Supply Chain Management
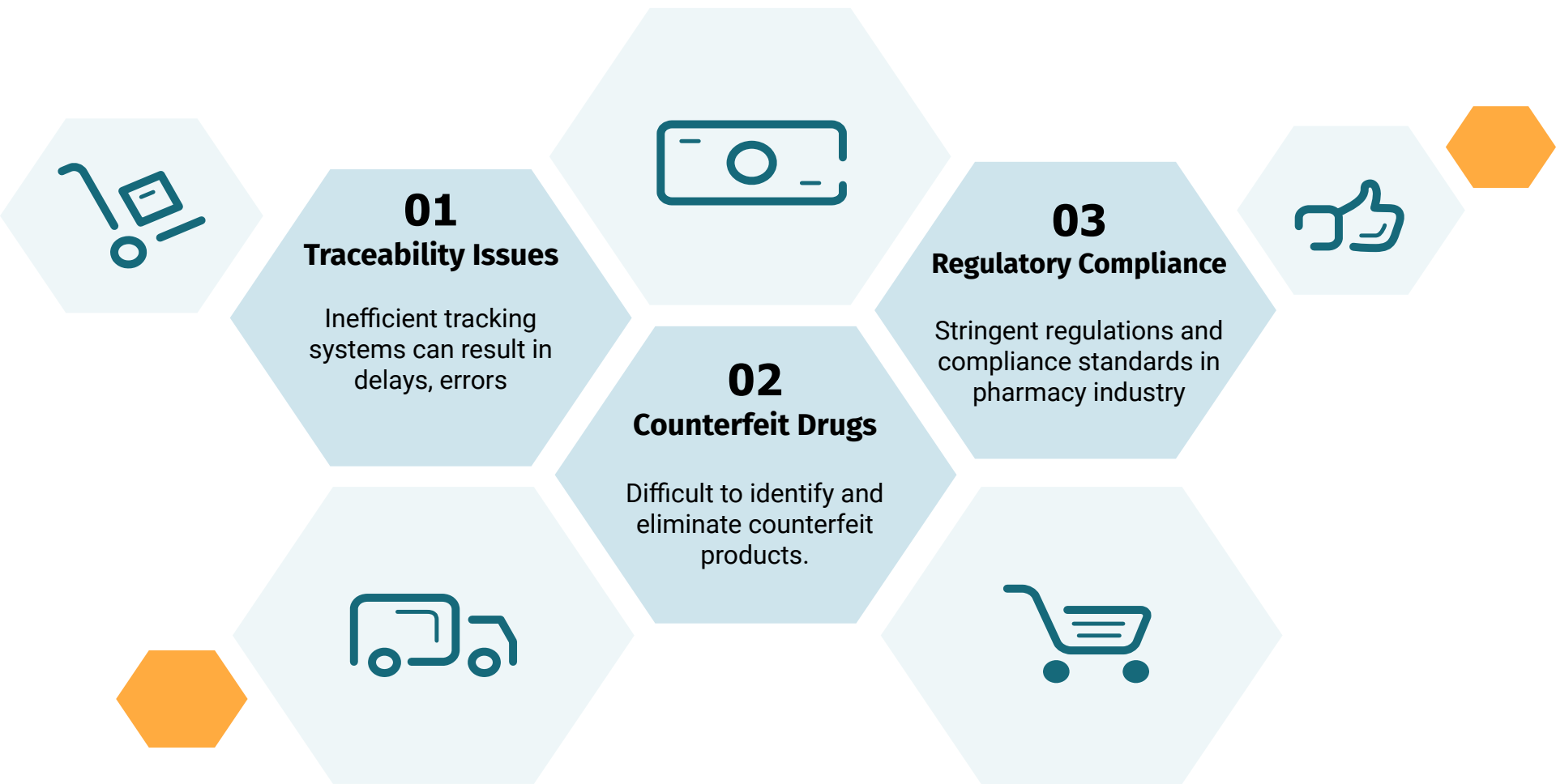
Hyperledger Fabric use case in pharmaceutical industry
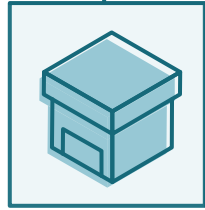
Challenges in current pharmaceutical industry

01
Traceability Issues

Inefficient tracking systems can result in delays, errors

02
Counterfeit Drugs

Difficult to identify and eliminate counterfeit products.

03
Regulatory Compliance

Stringent regulations and compliance standards in pharmacy industry

Why this use cases fit hyperledger fabric?
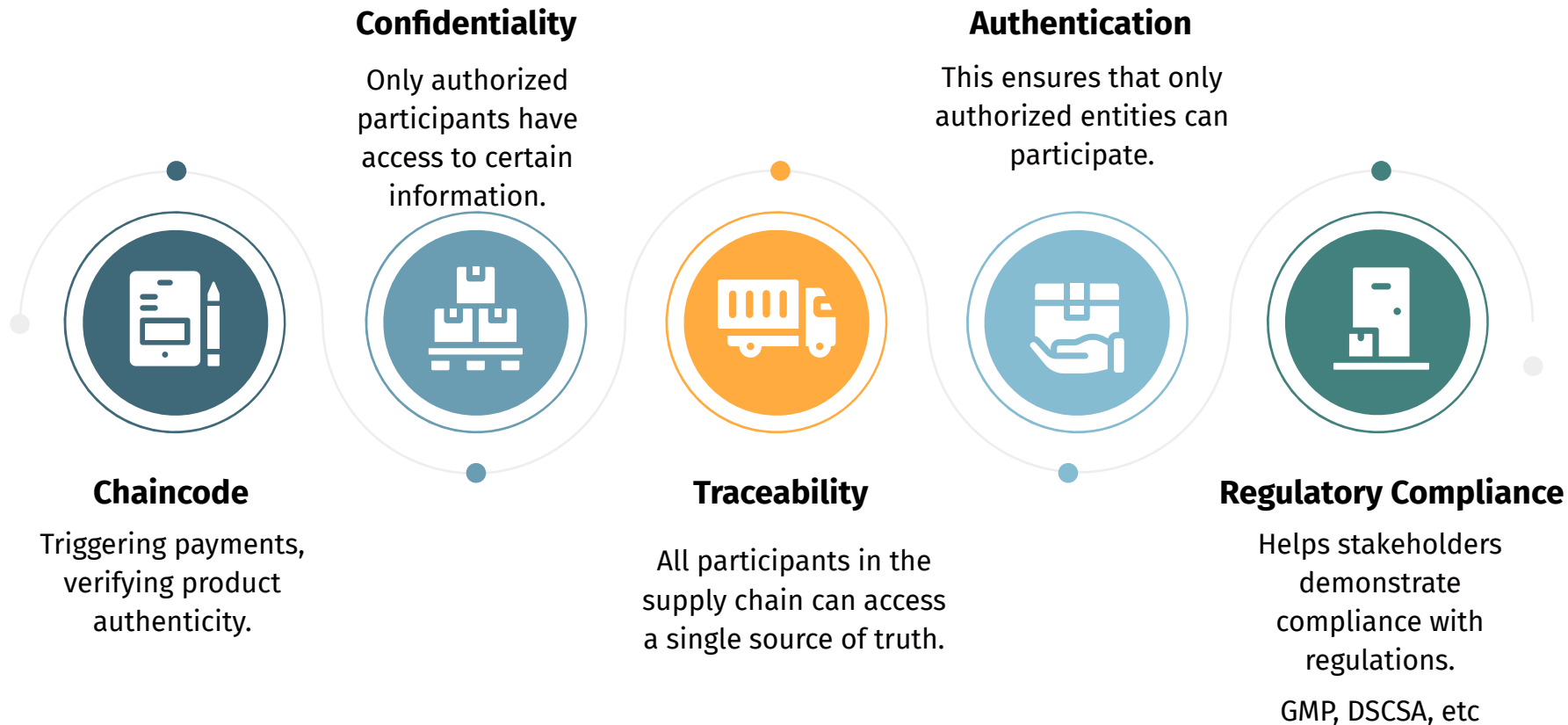
Confidentiality

Only authorized participants have access to certain information.

Authentication

This ensures that only authorized entities can participate.

Chaincode

Triggering payments, verifying product authenticity.

Traceability

All participants in the supply chain can access a single source of truth.

Regulatory Compliance

Helps stakeholders demonstrate compliance with regulations.

GMP, DSCSA, etc
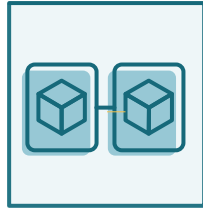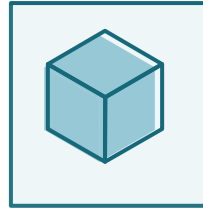
State Diagram

Organization 1 - Drugs Manufacturer
Organization 2 - Drugs WholeSaler
Organiztion 3 - Drugs Pacakger
Organization 4 - Drugs Distributor
Organization 5 - Pharmacy
MSP - Membership Service Provider
CC - ChainCode Smart Contract
CA - Certificate Authority

Blockchain Network

ORG1 MSP

ORG2 MSP

CC1 CA1

CC2 CA2

Peer 1 Ledger

Peer 2 Ledger

Raft Orderer

Fabric SDK

channel

Peer 5 Ledger

Peer 4 Ledger

Peer 3 Ledger

CA5 CC5

CA4 CC4

CA3 CC3

Fabric API

ORG5 MSP

ORG4 MSP

ORG3 MSP

User Interface

Blockchain Network
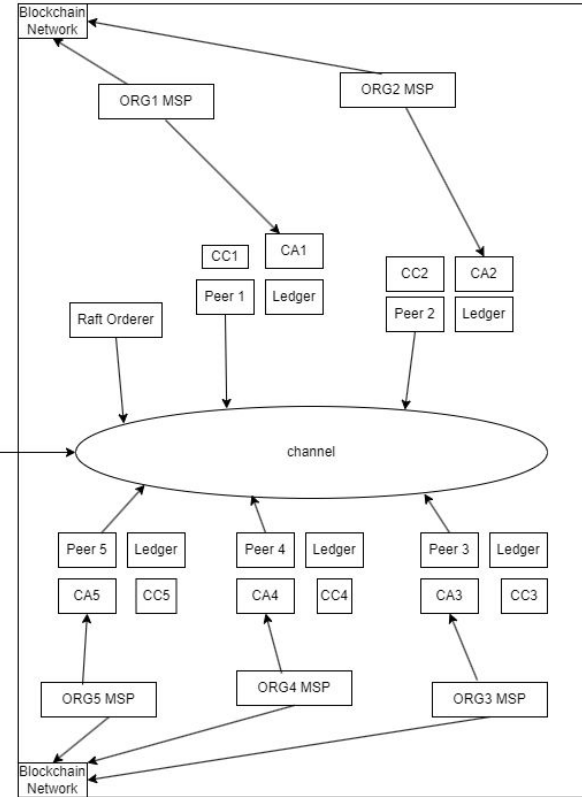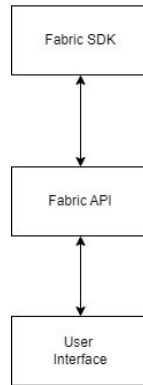
Chaincode

01

Invoke

Update the holder of the drug products

02

Query One

Real one specific product

03

Query

Read all the status of the product in the supply chain

```go
type Drug struct {
    Date      string `json:"id"`
    Timestamp string `json:"timestamp"`
    Location  string `json:"location"`
    Holder    string `json:"holder"`
}
```

```go
func (s *SmartContract) recordDrug(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {

    if len(args) != 4 {
        return shim.Error("Incorrect number of arguments. Expecting 5")
    }

    var drug = Drug{ Date: args[1], Location: args[2], Timestamp: args[3], Holder: args[4] }

    drugAsBytes, _ := json.Marshal(drug)
    err := APIstub.PutState(args[0], drugAsBytes)
    if err != nil {
        return err
    }

    return shim.Success(nil)
}
```

```go
func (s *SmartContract) changeDrugHolder(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {

    if len(args) != 2 {
        return shim.Error("Incorrect number of arguments. Expecting 2")
    }

    drugAsBytes, _ := APIstub.GetState(args[0])
    if drugAsBytes == nil {
        return shim.Error("Could not locate drug")
    }
    drug := Drug{}

    json.Unmarshal(drugAsBytes, &drug)
    drug.Holder = args[1]

    drugAsBytes, _ = json.Marshal(drug)
    err := APIstub.PutState(args[0], drugAsBytes)
    if err != nil {
        return shim.Error(fmt.Sprintf("Failed to change drug holder: %s", args[0]))
    }

    return shim.Success(nil)
}
```

Our frontend sample

Hyperledger Fabric Drug Application

Query All Drug

[Query]

ID	Timestamp	Holder	Location	Date
1	12:34:56	ABC Pharmaceuticals	Canada	2024-02-21
2	14:54:56	Wholesaler	USA	2024-02-22

Query a Specific Drug

Enter a Product number:

[Ex: 3]

[Query]

Our frontend sample

Query a Specific Drug

Enter a Product number:

| 1 |

[Query]

Timestamp	Holder	Location	Date
1	ABC Pharmaceuticals	Canada	2024-02-21

Our frontend sample

Create Drug Record

Enter id:

```
3
```

Enter name of date:

```
2021-02-23
```

Enter location:

```
Canada
```

Enter timestamp:

```
4927497276
```

Enter holder:

```
Manufacturer
```

Create

Success!

Our frontend sample

Query All Drug

Query

ID	Timestamp	Holder	Location	Date
1	12:34:56	ABC Pharmaceuticals	Canada	2024-02-21
2	14:54:56	Wholesaler	USA	2024-02-22
3	16:45:24	Manufacturer	Canada	2021-02-23

Query a Specific Drug

Enter a Product number:

Ex: 3
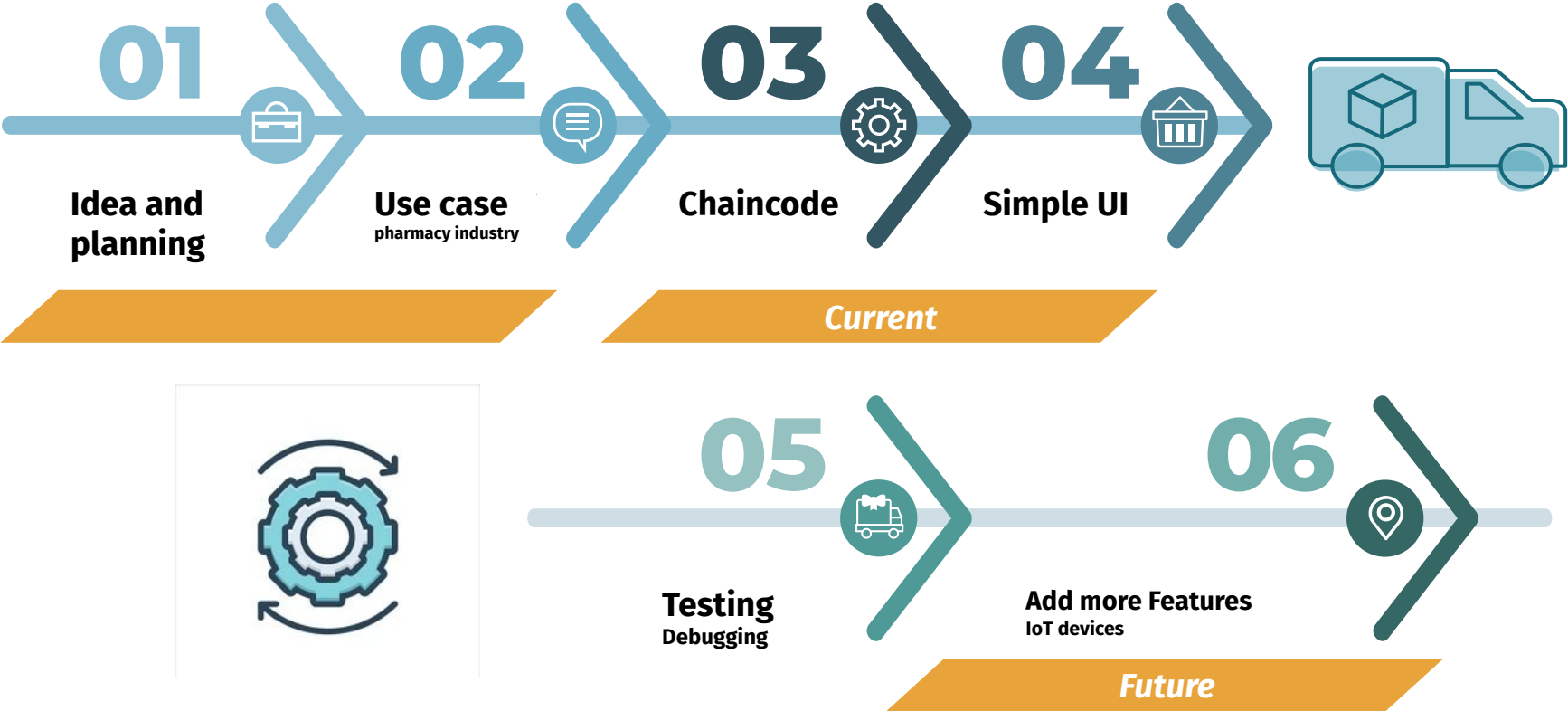
Query

Our frontend sample

Change Drug Holder

Enter an id:

> 3

Enter name of new holder:

> Wholesaler

[Change]

Success! Tx ID: f21a0ea5925aed6fbc604325303c41b445ea525570d2bb49e8f755cc9e584e33

Future planning

01
Idea and planning

02
Use case
pharmacy industry

03
Chaincode

04
Simple UI

Current

05
Testing
Debugging

06
Add more Features
IoT devices

Future

Thank You
END