform
 - การออกแบบเข้าใจง่ายและแบ่งงานให้ทีมพัฒนาได้ง่าย
- ข้อด้อย
 - แต่ละ service ไม่ได้ถูกพัฒนาให้เป็นอิสระต่อกัน ยังมีส่วนที่ใช้ร่วมกัน เช่น database อยู่
 - ESB อาจกลายเป็น Single Point of Failure

Model-View-Controller (MVC)

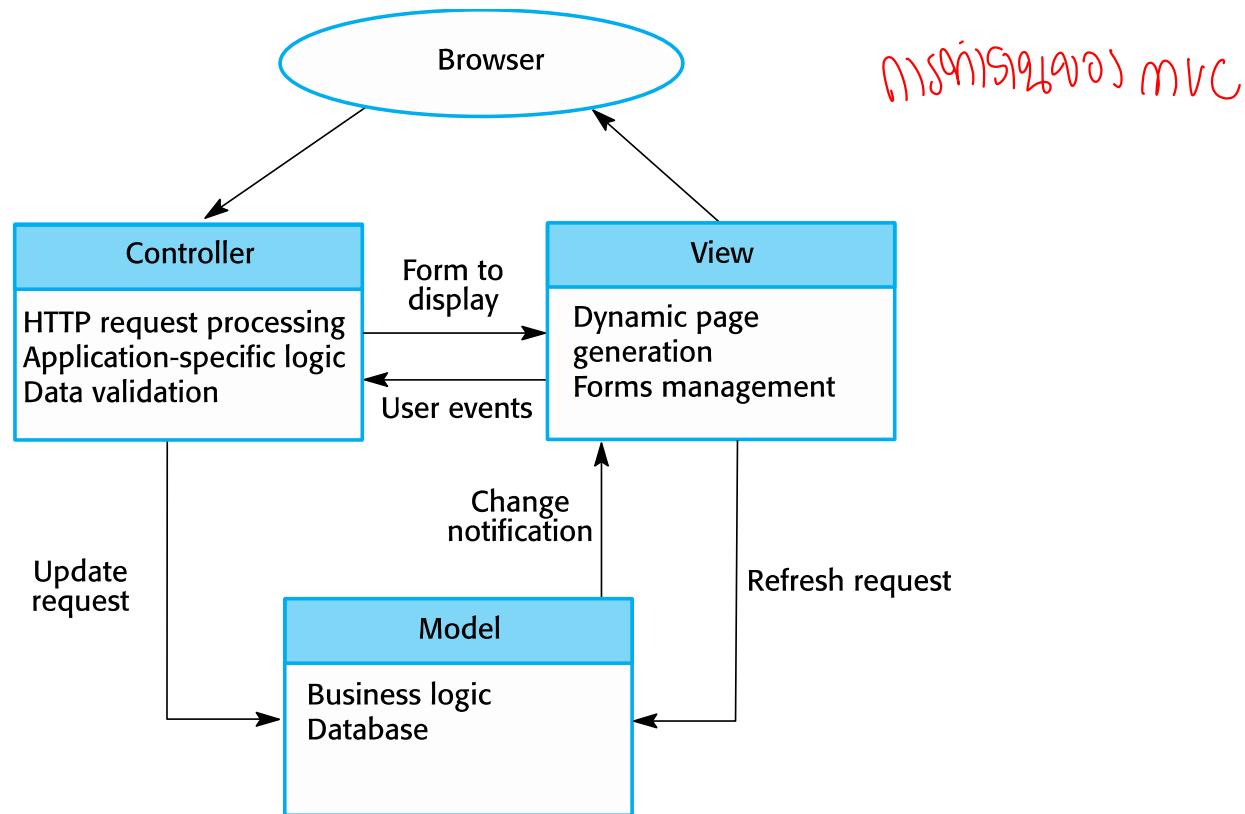
ເພື່ອກົງ standalone

ຂອງ framework ຂອງຕົວຢີ

MVC



Web Application Architecture using MVC

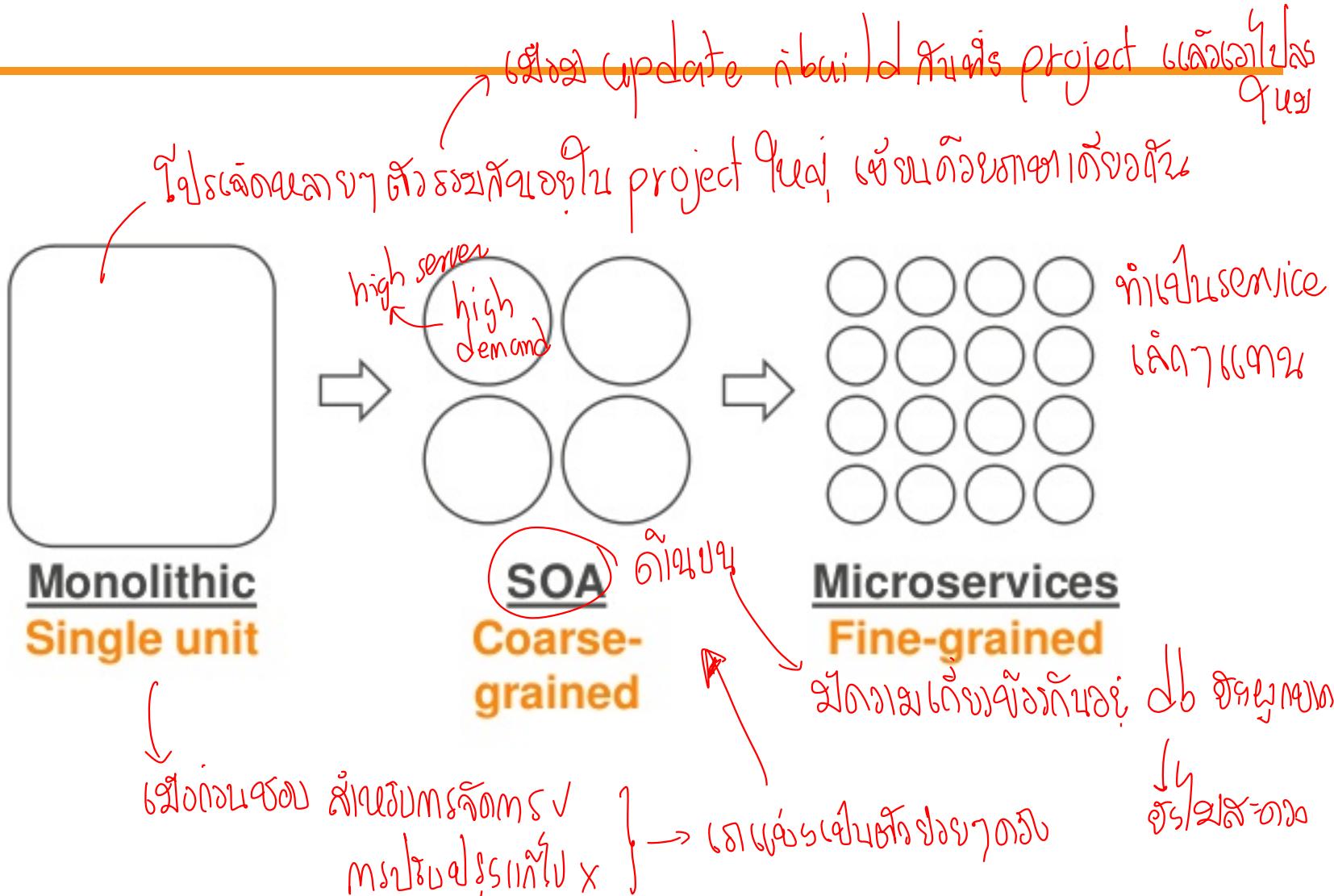


MVC

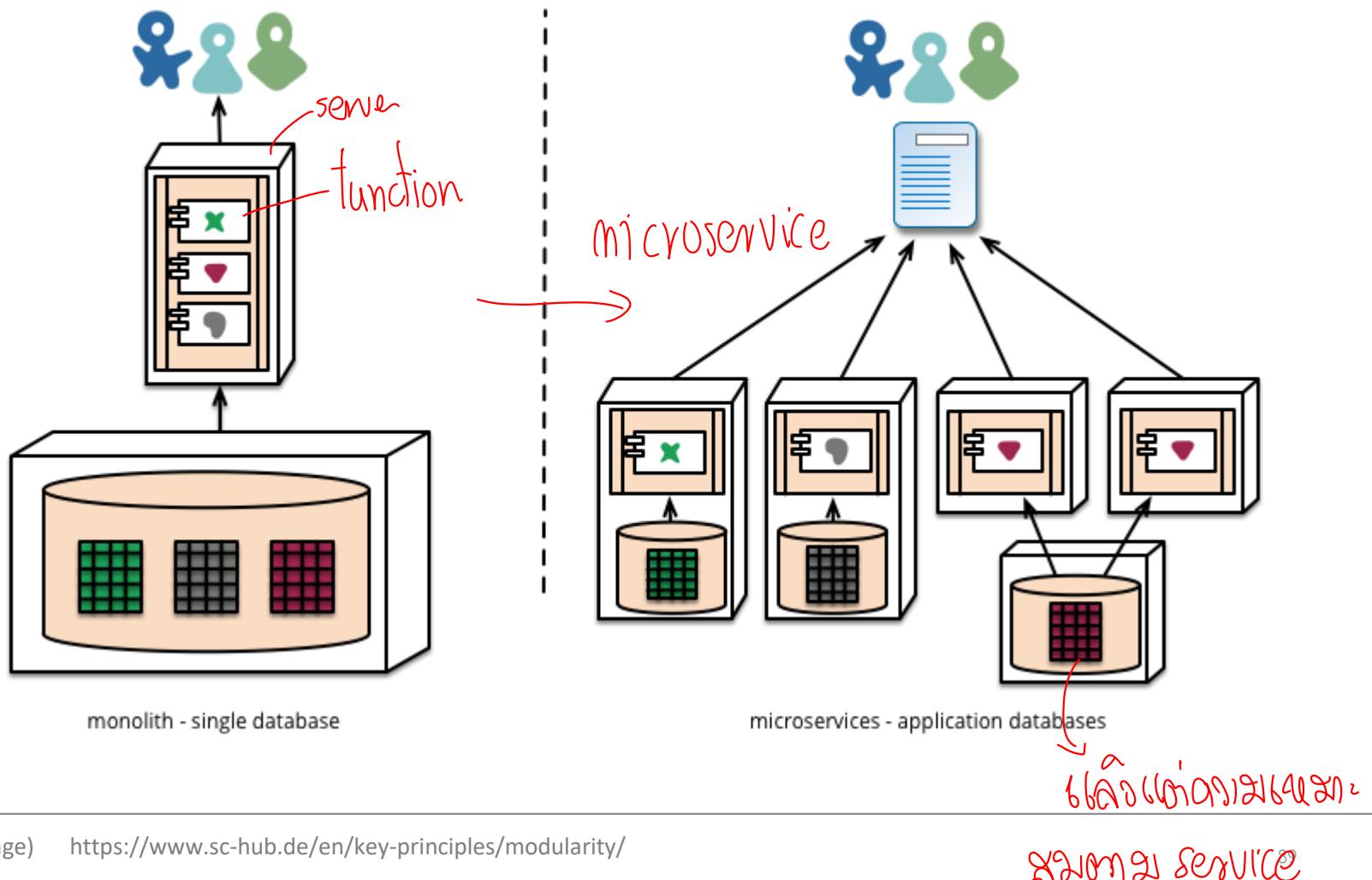
- รายละเอียด
 - แยกส่วน Presentation และ Interaction ออกจาก Data
 - มีใช้ใน Django, ASP.NET MVC, Rails, AngularJS, EmberJS, Backbone เป็นต้น
- ข้อดี
 - สามารถเปลี่ยน UI (view) ได้ โดยใช้ controller และ model ของเดิม
- ข้อด้อย
 - การเขียนโปรแกรมซับซ้อนขึ้น เพราะต้องแยก View ออกจาก Controller อย่างเด็ดขาด

The word "Microservice" is written in a bold, dark brown sans-serif font. It is partially obscured by a light green, semi-transparent cloud-like shape that has irregular edges and overlaps the letters.

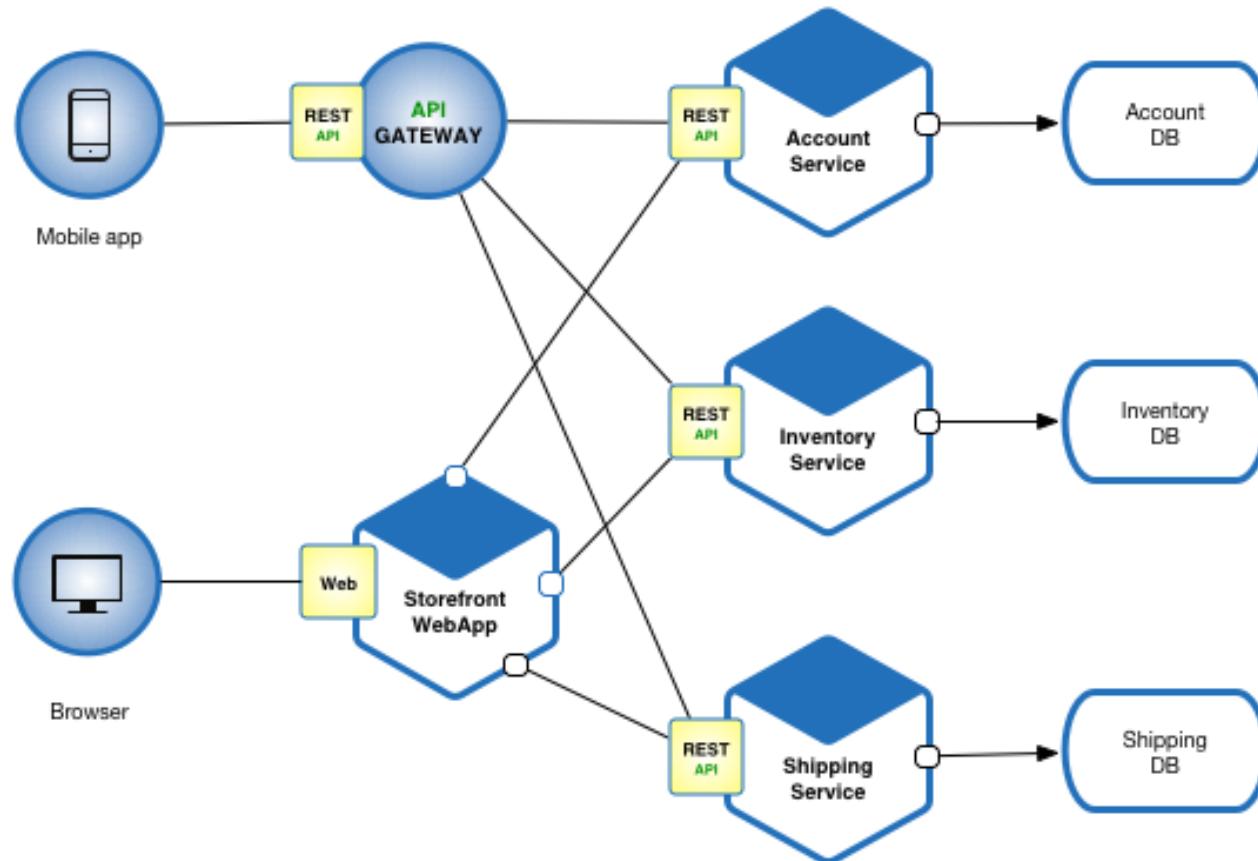
To be Microservice



Monolith vs. Microservice

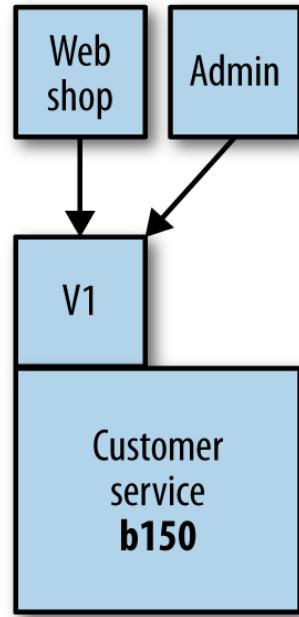


Microservice

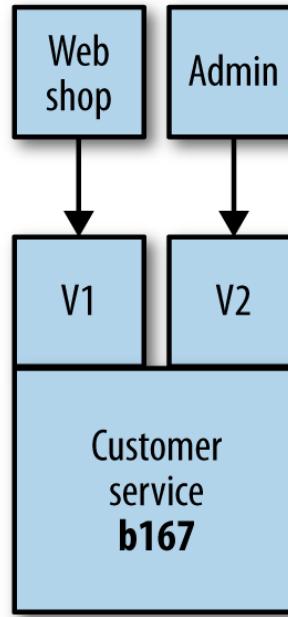


Services from Versions

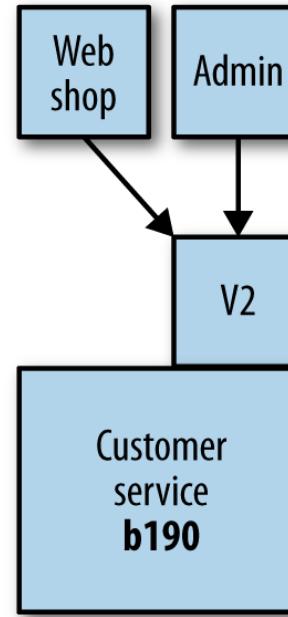
នៅលើមសនិយករាយសេវា ទាំងឡាយ version



Single endpoint supported



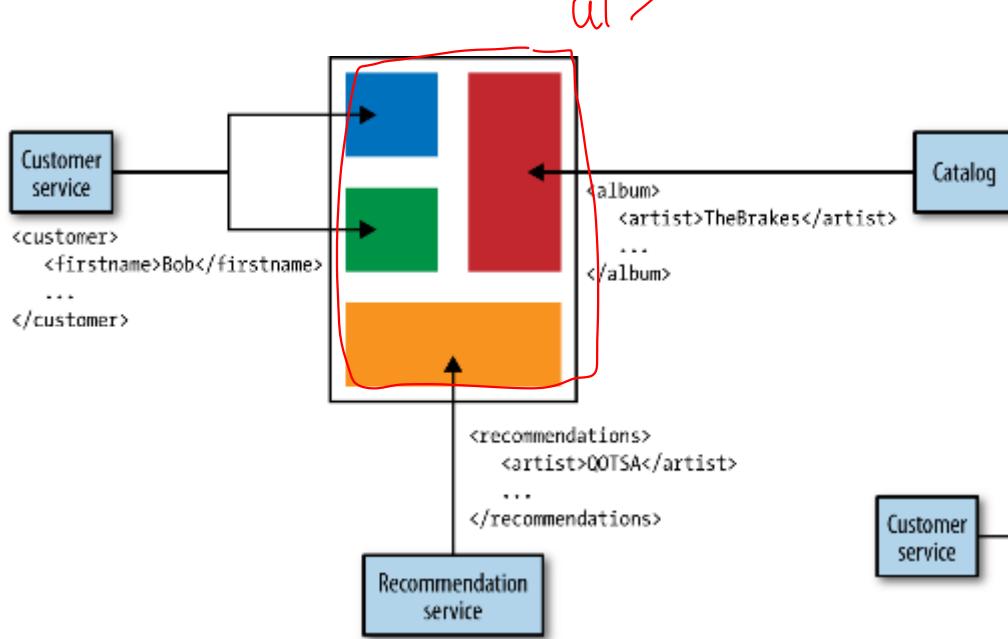
New release exposes
and additional
endpoint



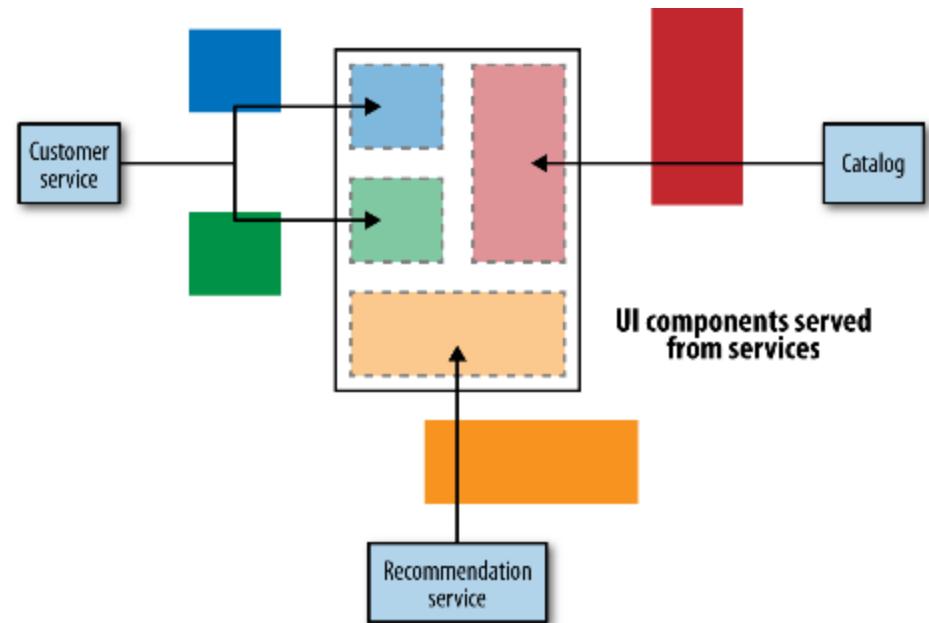
Once the old endpoint
is no longer used, a new
release of the service
can remove it

នៃសម្រាប់
V2

Microservice and UI



ເພື່ອໃຊ້ ກໍາລັງດາຕາ ຂໍມູນຂີ້າ = service ລົງທະບຽນ
ກໍາລັງ ເປົ້າ ສິນຄ້າ ໂດຍມີມູນຄວາມຮັດວຽກ
ກໍາລັງ ໃຫ້ ອົບສິນ ທີ່ໄດ້ຮັດວຽກ



Microservice

- รายละเอียด

- แบ่งระบบเป็น service ย่อยๆ โดยที่แต่ละ service จะติดต่อ data storage ที่เหมาะสมกับงาน

- ข้อดี

- ซอฟต์แวร์มีขนาดเล็ก
- สามารถพัฒนาคนละ programming language ได้ → ตาม需求数量 ตามความเชี่ยวชาญ
- แต่ละ service ใช้ data storage คนละประเภทตามความเหมาะสมได้
- สามารถขยายหรือปรับสัดส่วนการใช้ทรัพยากรของแต่ละ service ได้

- ข้อด้อย

- การบริหารจัดการความถูกต้องของข้อมูลใน database ยากขึ้น
- ต้องจัดการเรื่อง service version ให้รอบคอบ

จัดการ database ยากขึ้น หลายตัว
หลายตัว

เข้าชุด controller → ภายใน micro service ตัวเดียว → 1 container 1 service

↳ container server

Distributed Systems

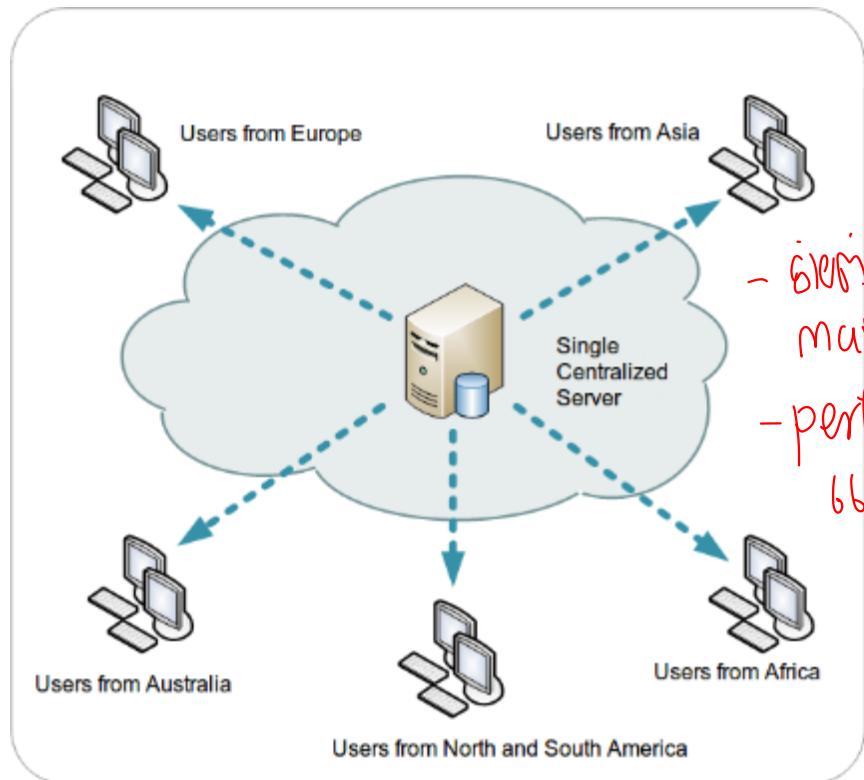
↳ ឧបតម្យលេង ឬជារុណាស់សំគាល់លេង

កន្លែងដ្ឋានការសេវា និង ផែនការ

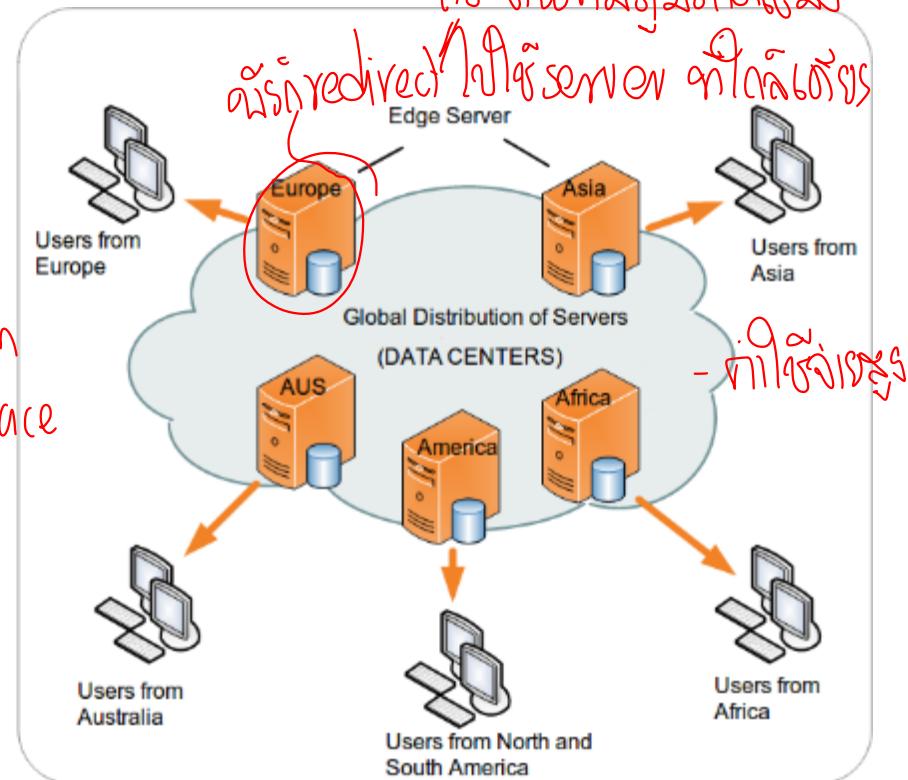
Centered Server vs. Distributed Servers

↳ ជួយក្រសាន្តរៀង ត្រូវបានចិត្តចាប់ផ្តើមទៅស្ថិត ដោយ
↳ server farm ការណា

- ហាំង video hosting services យូរដែល YouTube ត្រូវបានចិត្តចាប់ផ្តើមទៅស្ថិត ដោយប្រើប្រាស់ bandwidth មហាផាល



- គិតថាអាមេរិក
- performance
66%

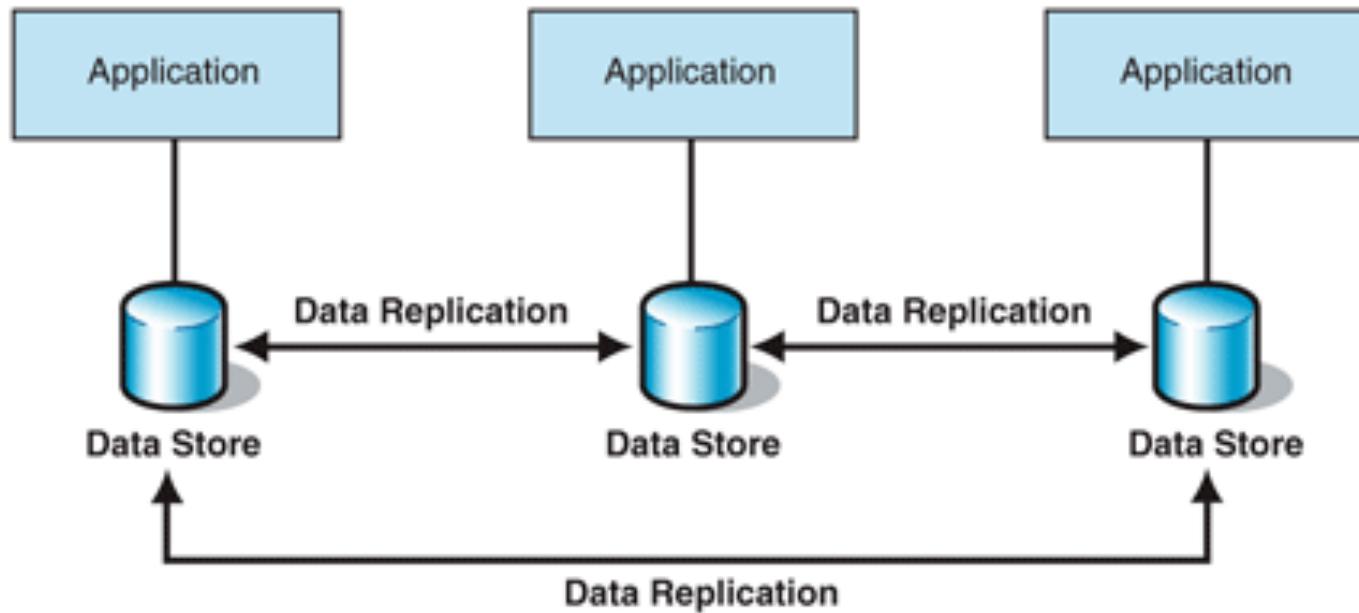


- ms syn ជីវិនុល (អាជីវកម្ម)

Data Replication

ໃຊ້ລາຍງືນທີ່ຈະຢູ່ຕະຫຼາມໄວ ບໍລິຫານດັບສິນລົດໂຮງ, ວິຊີ່

- การ update ຂໍ້ມູນໃຫ້ແກ່ມືອນກັນທຸກໆ site



Distributed Systems

- รายละเอียด
 - มีการแบ่งระบบเป็น site โดยที่จะมี
 - ในแต่ละ site จะมี database server เป็นของตนเอง และมีการ update ระหว่างกัน
 - ข้อมูลในแต่ละ site อาจจะไม่จำเป็นต้องเหมือนกันก็ได้
- ข้อดี
 - สามารถรองรับผู้ใช้ตามภูมิภาคได้อย่างดี
 - หาก site หนึ่งมีปัญหา ก็ยังสามารถปรับให้ผู้ใช้ไปใช้ site อื่นได้
- ข้อด้อย
 - การทำ data replication ตลอดเวลาเป็นเรื่องที่ดี เพราะผู้ใช้จะได้รับ data ที่ทันสมัยและถูกต้อง แต่จะทำให้เกิด traffic และสร้างปัญหาทาง network ได้ ดังนั้นจึงต้องกำหนด policy การทำ replication ให้รอบคอบ

Summary

Summary

- Software Architecture
- Layered Architecture
- Client-Server Model
- 3-Tier Architecture
- Service-Oriented Architecture (SOA)
- Model-View-Controller (MVC)
- Microservice
- Distributed Systems

“

Any work of architecture that has with
it some discussion, some polemic,
I think is good. It shows that people are
interested, people are involved.

”

Richard Meier

つづく