VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi-560014

An Internship Report

On

"Stellar Wallet Using Blockchain"

Submitted in partial fulfillment of the requirement of VI semester Internship

Submitted by,

R Shreyas **1DT17IS064**

Under the guidance of

Mrs. Rekha V.S
Asst. Professor
Dept. of ISE
DSATM, Bangalore.

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT
(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi)
All B.E branches Accredited 3 years by NBA, New Delhi, (Validity: 26-07-2018 to 30-06-2021)
Opp. Art of Living, Udayapura, Kanakapura Road, Bangalore- 560082
2020-2021

DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT

(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi)
All B.E branches Accredited 3 years by NBA, New Delhi, (Validity: 26-07-2018 to 30-06-2021)
Opp. Art of Living, Udayapura, Kanakapura Road, Bangalore- 560082

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

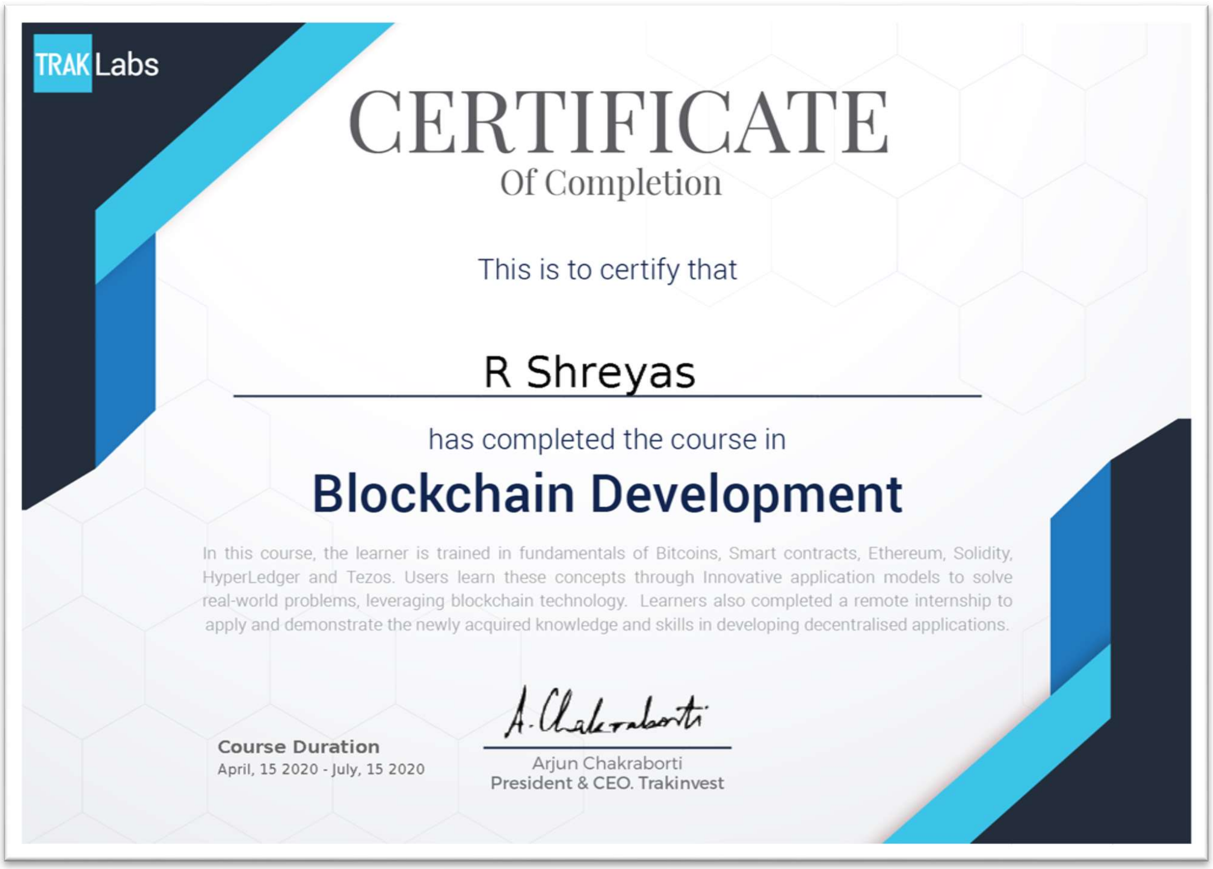
Certificate

This is to certify that the internship work for **"Stellar Wallet Using Blockchain"** is carried out by **R SHREYAS(1DT17IS064)** in partial fulfillment for the for the award of Bachelor of Engineering **in Information Science and Engineering** of the **Visvesvaraya Technological University, Belagavi** during the year 2020-2021. It is certified that all the corrections/ suggestions indicated for the given internal assessment have been incorporated in the report. The internship report has been approved as it satisfies the academic requirements in respect of internship work prescribed for the said Degree.

Signature of the Guide	**Signature of the HOD**
Mrs. Rekha V.S	**Dr. Sumithra Devi K A**
Asst. Professor, Dept. of ISE	Prof. & Head, Dept. of ISE
DSATM, Bangalore.	DSATM, Bangalore.

External Viva

Name of the Examiners	**Signature with date**
1.	
2.	



TRAK Labs

CERTIFICATE
Of Completion

This is to certify that

R Shreyas

has completed the course in

Blockchain Development

In this course, the learner is trained in fundamentals of Bitcoins, Smart contracts, Ethereum, Solidity, HyperLedger and Tezos. Users learn these concepts through Innovative application models to solve real-world problems, leveraging blockchain technology. Learners also completed a remote internship to apply and demonstrate the newly acquired knowledge and skills in developing decentralised applications.

Course Duration
April, 15 2020 - July, 15 2020

A. Chakraborti

Arjun Chakraborti
President & CEO. Trakinvest

LeewayHertz
You dream it. We build it.

To whomsoever it may concern

Date: 22 SEPT 2020

This is to certify that R Shreyas, a student of B.E from Dayananda Sagar Academy of Technology and Management, Bangalore has successfully completed the training with us. The training period was from 27 July, 2020 till 27 August, 2020. The Trainee has done Blockchain Internship with LeewayHertz Technologies Pvt. Ltd.

During the training period the trainee was found to be punctual and hardworking. We wish you all the best for your future endeavours!

For LeewayHertz Technologies Pvt. Ltd.

[illegible]

Sakshi Ralhan
Sr. Executive - Human Resources

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the project phase I and seminar would be incomplete without the mention of those people, who made it possible through their guidance, support and encouragement.

We would like to thank the **VTU, Belagavi**, for having this project work as part of its curriculum, which gave us a wonderful opportunity to work on our research and presentation abilities.

We extend our deep sense of sincere gratitude to **Dr. B R Lakshmikantha, Principal**, DSATM, Bengaluru, for providing us an opportunity to finish the necessary work.

We are thankful to all the people who have directly and indirectly involved in this project work.I/We am/are grateful to **Dr. Sumithra Devi K A, Dean Academics and Head of Department, Information Science and Engineering, DSATM, Bangalore,** for her valuable suggestions and advice throughout my/our work period.

I would like to express my deepest gratitude and sincere thanks to our guides **Mrs. Rekha V.S, Assistant Professor, Department of Information Science and Engineering, DSATM, Bangalore,** for her keen interest and encouragement in the project whose guidance made the project into reality.

I/We would like to thank all the staff members of **Department of Information Science and Engineering** for their support and encouragement during the course of this project.

Definitely most, I/We would like to thank my/our parents, all my family members and friends, without whose help and encouragement this project would have been impossible.

R SHREYAS 1DT17IS064

ABSTRACT

Blockchain is a decentralized, digital ledger system that serves as the foundation upon which Bitcoin and other cryptocurrencies operate. Once the transaction is encoded in the Blockchain and becomes part of the network, the parties cannot reverse or alter it without mutual agreement. Smart contracts are used to exchange money or ownership, store data, make decisions, and interact with other contracts. They are pieces of computer code that are programmed to perform operations on an outcome of an event based on predetermined criteria set by the creators of the contract. Often these smart contracts exist on distributed ledger technology, as of now, they primarily use blockchain. Ethereum is a well-known blockchain network that uses smart contracts to create decentralized applications, issue tokens, and manage governance. While at first smart contracts may seem mysterious, at a second glance their operations and applications start to make a little more sense.

Stellar is an open blockchain network based on a trust model with open membership. Its native cryptocurrency, Lumen (XLM), is a top ten cryptocurrency in market capitalization. Stellar uniqueness comes from its consensus protocol called Federated Byzantine Agreement (FBA), which is an open membership variant of Byzantine Agreement. The security of the system is achieved by nodes forming a network of trust. In contrast to proof-of-work based consensus algorithms, SCP finalizes transactions in a matter of seconds and is secure against adversaries with vast computational power. Although Stellar does not offer smart contracts in the Turing-completeness sense, it allows creating complex transactions that represent some subset of simple smart contracts. In the following sections, we present the smart contract used in the i-voting protocol.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
COMPANY PROFILE

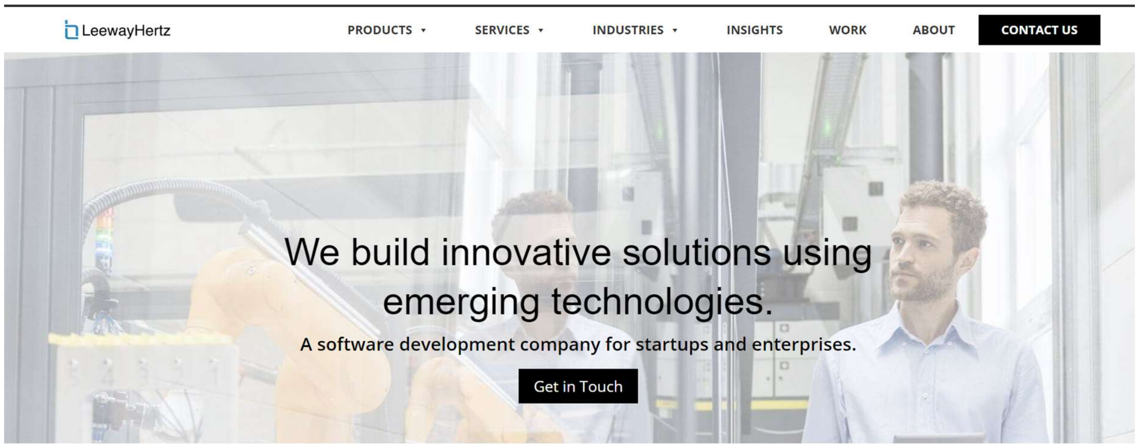

LeewayHertz
PRODUCTS
SERVICES
INDUSTRIES
INSIGHTS
WORK
ABOUT
CONTACT US
We build innovative solutions using emerging technologies.
A software development company for startups and enterprises.
Get in Touch

Figure 1.1: Shows LeewayHertz Company website

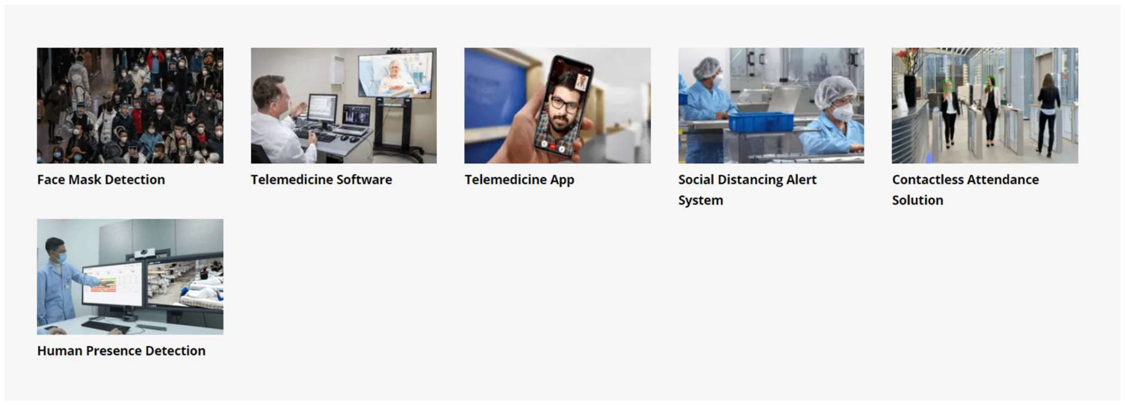


Figure 1.2: Shows Products developed by LeewayHertz

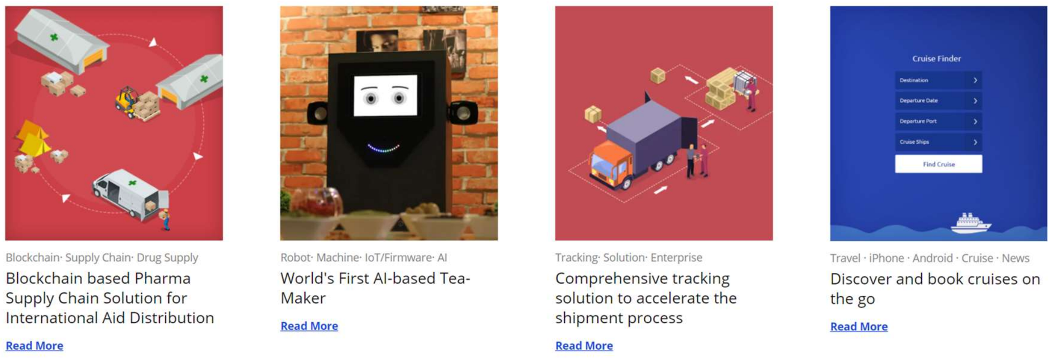

Blockchain· Supply Chain· Drug Supply
Blockchain based Pharma Supply Chain Solution for International Aid Distribution
Read More
Robot· Machine· IoT/Firmware· AI
World's First AI-based Tea-Maker
Read More
Tracking· Solution· Enterprise
Comprehensive tracking solution to accelerate the shipment process
Read More
Travel · iPhone · Android · Cruise · News
Discover and book cruises on the go
Read More

Figure 1.3: Shows recent Projects developed by LeewayHertz

LeewayHertz provides enterprise application development, designing and consulting services. LeewayHertz is a Private company. Akash Takyar is the Founder & CEO of LeewayHertz. Headquartered at San Francisco and founded in 2007, LeewayHertz is one of the first few companies to build and launch a commercial app on Apple's App Store. Our team of certified designers and developers has designed and developed more than 100 digital platforms on Mobile, Cloud, AI, IoT and Blockchain. They design, develop, and deploy enterprise-grade applications to help businesses digitize their operations with the latest technology. With 100+ platforms successfully delivered to start-ups and Fortune 500 companies, they are proven experts with 10 years of experience.

At LeewayHertz, we have developed digital solutions for Fortune 500 companies and start-ups to ease their business functions with the latest technologies. Being an award-winning software development company, we have also proven our expertise in blockchain development and worked on more than 20+ blockchain projects. Focusing on Blockchain, AI, mobile apps, IoT and Cloud platforms, we build end to end solution for businesses. Marketing growth hack team at LeewayHertz has helped companies reach the potential of acquiring millions of users for B2C platforms. With a portfolio of award-winning apps, we have launched over 100+ digital platforms. We have created a workforce of blockchain developers who can build blockchain apps on different blockchain platforms such as Ethereum, Hyperledger Fabric, Hyperledger Sawtooth, Hyperledger Iroha, Hyperledger Indy, EOS, Stellar, Tron and Corda.

They design, develop, deploy and maintain technology products. They also helped with conceptualization, wireframes, and a platform migration per their suggestion. Uber and Twitter are using our inventions and patents. They work with tech geeks and passionate technologists who are trained by the experts at Apple and Google and always remains at the cutting edge of technology.

Some of our reputed clients include ESPN, NASCAR, Hershey's, McKinsey, P&G, Siemens, 3M, Pearson and more. The management team echoes LeewayHertz’s motto of ‘You Dream It: We Build It.’ and knows how to take custom software application ideas from concept to delivery. Holding the experience of working with 30+ fortune 500 companies. They have developed applications that are now being used by millions of consumers worldwide.

CHAPTER 2

TASK PERFORMED

Day 01: Introduction to Database

Database:

Database, also called electronic database, any collection of data, or information, that is specially organized for rapid search and retrieval by a computer. Databases are structured to facilitate the storage, retrieval, modification, and deletion of data in conjunction with various data-processing operation. A database is stored as a file or a set of files. The information in these files may be broken down into records, each of which consists of one or more fields. Fields are the basic units of data storage, and each field typically contains information pertaining to one aspect or attribute of the entity described by the database.

Depending upon the usage requirements, there are following types of databases available in the market :

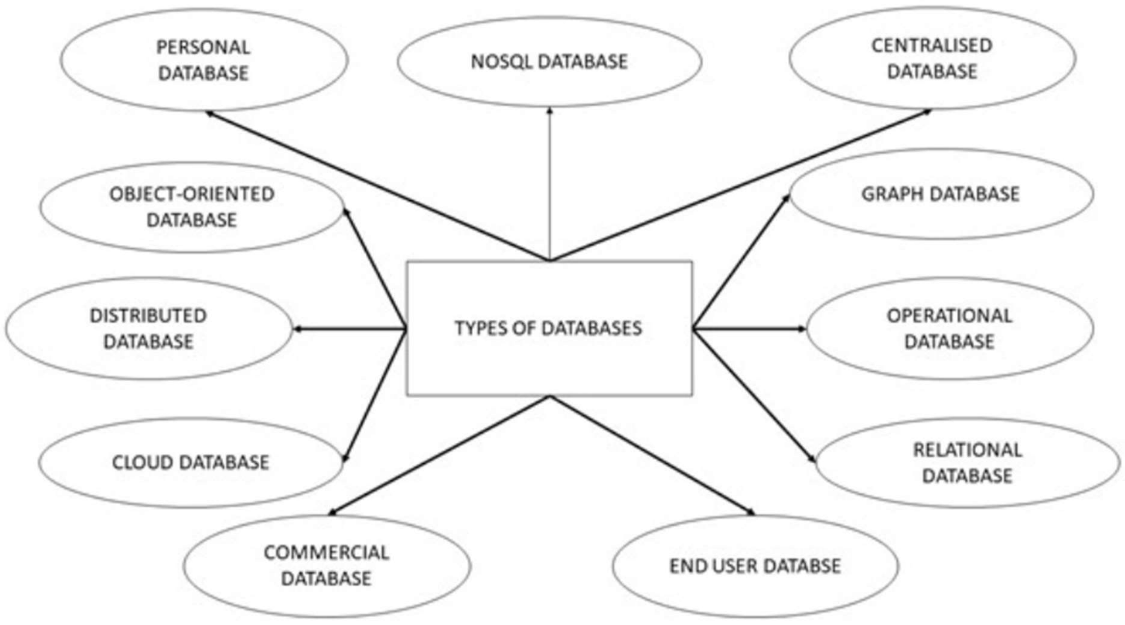


Figure 2.1: Shows different types of Databases

Ledger:

A ledger is a key concept in Hyperledger Fabric; it stores important factual information about business objects; both the current value of the attributes of the objects, and the history of transactions that resulted in these current values. The blockchain is structured as sequential log of interlinked blocks, where each block contains a sequence of transactions, each transaction representing a query or update to the world state. The exact mechanism by which transactions are ordered is discussed elsewhere; what's important is that block sequencing, as well as transaction sequencing within blocks, is established when blocks are first created by a Hyperledger Fabric component called the ordering service.

Day 02: Introduction to Blockchain

Blockchain:

Blockchain is a public digital ledger of past transactions in order. For the rest of the article, we will consider this to be a ledger of bitcoin transactions. The blockchain is called so because it is a chain of blocks. A blockchain is a hash-linked data structure. This ledger is stored on a decentralized network where all the hashing is enabled though cryptography.

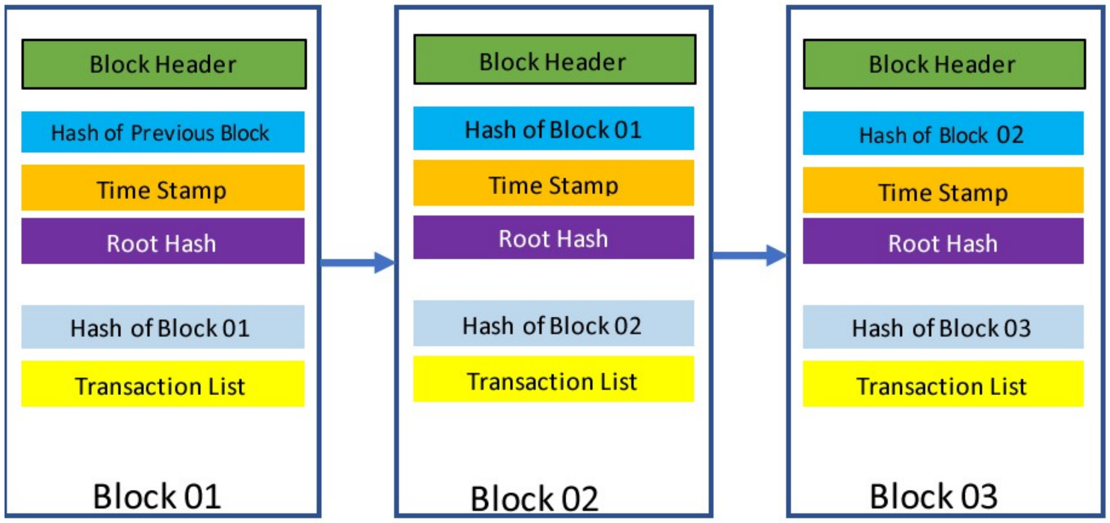


Figure 2.2: Shows Structure of Blockchain

The blockchain data structure is a back-linked list of blocks of transactions, which is ordered. It can be stored as a flat file or in a simple database. Each block is identifiable by a hash, generated using the SHA256 cryptographic hash algorithm on the header of the block. Each block references a previous block, also known as the parent block, in the "previous block hash" field, in the block header.

A hash, also known in long form as cryptographic hash function, is a mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size. In the case of SHA 256, the result is a string of 32 bytes. A peer-to-peer computer process, Blockchain mining is used to secure and verify bitcoin transactions. Mining involves Blockchain miners who add bitcoin transaction data to Bitcoin's global public ledger of past transactions. In the ledgers, blocks are secured by Blockchain miners and are connected to each other forming a chain. At its core, the term 'Blockchain mining' is used to describe the process of adding transaction records to the bitcoin blockchain. This process of adding blocks to the blockchain is how transactions are processed and how money moves around securely on Bitcoins. This process of Blockchain mining is performed by a community of people around the world called 'Blockchain miners.'

Proof of Work (Consensus Algorithm):

Proof of Work is one of the consensus algorithms. As the name suggests for validating a transaction a node should publicly prove that it did a certain amount of work. Node has to show proof of its work and they do it by solving a complex cryptographic puzzle. Before going into details about POW, let's see some of the important terms used in POW. We can take Bitcoin blockchain as an example:

Block: A block consist of a set of transactions, nonce, timestamp, hash of the previous block, the answer of the puzzle (Nonce) and index value. Each block in blockchain is linked to its previous block.

Miners: In bitcoin blockchain POW is conducted through miners, miners are the people keeping the blockchain running by providing a huge amount of computing resources competing to solve a cryptographic puzzleand upon solving the puzzle they generate a block and they also get rewarded. Miners compete with each other to generate a valid block of transactions. Miners collect all pending transactions from the decentralized network then they guess a random number (nonce) to solve cryptographic puzzle, on successfully solving the puzzle they generate a block then they push that block into the network for verification from other nodes so that other nodes after verification can add that block in there copy of blockchain.

Nonce: The cryptographic puzzle that miners solve is to identify the value of nonce. A nonce is a random number which can be used only one time. Mostly it is a random number with combination of some data. Blockchain adds a value called nonce in each block. This nonce is like a salt added to the contents of a block. By adding nonce, the hash output of the contents of the block will change.

Hash: Hashing is a cryptographic technique which maps input data to data of a fixed size output. Bitcoin uses the SHA-256 algorithm for it. SHA-256 output a fixed length number. A slight change in input would change the complete output but the output would always of the same length.

Day 03: Introduction to Bitcoin

Bitcoin:

Bitcoin is a digital currency created in January 2009 following the housing market crash. It follows the ideas set out in a whitepaper by the mysterious and pseudonymous Satoshi Nakamoto. The identity of the person or persons who created the technology is still a mystery. Bitcoin offers the promise of lower transaction fees than traditional online payment mechanisms and is operated by a decentralized authority, unlike government-issued currencies. There are no physical bitcoins, only balances kept on a public ledger that everyone has transparent access to, that – along with all Bitcoin transactions – is verified by a massive amount of computing power.

You'll need a Bitcoin wallet to manage your bitcoins for you because when you send someone bitcoins, the rest of the Bitcoin network needs to know so that the value can be tracked. The only way that the Bitcoin network knows that your address has some unspent bitcoins is because everyone on the Bitcoin network agrees that you do. This is where the value of Bitcoin lies—the value is there because people have placed a value on it.

A bitcoin exchange is a digital marketplace where traders can buy and sell bitcoins using different fiat currencies or altcoins. A bitcoin currency exchange is an online platform that acts as an intermediary between buyers and sellers of the cryptocurrency. When you send that bitcoin on to someone else, your wallet creates a transaction output, which is the address of the person you're sending the coin to. That transaction will then be registered on the bitcoin network with your bitcoin address as the transaction input.

Bitcoin mining is the processing of transactions in the digital currency system, in which the records of current Bitcoin transactions, known as a block, are added to the record of past transactions, known as the block chain. A Bitcoin is defined by the digitally signed record of its transactions, starting with its creation. The block is an encrypted hash proof of work, created in a compute-intensive process. Miners use software that accesses their processing capacity to solve transaction-related algorithms. In return, they are awarded a certain number of Bitcoins per block. The block chain prevents attempts to spend a Bitcoin more than once -- otherwise the digital currency could be counterfeited by copy and paste.

Bitcoin's P2P network architecture is much more than a topology choice. Bitcoin is a peer-to-peer digital cash system by design, and the network architecture is both a reflection and a foundation of that core characteristic. Decentralization of control is a core design principle and that can only be achieved and maintained by a flat, decentralized P2P consensus network.

Day 04: Introduction to Ethereum

Introduction to Ethereum:

Ethereum is an open blockchain platform that lets anyone build and use decentralized applications that run on blockchain technology.

Ether: Ether is a cryptocurrency whose blockchain is generated by the Ethereum platform.

Ethereum Blockchain

A blockchain is best described as a public database that is updated and shared across many computers in a network.

"Block" refers to the fact that data and state is stored in sequential batches or "blocks". If you send ETH to someone else, the transaction data needs to be added to a block for it to be successful.

"Chain" refers to the fact that each block cryptographically references its parent. A block's data cannot be changed without changing all subsequent blocks, which would require the consensus of the entire network.

Ethereum Virtual Machine (EVM)

Contracts written in smart contracts specific programming language compiled into bytecode. EVM is the Ethereum smart contract bytecode execution environment. Every node in the network runs EVM. All the node executes all the transactions that points to smart contracts using EVM so every node does the same calculation and store the same values.

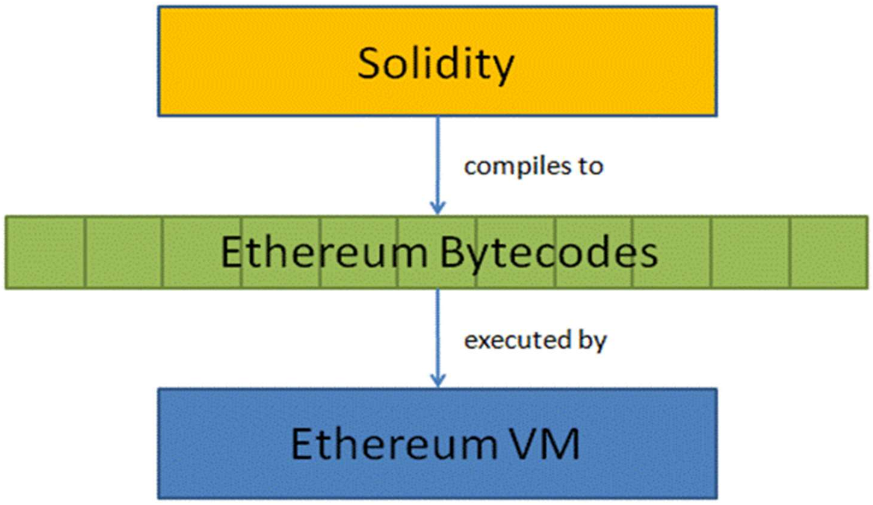


Figure 2.3: Shows Ethereum Structure

Transaction cycle in Ethereum:

1. Construct the raw transaction object
2. Sign the Transaction
3. Transaction is validated locally
4. Transaction is broadcast to the network
5. Miner Node accepts the transaction
6. Miner Node finds a valid block and broadcasts to the network
7. Local Node receives/syncs the new block

Day 06: Introduction to Smart Contracts

Smart contracts are lines of code that are stored on a blockchain and automatically execute when predetermined terms and conditions are met. At the most basic level, they are programs that run as they've been set up to run by the people who developed them. The benefits of smart contracts are most apparent in business collaborations, in which they are typically used to enforce some type of agreement so that all participants can be certain of the outcome without an intermediary's involvement.

Work of Smart Contracts:

The easiest way to explain what a smart contract does is through an example. If you've ever bought a car at a dealership, you know there are several steps and it can be a frustrating process. If can't pay for the car outright, you'll have to obtain financing. This will require a credit check and you'll have to fill out several forms with your personal information to verify your identity. Along the way, you'll have to interact with several different people, including the salesperson, finance broker and lender. To compensate their work, various commissions and fees are added to the base price of the car.

What smart contracts on blockchain can do is streamline this complex process that involves several intermediaries because of a lack of trust among participants in the transaction. With your identity stored on a blockchain, lenders can quickly make a decision about credit. Then, a smart contract would be created between your bank, the dealer and the lender so that once the funds have been released to the dealer, the lender will hold the car's title and repayment will be initiated based on the agreed terms. The transfer of ownership would be automatic as the transaction gets recorded to a blockchain, is shared among the participants and can be checked at any time.

Architecture of Smart Contract:

Smart contracts work by following simple "if/when…then…" statements that are written into code on a blockchain. A network of computers executes the actions (releasing funds to the appropriate parties; registering a vehicle; sending notifications; issuing a ticket) when predetermined conditions have been met and verified. The blockchain is then updated when the transaction is completed.

Sample Smart Contract:

```
pragma solidity ^0.4.21;

contract Coin {
   // The keyword "public" makes those variables
   // readable from outside.
   address public minter;
   mapping (address => uint) public balances;

   // Events allow light clients to react on
   // changes efficiently.
   event Sent(address from, address to, uint amount);

   // This is the constructor whose code is
   // run only when the contract is created.
   function Coin() public {
      minter = msg.sender;
   }

   function mint(address receiver, uint amount) public {
      if (msg.sender != minter) return;
      balances[receiver] += amount;
   }

   function send(address receiver, uint amount) public {
      if (balances[msg.sender] < amount) return;
      balances[msg.sender] -= amount;
      balances[receiver] += amount;
      emit Sent(msg.sender, receiver, amount);
   }}
```

Day 06: Introduction to Solidity

Solidity:

Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behavior of accounts within the Ethereum state. Solidity was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM). Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.

Solidity is a statically-typed programming language designed for developing smart contracts that run on the Ethereum Virtual Machine, also known as EVM. As specified by Wood it is designed around the ECMAScript syntax to make it familiar for existing web developers. Compared to other EVM-targeting languages of the time such as Serpent and Mutant, Solidity contained a number of important differences. Complex member variables for contracts including arbitrarily hierarchical mappings and structs were supported. Contracts support inheritance, including multiple inheritance with C3 linearization. An application binary interface (ABI) facilitating multiple type-safe functions within a single contract was also introduced.

Trust in smart contract can be better established if their source code is available. Since making source code available always touches on legal problems with regards to copyright, the Solidity compiler encourages the use of machine-readable SPDX license identifiers. Every source file should start with a comment indicating its license:

```
// SPDX-License-Identifier: MIT
```

Pragmas

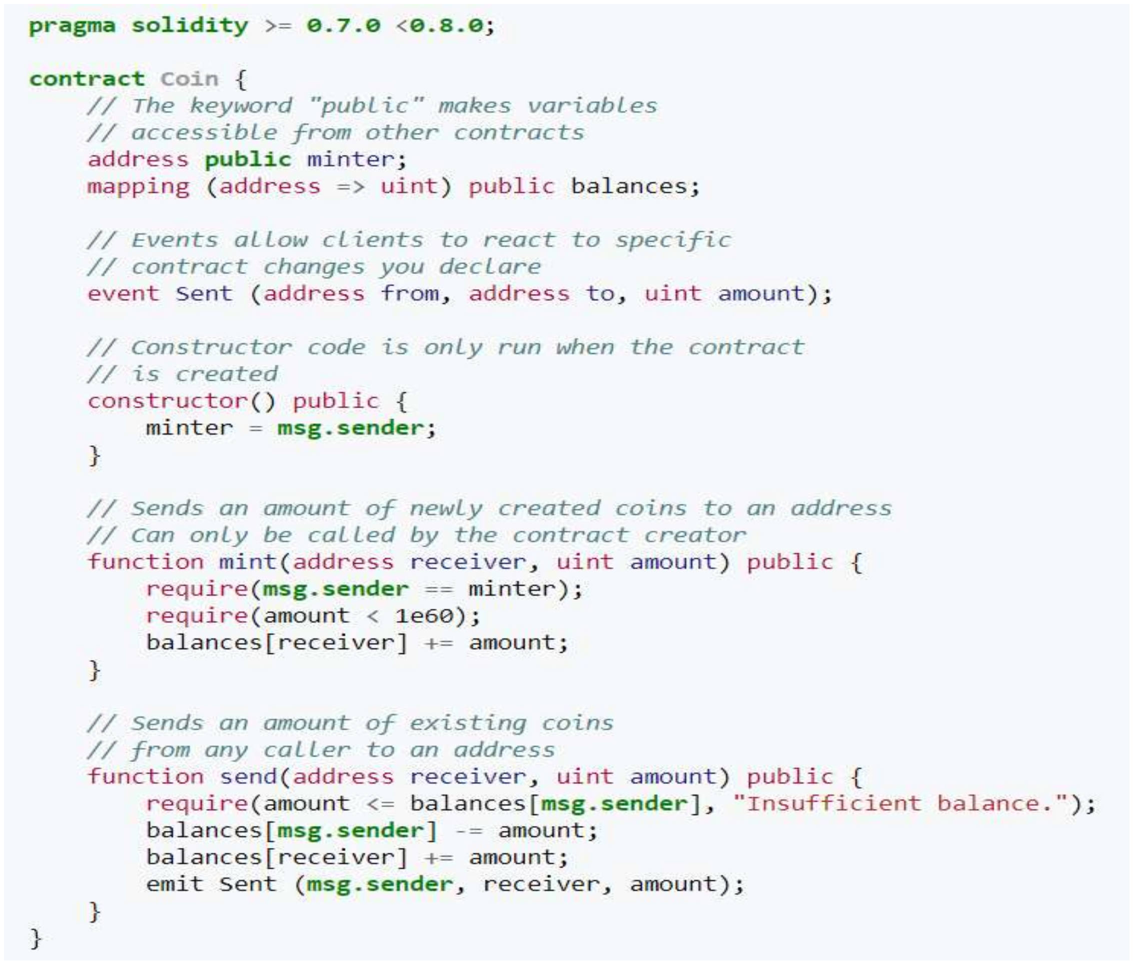
The pragma keyword is used to enable certain compiler features or checks. A pragma directive is always local to a source file, so you have to add the pragma to all your files if you want enable it in your whole project. If you import another file, the pragma from that file does not automatically apply to the importing file.

The version pragma is used as follows:

```
// pragma solidity ^0.5.2;
```

Remix IDE:

Remix IDE is an open-source web and desktop application. It fosters a fast development cycle and has a rich set of plugins with intuitive GUIs. Remix is used for the entire journey of contract development as well as for learning and working with Ethereum. Remix IDE is part of the Remix Project which is a platform for development tools that use a plugin architecture. It encompasses sub-projects including Remix Plugin Engine, Remix Libs, and of course Remix-IDE. Remix IDE is a powerful open-source tool that helps you write Solidity contracts straight from the browser. It is written in JavaScript and supports both usage in the browser, in the browser but run locally and in a desktop version. Remix IDE has modules for testing, debugging and deploying of smart contracts and much more.

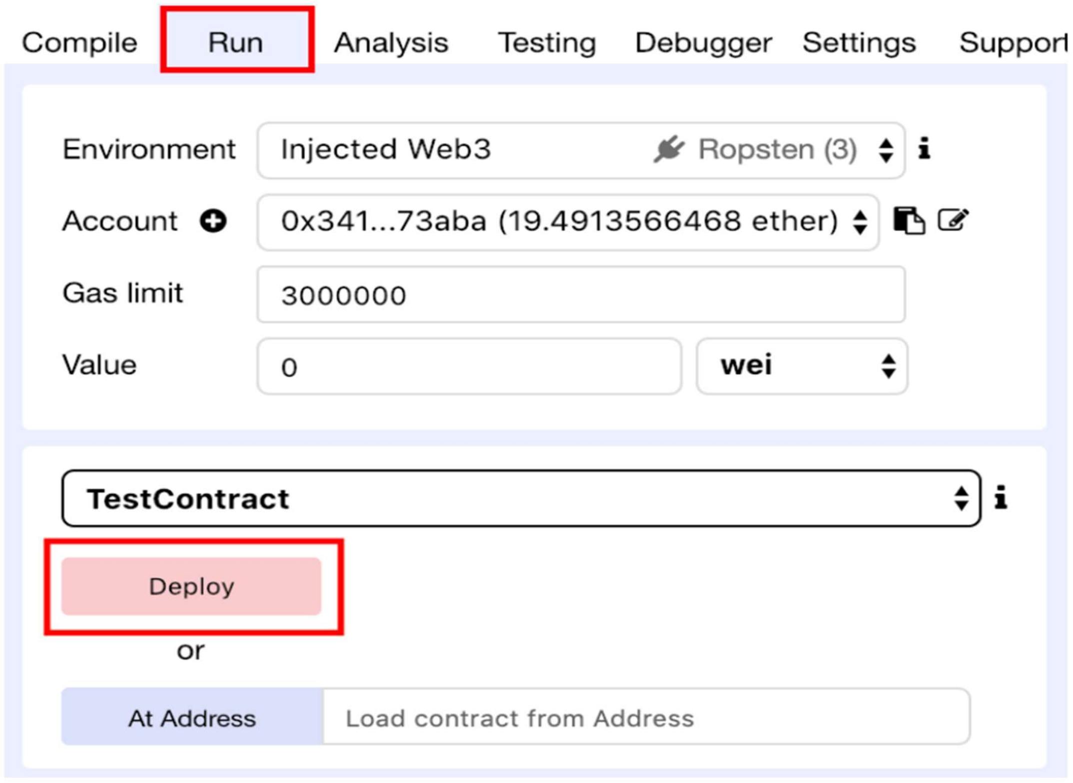


```
pragma solidity >= 0.7.0 <0.8.0;

contract Coin {
    // The keyword "public" makes variables
    // accessible from other contracts
    address public minter;
    mapping (address => uint) public balances;

    // Events allow clients to react to specific
    // contract changes you declare
    event Sent (address from, address to, uint amount);

    // Constructor code is only run when the contract
    // is created
    constructor() public {
        minter = msg.sender;
    }

    // Sends an amount of newly created coins to an address
    // Can only be called by the contract creator
    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        require(amount < 1e60);
        balances[receiver] += amount;
    }

    // Sends an amount of existing coins
    // from any caller to an address
    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender], "Insufficient balance.");
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent (msg.sender, receiver, amount);
    }
}
```

Figure 2.4: Shows an Example Solidity program

Day 07: Deployment over RemixIDE

After successfully compiling your contract, you can now deploy it. Deploying a contract will let you test it and use it to your applications. Go to the Run Tab located at the side of the Compile Tab:



Environment: Injected Web3 — Ropsten (3)
Account: 0x341...73aba (19.4913566468 ether)
Gas limit: 3000000
Value: 0 wei
TestContract

Figure 2.5: Shows Deployment in Remix IDE

JavaScript VM - The instance will be deployed in a sandbox blockchain in the browser. This means nothing will be saved when the page reloads, a new instance will be created.

Injected Provider - Remix will connect to an injected web3 provider. Metamask and Mist are example of injected web3 providers.

Web3 Provider - Remix will connect to a remote instance. You will need to provide the URL address of your selected provider.

Account - the list of accounts that will be associated with the current instance.

Gas Limit - the maximum gas amount of each transaction.

Value - the amount of value for the next created transaction.

Wallets:

A crypto wallet that interacts with smart contracts is called a smart contract wallet. In other words, these wallets enable different types of activities such as buying or selling tokens, token exchange, dapp interaction, trade, borrow or lend, etc.

Transaction in smart contract:

A smart contract is an agreement between two people in the form of computer code. They run on the blockchain, so they are stored on a public database and cannot be changed. The transactions that happen in a smart contract are processed by the blockchain, which means they can be sent automatically without a third party. This means there is no one to rely on it. The transactions only happen when the conditions in the agreement are met — there is no third party, so there are no issues with trust.

Day 08: Introduction to HyperLedger

HyperLedger Fabric:

Hyperledger Fabric is one of the blockchain projects within Hyperledger. Like other blockchain technologies, it has a ledger, uses smart contracts, and is a system by which participants manage their transactions. Where Hyperledger Fabric breaks from some other blockchain systems is that it is private and permissioned. Rather than an open permission-less system that allows unknown identities to participate in the network (requiring protocols like "proof of work" to validate transactions and secure the network), the members of a Hyperledger Fabric network enrol through a trusted Membership Service Provider (MSP). Hyperledger Fabric also offers several pluggable options. Ledger data can be stored in multiple formats, consensus mechanisms can be swapped in and out, and different MSPs are supported.

A Hyperledger Fabric channel is a private "subnet" of communication between two or more specific network members, for the purpose of conducting private and confidential transactions. A channel is defined by members (organizations), anchor peers per member, the shared ledger, chain code application(s) and the ordering service node(s). Each transaction on the network is executed on a channel, where each party must be authenticated and authorized to transact on that channel. Each peer that joins a channel, has its own identity given by a membership services provider (MSP), which authenticates each peer to its channel peers and services.

Day 09: Introduction to HyperLedger Composer:

Hyperledger Composer is an extensive, open development toolset and framework to make developing blockchain applications easier. Our primary goal is to accelerate time to value, and make it easier to integrate your blockchain applications with the existing business systems. You can use Composer to rapidly develop use cases and deploy a blockchain solution in weeks rather than months. Composer allows you to model your business network and integrate existing systems and data with your blockchain applications. Hyperledger Composer supports the existing Hyperledger Fabric blockchain infrastructure and runtime, which supports pluggable blockchain consensus protocols to ensure that transactions are validated according to policy by the designated business network participants.

Blockchain package management tooling contributed by IBM. Composer is a user-facing rapid prototyping tooling, running on top of Hyperledger Fabric, which allows the easy management of Assets (data stored on the blockchain), Participants (identity management, or member services) and Transactions. The resulting application can be exported as a package (a BNA file) which may be executed on a Hyperledger Fabric instance, with the support of a Node.js application (based on the Loopback application framework) and provide a REST interface to external applications.

Development environment:

Composer-CLI is the most important tool for Composer deployment; it contains all the essential command-line operations. Other very useful tools include Composer REST server, generator Hyperledger Composer and Playground. Composer CLI provides many useful tools for developers.

Step 1: Install the CLI tools.
Step 2: Install Playground.
Step 3: Set up your IDE. & Step 4: Install Hyperledger Fabric.

Day 10: Introduction to Finance use cases

Blockchain technology is a decentralized ledger and is a form of distributed ledger technology (DLT). It is a "chain of blocks" where each block holds timestamped digital data and its own along with its previous blocks' unique identity known as a hash. Since its inception, blockchain in financial services has been considered as the technology's primary use case. The technology gained a lot of traction back in 2009 when it was used for the cryptocurrency Bitcoin. The unique features of blockchain have the potential to benefit the finance industry significantly.

The financial services industry is estimated to reach USD 2.6 Trillion by 2022. The global financial system deals with trillions of dollars a day and serves billions of individuals. With such great heights come many challenges that the finance industry has been facing for a long time. Ranging from the high cost of multiple stakeholders to delays, excessive paperwork, and data breaches, these challenges have been the root cause of massive amounts of losses the industry faces every year. As per a PWC report, 45% of the financial intermediaries like stock exchanges, money transfer services, and payment networks face economic crimes every year. Blockchain technology can be a possible solution to the challenges of the global financial system.

CYBER SECURITY USE CASES:

Securing Private Messaging: With the internet shrinking the world into a global village, more and more people are joining social media. The number of social media platforms is also on the rise. More social apps are being launched with each dawn as conversational commerce gains popularity. Huge amounts of metadata are collected during these interactions. Most social media platform users protect the services and their data with weak, unreliable passwords. Most messaging companies are warming up to blockchain for securing user data as a superior option to the end-to-end encryption which they currently use. Blockchain can be used to create a standard security protocol. For enabling cross-messenger communication capabilities, blockchain can be used to form a unified API framework.

CAPITAL MARKET USE CASE:

Despite the fact that the blockchain was initially developed as a freely-accessible, utility-like alternative to traditional means of recording and storing the transfer of assets between counterparties within a distributed, shared network, many fintech start-ups are focused on developing private blockchains in 2015 that can only be accessed by pre-approved participants. Grey Spark believes this trend shows a disconnect between the business imperatives of fintech start-ups to create blockchain solutions that garner widespread uptake and the desire of banks and buyside firms to support a DLT solution designed to service every aspect of the pre- and post-trade lifecycle.

This report defines DLT and the blockchain protocol, and it analyses how banks and exchange platform operators are testing their use within both proprietary environments and on a consortium basis. The report also analyses the potential for a raft of blockchain applications developed by fintech start-ups that could expand the functionality of those solutions within the capital markets industry in the future. These solutions aim to bridge the gap between distributed ledger systems and the world of mainstream financial infrastructure.

Day 11-12: Introduction to JAVASCRIPT and REACT

JAVASCRIPT:

JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else. JavaScript allows us to implement complex features on web pages every time a web page does more than just sit there and display static information for you to look at displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. you can bet that JavaScript is probably involved. It is the third layer of the layer cake of standard web technologies.

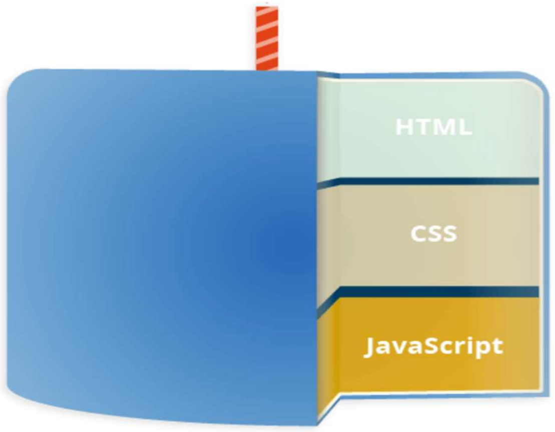


Figure 2.1: Shows JavaScript structure

JAVASCRIPT ES6:

1. **Arrow Functions:**
   - Arrow functions allows a short syntax for writing function expressions.
   - Arrow functions do not have their own this. They are not well suited for defining **object methods**.
   - Arrow functions are not hoisted. They must be defined **before** they are used.
2. **Classes:**
   - JavaScript Classes are templates for JavaScript Objects.
   - They encapsulate data with code to work on that data. Classes in JS are built on prototypes but also have some syntax and semantics that are not shared with ES5 class alike semantics.
3. **Promise**
   - The Promise object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.
   - It allows you to associate handlers with an asynchronous action's eventual success value or failure reason.
   - A Promise is in one of these states:
     i. ***pending***: initial state, neither fulfilled nor rejected.
     ii. ***fulfilled***: meaning that the operation was completed successfully.
     iii. ***rejected***: meaning that the operation failed.

ADVANTAGES OF JAVASCRIPT:

- **Speed -** Client-side JavaScript is very fast because it can be run immediately within the client-side browser. Unless outside resources are required, JavaScript is unhindered by network calls to a backend server.
- **Simplicity -** JavaScript is relatively simple to learn and implement.

- **Interoperability -** JavaScript plays nicely with other languages and can be used in a huge variety of applications.
- **Server Load -** Being client-side reduces the demand on the website server.

Day 13-14: INTRODUCTION TO REACT

REACT:

React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing.

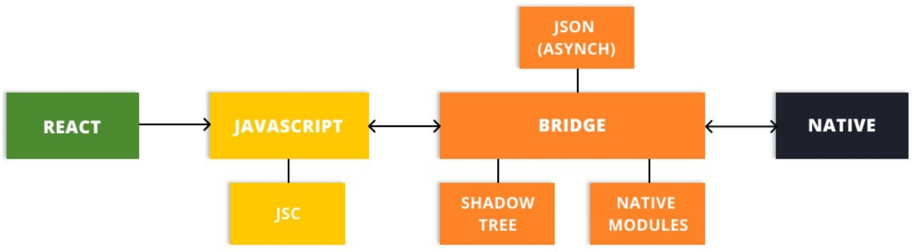


Figure 2.2: Shows the flow chart of React

Components:

React Router is an example of such a library. React code is made of entities called components. Components can be rendered to a particular element in the DOM using the React DOM library. When rendering a component, one can pass in values that are known as "props".

ReactDOM.render(<Greeter greeting="HelloWorld!"/>,
document.getElementById('myReactApp'));

The two primary ways of declaring components in React is via functional components and class-based components.

Functional components:

Functional components are declared with a function that then returns some JSX.

```
const Greeting = (props) => <div>Hello, {props.name}!</div>;
```

Class-based components:

Class-based components are declared using ES6 classes.

```
class ParentComponent extends React.Component {
  state = { color: 'green' };
  render() {
    return (
      <ChildComponent color={this.state.color} />
    );
  }
}
```

Adding React to an HTML Page:

```
<!DOCTYPE html>
<html lang="en">
<title>Test React</title>
<!-- Load React API -->
<script src= "https://unpkg.com/react@16/umd/react.production.min.js"></script>
<!-- Load React DOM-->
<script src= "https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>
<!-- Load Babel Compiler -->
<script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
<body>
<script type="text/babel">
  //  JSX Babel code goes here
</script>
</body>
</html>
```

Advantages of react:

Speed: The React basically allows developers to utilize individual parts of their application on both client-side and the server-side, which ultimately boosts the speed of the development process.

Flexibility: Compared to other frontend frameworks, the React code is easier to maintain and is flexible due to its modular structure. This flexibility, in turn, saves huge amount of time and cost to businesses.

Day 15: Create a User Registration form and integrate API to register user and store User data in-store and show it in the new route.

Most of the time, we talk about logging in and signing up as though it's obvious that everyone will do it. However, asking users to sign up for your product isn't always an easy decision to make. Some people are worried about the friction it causes, or if it's necessary for their product. Sometimes, worries about maintaining secure logins make asking for signups seem like more trouble than not asking users to sign up at all.

FINAL OUTCOME OF WEEK1:

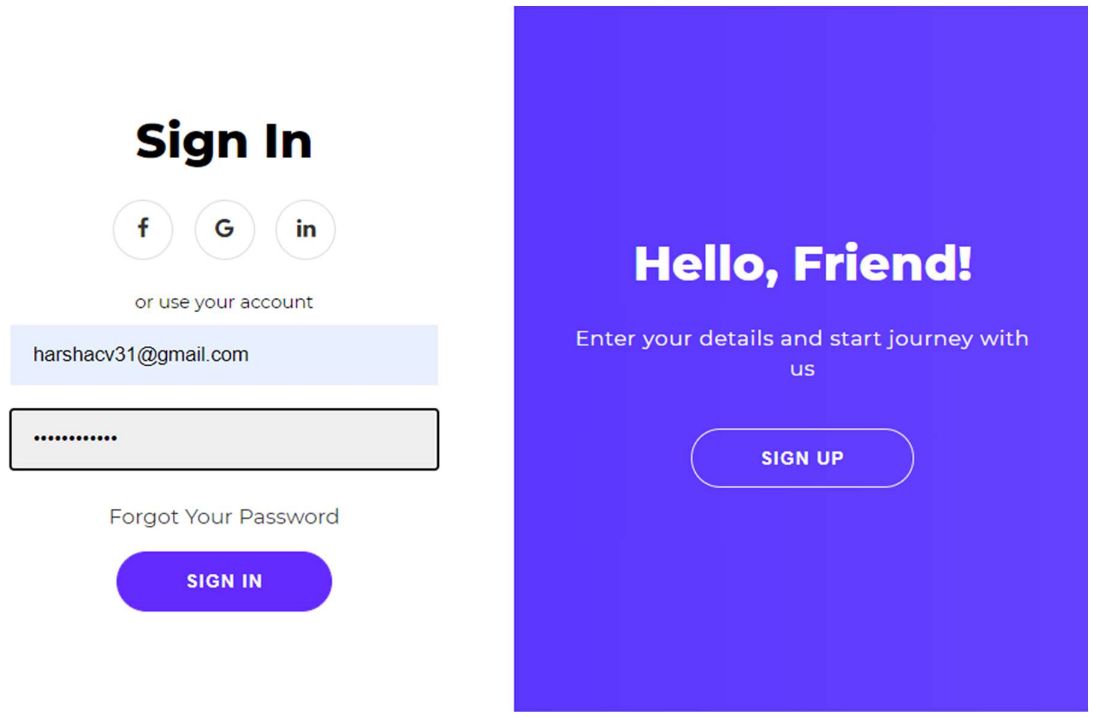


Figure 2.3: shows the Sign-In page

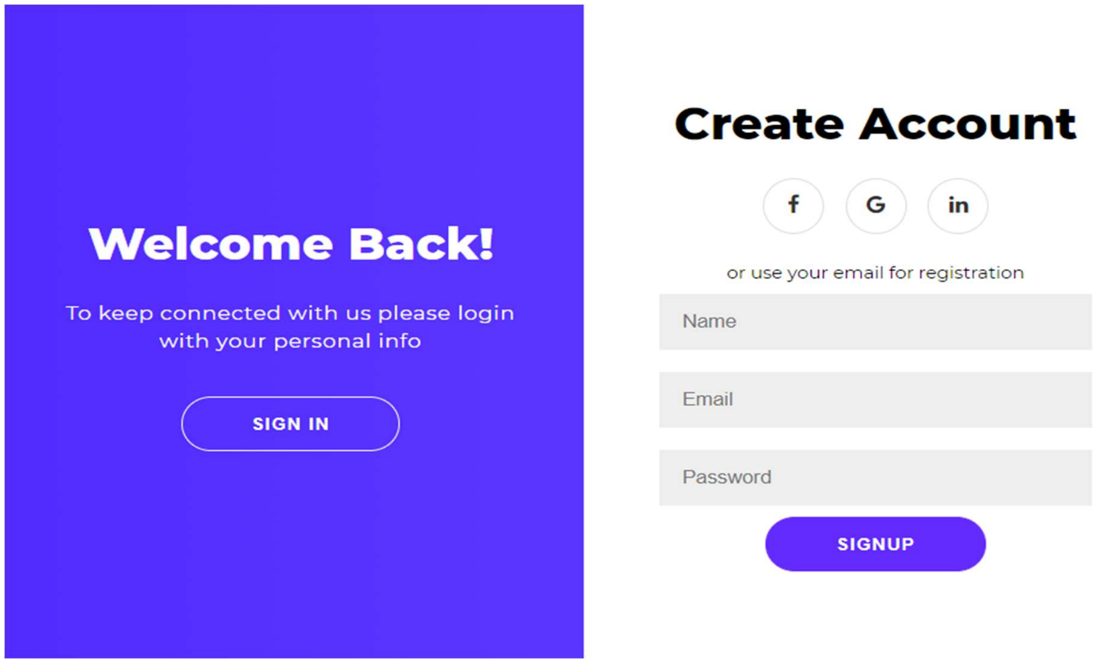
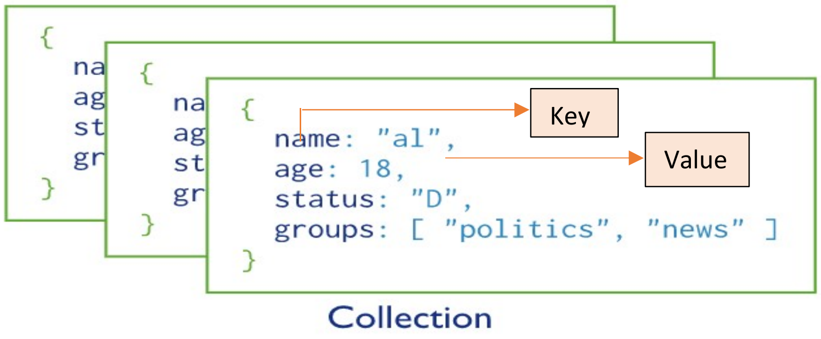

Welcome Back!
To keep connected with us please login with your personal info
SIGN IN
Create Account
f
G
in
or use your email for registration
Name
Email
Password
SIGNUP

Figure 2.4: shows the Sign-Up page

Day 16-17: Introduction to MongoDB

MongoDB:

MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc.



```
{
   name: "al",
   age: 18,
   status: "D",
   groups: [ "politics", "news" ]
}
```

Figure 2.5: Shows the MongoDB Collections

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

The advantages of using documents are:

- Documents (i.e. objects) correspond to native data types in many programming languages.
- Embedded documents and arrays reduce need for expensive joins.
- Dynamic schema supports fluent polymorphism.

Data in MongoDB has a flexible schema, documents in the same collection. They do not need to have the same set of fields or structure Common fields in a collection's documents may hold different types of data.

Editions of MongoDB:

- **MongoDB Community Server**
  The MongoDB Community Edition is free and available for Windows, Linux, and OS X.

- **MongoDB Enterprise Server**
  MongoDB Enterprise Server is the commercial edition of MongoDB, available as part of the MongoDB Enterprise Advanced subscription.

- **MongoDB Atlas**
  MongoDB is also available as an on-demand fully managed service. MongoDB Atlas runs on AWS, Microsoft Azure, and Google Cloud Platform.

Advantages of MongoDB:

1. **High Performance**
   MongoDB provides high performance data persistence. In particular,

- Support for embedded data models reduces I/O activity on database system.
- Indexes support faster queries and can include keys from embedded documents and arrays.

2. **Rich Query Language**

   MongoDB supports a rich query language to support read and write operations (CRUD) as well as:

- Data Aggregation, Text Search and Geospatial Queries.

3. **High Availability**

   MongoDB's replication facility, called replica set, provides:

- Automatic failover and data redundancy.
- A replica set is a group of MongoDB servers that maintain the same data set, providing redundancy and increasing data availability.

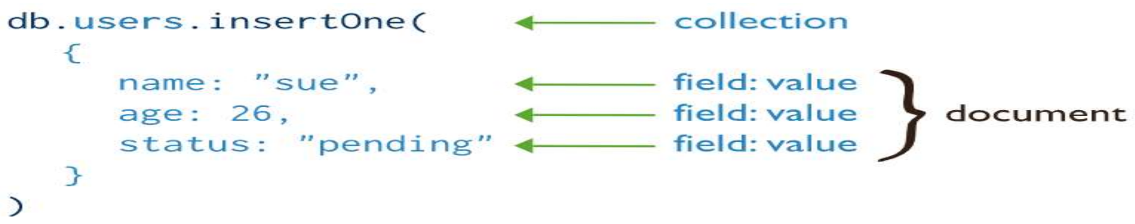
4. **Schema less**

   MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.

5. **Deep query-ability**

   MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.

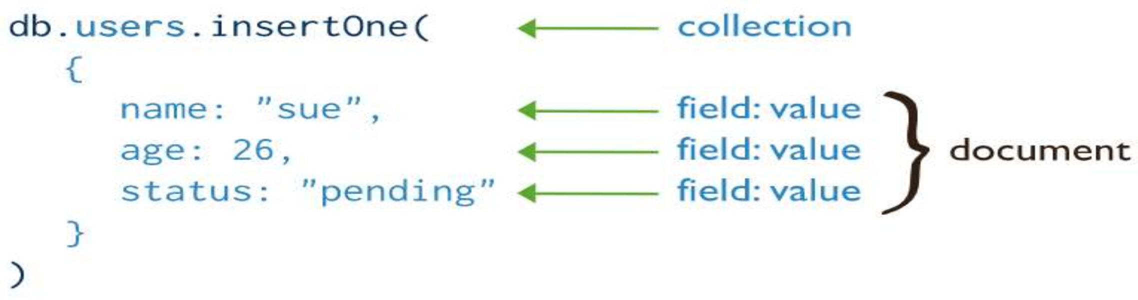
Insert Operations: db.collection.insertOne(), db.collection.insertMany()



```
db.users.insertOne(
   {
      name: "sue",
      age: 26,
      status: "pending"
   }
)
```

Figure 2.6: Shows Insertion operation in MongoDB

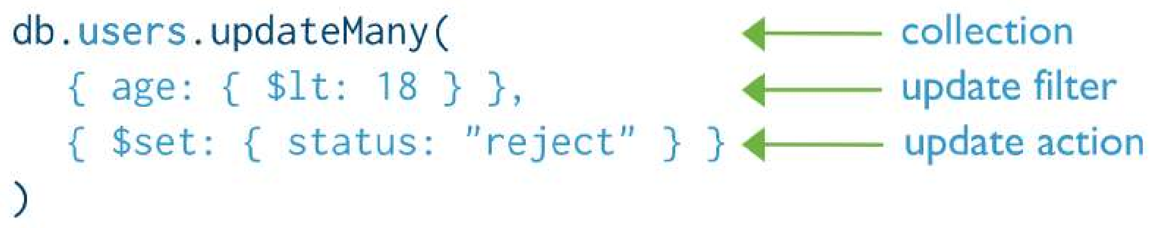
Search Operations: db.collection.find()



```
db.users.insertOne(
   {
      name: "sue",
      age: 26,
      status: "pending"
   }
)
```

Figure 2.7: Shows Search operation in MongoDB

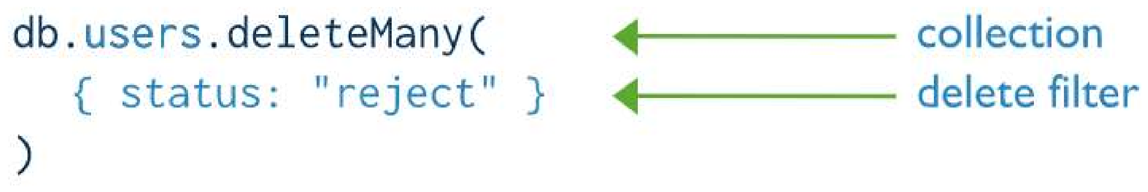
Update Operations: db.collection.updateOne(), db.collection.updateMany(), db.collection.replaceOne()



```
db.users.updateMany(
   { age: { $lt: 18 } },
   { $set: { status: "reject" } }
)
```

Figure 2.8: Shows Update operation in MongoDB

Delete Operations: db.collection.deleteOne(), db.collection.deleteMany()



```
db.users.deleteMany(
   { status: "reject" }
)
```

Figure 2.9: Shows Delete operation in MongoDB

Day 18: Introduction to Node

Node:

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36. Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent. Node.js = Runtime Environment + JavaScript Library Features of Node.js. Following are some of the important features that make Node.js the first choice of software architects. Asynchronous and Event Driven – All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.

Very Fast: Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.

Single Threaded but Highly Scalable: Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly calable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.

No Buffering:
Node.js applications never buffer any data. These applications simply output the data in chunks.

Day 19-20: Create API to store create and update user.

FINAL OUTCOME OF WEEK2:

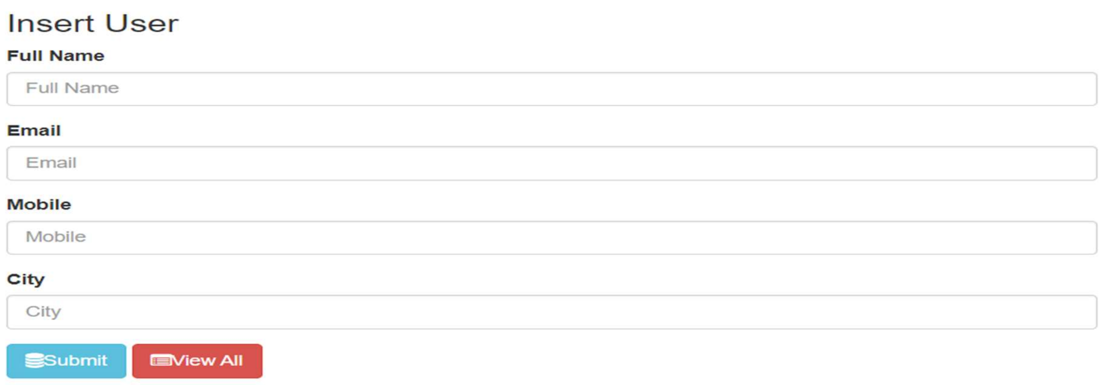


Figure 2.10: Shows the CRUD System

Day 21-22: Introduction to Blockchain

Blockchain refers to a technology that brings in the solution to the age-old human trust problem. It emerged in the market with the renowned cryptocurrency Bitcoin. It provides an architecture that allows us to trust on a decentralized system (Internet or Web) rather than trusting any actor within it. It runs on top of a peer-to-peer network and holds the identical copies of the ledger of transactions. This helps to avoid any middleman and the entire process of transaction takes place through machine consensus. It is a ledger that is shared between multiple entities that everyone can inspect but not any single user can control it. It is a distributed cryptographically secured database that keeps the record of every transaction from the very initial one. let us discuss the introduction to blockchain in detail in this post.

Major Components of Blockchain:

The information in a Blockchain is stored in cryptographically encrypted chunks called blocks. The next successive block contains information about the previous block and hence forms a chain. Thus, the name comes as Blockchain. The privacy of the Blockchain is maintained by high-end cryptographic hash functions and public key cryptography. It also helps to achieve transparency.

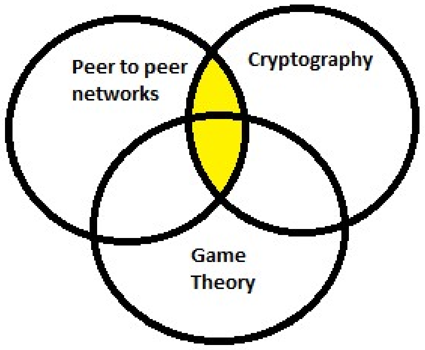


Figure 2.11: Shows the major components of Blockchain

The new transactions are added to the existing information on a consensus of the miners participating in the network. The rules of validating the transactions are coded in form of algorithms and get implemented by the miners who also gets rewarded a native token as per existing economic mechanisms like Proof of Work, Proof of Stake, etc. Here Game Theory comes into play. The ledger runs on a peer-to-peer network and thus all the nodes participating in the network gets a copy of the original information. Here every node in the network is a client as well as server depending on scenarios.

Types of Blockchain:

There are basically two types of blockchain network related to access control.

- Public Blockchain
- Private Blockchain

1. Public Blockchain:

Public Blockchain is accessible to everyone – anyone who wants to read, write can join the blockchain and can perform respective operations. The information once validated in the network cannot be changed and no single entity can have control over the network. Bitcoin is one of the first private blockchain networks to prove the value can be moved anywhere around the world without banks or other third parties.

2. Private Blockchain:

These blockchains work in the same way as public blockchain but with restricted access. The restriction is applied to the users who are authorized to join the network and operate. They may have one or more entities that control the network. Hyperledger is one of the renowned private blockchain networks.

Major Characteristics and Application of Blockchain:

It runs on peer-to-peer network thus, if any of the nodes tampers any information, that will not match the existing copy on the other nodes. As a result, the tampered copy will get discarded as any decision is made on majority basis in the network and here majority will not agree on the tampered copy to be true. Hence, any middleman or broker can be removed by building secured trusted peer to peer network and implementing rules based on a consensus basis. In case of Bitcoin, when sending money from A to B – in place of bank validating the transaction, the same is done by a peer-to-peer network running the validity protocol on a consensus of the majority.

Smart contracts are one of the most used applications where transactions are automatically triggered when certain preconditions are met. This is achieved by decoupling the contract layer from the blockchain layer – here smart contract use ledger only to trigger transactions. It has been introduced by Ethereum blockchain.

Essentially smart contracts are a set of codes that run in a blockchain network and implements a predefined set of rules or contractual agreement which in turn controls the transactions. When all the actors in the contract meet the set of predefined rules the transactions are auto-executed.

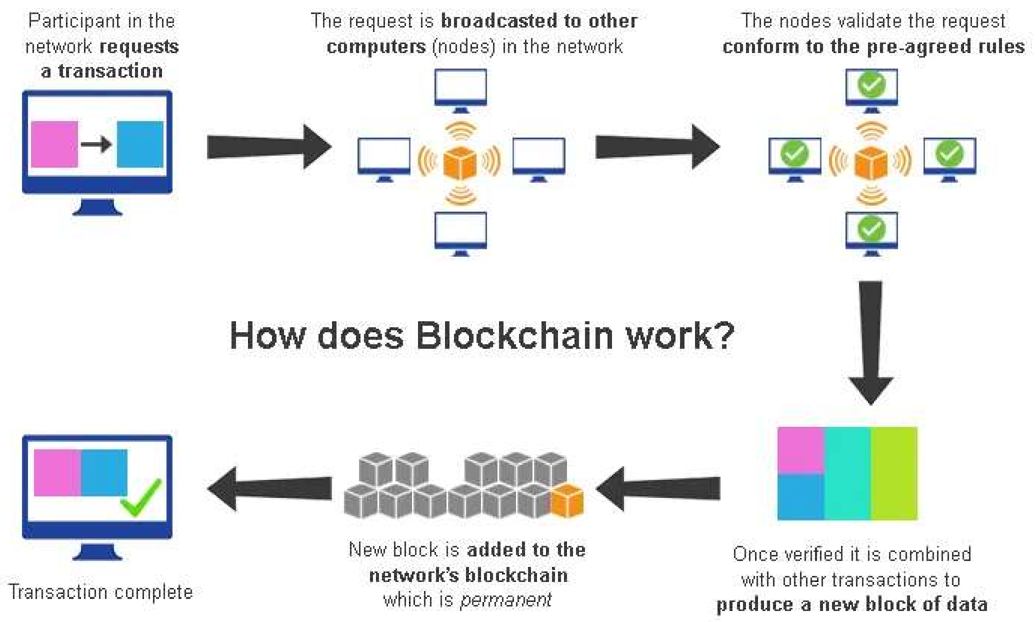

Participant in the network **requests a transaction**
The request is **broadcasted to other computers** (nodes) in the network
The nodes validate the request **conform to the pre-agreed rules**
How does Blockchain work?
Once verified it is combined with other transactions to **produce a new block of data**
New block is **added to the network's blockchain** which is *permanent*
Transaction complete

Figure 2.12: Shows how Blockchain works

Day 23-24: Introduction to Stellar

The stellar network is a blockchain-based distributed ledger network that connects banks, payments systems and people to facilitate low-cost, cross-asset transfers of value, including payments. Stellar has its native cryptocurrency called lumens, which is denoted by the symbol xlm. All three terms stellar, lumens and xlm are used interchangeably, though stellar is blockchain network, lumen is the cryptocurrency and xlm is its trading symbol. Stellar is operated by a non-profit organization called stellar.org and was founded by jed mccaleb, who also co-founded another popular cryptocurrency, ripple. Similar to ripple, stellar is also a cross-border transfer and payment system that connects financial entities with an aim to significantly reduce the transaction costs and time lags. Each transaction has a standard mining fee of 0.00001 lumens.

Stellar's primary focus is on the developing economies in the areas of remittances and bank loans to those who are still outside of the scope of the banking services. It doesn't charge individuals or institutions for using the stellar network. It received initial funding from the payment's startup. Stripe and donations from organizations like blackrock, google.org and fast-forward. It covers the operational costs by accepting tax-deductible public donations, and by using the 5% of lumens set aside initially for the purpose.

Working of Stellar:

While stellar works similar to most decentralized payment technologies like bitcoin, its key distinguishing feature is its consensus protocol. The present-day stellar is a result of a 2014 fork that created the stellar consensus protocol (scp) following which stellar became an open-source system. Under this protocol, the transaction authentication process is confined to a select set of trustworthy nodes rather than being left open to the whole network of nodes. Each node on the network selects a set of such trustworthy nodes, and a transaction is considered approved once authenticated by all nodes that are part of this select group. This shortened approval cycle allows the stellar network to process transactions faster and keep transaction costs lower.

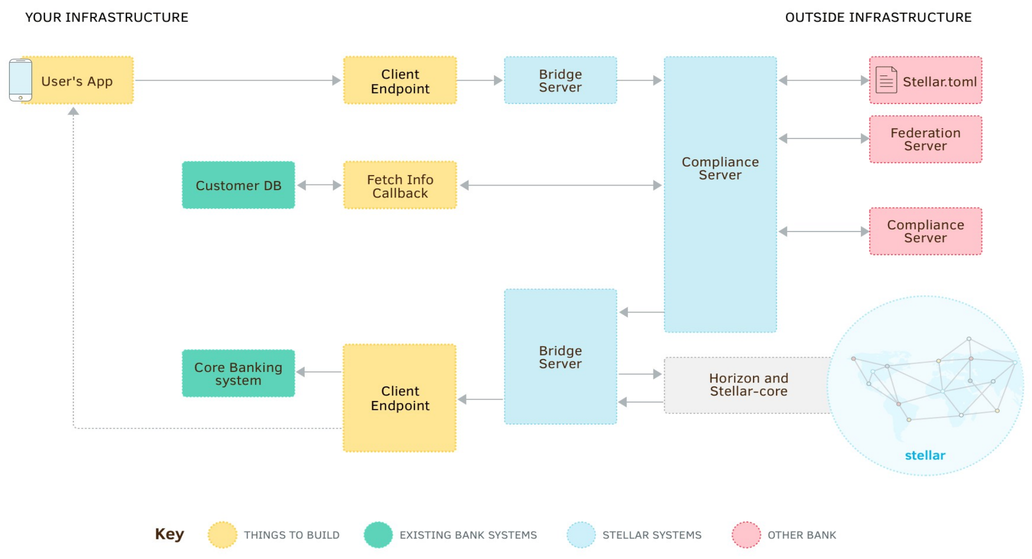


Figure 2.13: Shows the flow chart of Stellar

Stellar supports a distributed exchange model, which allows a user to send payment in a particular currency (like EUR) though they may be holding USD credit. The network automatically performs the forex conversion at the best available rates. The receiver can withdraw the EUR equivalent through the partner institute like a bank.
During 2018, Stellar signed a deal with Transfer To for cross-border payments to more than 70 nations. It also became the first distributed technology ledger to obtain a Shariah-compliance certificate for payments and asset tokenization, and was selected as a partner by International Business Machines Corp. (IBM) for a double-pegged stable coin project.

Day 25-26: Introduction to Front-End Using React:

Environment Setup:

Before we start you should have Node.js with npm installed. Let's begin by creating a new project and installing Network JS by Open Zeppelin to get library methods for working with web3:

```
mkdir web3-infura && cd web3-infura
npm init -y
npm install @openzeppelin/network
```

Creating a Dapp:

Using a common React boilerplate called create-react-app we create the initial version of our dapp. This is easy to setup with npx (npm's package runner). One line and we'll have the scaffolding files for our dapp:
npx create-react-app client
In order to use Infura inside your dapp, you'll need to register to set up an account and create a Project.In the client/src/App.js file, we find the react project and look for the placeholder code written in App.js. We replace the placeholder code with the following:

```
import React from 'react';
import './App.css';
import { useWeb3 } from '@openzeppelin/network/react';
```

```
const infuraProjectId = '<YOUR_INFURA_PROJECT_ID';
function App() {
const web3Context = useWeb3(`wss://mainnet.infura.io/ws/v3/${infuraProjectId}`);
const { networkId, networkName, providerName } = web3Context;
return (
<div className="App">
    <div>
    <h1>Infura/MetaMask/OpenZeppelin Dapp</h1>
        <div>
    Network: {networkId ? `${networkId} – ${networkName}` : 'No connection'}
        </div>
        <div>
        Provider: {providerName}
        </div>
    </div>
</div>
);
}
```

Our dapp will now reveal which Ethereum network (mainnet or a testnet) it is currently connected to, and the web3 provider it's using.Let's test this out by saving and starting the browser by running npm start from within the /client directory. Test it out by replacing the mainnet in the Infura websocket URL to rinkeby. If MetaMask is already installed, disable the extension here as you will see it takes precedence.

How this works is that in the above code we import useWeb3 from the React implementation of Network JS (@openzeppelin/network/react) in order to get a web3Context. This is a Javascript hook that will attempt to retrieve an injected web3 provider which, by default, is MetaMask. If no provider is injected, it will use the Infura URL set as web3Context.   The term injected means code or data that comes from a user's browser and is available for the website to use.

Please note there are a variety of Ethereum web3 libraries available in many languages, and while the implementation specifics differ, they all implement creating a web3 connection through a 'provider' or 'signer' so this step is important to understand.

The useWeb3 hook attempts to obtain an injected web3 provider first, before falling back to a network connection. Alternatively use useWeb3Injected for an injected web3 provider or useWeb3Network for a network provider such as Infura or a private node.

Add a React Component:

Our next goal is to move the display of the current Ethereum network to a component and see how components are re-rendered when changes are made, such as the network.

To do this, the first step is to select a web3 provider to be injected, and install it. We're going to use MetaMask as our web3 provider in this tutorial, so head to metamask.io to install it. In the code below, the Web3Data component expects a web3Context. The networkId, networkName and providerName are obtained from the web3Context and displayed in the component. We then create a components directory in the client/src directory, and create a Web3Data.js file with the code below:

```
import React from 'react';
export default function Web3Data(props) {
const { web3Context } = props;
const { networkId, networkName, providerName } = web3Context;
return (
<div>
    <h3> {props.title} </h3>
    <div>
    Network: {networkId ? `${networkId} – ${networkName}` : 'No connection'
    </div>
    <div>
    Provider: {providerName}
    </div>
</div>
);
}
```

Now back inside App.js, we'll add the Web3Data component to provide a web3Context. In the client/src/App.js file, add this import for the Web3Data component:

import Web3Data from './components/Web3Data.js';

Then replace the entire contents of the App function in App.js with the following code:

```
const web3Context = useWeb3(`wss://mainnet.infura.io/ws/v3/${infuraProjectId}`);
return (
<div className="App">
    <div>
    <h1>Infura React Dapp with Components!</h1>
    <Web3Data title="Web3 Data" web3Context={web3Context} />
    </div>
</div>
);
```

Let's run npm start from within the client directory to start the dapp again to see how React components are re-rendered by Network JS when changes are made to accounts, connection type or network. You can test this out by changing networks in MetaMask.

Day 27-30: IMPLEMENTATION OF STELLAR WALLET

The Stellar Stack:

Fundamentally, Stellar is a collection of Stellar Core nodes, which are computers that keep a common ledger of accounts and balances, listen for incoming transactions, and, using the Stellar Consensus Protocol, agree to apply a valid set of those transactions to update the ledger.

Stellar SDKs:

SDKs make it easy to craft code and handle network queries and transaction submissions. They're linked to in the SDK section of the docs, and each is robust, and has its own documentation showing you how to request data and create and submit transactions. When you start developing on Stellar, the first step is usually to find the SDK in your language of choice and familiarize yourself with how it works.

API: Horizon:

Horizon is a RESTful HTTP API server that provides a straightforward way to submit transactions, check accounts, and subscribe to events. Because it's HTTP, you can communicate with Horizon using an SDK, but you can also use your web browser, or simple command line tools like URL. At the moment, Horizon requires access to Stellar Core's database to function properly. So, every Horizon instance connects to a Stellar Core node but we are increasing its independence from Stellar Core.

Network Backbone: Stellar Core:

The Stellar Core software does the hard work of validating and agreeing with other instances of

Core on the status of every transaction through the Stellar Consensus Protocol (SCP). The ledger, transactions, results, history, and even the messages passed between computers running Stellar Core are encoded using XDR, which is incredibly efficient, but not human readable.

The Public Network and the Test Network:

There are two different versions of the Stellar network: one for testing and one for real-world deployments. The Stellar Development Foundation provides a free public Horizon instance for each, which you can use to submit transactions or query network data. The testnet gets reset every quarter. Additionally, it has a lower ledger limit than the public network: currently, the testnet tops out at 100 operations/ledger; the public network at 1,000 operations/ledger.

CHAPTER 3

REFLECTION

PROJECT UNDERTAKEN: STELLAR WALLET DEVELOPMENT

4.1 Design and Implementation

The step wise development of stellar wallet is shown below:

1. Install SDK based on the technology you want to use:
The first step is to install an SDK based on the technology you want to use for building your app. For example, if you want to develop a Python-based app, you will be installing the Python SDK.

```
| pip install stellar-base
```

Using this utility, the main SDK will be installed along with all dependencies. The Python SDK for Stellar apps provides a networking layer API for Horizon endpoints. It also allows you to build and sign transactions, communicate with a Stellar Horizon instance, query network history and communicate with a Stellar Horizon instance.

2. Connect the SDK with the Horizon Network:
the next step is to connect it with the Horizon network. Horizon is the client-facing API Server for the Stellar blockchain ecosystem. It acts as a bridge between Stellar core and applications that want to access the Stellar network. Horizon enables you to submit transactions to the network, subscribe to event streams and check the accounts' status.

```
| from stellar_base.horizon import Horizondef data_handler(response):
| print(response)
| def get_ledger_data():
| if _name_ == '_main_':
| get_ledger_data()
```

3. Create an account:
Once the SDK gets connected to the Horizon Server, the next step is to create an account on the Stellar Network. Accounts are used for holding all your money inside Stellar and enable you to receive and send payments.

The first step in creating an account is to create your own key and seed. Use the following command to generate the key and seed:

```
|from stellar_sdk.keypair import Keypair
|pair = Keypair.random()

|print(f'Secret: {pair.secret}")
|#Secret:
|SCMDRX7A7OVRPAGXLUVRNIYTWBLCS54OV7UH2TF5URSG4B4JQMUADCYU
|print(f'Public Key: {pair.public_key}")
|#PublicKey:GAG7SXULMNWCW6LX42JKZOZRA2JJXQT23LYY32OXA6XECUQG7RZ
TQJHO
```

4. Creating different assets:

The Stellar Distributed Network is used to hold, track and transfer any kind of asset: euros, dollars, stocks, gold, bitcoin and other tokens of value. Any asset on the Stellar network can be exchanged and traded with any other asset. The issuing asset makes a payment to the distribution account with the newly named asset. As long as the issuing account is unlocked, it can create new tokens by doing payments to the distribution account. If your plan is to do anything with the asset, the next step is to complete a stellar.

5. Define transaction fees:

To maintain the efficiency of the network and ledger spam, Stellar needs small transaction fees and a minimum account balance. Transactions on the Stellar can range from 1 to a defined limit of 100 operations. The fee for a transaction is calculated as:

```
| Transaction fee = number of operations * base fee
```

Stellar deducts the fee from the transaction's source account regardless of which accounts are involved in each operation or who signs the transaction.

6. Perform transactions:

Transactions are the commands used to modify the state of the ledger that include sending payments, making changes to account configurations, creating offers and so on. Each transaction has a source account that pays the fee and uses a sequence number for the transaction.

Transactions comprise one or more operations where each operation has a source account that is default to the transaction’s source account.

```
from stellar_sdk import TransactionBuilder, Network, Keypair, Account root_keypair =
Keypair.from_secret(“SA6XHAH4GNLRWWWF6TEVEWNS44CBNFAJWHWOPZCVZOU
XSQA7BOYN7XHC”)
|# Create an Account object from an address and sequence number.
|root_account = Account(account_id=root_keypair.public_key, sequence=1)
|transaction = TransactionBuilder( source_account=root_account,
|# If you want to submit to pubnet, you need to change `network_passphrase` to `Network.
|PUBLIC_NETWORK_PASSPHRASE`
|network_passphrase=Network.TESTNET_NETWORK_PASSPHRASE, base_fee=100)
|\ .append_payment_op(
|destination=”GASOCNHNNLYFNMDJYQ3XFMI7BYHI
|OCFW3GJEOWRPEGK2TDPGTG2E5EDW”, asset_code=”XLM”, amount=”125.5″)
|\.append_set_options_op( # add a set options operation to the transaction
home_domain=”overcat.me”) \ .set_timeout(30) \ .build()
|# mark this transaction as valid only for the next 30 seconds
```

7. Creating a payment transaction:

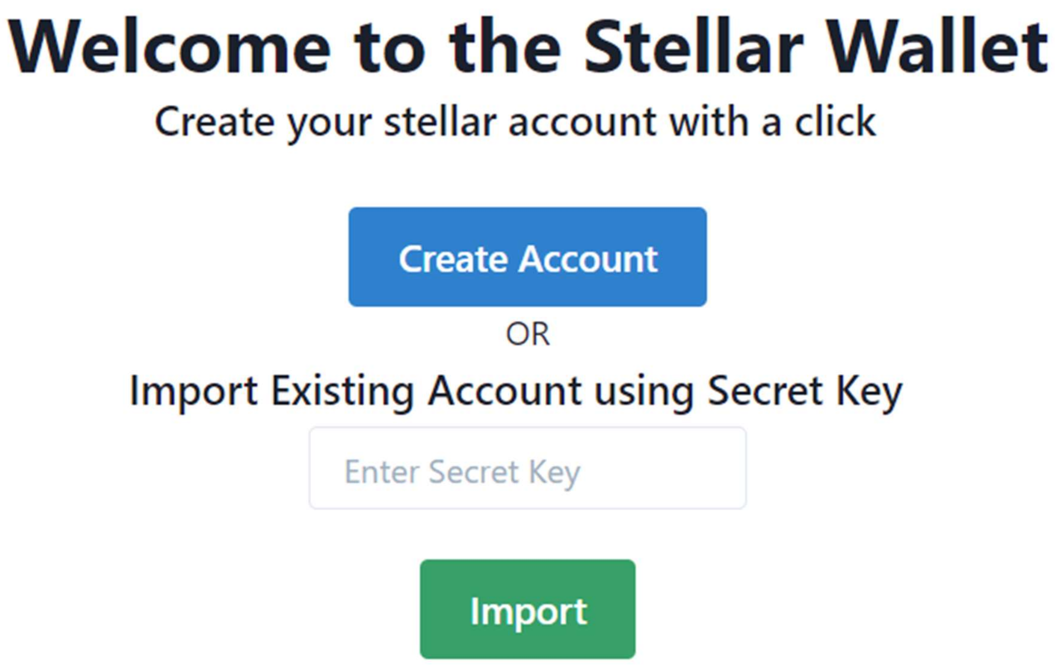
The payment transaction steps are shown below:

1. Secret key of a funded account to be the source account
2. Public key of an existing account as a recipient These two keys can be created and funded by the friend bot at https://www.stellar.org/laboratory/ under the heading “Quick Start: Test Account”
3. Access to Stellar SDK

8. Monitoring:

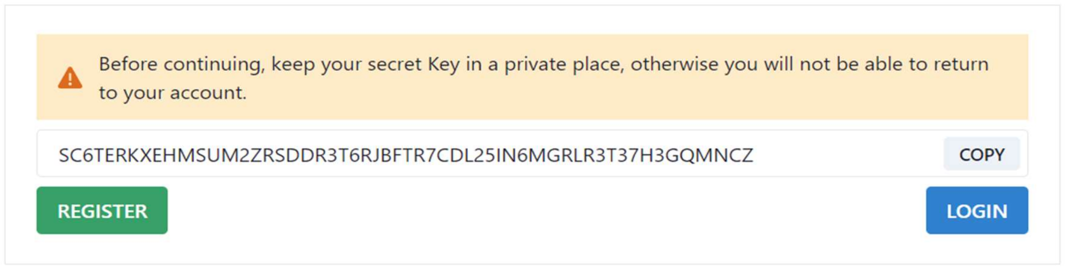
Once the network is set up and the node is up and running, it is essential to keep an eye to ensure the network is running efficiently. To help with monitoring, Stellar Core reveals vital information that you can use to monitor your node and identify potential problems. You have to run $ stellar-core http-command ‘info’ command to check the overall health of the Stellar network.

Final outcome of the System:


Welcome to the Stellar Wallet

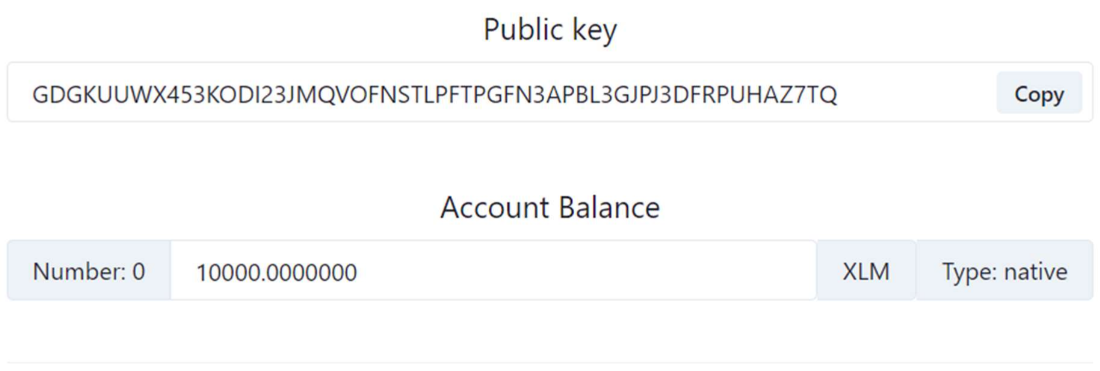
Create your stellar account with a click

Figure 3.1: Shows the Home page of the stellar wallet


Before continuing, keep your secret Key in a private place, otherwise you will not be able to return to your account.

SC6TERKXEHMSUM2ZRSDDR3T6RJBFTR7CDL25IN6MGRLR3T37H3GQMNCZ

Figure 3.2: Shows the Registration page of the stellar wallet


Public key

GDGKUUWX453KODI23JMQVOFNSTLPFTPGFN3APBL3GJPJ3DFRPUHAZ7TQ

Account Balance

Number: 0 | 10000.0000000 | XLM | Type: native

Figure 3.3: Shows the Account Details page of the stellar wallet

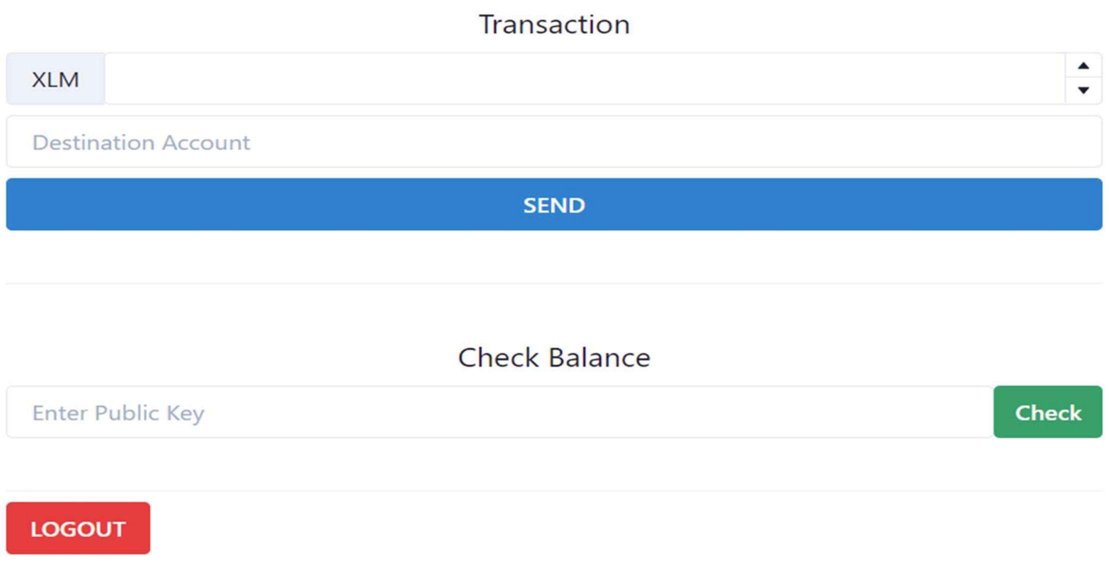


Figure 3.4: Shows the Transaction page of the stellar wallet

CHAPTER 4

CONCLUSION

Working as an intern at leewayHertz Pvt. Ltd. provided us an opportunity to gain a deep insight of the industry. Working and interacting with experienced trainers helped us in learning a lot of new things and provided us a great platform to get into the Blockchain World. During the course of this internship, we were able to understand how the environment will be. Through this internship, we were able to learn about the development of wallets using stellar and how to code for it. Along the training period, we realized that observation is a main element to find out the root cause of a problem. Not only for the project but daily activities too.

Stellar opens the magnitude of new opportunities for entrepreneurs, but you may need technical expertise to build solutions on the Stellar network Though Stellar is not currently experiencing mass adoption, it is one of the most significant open-source projects that you might need to integrate with your business. Stellar also supports SDKs in different languages, including Python, Go, C++, JavaScript and more. Using Stellar, you can develop cost-effective and easy to integrate payment solutions. Stellar opens the magnitude of new opportunities for entrepreneurs, but you may need technical expertise to build solutions on the Stellar network.

After completing the internship, we have been exposed to a programmer working life. During the internship extension, we were able to apply what we had learnt, in a new project developed using the same. The project indirectly helps to learn independently, discipline our self, be considerate/patient, self-trust, and take initiative and the ability to solve problems. In sum total, the activities that we have learned during internship are useful for us in future to face challenges in a working environment. This internship also helped us better our communication skills, work together in a team, learn from experienced trainers and provided us an exposure to corporate world and its work-environment.