UNIT- III

Architecting Block chain solutions: Introduction, Obstacles for Use of Block chain, Block chainRelevance Evaluation Framework, Block chain Solutions Reference Architecture, Types of Blockchain Applications. Cryptographic Tokens, Typical Solution Architecture for Enterprise Use Cases,Types of Block chain Solutions, Architecture Considerations, Architecture with Block chainPlatforms, Approach for Designing Block chain Applications.

Obstacles for Use of Blockchain

Introduction: -

Blockchain is not going to replace existing technologies; rather, it is going to complement or work in tandem with existing technologies.

The primary driver in using blockchain for most of the use cases would be to create a new business model or to optimize a business process; blockchain comes with its own benefit and challenges.

It is extremely important to perform the assessment to check the fitment of the use case since blockchain comes with its added complexities that are either absent or have minimal impact on other technologies.

Obstacles for use of Blockchain include the following:

-Collaboration between Competing Stakeholders
-Change in Business Process
-Cost of Implementation
-Time-to-Market (TTM) is High

Collaboration between Competing Stakeholders: -

Blockchain most often involves bringing together multiple stakeholders that may not fully trust each other for enterprise use cases. This is one of the most important challenging factors in blockchain implementation.

When there is a need to share data between competing stakeholders, they will be very skeptical of sharing the data with the competitors and they have to work in close collaboration for the use case to be successfully implemented in blockchain.

Rather, bringing these stakeholders to a table for discussion in itself can be a challenge. Added to this is the requirement that participants need to convince their sponsors of business case to fund the initiative.

Change in Business Process: -

Introducing blockchain may sometimes requires a modification to the business process. For example, a Provenance use case may require an additional step or step modified on bow the asset information is captured.

These business process leads to the aspect of organizational change management, which creates challenges in realizing the business value.

Cost of Implementation: -

The overall cost of implementing and testing a blockchain application is relatively high compared with other technologies.

Also, extreme care has to be taken in coding and testing the smart contracts. Proper auditing needs to be performed for the smart contracts.

In case of enterprise use cases, each organization needs to dedicate resources.

Time-to-Market (TTM) is High: -

The time needed for releasing blockchain-based solution is more than traditional software solution. Coding and testing a smart contract consume more time compared to other technologies.

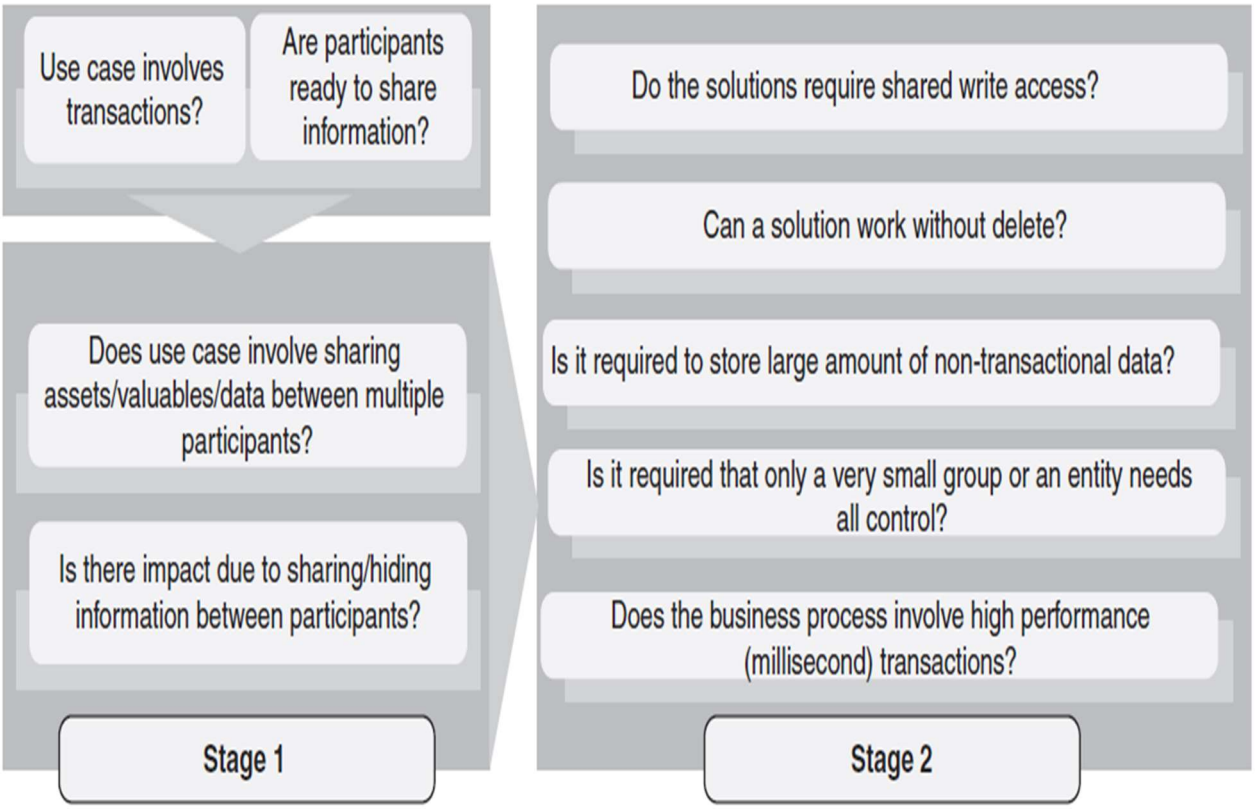
More than Software development related activities, identification and articulating the problem statement and laying the overall solution take additional time in blockchain-related solutions.

Block chain Relevance Evaluation Framework

Introduction: -

- The blockchain relevance evaluation framework is a two-stage framework, and if the first stage is not passed, we need not go to the second stage.
- Many criteria can be used in the assessment of blockchain for the use case.

- On the foundation level, some key features need to be considered in the assessment.
- The framework mentioned here is more techno functional than a pure business focus, which helps as there are business as well as technology levers that impact realization of blockchain solution.



- There are multiple solutions that are being developed to accommodate more scenarios that can be supported at stage 2 questionnaire.
- It needs to be noted that while it is kind of mandatory to pass stage 1 set of questions, it is not really required that all stage 2 answers are positive.
- In short, stage 1 can be considered the filtering criteria, but stage 2 is more a of selection criteria.

Before starting, it is important that the use case under consideration involves some kind of transactions where three or more participants are ready to share information with each other, if it is not true then blockchain is the wrong technology under consideration.

Stage 1.a: Does Use Case Involve Sharing Assets/Valuables/Data Between Multiple Participants?

Blockchain makes the most sense when the business process involves sharing assets or data between participants from different organizations. In case the complete use case does not involve participants external to the organization, then it is better to implement the use case using a centralized database solution since the organization controls the data and everybody inside it trust the employer. Besides, it is cheaper and faster to implement using a centralized solution. However, if there are use cases where the organization shares the data with the external participants, then blockchain makes sense. We will explain in subsequent sections about the type of blockchain to be used based on the use case.

Stage 1.b: Is There Impact due to Sharing/Hiding Information Between Participants?

Multiple complexities. are involved in today's business processes. Some of them are as follows:

1. Enterprise has to collect, transform, and share data with multiple sources. For example, the data shared between provider systems (e.g., hospitals and insurance payers) needs to be scrubbed for errors before sub- mission. Also, the provider may need to submit the data to multiple insurers. This is where a clearing house comes in to facilitate the job of checking, correcting, and submitting the data to various parties.

2. Enterprise has to deal with varying processes and formats of data. For example, the data that needs to be submitted to a regulatory body must adhere to all the processes and the formats prescribed by regulatory bodies. For example, the pharma industry must adhere to all the regulations imposed by the country and submit periodic reports of their adherence. Pharmacy regulatory services help the pharma companies with publishing, submission, dossier preparation, medical writing, etc.

3. Due to the increasing complexity in business processes, more intermediaries have arisen in various industries, which is the key role in the overall success. However, with the introduction of intermediaries, the overall cost and time needed to fulfill the business process have increased.

4. Intermediaries act as walls that need participants to hide information for survival. If the information flow is streamlined, most of the ecosystem may benefit albeit a loss to minority.

Hence, any use case where intermediaries play a role of data sharing only is a good candidate for blockchain. There could be cases where there will be no intermediaries or the role played by the intermediaries is either minimal or there is significant value add. In such cases, you need to look at other factors to decide if the blockchain is needed or not.

There could also be a case where the role of intermediaries cannot be eliminated or reduced. In such a case, we need to look at other factors such as return on investment by speeding the business processes to decide if blockchain is suitable or not. Here, even the intermediaries can be benefitted out of the change.

Though the above-mentioned criteria are the most basic to be considered in the assessment, there are other points that need to be considered.

Stage 2.a: Do Solutions Require Shared Write Access?
Knowing whether the different members of the network require shared write access will help us decide if blockchain is the only solution or if it can be done with other technologies.

If the solution does not require shared write access to a common ledger, then it may also be implemented with other technologies. While blockchain technically does not write to a single common ledger, with shared ledger, the effect is that participants are writing to a common ledger.

Stage 2.b: Can a Solution Work without Delete?
There are use cases where delete is necessary to clear the history of what has happened, as such information may not be desirable for all participants to log. Many times, such information is not captured electronically. In such cases, like "right to be forgotten" regulations, the information will have to be removed by the organizations [Link 3].

Blockchain does not allow deletion of the information that is captured once, information can only be appended. In cases where there is a need for removal of certain information, blockchain may not be a viable solution.

Stage 2.c: Is it Required to Store Large Amount of "Non-Transactional" Data?
Blockchain u uses distributed ledger that is stored on all the nodes of the network, and the transactions are constantly updated on the ledger. Hence, dealing with large non-transactional data will increase the size of the ledger leading to performance issues. There are some use cases such as electronic health record, which deal with large amounts of patient data. Storing such data on blockchain would make the ledger so large, thus leading to performance issues in syncing the ledger as well in the consensus process.

Hence, in such cases, non-transactional data needs to be stored outside the blockchain and only the transactional data can be stored on the blockchain. Storing this data outside the blockchain would mean that it has to be stored in a central location that goes back to the old centralization issues such as censorship. But there is a lot of work in progress on achieving this with blockchain. People are also working on decentralized storage solutions that will help store huge amounts of data using the same decentralized technology.

Stage 2.d: Is it Required That Only a Very Small Group or an Entity Needs All the Control?
If the use case demands that functionality need to be controlled by a limited group of participants, then it could be a good fit for a consortium blockchain utilizing permissions.

A good example of permissioned consortium blockchain would be a group of banks coming together to maintain a list of defaulters and they decide how the data can be shared. This is one of the key criteria in deciding what type of block- chain network needs to be used. While this is true, if it is a very small group or entity that needs all the control, then blockchain might not be a solution that is meaningful.

Stage 2.e: Does Use Case Involve High Performance (Millisecond) Transactions?
Currently, blockchain is not the best option for use cases involving millisecond transactions. This is because of the consensus mechanism used in blockchain that achieves transaction finality as well as network band- width required for ledger synchronization.

Though work in progress on making consensus mechanisms work faster, it will take some time to achieve milliseconds consensus. Till that time, such cases are not a good fit for blockchain.

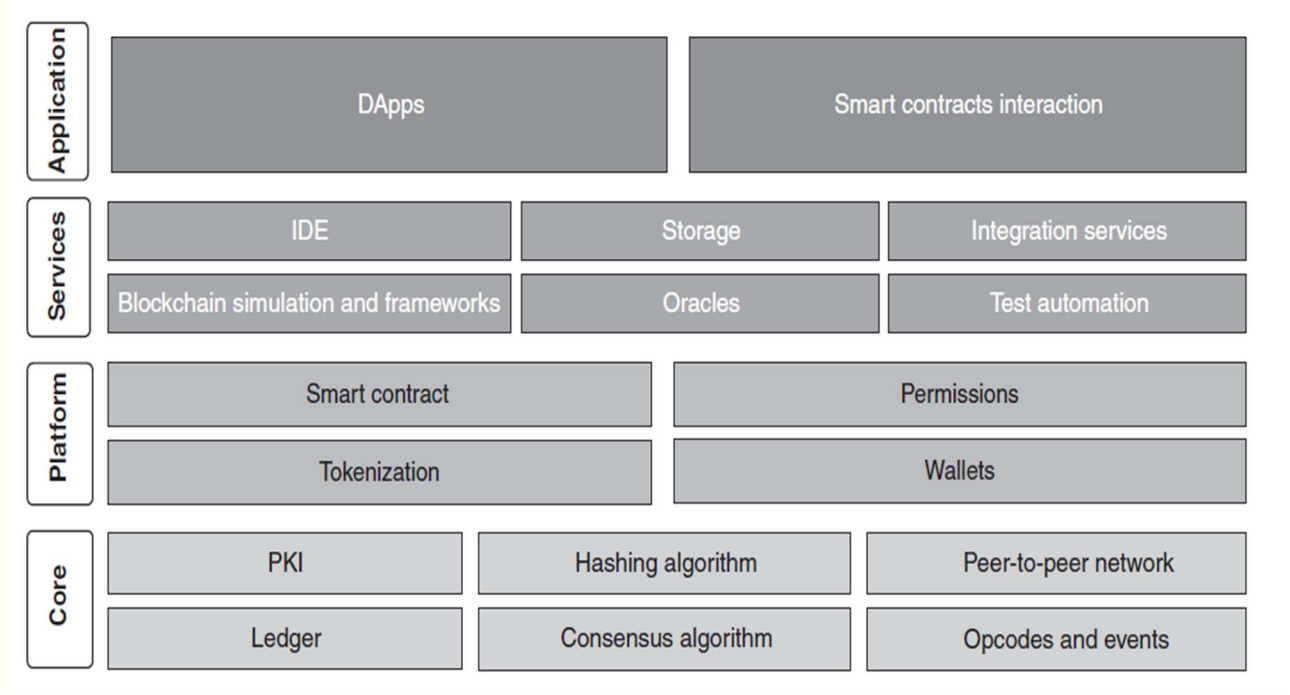
Apart from these factors, it is also important to consider what information is being shared and with whom This information will help us decide which type of blockchain to be used, if other factors indicate that blockchain is the right solution for the use case. For example, if the transactions cannot be made public. then a private or consortium blockchain can be used.

Block chain Solutions Reference Architecture

Introduction: -

The high-level reference architecture for blockchain solutions typically consists of four layers. -Blockchain Core
-Blockchain Platform
-Blockchain Services
-Blockchain Applications

Blockchain Solution Layered Reference Architecture is shown below:



Each of the layers in the reference architecture consists of technical components performing a set of technical functions.

While functionalities are abstracted in logical layers, the underlaying components are interoperable across layers. Let us understand each of the layers in little more detail.

Blockchain Core:

Blockchain core refers to the core elements that are required to build and run a blockchain network. These modules implement core characteristics such as distributed ledger and consensus that constitute a blockchain network.

Concepts associated with components of this layer are shown below:

1.Public-private key infrastructure (PKI): This is also known as asymmetric cryptography It uses public and private keys to encrypt and decrypt data. This is one of the fundamental aspects that enable a blockchain to verify the integrity of transactions. This allows secure sending of message from one account to another.

2.Hashing algorithm: This algorithm is used to create a hash of a fixed size irrespective of the size of input data. These algorithms help in implementation of security and privacy features of blockchain. Hashing and digital signatures ensure that the ledger is not corrupted or compromised.

3. Peer-to-peer (P2P) network: This network considers every peer in the network to be equally privileged. There is no central party controlling or coordinating the network and all the peers communicate directly with each of the other peers. These components help implement protocols and communication between peers in the network.

4. Consensus algorithm: This algorithm is at the heart of blockchain that ensures that each and every peer in the blockchain network reaches an agreement about the current state of the ledger. That helps achieve transaction finality and ledger synchronization.

5. Ledger: This is storage where details of each and every transaction is recorded uang the concept of blocks and chaining of blocks.

6. Opcodes and events: Blockchain networks provide infrastructure to execute certain instruction on the network as well as create and manage events. Instructions are sometimes called *Operational Codes or Opcodes*.
Opcodes are eventually support execution of smart contracts. Events help in the identification of occurrence of something meaningful in order to tigger additional processing.

Blockchain Platform:-

A blockchain platform comes bundled with a set of predefined functionalities such as smart contracts, wallets, etc., with guidelines and structures.

Blockchain platforms provide abstraction of core services. It should be noted that each platform has a core implementation. Key elements of this layer are listed below:

Smart contract: Smart contract is a piece of custom written code implementing business logic. This is a feature that makes blockchain useful as it allows for business logic to be implemented in response to certain event or transactions.

Blockchain platforms provide libraries, compilers, and other language elements for writing smart contract. These also support smart contracts to get access to blockchain storage and other functionality. The platform also helps to deploy and execute start contracts on the network

Wallet Support: Wallet support is a software program that allows secure management of user accounts. A blockchain wallet helps users to send messages, sign messages, and in case of cryptocurrency applications allowed users to send and receive cryptocurrency. While this is technically part of the outermost layer, platforms supporting cryptocurrency have components supporting wallets.

Permissions: Permission is one of the features that was needed in many of the enterprise use cases. Specifically, permission-based access enabled privacy was needed in many use cases:

Tokens: When we discuss about token, it is important that we first understand the meaning of a coin. A **coin** is a digital asset that is native to the blockchain network. Native refers to the fact that com built into the blockchain platform.
A **token** is also a digital asset but it is coded by the developers of blockchain applications. So, there will be only one coin per blockchain network whereas there can be many tokens in a blockchain network.

Blockchain Services:

Although we use a blockchain platform to create applications, many times additional services make our life easier so that we concentrate only on building the business functionality and not worry about some of the technical intricacies.

The services that we would be using will be generally driven by the blockchain platforms that we selected for implementation. The Service Layer includes the following components:

-Blockchain Simulation and Framework
-Integrated Development Environment
-Test Automation
-Blockchain Integration Services
-Oracles
-Storage

Blockchain Simulation and Framework: A blockchain application runs on multiple nodes but development cannot be done in this manner as it would be practically very expensive to have multiple nodes for development.

Also, consensus takes significant time and effort, developers have to wait a few seconds to make changes and test their functionality. Hence, we need a simulated environment where blockchain can run on single storage node and where consensus would happen instantaneously.

These are also known as personal blockchain networks. For example, building applications using Ethereum platform is made easier using a tool called "**Ganache**" and **Truffle** framework.

Integrated Development Environment: Any application development is made easier when we use an integrated development environment. This helps the developer to build and test applications faster and focus on business functionality than technical details.

For example, building applications using Ethereum platform is made simpler by using solidity plugin for **Visual studio Code**.

Test Automation: Smart contracts implement business functionality on the blockchain. They go by the rule code and will be deployed on all nodes of the network.

Since blockchain is a decentralized application (DApp) not controlled by a single entity, extreme care is needed to test smart contracts before deploying on blockchain.

Testing the contracts thoroughly is not only inevitable but also essential to automate the process to deliver applications within a reasonable amount of time. **Mocha** and **Chai** are popular frameworks that work along Truffle to provide testing functions.

Blockchain Integration Services: When it comes to public blockchain networks, we need to have a node that is in sync with the blockchain network. Thus node is needed to deploy smart contracts or updates to your contracts. If the node is not in sync, then we need to wait for the ledger to be synced.

For example, a new node may take many hours to sync with the test network and it may take more than a day to sync with the live network. Node clusters like "**Infura**" make this easier for developers by providing blockchain nodes that can be used by the application developer.

The developers can connect to this node and deploy their changes or make transactions using this node. There are also frameworks and tools that are provided by blockchain platforms that help developers interact with blockchain networks.

Oracles: Oracles fetch reliable inputs from the outside world and provide it to the smart contracts. Many business use cases need inputs from the outside world and oracles makes this possible.

For example, smart contracts that make payment based on flight delay gets flight timing inputs from oracles. **Augur** and **Ramp** are examples of open-source oracles that support specific industry segments

Storage: Blockchain networks of today are not best suited for storing large amounts of data. Hence, if the use cases involve large amounts of data, then it is better to keep non-transactional data outside of blockchain.

Some- times due to compliance requirements, the data cannot be stored on blockchain. In such cases, the data needs to be stored outside the blockchain. While this is true, business requirement may demand decentralization for storage.

This is where decentralized storage services come in handy, where the data can be stored outside blockchain but it is still using decentralized services. One such Storage example is Inter Planetary File System (IPFS).

In the rest of the case, regular SQL or NoSQL database might be utilized by each node to store node specific data.

Blockchain Applications: Blockchain applications is an area that would be of interest to application developers and application architects.

Types of Block chain Applications

Introduction: -

Blockchain Technology has been evolving on various dimensions that lead to different Architecture Types each of focusing on unique set of requirements, limitations, and benefits.

We can Classify blockchain applications into four categories based on the focus areas:

-Decentralized Applications
-Simple Blockchain Solutions
-Enterprise Solutions
-Enterprise Blockchain Ecosystems

Fully Decentralized Applications: Most public blockchain solutions fall into this category. Cryptocurrency as well as other asset trading can be done using blockchain technology and solutions can be provided to end user using existing public blockchain networks. Cryptocurrencies are example of these applications.

Simple Blockchain Solutions: Simple blockchain solutions refer to simple and independent blockchain solutions that help realization of certain business processes.
Many times, these solutions are done as experimentation to evaluate or build confidence of stakeholders.

Enterprise Solutions: Once simple solutions build confidence among stakeholders and proof-of-value is established, enterprise systems get integrated with the blockchain solution that has been built.
This helps automate significant work as existing enterprise assets supporting business processes get utilized to initiate transactions and respond to events coming from blockchain.

Enterprise Blockchain Ecosystems: With the success of blockchain solutions, more stakeholders start joining the network to realize their business value.
It reaches a critical mass to create enterprise ecosystem using blockchain to deliver value to end customers and create value for stakeholders.

Cryptographic Tokens

Introduction: -

A cryptographic token represents a programmable asset that is created and managed by a smart contract.

The smart contract decides the number of units of tokens to be created and handles the transactions involving the token and each token holder's balance.

While a cryptocurrency such as ETH will be available for all the applications implemented on Ethereum, the cryptographic token is implemented by a particular application and can be accessed and controlled only by the smart contract belonging to that particular application.

It is important to remember the difference between a coin and a token. Coins are native to the platform and are accessible to all applications developed using the platform whereas token are accessible to the application that creates token on the platform.

In short, coin is an out of the box currency provided by the platform. On the other hand, tokens are not out of the box, they are explicit implementations by blockchain applications and smart contracts.

Utility Tokens:

- Utility tokens are user tokens that provide access to the products or services offered by the company. They can help in driving the internal economy within a blockchain project.
- They can be used to provide advantages to the holders of the utility tokens.

- For example, Golem Network Token (GNT) provides the holders access to the Golem ecosystem.

Security Tokens:

- Security tokens are tokens that represent legal ownership of a physical or digital asset. They are generally tied to the company's profit and loss valuation.
- These tokens are heavily regulated, and failing to comply with the regulations shall have serious consequences which can be even lead to the shutdown of the complete blockchain project.
- Some examples of security tokens are Siacoin token, Blockstack STX token, etc

ERC Tokens:

- Token management can be significantly standardized as it follows very similar functionality and attributes irrespective of domains where these are used.
- The Ethereum community has taken a lead on this and EIPs standardized definition for developers to code and deploy.
- ERC tokens define rules that will enable the tokens to be handled and exchanged between various DApps and wallets.

Typical Solution Architecture For Enterprise Use Cases

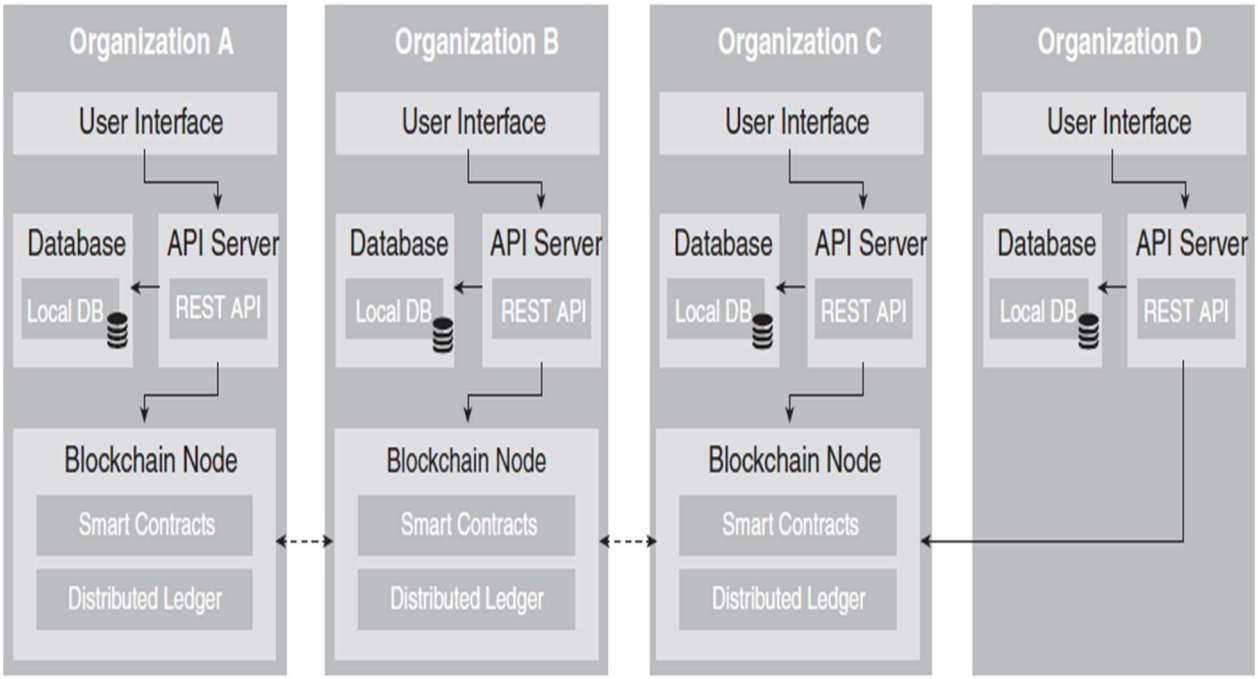
Introduction:-

In enterprise solutions, the user interface reflects the functionality provided by the complete business process and not just the functionality achieved through blockchain.

Besides, enterprises would require better control of the functionality exposed to user interfaces.

Blockchain functionality needs to be encapsulated and exposed through web services or APIs. These services may expose the blockchain functionality or may orchestrate a group of functionalities along with the blockchain functionality.

Typical Solution Architecture:



Blockchain Node:

Every organization that is participating in solution would either run a blockchain node or connect blockchain node set up by another organization.

The blockchain platform selected for the use case will be running on this node. The smart contracts developed by the application developers are deployed on the blockchain platform.

This would mean that each organization would have a blockchain node running smart contracts on that node. All these nodes are connected together to form a blockchain network

Web Services:

There would be an API server exposing blockchain functionalities using REST API or web services. These APIs will also maintain the details of the smart contracts to be used, that is, the address at which the smart contract is reachable, the version of the smart contract to be used, etc.

The information would be maintained as configurations. These APIs may not just expose blockchain functionalities. Depending on the use case, the organization may have its own functionalities outside blockchain that would be exposed through these APIs.

Database:

Generally, the organization shall have a database server housing a local database that would store non-transactional data and PII data that cannot be stored on blockchain.

Types Of Blockchain Solutions

Introduction: -

Blockchain Solutions can be broadly classified into the following:

-Value Transfer
-Provenance
-Identity Management
-Data Sharing

Value Transfer:

- Value transfer is one of the first use cases that blockchain was used for, starting with Bitcoin.
- These can be categorized as payment and transaction solutions. Value transfer use cases deal with fungible tokens.
- Some of the probable use cases are
  -P2P payments,
  -cryptocurrency,
  -insurance payments, etc.

Provenance:

- Blockchain plays a pivotal role in provenance use cases. Generally, provenance refers to identification of the place of origin or finding history about something.
- Blockchain can help document immutable record of change in ownership as well as movement of assets.
- The assets can be any physical asset that is digitally represented on the blockchain.
- Some of the probable use cases are shown below:
  -Documented Journey of food items and consumables
  -Counterfeit detection of drugs
  -Ethical sourcing of diamonds

Identity Management:

- Identity management refers to the process of identifying individuals or entity.
- Once a user's identity is established, various authentication and authorization services can be provided by associating role and privileges for the individual.
- Access to applications and systems can be provided and controlled based on the access rights
- Some of the prominent use cases are:
  - -Identity verification solutions such as KYC
  - -Non-custodian login solutions
  - -Real Estate or other title transfer use cases etc.

Data Sharing:

- Blockchain can provide secure data sharing between competing stakeholders without comprising on privacy.
- More than sharing data between competing stakeholders, blockchain gives complete control and rights to the owner of the data.
- Some of the prominent use cases are:
  - -Electronic health records
  - -Provider data management

Architecture Considerations

Introduction: -

Solutions that get implemented using Blockchain need to take care of considerations that would make solutions usable and relevant for end users and stakeholders.

- Performance
- Storage Management
- Distributed Computing
- Privacy
- User Experience
- User Experience
- Integration with Other Systems
- Compliance
- Standardization
- Costing
- Security

Performance: Blockchain applications pose greater performance challenges due to the delay resulting from consensus process. There are two important performance factors: transaction throughput and data retrieval.

Storage Management: Blockchain uses a distributed ledger, that is, same copy of the ledger is stored on all the nodes of the blockchain network. Data is written to the ledger only when the nodes agree on the correct transaction. Also before each transaction, the ledger is validated. it becomes extremely important to store only data that is mandatory for transaction.

Distributed Computing: It is a computing methodology involving a network of computers working together as one system. Blockchain has to be considered as a pure distributed computing system. It employs consensus where all the computers participate in making a decision for adding a new block.

Privacy: Here, privacy relates to data privacy. when personal identifiable information is involved, extreme caution is needed to make sure that data privacy is fulfilled else one can end up violating some laws. Personal data access needs to be managed across nodes which involve task on blockchain.

User Experience: Though end users may not even be aware that an application is implemented using blockchain, it will be revealed in the user interface of a blockchain-based application. Any transaction triggered from the frontend is first submitted to the blockchain network, transaction finality is achieved using consensus. .

Integration with Other Systems: Blockchain applications may need to integrate with other systems such as Web Services, IoT etc. depending upon the use case. For example, real-time pharma supply chain networks shall integrate with IoT-enabled track and trace systems.

Compliance: Blockchain applications are also expected to be compliant with regulations imposed by the respective industries such as finance, healthcare, etc. For example, GDPR is one of the most widely discussed regulations that makes blockchain adoption very challenging for enterprises. One of the tenets of GDPR describes where the customer data is stored, that is, the physical location of the digital data.

Standardization: Blockchain platforms and protocols are getting introduced to resolve issues, but at the same time maturity is happening in some isolation also. Each platform is building their own consensus, P2P protocols, storage mechanism, etc. Unless standardized, integration between blockchain networks, can pose a challenge and reduce the full potential of the blockchain.

Costing: Generally, blockchain implementation is expensive compared to other technologies. We need to understand that blockchain will reduce the overall cost involved in the business process, but the initial investment cost will be relatively higher compared to traditional technologies.

Security: Although blockchain is known to be highly secure when it comes to the distributed ledger that is extremely tamper resistant, blockchain also has its own security risks in other areas.

- **Perimeter Security:** One of the most vulnerable risks for blockchain solutions actually lies outside the core blockchain system. These are areas where humans or systems interact with blockchain. Generally, these are computers used for accessing blockchain systems.
-
- **Credential Security:** People mostly hack into computers to get hold of credentials. In blockchain parlance, this would be the public and private keys. The public–private key pair should be secured using hardware security module (HSM). HSM is a physical device that securely stores digital keys.
-
- **Smart Contract Security:** Smart contract security is another key area that needs proper attention. We already discussed the importance of security audit since code is law in blockchain and the vulnerabilities in smart contracts can lead to drastic results.

Architecture With Blockchain Platforms

Introduction: -

There are various blockchain networks and platforms that have sprung out in the last few years, each with their own followership, developer community, and relevance.

We will focus on two blockchain platforms used for Enterprise:

-Ethereum and

-Hyperledger.

Ethereum is a balanced platform with good public blockchain capabilities while providing smart contracts and other constructs that are very useful for solutions focused on end users.

Hyperledger can be considered as enterprise blockchain platform with more capabilities of a shared ledger than pure decentralized blockchain.
Hyperledger also provides very good programming and development capabilities through a chaincode (a term for smart contracts in Hyperledger).

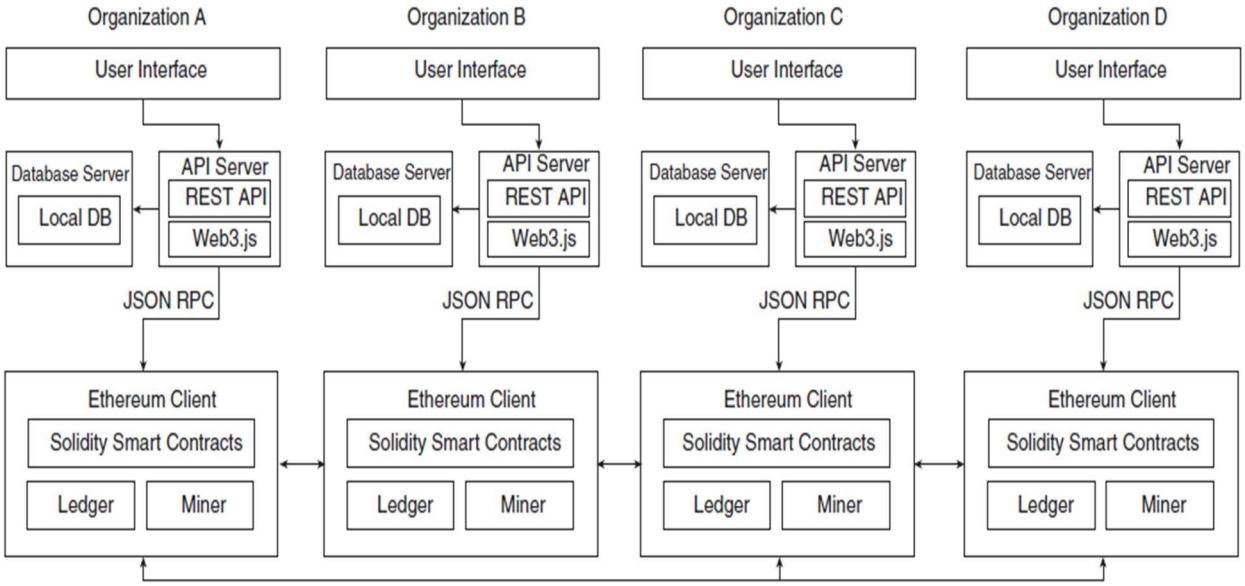
Ethereum Architecture

Ethereum is a public blockchain network that combines capability of cryptocurrency trading along with execution of smart contracts. An organization named Consensys is one of the founding members.

To execute smart contracts, Ethereum utilizes something called Ethereum virtual machine (EVM), which is a kind of isolated sandbox for the smart contract code running on blockchain. This restriction where smart contract cannot access network, file system or other node assets directly provides secure and independent environment for logic execution.

On Ethereum, smart contracts are typically coded in a language called Solidity, complied using an EVM compiler and tested and deployed utilizing provided tool set like the Truffle. The tool set includes capability of providing a sample node with unlocked accounts, so that developers can utilize their own machines to develop, deploy and test code before it is ready to be tested on the real network.

DApps and oracles can interact with the network utilizing a library called web3.js for cryptocurrency trading or smart contracts. Typical solution architecture applied to create solution architecture for Ethereum solutions is explained as follows.



Ethereum typical solution architecture for enterprise use cases.

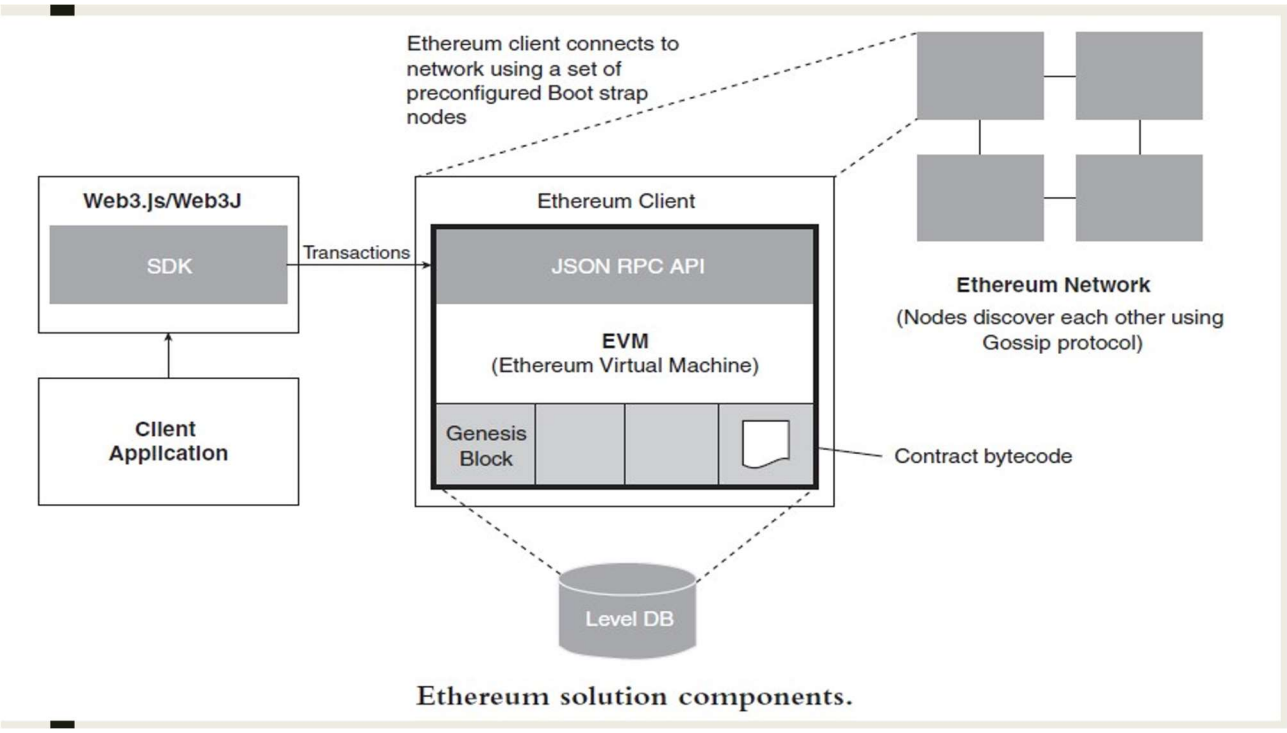
In Ethereum platform implementation, every blockchain node will be an Ethereum client. Various Ethereum clients available in the market are Geth, Aleth, Trinity, Parity, EthereumJS, Ethereum), etc.

The most widely used Ethereum client is Geth. Every Ethereum client consists of miner functionality that participates in the consensus process. Client applications connect to the Ethereum Client using an Ethereum library that does the heavy lifting of handling the communication between them.

This allows developers to focus more on implementing the functionality rather than the communication between client application and Ethereum client. Though there are various libraries available, the most widely used library is web3.js.

The smart contracts are implemented using Solidity language, and these smart contracts will be executed on EVM.

A similar view can be created for other platforms by replacing Ethereum specific components with corresponding components from other platforms.



Ethereum solution components.

Hyperledger Fabric Architecture:

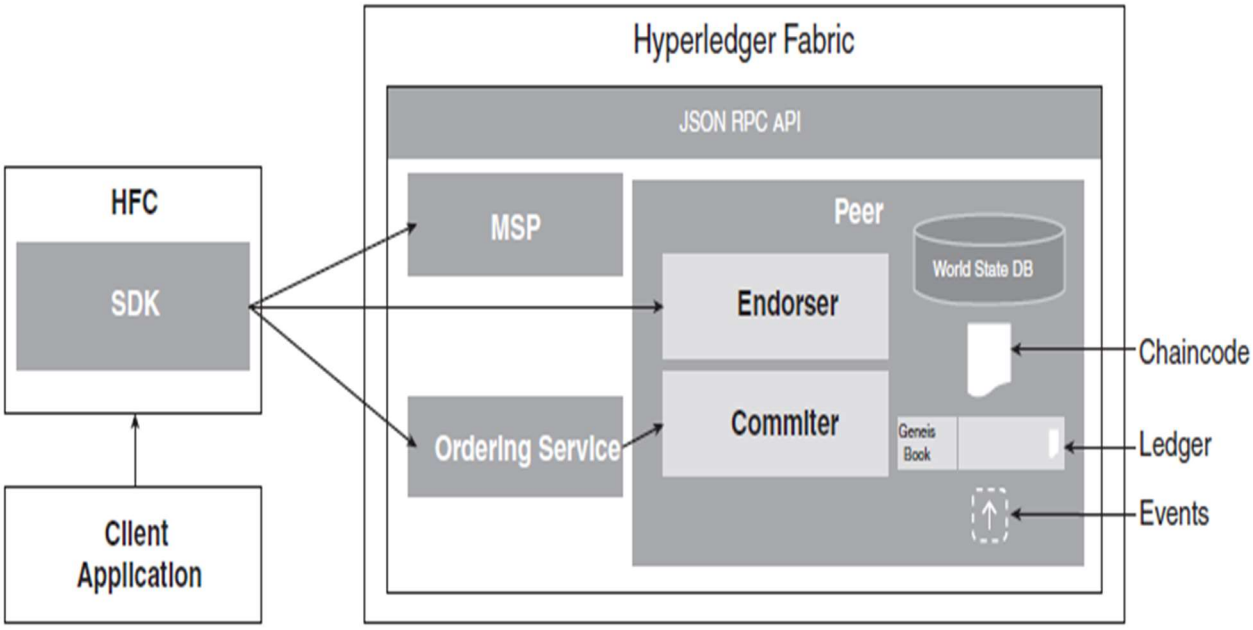
- Hyperledger is an open-source initiative by 'The Linux Foundation' with focus on enterprise blockchain use cases across industry domain and includes multiple projects under an umbrella offering. One of the projects 'Hyperledger Fabric' provides permissioned distributed ledger technology (DLT) platform.

- DLTs are different from traditional blockchain as they focus on maintaining ledger of transactions in immutable way rather than creating immutable platforms for currency trading.

- Enterprise focus allows final decisions on ledger commit to be made by dedicated nodes, which in a way deviates from full decentralization.

- However, a provision for pluggable consensus protocols makes Hyperledger applicable across industry domains.

- Hyperledger Fabric components are shown below:



Hyperledger solution components.

Membership Service Provider:

Hyperledger Fabric focuses on building permissioned blockchain solutions for an enterprise. Unlike public blockchain network where anyone can join, permissioned networks require the identity of participant to be known beforehand.

For this purpose, Fabric has a component known as membership provider (MSP) As the name indicates, this component manages registration, validation, and dentification of the members of network. The component known as certificate authority (CA) issues new membership certificate and also revokes an existing member's certificate.

Peer:

Any participating node in the Fabric network is known as peer. There are two types of peers - Endorser and Committer.

Each peer obtains an identity certificate via MSP before joining the network. Each peer maintains a copy of the ledger.

An **endorser** performs checking the transaction validity before committing. External applications communicate with the Fabric network through these endorsers. When an endorser receives a transaction, it simulates that transaction and in response sends the current state, future state, and endorsement metadata to the caller.

A **committer** does some simple checks on the state of ledger and then writes a particular transaction to ledger. All the nodes perform commit, that is, write changes to keep the ledger in sync.

A transaction may need endorsements from multiple endorsers before it can be committed. This decided by the rules defined in an access-control list (ACL) within the Fabric network. After getting endorsement from all the required endorsers, the transaction is sent to an orderer that broadcasts it to all the peers in the network.

The peers verify whether the actual current state is the same as the one present in the transaction metadata returned by the endorser. If everything looks good the transaction written onto the ledger. After a transaction is committed, peers may optionally emit an event for external applications.

Orderer:

Ordering the transactions is important to maintain ledger state and consistency. In a public network such as Bitcoin, ordering transactions becomes part of mining itself. On the other hand, Fabric implements transaction ordering as a pluggable service. Every Fabric network will have one orderer, which is basically a cluster of nodes.

The orderer does not validate the transaction against the state of ledger, but just groups the transactions that occurred within a timeframe to create a block. This block will be transmitted to the peers.

It should be noted that the identity of the orderer is also established by the membership services provider (MSP).

Fabric provides three ordering mechanisms, which are as follows:

1. SOLO: In this method, there will be only one orderer node and it is intended to be used only in the development environments.

2. Kafka: In this method, Apache Kafka, which is an open source stream processing platform, is used as the ordering component. It provides high throughput, low latency, and crash fault-tolerant features. It is ideal for production use.

3. Simplified Byzantine Fault Tolerant (SBFT): This mechanism provides both crash fault-tolerant and Byzantine tolerant features. It can arrive at an agreement even in the presence of malicious of faulty nodes

Ledger:

Ledger is the distributed database where Fabric blockchain network transaction data gets stored. Apart from the ledger, Fabric also maintains world state on peers.

It provides two options to implement the world state-LevelDb or CouchDb.
The default option is **LevelDb**, which stores data as a key-value pair. This can be replaced with **CouchDb** while setting up the Fabric network.

CouchDb stores objects in JSON format and also offers feature-rich queries. Every peer maintains a copy of the ledger.

Along with the ledger, Fabric maintains another indexed database known as state database or world state. This is a collection of the current state of all the assets on the chain. This database helps to make the smart code interactions and transaction validation process efficient

Chaincode:

Smart contracts in Fabric are known as chaincode. They contain the logic to perform transactions and change the state of an asset.

Currently, Fabric supports four programming languages to write

-GO,
-Node.js,
-Java, and
-Solidity

The Hyperledger Composer project by IBM under Hyperledger umbrella has its own language for chaincode creation. It is known as **Composer Modeling Language**. As of January 2019, IBM has announced that there will be no further enhancements to this language and the focus would be more on the core Fabric component

Channels:

Channels offer an additional layer of privacy within a Fabric network. It allows organizations to use the Channel policy can be defined while setting up the Fabric network or at a later point of time as well.

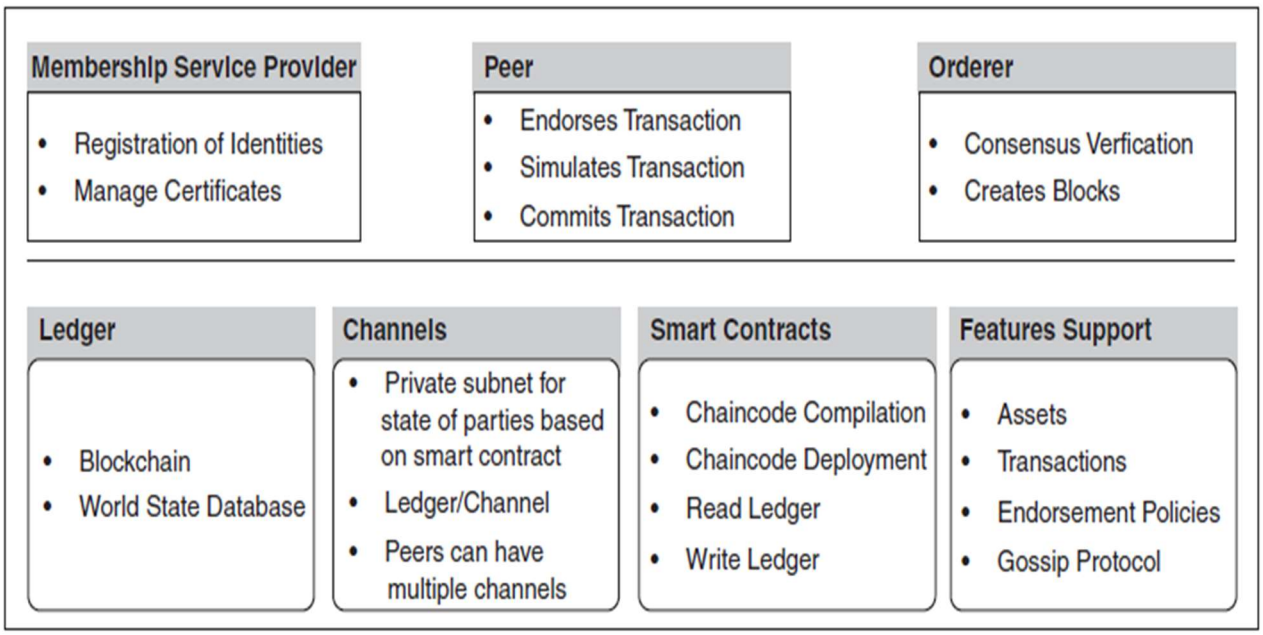
The following points should be remembered about channels:

1. Each channel has its own ledger and endorsing policies.
2. A peer can be part of multiple channels within a network.
3. The same chaincode logic can be applied to multiple channels.
4. The ordering service is the same across channels.

Access Control List:

Access control list (ACL) is a set of policies that controls different activities within a Fabric network such as who can deploy the chaincode, who can execute chaincode, and what type of endorsements are required for transactions.

The Hyper Ledger Solution Components are shown below:



Hyperledger fabric components.

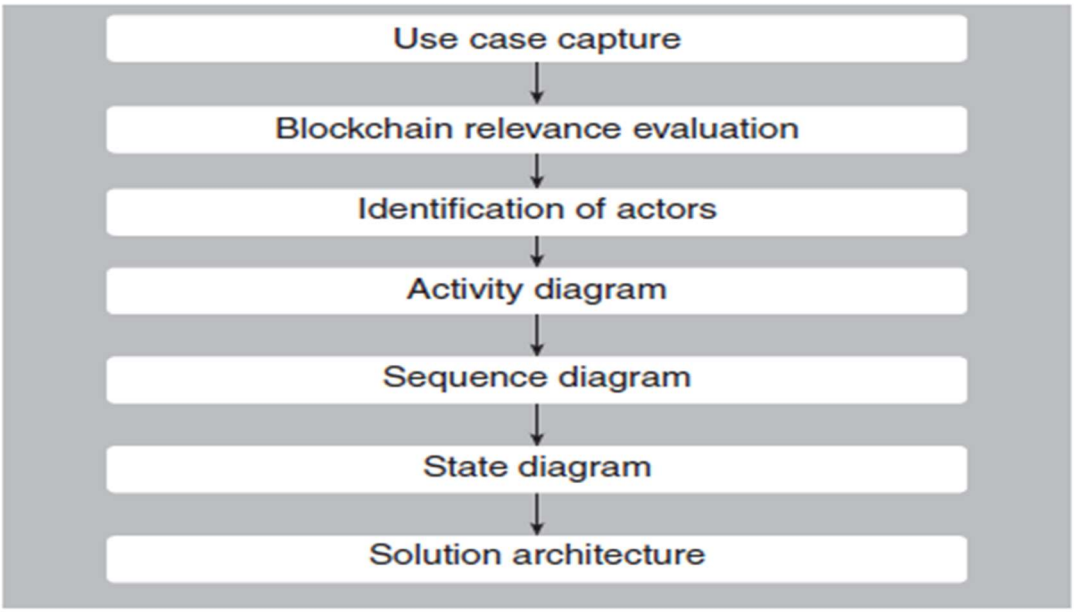
Approach For Designing Blockchain Applications

Introduction: -

Design of blockchain applications can be simplified if a set of well-defined steps are followed in order.

These steps help in capturing all the necessary information that are needed to design the application, without missing on any critical information needed for the design.

These also help in quick drafting of a design document without spending too much time on document writing.



Steps in designing blockchain application.

Use Case Capture: To solve enterprise problems using blockchain, it is important to capture use case or list of potential use cases that can be solved using blockchain. These use cases need collaboration or information sharing.

Blockchain Relevance Evaluation: Framework to evaluate relevance of blockchain for a particular use case needs to be applied on the use case identified to judge if it is worth exploring the use of blockchain in an overall solution.

Identification of Actors: Like every other software solution, the key is to identify actors in the given problem statement. In the case of blockchain, some of the actors might be enterprise systems or devices. These should also be explicitly captured so that subsequent design happens in good amount of details.

Activity Diagram: Activity diagrams will document all the interactions actors will have with the system and with each other. It will be good if these activities are captured, in order to mention activities that would be automated and those that would not be automated.

Sequence Diagram: Sequence diagrams list details of sequences for each of the transactions. These diagrams should be detailed in iterative fashion to capture the detailed technical interactions and their sequences. These help in identification of methods class objects will need to have as well as orchestration of these methods in the solution.

State Diagram: State diagrams help identify workflow or workflows that need to be taken care of during implementation of the blockchain solution. These will also identify state database that needs to be maintained at node, and the conditions and events that a smart contract will need to have in an eventual solution.

Solution Architecture: Solution architecture lays down components and their interaction in the overall network. This will help understand building blocks of the solutions and will have different views so as to interact with relevant stakeholders.

Architecture for Sample Use Case using Ethereum

Introduction: -

We will discuss about the **Tuna Fish supply chain use case** and how this solution can be architected. We have chosen a multi-actor supply chain for tracking tuna fish.

One of the persistent problems in tuna fishing industry is illegal, unreported, unethical, and unregulated fishing in the Pacific region.

Consumers are now increasingly looking for transparency, ethical sourcing, and full traceability to make sure that the fish they purchase does not come from any illegal fisheries.

A simple QR de scan should reveal the story of tuna fish to its consumers. The data collected from RFID tracking and IoT devices can be inserted into the blockchain.

This gives details on when and where the fish was caught processing temperatures, result of quality checks by auditors, etc.

i)Use Case Description:

As very simple level, the use case will explain the flow of events and activities that need to happen. The following are the steps that have been identified:

1.Workflow:

(a) Fishermen capture tuna and send it to the supplier.
(b) Supplier will send the tuna received from the fishermen to the fish-processing unit.
(c) The processed fish is then sent to the distributor with the help of a transporter.
(d) The distributor now splits and distributes the fish to various retailers.
(e) The retailer sells the fish to restaurants and supermarkets.
(f) Customers buy tuna from the restaurants and supermarkets.
(g) Flow of tuna through the various parties can be inspected by the regulator as well as customers.

2 Benefits for players:

(a)Customers get to make informed decision regarding the food they are consuming, such as if it is coming from a source that fits their ethical framework. Customers are happy to pay more for this transparency.
(b) Law abiding and ethical fishermen get the right price and do not need to depend on unethical. practices for the sake of needing to make ends meet.
(c) Regulators get a full view of the supply chain and can trap any incorrect entry early.
(d) All other participants benefit through the value generated and extra money spent by customers.

Only a few main scenarios have been captured to depict how to approach the design of blockchain applications. The actual solution will need more significant details

ii)Blockchain Relevance Evaluation Framework:

We will perform the fitness assessment in this section. Let us go through the questions listed in the assessment. These are not the final set of criteria but a good set of primary criteria to be considered.

1. **Does use case involve sharing assets/valuables/data between multiple participants?**
   Yes. Multiple participants who are not confined to a single organization are involved. The fish is the asset that is shared between various participants.

2. **Is there impact due to sharing/hiding information between participants?**
   Yes, intermediaries are involved. The fish does not reach the restaurant or the supermarkets directly. They are passed through intermediaries. In this example, the intermediaries can also be accommodated in the blockchain. It should be noted that not all blockchain use cases focus on eliminating intermediaries. Some of the use cases may still involve intermediaries, but the reliance on intermediaries is reduced.

3. **Does the solution require shared write access?**
   Yes. Fish is the asset that moves through the supply chain, that is, the ownership of the fish is changed as it moves through the supply chain.

4. **Can the solution work without delete?**
   Yes. As of today, the tuna supply chain does not mandate delete of data. Also, delete of change of hands will not help build the desired transparency.

5. **Is it required to store large amount of non-transactional data?**
   No. The properties of fish can be generally digitized and stored on the blockchain. Any related documents shall be stored on a central database by the needed parties outside the blockchain.

6. **Is it required that only a very small group or an entity needs all the control?**
   Yes. The parties participating in the supply chain can control the supply chain functionality.

7. **Does use case involve high-performance (millisecond) transactions?**
   No. This process spans through various participants over a period of few days to few weeks and does not demand any high-performance transactions.

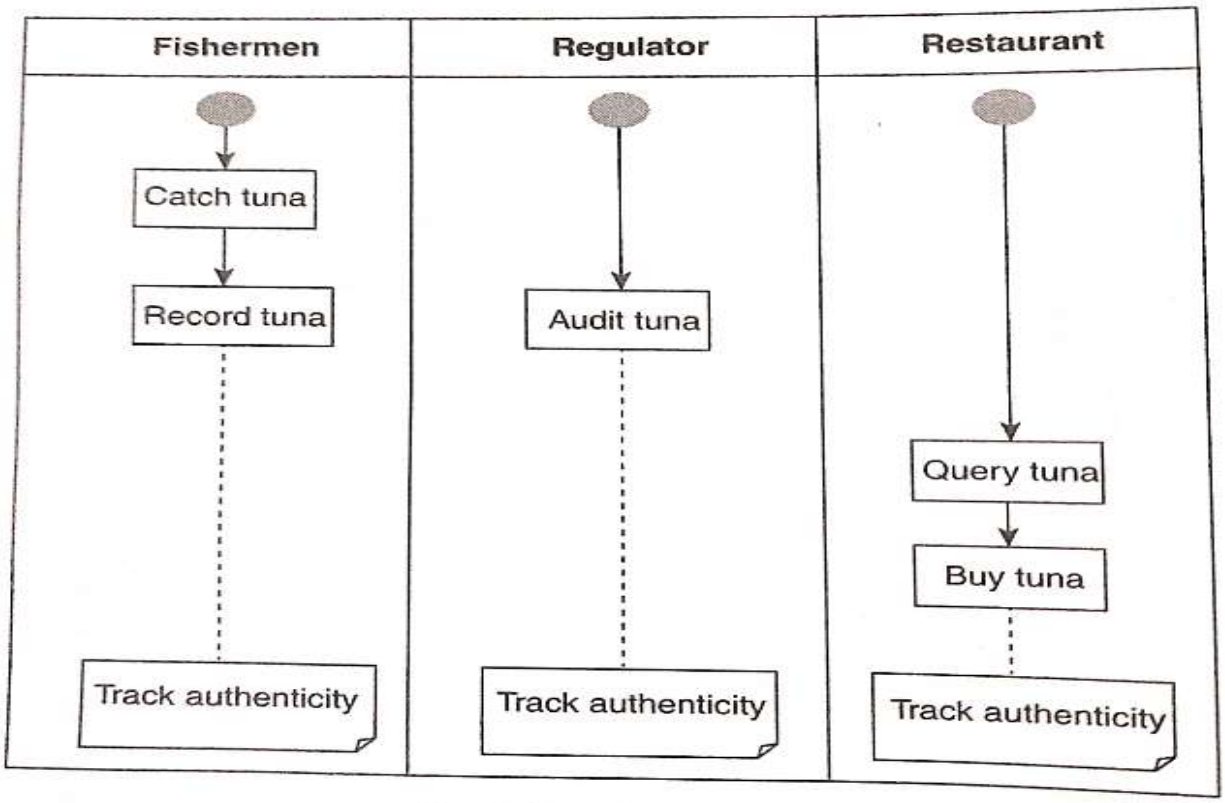
Now, we know that the assessment revealed that it is a very good use case for blockchain.

iii)Actors, Roles and Permissions:

a) Producer (Fishermen)
b) Supplier
c) Fish-Processing Unit
d)Transporter
e) Distributor
f) Retailer
g) Restaurant
h) Supermarket
i) Restaurant Customers
j) Regulator

Now we will capture the various activities that shall be performed by each of the actors. For the sake of simplicity, only free actors are chosen for the subsequence section-fishermen, regulator, and restaurant.
However, the solution approach and concepts can be extended to more number of actors as well.
We will be utilizing Unified modeling language (UML) diagrams to capture these details.

iv)Activity Diagram:



Activity diagram.

v)Sequence Diagram:

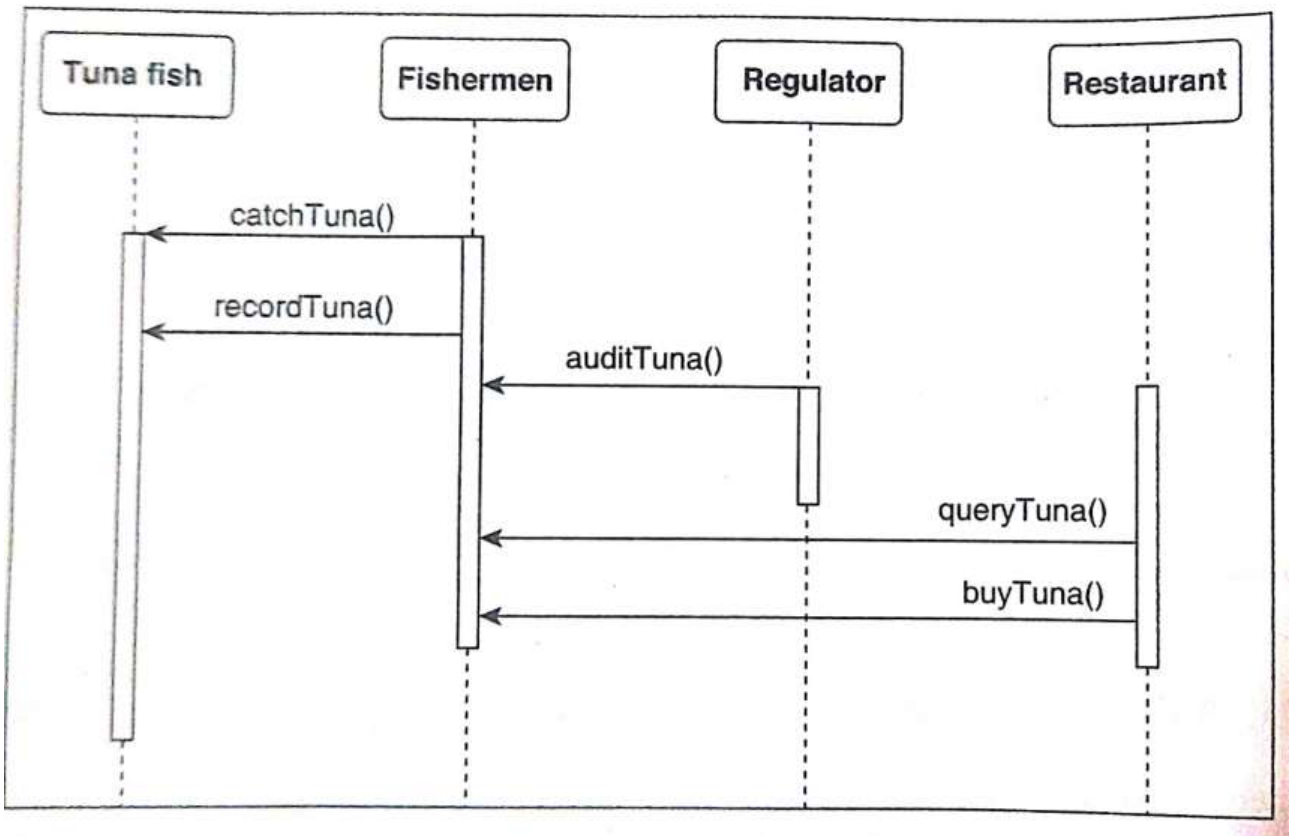


Figure 5.6 Sequence diagram.

Vi)State Diagram

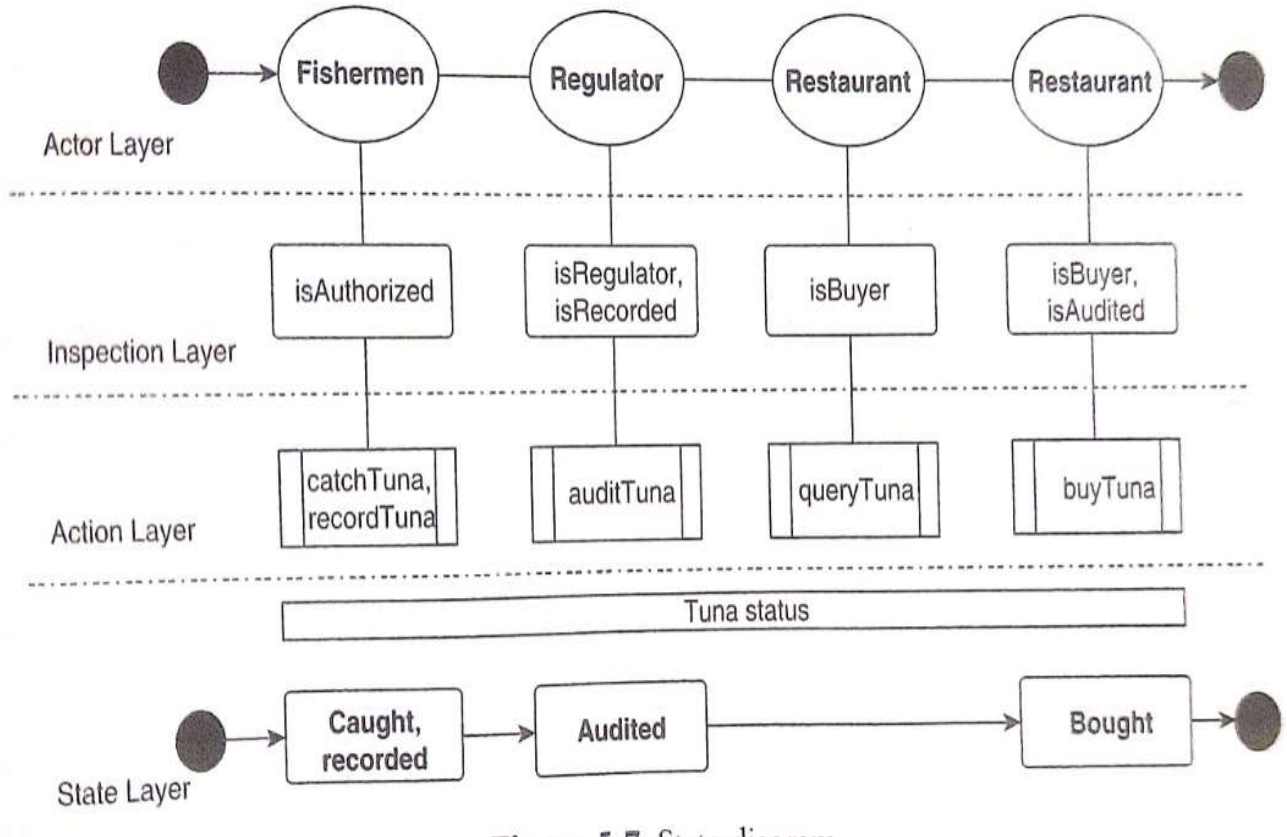


Figure 5.7 State diagram.

For further understanding, the **state diagram** is split into multiple layers. The bottom layer of Figure 5.7 marked as State Layer shows hoy the state of tuna fish changes during its journey. This gets recorded on blockchain via smart contracts.

The layer above this is represented as **Action Layer**, which shows the corresponding action that results in that particular state change.

This is accompanied by another layer known as **Inspection Layer**, which performs some simple checks to ensure that an actor is authorized to perform particular action and the current state of tuna is valid to perform that action.

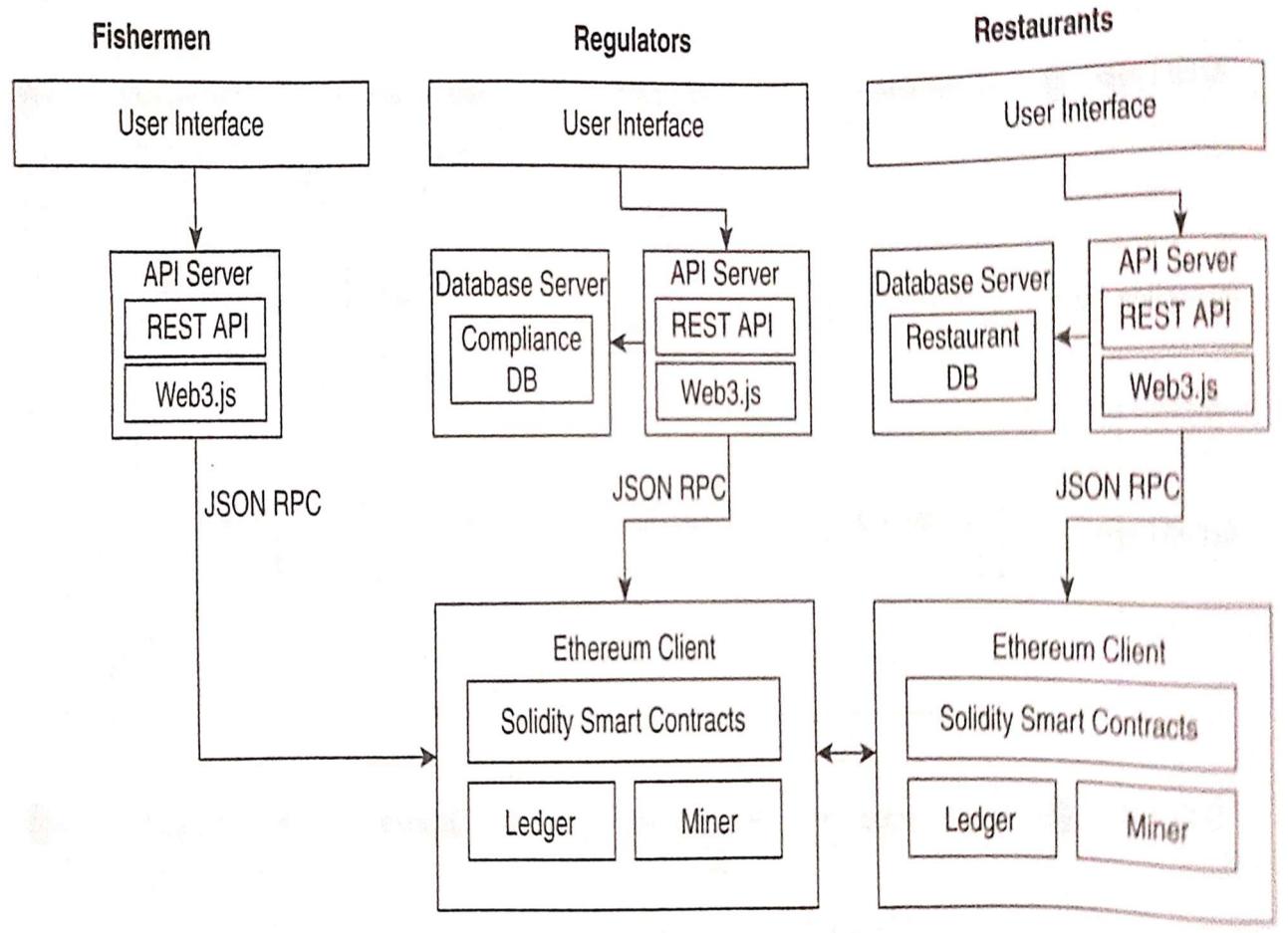
For example when a regulator wants to carry **AuditTuna** activity, the following checks occur:

1.isRegulator: This will check whether the user is a Regulator.

2.isRecorded: This will ensure that the current status is "Recorded", which means that the metadata related to tuna fish is recorded on blockchain.

The topmost layer of the state diagram represented in Figure 5.7 as **Actor Layer**, shows different actors and the sequence in which they perform activities.

vii)Solution Architecture:



Architecture for Sample Use Case using Hyperledger

Introduction: -

Here, we will discuss about **vehicle identification use case** and how this solution can be architected. We have chosen a multi-actor network for vehicle identification.

One of the key problems in vehicle business is the missing information about ownership and the complete history of the vehicle. This result in losses for customers and at times legal issues when the ownership of the vehicle changes multiple times.

Consumers are now increasingly looking for transparency and full traceability to make sure that the car they purchase does not have any legal complications and is indeed genuine.

We will consider a **Fabcar network** that will track the ownership of the car, starting from the manufacture till it is sold to various customers. An actual network shall involve multiple players such as car manufacturer, distributor, dealer, agent, road transport authorities, service center, auto insurer, customer, etc.

Let us now go through the fitment assessment.

i)Use Case Description:

The following are the steps that have been identified for Car Ownership Tracking use case:

1. Workflow:

a) Car manufacture can create a new car and send it to the distributor with the help of a transporter.
b) Manufacturer updates regional transport office with details on new car.
c) The distributor now splits the stock received and distributes the cars to various dealers.
d) The dealers sell the car to the customers.
e) The car is now registered at the transport office against the customer.
f) Customer can approach the service center for servicing the car.
g) Customer can sell the car to another customer.
h) Ownership changes are approved and updated in transport office records.

2 Benefits for players:

(a) Availability of complete journey of the car, from the manufacturer till the customer, helps in reducing audit costs for manufacturers, distributors, and dealers.
(b) The Audit Trail can simplify loan and insurance processes.
(c) Helps in quicker recall in case of any faulty equipment that gets discovered at a later stage.
(d) Customers get to make informed decision regarding the car they are purchasing, such as if it is coming from a source that fits their ethical framework. Customers are happy to pay more for this transparency.
(e) Customers buying a used car from another customer can be assured that the used car is genuine and free from any legal complications.
(f) All other participants benefit through value generated and extra money spent by customers.

ii)Blockchain Relevance Evaluation Framework:

1. Does use case involve sharing assets/valuables/data between multiple participants?

Yes. Multiple participants are involved who are not confined to a single organization. Car is the Asset that is shared between various participants.

2. Is there impact due to sharing/hiding information between participants?

Yes, intermediaries are involved. The car does not reach the customers directly. They are passed through intermediaries such as dealers and agents.
In this example, intermediaries can also be accommodated in the blockchain.
It should be noted that not all blockchain use cases focus on eliminating intermediaries. Some of the use cases may still involve intermediaries, but the reliance on intermediaries is reduced.

3. Does the solution require shared write access?

Yes. Car is the asset that moves through the supply chain, that is, the ownership of the car is Changed as it moves through the supply chain.

4. Can the solution work without delete?

Yes. As of today, car supply chain does not mandate delete of data. Also, delete of change of hands will not help build the transparency desired.

5. Is it required to store large amount of non-transactional data?

No. The properties of car can generally be digitized and stored on the blockchain. Any related documents shall be stored on a central database by the needed parties outside the blockchain.

6.Is it required that only a very small group or an entity needs all the control?

Yes. The supply chain functionality is controlled by the parties participating in the supply chain.

7. Does use case involve high-performance (millisecond) transactions?

No. This process spans through various participants over a period of few days to few weeks and does not demand any high-performance transactions.

Now, we know that the assessment revealed that it is a very good use case for blockchain Also, since a small group needs to control the network, unlike a public network such as Bitcoin, a permissioned blockchain such as Hyperledger Fabric would be a good fit for this use case.

iii)Identification of Actors, Roles and Permissions:

Actor	Role Permission
Car Manufacturer	Can create a new car and record it on blockchain. Can sell the car to the customer. Can trace the complete ownership history of the car.
Transporter	Can deliver the car to the distributor. Can trace the complete ownership history of the car.
Distributor	Can send the car to the Dealer. Can trace the complete ownership history of the car.
Transport Authority	Can approve car registration. Can approve car ownership changes.
Dealer	Can sell the car to the Customer. Can trace the complete ownership history of the car.
Agents	Can sell the car to the customer. Can trace the complete ownership history of the car.
Customer	Can buy or sell the car. Can trace the complete ownership history of the car.
Service Center	Can service the car. Can trace the complete ownership history of the car

iv)UML Diagrams:

For the sake of this example, we will consider only 3 three parties - car manufacturer, dealer, and customer. However, the solution approach and concepts can be extended to more number of actors as well. We will be utilizing UML diagrams to capture these details as shown below:

ACTIVITY DIAGRAM:

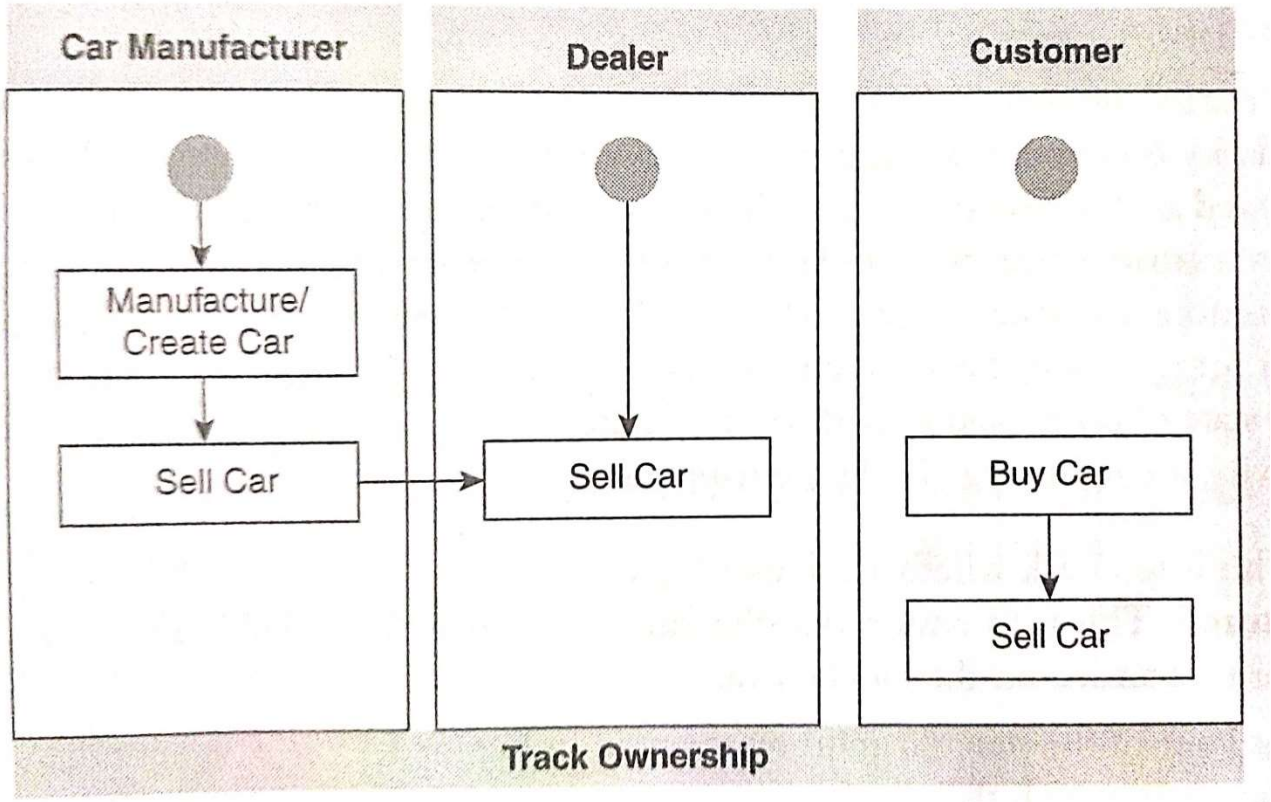


Figure 4.9 Car ownership tracking use case activity diagram.

SEQUENCE DIAGRAM:

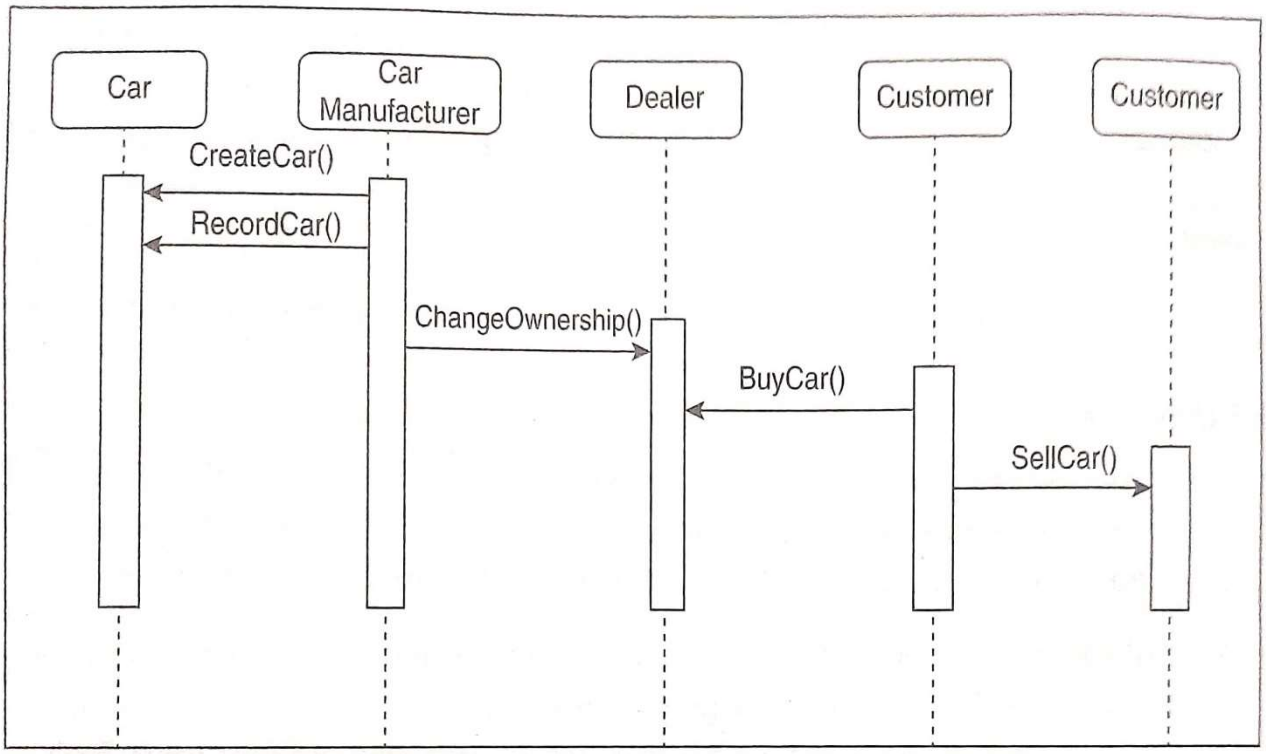


Figure 4.10 Car ownership tracking use case sequence diagram.

STATE DIAGRAM:

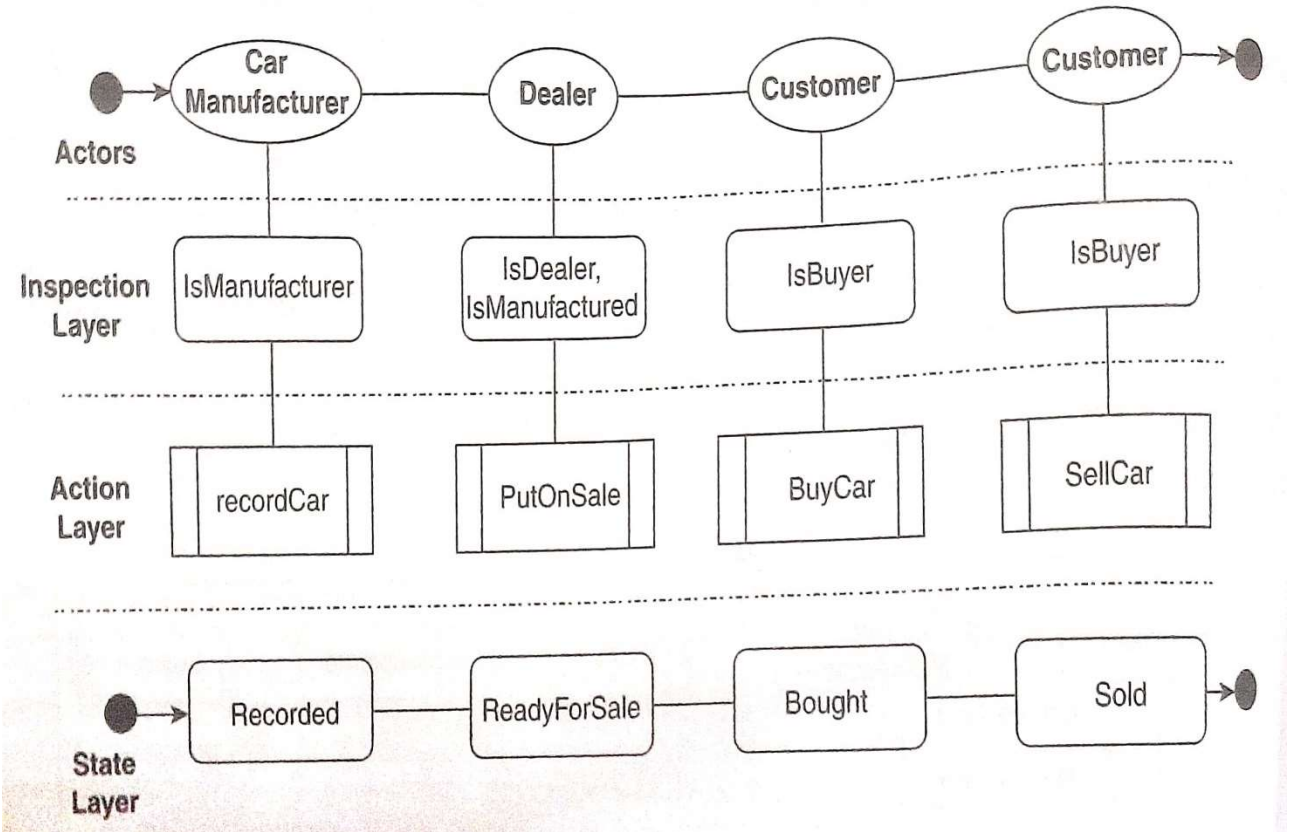


Figure 4.11 Car ownership tracking the use case state diagram.