upGrad
#LifeKoKaroLift

Further Scoping of DeFi

upGrad

Course: PGD Software Development (Blockchain)

Lecture On: Introduction to DeFi

Instructor: Jitender Bhutani

upGrad

Topics covered in the previous class...

1. Understand the Architecture of Defi Application
2. Create and Build a Defi App from scratch
    a. Smart Contract
    b. Backend

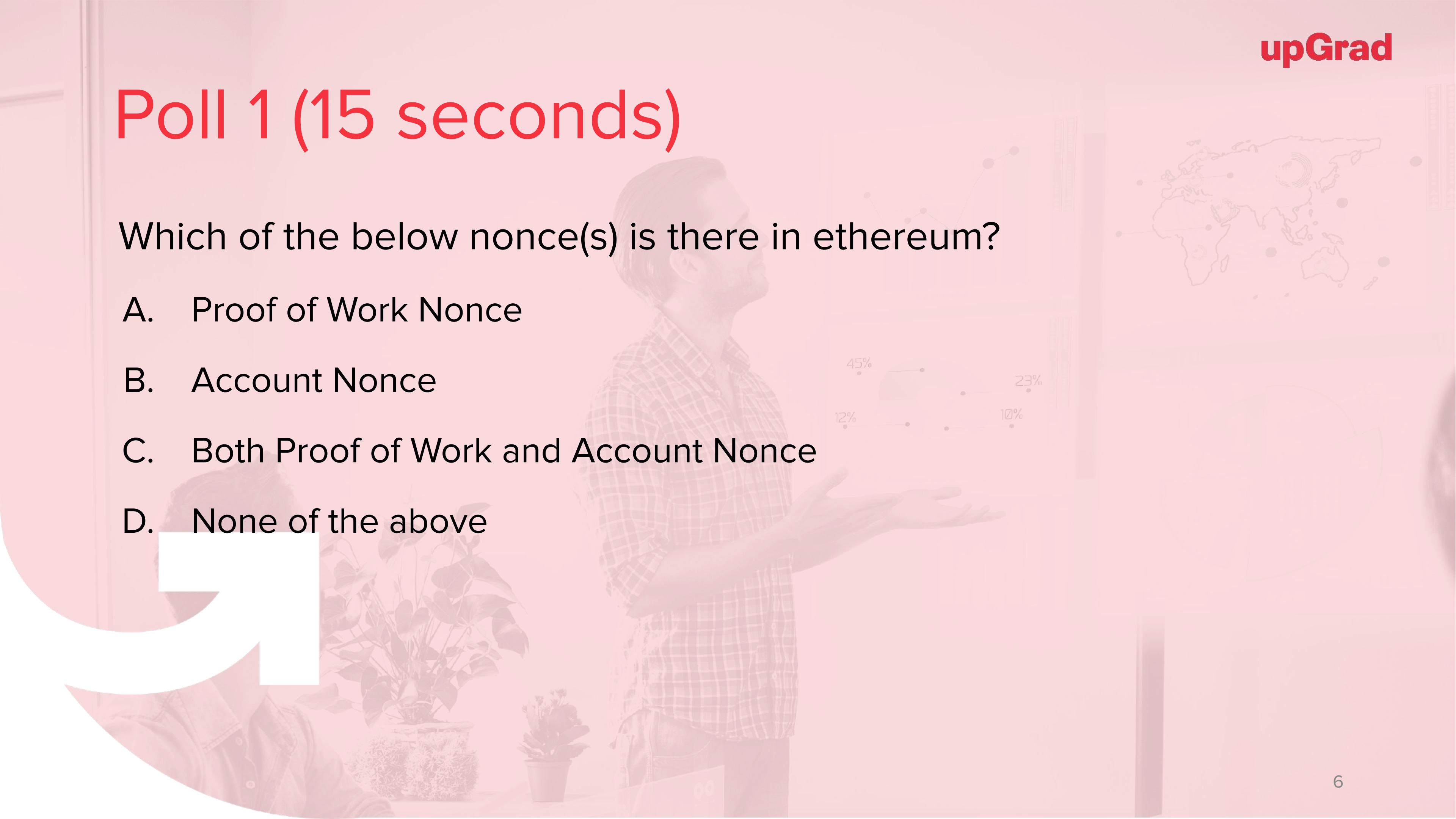
upGrad

Overview

- Session 1 : Understand the Architecture of DeFi Application
- Session 2 : Develop DeFi App Smart Contract
- Session 3 : Develop DeFi App Backend
- **Session 4 : Further scope**
    - **Frontend**
    - **Upgradable smart contract**
    - **Administration**
    - **Scalability**

Today’s Agenda

- Understand UI for DeFi
- Understand admin panel requirement
- Understand Upgradable smart contract
- How to scale app for better TPS



Poll 1 (15 seconds)

Which of the below nonce(s) is there in ethereum?

A. Proof of Work Nonce

B. Account Nonce

C. Both Proof of Work and Account Nonce

D. None of the above

upGrad

Poll 1 (15 seconds)

Which of the below nonce(s) is there in ethereum?

A. Proof of Work Nonce

B. Account Nonce

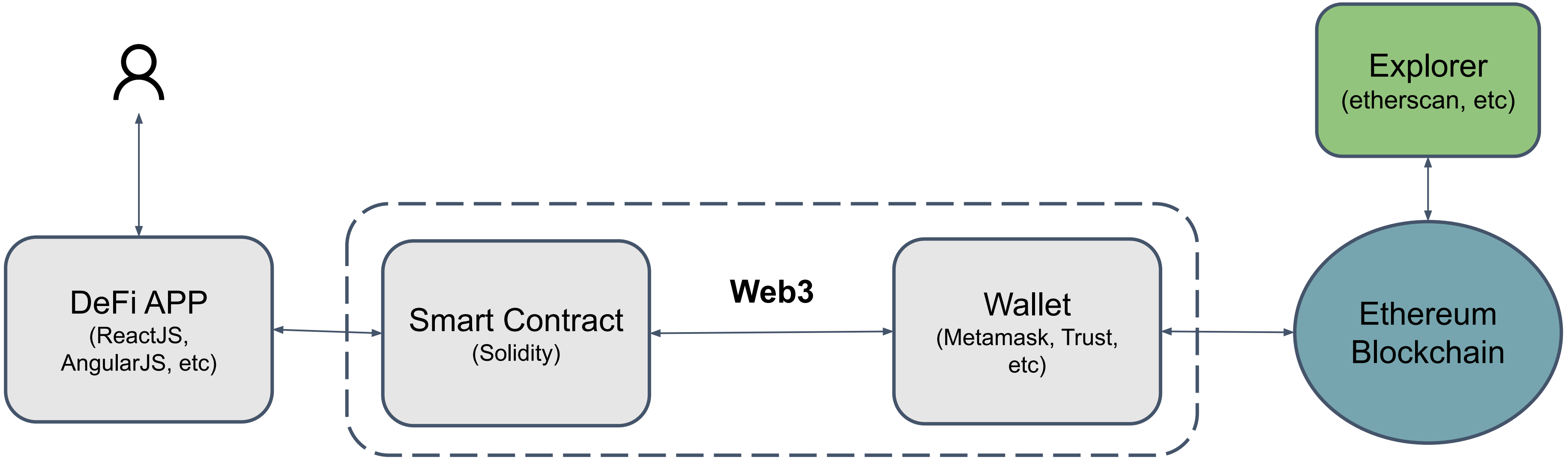
C. Both Proof of Work and Account Nonce

D. None of the above

FRONTEND

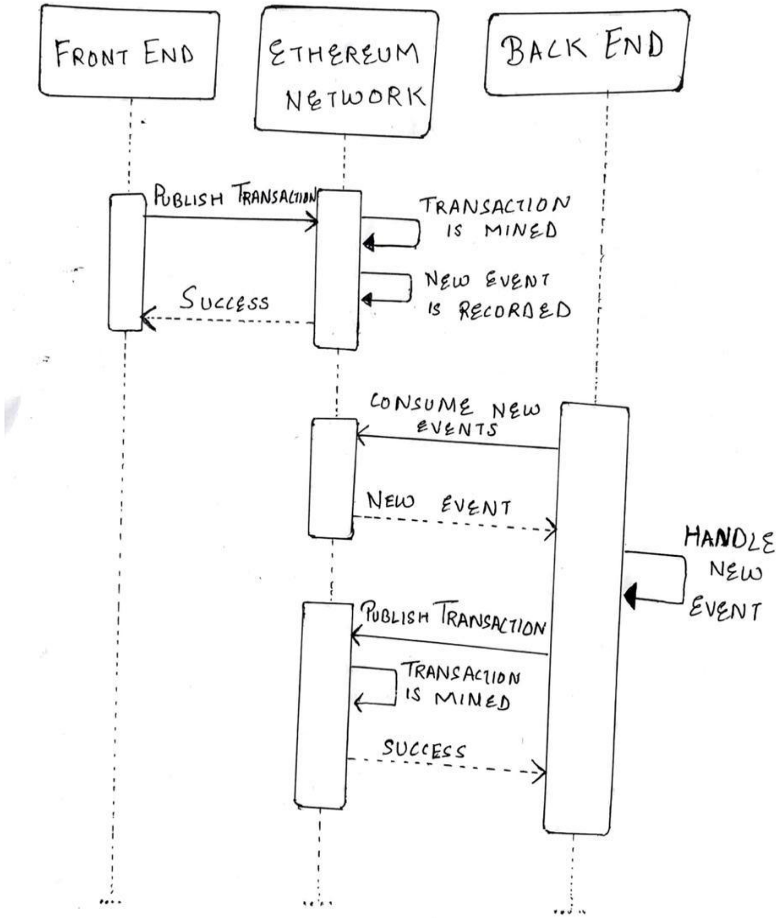
DeFi ARCHITECTURE

upGrad



DeFi ARCHITECTURE

upGrad

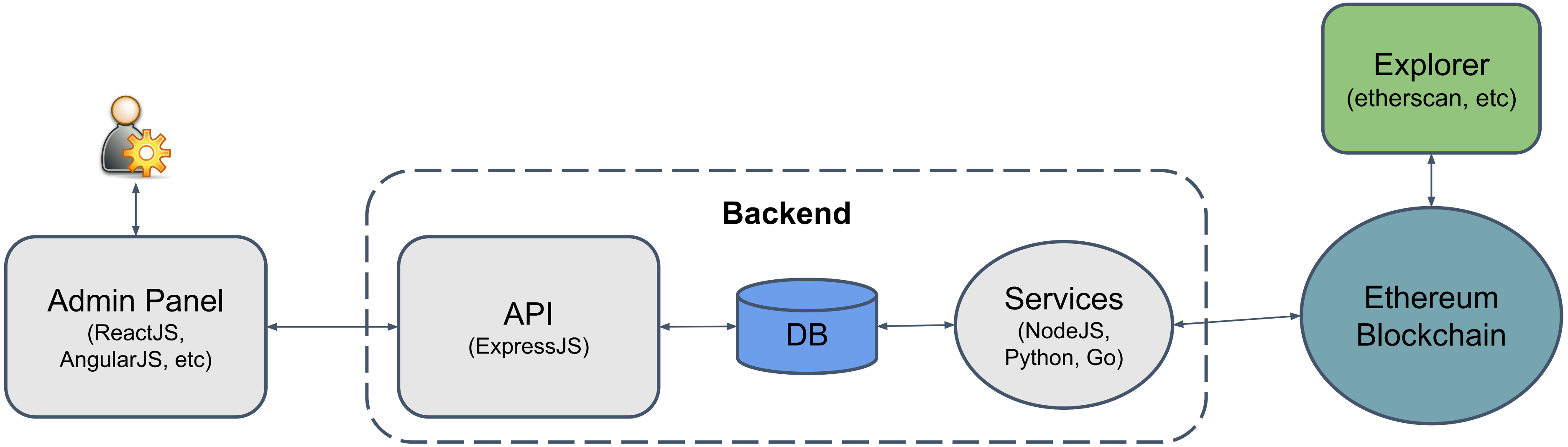


FRONTEND APP DEMO

upGrad

ADMIN PANEL

DeFi ARCHITECTURE



ADMIN PANEL

- App (Web + API)
  - Login
  - Get Smart Contract Transaction Data
  - Analysis
    - Charts
    - Graphs
  - Administrative work (if any)
    - Managing Permissions

ADMIN PANEL

- **Services** are used:
  - To capture transaction data from blockchain for our contract
    - From address and to address
    - Gas limit and price
    - Transaction Hash
    - Block number, etc
  - To save in database such as:
    - MongoDB
    - MySQL, etc

upGrad

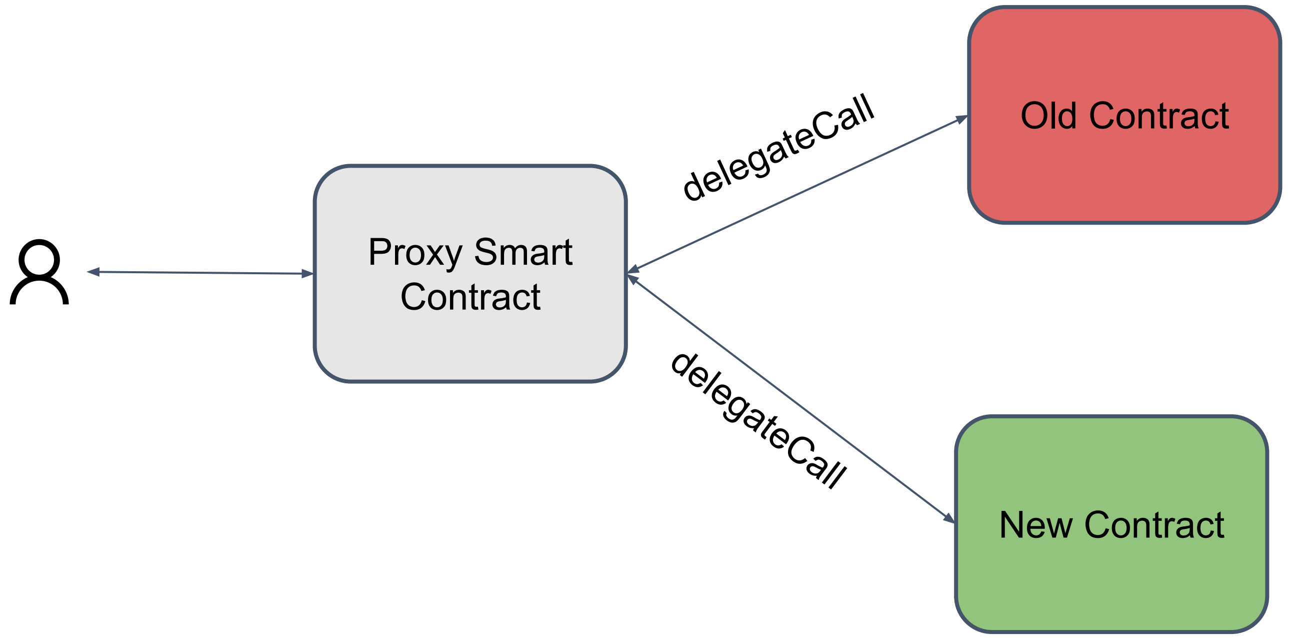
UPGRADABLE SMART CONTRACT

HOW TO UPGRADE A SMART CONTRACT ?

upGrad

- Deploy a new version of that same contract
- Now you have to migrate all the **state** from the old smart contract to the new one you deployed
- Now you'll have to update all the smart contracts that interacted with the old contract, and now use the address of the new one
- And now reach out and convince all the users to start using the new deployment

UPGRADABLE SMART CONTRACT



UPGRADABLE SMART CONTRACT

upGrad

- Smart Contracts are immutable in the Ethereum Blockchain and there is no way to update or alter them after deployment
- But the smart contracts that are deployed using the **OpenZeppelin Upgrades Plugins,** they can be upgraded to modify their contact's code. The contracts may upgrade the code but the **address, state and balance** of the contract is preserved.
- Therefore, this thing allows us to add new features to our projects and we can also fix bugs if needed

HOW UPGRADE WORKS?

Openzeppelin Upgrades Plugins actually deploys three smart contracts when we create a new upgradable smart contract instance. The three contracts are:

- The smart contract that we wrote and it’s also known as the implementation contract which contains all the logic
- A **ProxyAdmin** to be the admin of the proxy.
    - That is in charge of all the proxies you create via the Upgrades plugins
    - Interact with the proxies from any of your node’s accounts
- And a **proxy** to our implementation contract, which is the smart contract that we’ll interact with.

HOW UPGRADE WORKS?

upGrad

- Upgrade usually require the following steps:
    - Deploying our new implementation smart contract
    - And then sending transaction to the proxy that will update its implementation address to the new one

UPGRADE LIMITATIONS

- **Constructors** can’t be used in upgradable smart contracts
- When we upgrade our smart contract to new one, we can’t change the **storage layout** of the smart contract
- Therefore, if we have already declared a state variable in our contract, then we **can’t remove it** and also **we can not change variable’s type or even declare another variable before it**
- However, we can change the smart contract’s functions and events too. This limitation **only affects the state variables**

UPGRADABLE SMART CONTRACT DEMO (USING OPENZEPPELIN SDK)

Poll 1 (15 seconds)

Which of the following statement(s) is not true w.r.t to upgradable smart contracts

A. Add new functions

B. Add new state variables before existing variables

C. Remove existing states variables

D. Change data type of state variables

upGrad

Poll 1 (15 seconds)

Which of the following statement(s) is not true w.r.t to upgradable smart contracts

A. Add new functions

B. Add new state variables before existing variables

C. Remove existing states variables

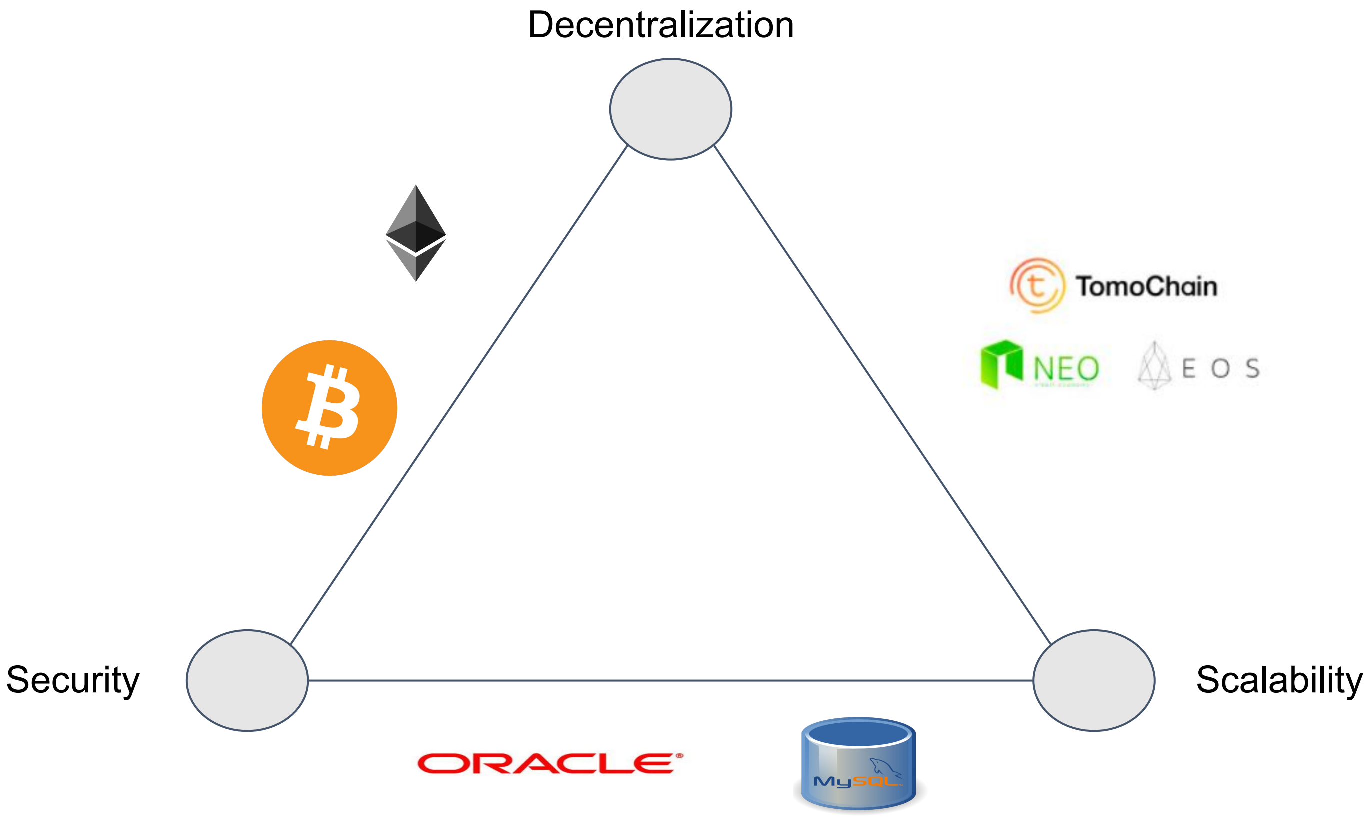
D. Change data type of state variables

SCALING APPS

SCALING TRILEMMA

- The scalability trilemma is a term stated by Ethereum's co-founder, Vitalik Buterin to describe that blockchain cannot **equally maximize** all three**:**
  - Scalability
  - Decentralization
  - Security
- The trilemma states that the blockchain system can only **maximize two** of the above listed, **at the cost of third**
- Therefore, every blockchain system has to make **trade-offs** depending on the functionality and **specific uses cases** for that particular blockchain system

Scaling Trilemma



SCALING TRILEMMA

- **Decentralization**
  - Data and its control is decentralised and distributed across **several** participating nodes in the network.
- **Scalability**
  - In a highly distributed blockchain system, it is very **challenging** to **update ledgers** on all the nodes concurrently

- **Security**.
  - Our system should be **resistant to attacks** and should also be **immutable** (for example, resistant to 51% attack, DDoS attacks, etc.)

SCALING TRILEMMA

upGrad

- **Decentralized & Secure**
  - Bitcoin and Ethereum blockchains are decentralized and secure.
  - The transaction speed is very slow, and therefore difficult to use it as a payment systems
  - However in comparison, the Visa payment can process 65,000 transaction messages per second
- **Security & Scalability**
  - SQL and Oracle, SAP
  - **Hyperledger** would be the **closest blockchain equivalent**
  - Stored in a **few locations** as compared to thousands of node in a decentralised blockchain system.

SCALING TRILEMMA

- **Scalability & De-centralization.**
  - There are few blockchains which sacrifice some decentralization for scalability. Examples are EOS, NEO, Tomochain etc
  - Consensus : Proof-of-Stake or Delegated-Proof-of-Stake
  - The quantity or numbers of master nodes in the blockchain system basically defines how decentralized that blockchain system is.
  - Number of master nodes for respective blockchains like Neo, EOS and Tomochain are 7, 15, and 150.

SCALING

- Ethereum processes roughly **15** transactions per **second**
- **On-Chain Scaling**
  - Increase the amount of **data** that can be stored in each **block**
  - Optimize transaction **validation** process
- **Off-Chain Scaling**
  - These are the additional layers which can manage the transactions without the involvement of blockchain network
  - Examples :
    - We can batch multiple payments into one transaction
    - Sidechains are example of off-chain scaling
  - Core Idea
    - The blockchain should always be used as a **trust** and **arbitration** layer

SCALING : ETH 2.0

upGrad

- Ethereum is coming up ETH 2.0 which focus on three scalability solutions to improve and make ethereum more widely accepted.
  - **State Channels**
    - Users will be able to transact peer to peer using ‘**off-chain**’
    - They’ll only send messages onto the **main blockchain** when they actually want to **exit** the channel

  - **Plasma (layer 2)**
    - **Child** blockchains connected to Ethereum through a root contract.
    - Root contract defines the **rule** of the child chain and also creates a permanent record of the state
    - If the users want to move back to the main chain, then they must **follow the rules** which are set out in the root contact

SCALING : ETH 2.0

upGrad

        - Creating **smaller blockchain** with **infinite complexity** as long as the Ethereum network is able to verify it
    - **Sharding**
        - For On-chain performance
        - The entire Ethereum network will be splitted into **multiple portions** also known as “shards”
        - Shards can be considered as nothing but **a separate** blockchain which has its **own states**
        - Most technical complex to implement
            - Like cross-shard communication

SCALING TECHNIQUES

- **Bulk State Update in Single Transaction**
    - Writing batch functions in smart contract for multiple state changes in single call
- **Use Custom Nodes**
    - Use self deployed nodes to broadcast transaction
- **Sidechains for DApps**
    - Example : Binance
- **Lightning Network**

SCALING

upGrad

- What is the best scaling strategy for our use case?
    - Implement bulk state updates functions
    - Use side chains
    - Use multiple custom nodes
    - Use state channels

FUTURE SCOPE

upGrad

- At the end of the day, what could be a better sector than finances for cryptocurrencies to dominate, after all, they are currencies and the first use case for a blockchain was a financial one.

FUTURE OF DEFI

FUTURE SCOPE

upGrad

- The future is impossible to predict but the way things are resolving now, the future of DeFi seems good. With the exponential increase of projects being connected to each other, with tokenizing more stuff and consequently more funds being locked into DeFi ecosystem, this might just be the next gold rush in crypto space.
- The goal of the ecosystem should be that anyone can invest in anything regardless of the amount that they contribute as blockchain technology allows easy pooling to collect enough funds you could look for example at investing in real estate market with 1$.
- In traditional markets, this would be impossible as it would not be worth the time for any intermediary. Also, the returns in a decentralized system can be higher because there is no middle man taking their **cut of the profits.**

DOUBT CLEARANCE WINDOW

In this class, you learnt that:

1. Understand UI for dApp
2. How to integrate it with existing wallets like metmask
3. Understand Upgradable smart contract
4. Discussed admin panel development
5. How to scale app for better TPS

HOMEWORK

1. Write script to decode smart contract transaction data
2. Extract function names and input params from decoded transaction data
3. Check if the above transaction has failed while executing smart contract function

FUTURE SCOPE

upGrad

1. Write services to capture smart contract transaction in any database for admin panel
2. Write API to get data from database for your smart contract
3. Write batch functions in smart contract to speed up some of the functions in DeFi
4. Convert our defi smart contract to upgradable smart contract
5. Write further test cases for smart contract

upGrad
#RahoAmbitious

Thank You!