COPYRIGHT

Acknowledgment

The completion of this project and report has been possible due to support, guidance and cooperation of multiple helping hands and it would be unfaithful to mention them. We are sincerely grateful to IOE, Pashchimanchal Campus for this course on a major project and all the teachers and staff of Electronics and Computer Engineering for assisting us in the duration of the project with suggestions, and lectures on the context of our project goals.

We are also grateful to our Head of Department and Supervisor, Mr. Hari Prasad Baral for helping us throughout our project and giving us valuable suggestions and clear guidance all the way in this project.

Members of Project

Madhav Poudel (071/BCT/625)
Kiran Tiwari (071/BCT/622)
Nimbus Shrestha (071/BCT/628)
Ravi Tamang (071/BCT/633)

ABSTRACT

Unlike the physical versions of global currencies, cryptocurrencies predominantly exist in the digital world. Since a digital currency does not exist in the physical space, using it in a transaction does not remove it from someone's possession. While bitcoin had the power to make transactions untraceable, it was another innovation that promised to make every transaction transparent and permanent. The underlying technology used in bitcoin is blockchain which possesses the ability to have permanent records of the transactions in the form of blocks and is publicly available for everybody as proof. We have built a decentralized payment system that implements blockchain technology that allows users to send money from one person to another without the requirement of any middleman hence eliminating the need for a middleman in the transaction. Also, the implementation of blockchain technology effectively eliminates the problem of double-spending and other internet frauds.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

BTC:	Bitcoin Currency
DIY:	Do It Yourself
HTTP:	HyperText Transfer Protocol
IBM:	International Business Machines
JSON:	JavaScript Object Notation
P2P:	Peer to peer
RPC:	Remote Procedure Call
NPM:	Node Package Manager
Dapps:	Decentralized Applications

1. INTRODUCTION

1.1. Background

Decentralized Payment System generally refers to the type of payment system where there is no need for any central authority to control the end to end transactions occurring across the network. In the growing world of digital technologies, a decentralized payment system has been proved as a secure, reliable and effective way of transferring funds across the globe.

Our project aims at implementing the concept of a decentralized payment system to eliminate the hindrances caused due to centralized systems during the transaction process. The main purpose of this system has been to develop a decentralized payment system where every user can be in command of their own money and use it when they need to without any microtransactions in between from which others might benefit, allowing for minimum fee deduction.

1.2. Motivation

The Ethereum blockchain is currently the industry standard for issuing custom digital assets and smart contracts. The ERC20 token interface allows for the deployment of a standard token that is compatible with the existing infrastructure of the Ethereum ecosystem, such as development tools, wallets, and exchanges. The ability of Ethereum to deploy Turing-complete trustless smart contracts enables complex issuance rules for cryptocurrencies, digital financial contracts, and automated incentive structures. These advanced features and active ecosystem make Ethereum a natural fit for Sikka.

Sikka is implemented on Ethereum as a Smart Contract and includes an ERC20 compatible utility token, called SIK, which is used for payments.

SIK is a general-purpose cryptocurrency that is:

- Fixed supply
- Fractionally divisible
- Non-inflationary
- Fungible and transferable

Like Bitcoin, Ethereum is a distributed public blockchain network. Although there are some significant technical differences between the two, the most important distinction to note is that Bitcoin and Ethereum differ substantially in purpose and capability. Bitcoin offers one particular application of blockchain technology, a peer to peer electronic cash system that enables online Bitcoin payments. While the Bitcoin blockchain is used to track ownership of digital currency (bitcoins), the Ethereum blockchain focuses on running the programming code of any decentralized application. Using the Ethereum as our technology stack, we built a decentralized cryptocurrency payment system that can be used for general-purpose payment applications.

1.3. Problem Statement

We have till now used the system of centralized banking system where all the transactions take place in the central office i.e. the bank. All the transactions are recorded by them and the deposit and withdrawal of money occur in the bank itself. Due to this, there is often congestion due to too many transactions pending. Also because of this system, international transactions have to go through a number of banks between the sender and the receiver. This can cause a delay in the transaction and any error in one system can cause an error in the following systems as well.

The centralized system also has the problem of transparency since the overall transactions are not visible to the public. One can only view the transactions taken place in their own accounts. Thus each person is unaware of the rest of the transactions taking place simultaneously which can lead to various frauds and scams.

The centralized system also hampers the online payment system as that options are limited only to where the system allows us to. Anywhere beyond that is unavailable to the users even if they have the fund to carry out the transaction. Due to this major hindrance, we have seen e-commerce being unable to scale accordingly in a country like ours. The centralized system doesn’t only exist in the payment system but also in many other systems like insurance, e-commerce, government, online business and so on where there is involvement of middleman.

Centralized organizational structures rely on one individual to make decisions and provide direction for the company. Small businesses often use this structure since the owner is responsible for the company’s business operations. Decentralized organizational structures often have several individuals responsible for making business decisions and running the business.

On the other hand, Decentralized organizations rely on a team environment at different levels in the business. Individuals at each level in the business may have some autonomy to make business decisions. An advantage of decentralization is that there tends to be faster decision making and an ability to adapt to the demographic area of production. It also means that lower-level managers have the opportunity to gain valuable experience and develop more fully because there is more room to grow.

1.4. Objectives

1.4.1. General Objectives

- To provide a highly secure, reliable and efficient method of transaction
- To remove third party involvement
- To provide anonymity and transparency to all the users
- To establish trust between users and system

1.4.2. Specific Objective

To provide a complete decentralized payment system that can be used for general online purposes like paying in an e-commerce store, movie tickets, vehicle fare, hotel booking, etc.

1.5. Scope and Application

Sikka is an open-source cryptographic token that can be used for the general-purpose payment system in a decentralized manner. For Instance, Paying in an e-commerce store, movie tickets, vehicle fare, hotel booking and so on. The possibility is boundless. It is implemented on the public Ethereum blockchain as an ERC20 token. The Ethereum blockchain is currently the industry standard for issuing custom digital assets and smart contracts. The ERC20 token interface allows for the deployment of a standard token that is compatible with the existing infrastructure of the Ethereum ecosystem, such as development tools, wallets, and exchanges. Ethereum's ability to deploy Turing-complete trustless smart contracts enables complex issuance rules for cryptocurrencies, digital financial contracts, and automated incentive structures. These advanced features and active ecosystem make Ethereum a natural fit for Sikka.

2. LITERATURE REVIEW

2.1. Overview

Bitcoin was the first decentralized payment system. It is a decentralized digital currency without a central bank or single administrator that can be sent from user to user on the peer-to-peer bitcoin network without the need for intermediaries. It follows the ideas set out in a white paper by the mysterious Satoshi Nakamoto, whose true identity has yet to be verified. Bitcoin offers the promise of lower transaction fees than traditional online payment mechanisms and is operated by a decentralized authority, unlike government-issued currencies. Today's market cap for all bitcoin (abbreviated BTC or, less frequently, XBT) in circulation exceeds $7 billion. Nowadays, bitcoin is widely used all over the world as a very reliable means of digital currency. The core of bitcoin is 'blockchain' which is a revolutionary algorithm in the field of Information Technology [1].

Ethereum is another similar decentralized platform that runs smart contracts: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference. It allows apps to run on a custom-built blockchain, an enormously powerful shared global infrastructure that can move value around and represent the ownership of property. This enables developers to create markets, store registries of debts or promises, move funds in accordance with instructions given long in the past (like a will or a futures contract) and many other things that have not been invented yet, all without a middle man or counterparty risk. The project was bootstrapped via an ether presale in August 2014 by fans all around the world. It is developed by the Ethereum Foundation, a Swiss nonprofit, with contributions from great minds across the globe. Launched in 2015, Ethereum is the largest and most well-established, open-ended decentralized software platform that enables Smart Contracts and Distributed Applications (Dapps) to be built and run without any downtime,

fraud, control or interference from a third party. Ethereum is not just a platform but also a programming language (Turing complete) running on a blockchain, helping developers to build and publish distributed applications.
The potential applications of Ethereum are wide-ranging and run on its platform-specific cryptographic token, Ether. In 2014, Ethereum had launched a pre-sale for ether which received an overwhelming response. Ether is like a vehicle for moving around on the Ethereum platform and is sought by developers looking to develop and run applications inside Ethereum [2].

In its early days, the Web was obviously not as useful as it is today with the array of apps and services that do everything under the sun, but it did have a more DIY distributed feel to it. The Web was pretty decentralized from the outset. The HTTP protocol connected everyone on the planet with a computing device and an Internet connection. In the HTTP protocol guidelines, there are a set of trusted servers that translate the web address you enter into a server address. Furthermore, HTTPS adds another layer of trusted servers and certificate authorities. People would host personal servers for others to connect to, and everyone owned their data.

Mostly due to its revolutionary properties, cryptocurrencies have become a success their inventor, Satoshi Nakamoto, did not dare to dream of it. While every other attempt to create digital cash, the system did not attract a critical mass of users, Bitcoin had something that provoked enthusiasm and fascination. Sometimes it feels more like religion than technology. Cryptocurrencies are digital gold. Sound money is secure from political influence. Money promises to preserve and increase its value over time. Cryptocurrencies are also a fast and comfortable means of payment with a worldwide scope, and they are private and anonymous enough to serve as a means of payment for black markets and any other outlawed economic activity [3].

One challenge is how to introduce new coins into the system. Obviously, a viable payment network needs some way to create new coins, but if you let anyone create new coins whenever they want, the currency will quickly become worthless. The second challenge is known as the double-spending problem. The rules of bitcoin say that each transaction output can only be spent once. If someone tries to spend the same output twice, the bitcoin community needs some way to detect this double-spending attempt and reject the later transaction. The obvious solution is to have a company manage a shared record of all transactions. That's how conventional payment networks like MasterCard and PayPal work. But bitcoin inventor Satoshi Nakamoto wanted to build a network that wasn't controlled up by any single organization. So Nakamoto invented a shared ledger called the blockchain that is maintained by computers, called nodes, operating on a peer-to-peer network. Thousands of computers around the world keep separate copies of the entire blockchain, storing every transaction that has happened since the network was launched in 2009. The network rewards nodes who help to create the blockchain by allowing them to create new bitcoins—solving the coin-distribution problem while simultaneously creating an incentive to help solve the ledger-updating problem [4].

As the HTTP web grew larger, a new protocol was introduced by a developer named Bram Cohen, called BitTorrent. BitTorrent was a protocol created as a solution to the lengthy time to download huge media files via HTTP and as an improvement on some of the P2P proposals before it, like Gnutella, Napster, and Grokster. The problem was that downloading huge files took a very long time and as the Web grew, so did the size of files that were available. Meanwhile, hard-drive space was increasing and more people were connected. BitTorrent tried to solve this problem by making downloaders into uploaders, as well [5].

If there was a file you wanted, you would download it from not one, but multiple sources. The more popular the file, the more users who would be downloading it and subsequently uploading it, which meant you would be pulling from multiple sources. The more sources, the faster the download. Seeders were rewarded with faster download speeds, whereas leechers were punished with limited speeds. This tit-for-tat system of transferring data proved to be very useful for large media files like movies and TV shows [5].

BitTorrent grew and is for many the de facto way to download any sort of large media file like a game or movie. BitTorrent's speed, resilience, and reward mechanism proved to be better.

So, why doesn't the Web work this way?

Most likely because of HTTP's first mover advantage, its infrastructure, and all of the time and money already invested in it. There are currently active projects working on upgrading the HTTP web with BitTorrent-like technology, and they'll most likely be successful because of BitTorrent's huge value proposition. As soon as BitTorrent was introduced, developers began to use the technology to create nonprofit decentralized applications. Let's look through just a few examples of recent decentralized apps [5].

2.2. Existing Decentralized Applications

There are many decentralized applications built upon the Ethereum platform. Some of the most popular ones are as follows:

2.2.1. Openbazaar

OpenBazaar aims to be a decentralized version of eBay. No middleman can tell sellers what they can and can't sell or decide on the fees for using the service. It's built on the BitTorrent protocol, but the problem is that the sellers must host their own stores. They need to have their own server and leave it on in order for users to be able to see their items. Ideally, sellers could just upload their store data to the network, perhaps paying a small fee, without having to worry about it. This requires a decentralized system of incentivized storage miners. OpenBazaar uses BitTorrent's protocol for data transfer and Bitcoin as currency for transactions between sellers

2.2.2. Storj

Storj is a protocol, cryptocurrency, and suite of decentralized applications that allows users to store data in a secure and decentralized manner. It uses Bitcoin-inspired features like a transaction ledger, public/private key encryption, and cryptographic hash functions for security. Storj nodes, or average computers running the software, sell resources to store and transfer information and earn Storjcoin X (or other cryptocurrencies) in exchange for their services. You, in fact, could run the software and earn some extra cash for leasing your hard drive and bandwidth [7].

2.2.3. Weifund

WeiFund is a toolkit for running crowdfunding campaigns on the Ethereum blockchain. It's particularly useful for launching tokens that can be used in new applications and protocols. You can launch a campaign using one of WeiFund contract templates or integrate your own smart contracts. The WeiFund contracts implement the core mechanics of crowdfunding; contributions with refunds if the campaign objective isn't met. The WeiFund web app makes it easy to start accepting contributions and has hooks to allow you to reuse the user interface with custom contracts, if necessary [8].

2.2.4. Angur

Augur aims to combine the concept of prediction markets with the power of the decentralized network to create a forecasting tool, for potential trading gains. Standing at a market cap above 200 million USD, Augur is currently still under beta test. Eventually, it may be able to feed real-world truths into other applications and establish itself as the blockchain of facts [9].

2.2.5. Melonport

The Melonport protocol is a blockchain protocol for digital asset management. Participants can set up or invest in digital asset management strategies in an open and competitive manner. Using blockchain technology, time and costs are drastically reduced. By building an auditable and visible track record, Melonport enables a never-seen-before competitive environment in asset management [10].

2.2.6. Golem

The Golem project aims to create the first global market for idle computer power. Standing at a remarkable market cap of 220 million USD, Golem will release the first version, Brass Golem, in May. Brass Golem will be tested on its ability to tackle CGI rendering, its first use case. If it turns out to be sustainably successful, CGI artists will be able to rent computing resources from other users to render an image quicker. Likewise, an idle machine can also accept tasks from other users. In light of this, frictionless sharing and pooling of resources may be a reality sooner than we think [11].

3. REQUIREMENT ANALYSIS

3.1. Feasibility Study

Feasibility is the determination of whether or not a project is worth doing. Four tests for a feasibility study for a decentralized payment system are as follows:

3.1.1. Technical Feasibility

It is concerned with specifying equipment and software that will successfully satisfy the use considerably. Ethereum was the selected platform for building our system. Ethereum is an open-source, public, blockchain-based distributed computing platform and operating system featuring smart contract functionality. A smart contract was written in Solidity Programming language and front-end programming was done with javascript. In order to run the wallet software and perform transactions, one must be connected to the web3 network through Ethereum client applications or web browsers. In order to perform transactions in the web browser, an in-browser Ethereum wallet such as Metamask and Parity is used.

3.1.2. Economic Feasibility

Economic analysis is the most frequently used technique used for evaluating the effectiveness of a proposed system. More commonly known as cost/ benefit analysis, the procedure used to determine the benefits and savings that are expected from a proposed system and compared them with a cost. The resulting cost/ benefit ratio of our system is favorable.

In order to deploy a decentralized application, one of the ways is to go through the ICO (Initial Coin Offering) process through crowdfunding for which white paper is to be written and distributed to potential investors. The strong white paper with complete system details and plans makes it economically feasible to initiate a decentralized payment system over the internet.

3.1.3. Operational Feasibility

It is mainly related to human organizational as social aspects. The points to be considered are:

- The system interface is standard, user-friendly and provides ease for operating the application.
- The system runs on a web browser, it is easily accessible and user-friendly.
- Smart Contract is deployed at the beginning of the system use and it cannot be changed later even by the people who operate and runs the system, which builds up trust with the user.
- The system is backed by the Ethereum peer to peer network, the chance of network failure and system down is very rare.

3.1.4. Social Feasibility

Social feasibility is the determination of whether a proposed project will be acceptable to society or not. This project is for the ease for people to implement a decentralized payment system, eliminating the need of the third party for any transaction and does not reflect anything negative to the society, so this project is totally socially feasible. The interface for using the system is similar to the general online payment system despite the underlying technology used. So, there is almost a zero learning curve for users to use the system.

3.2. Functional Requirements

This system is able to perform the following functionalities

- Allow users to send and receive money instantly.
- Eliminate the need of any third party (i.e. bank) for the transaction.
- Remove the need for transaction fees.
- Easy to use client web interface.
- Integrate the system itself into another existing system for payment purpose

3.3. Non-Functional Requirements

3.3.1. User Experience

One particular barrier to blockchain technology is designing the user experience. The challenge here is how much of the underlying technology is to be exposed to the general user base.

3.3.2. Scalability

The platform like ours is tied to a blockchain implementation system like Ethereum and thus the scalability of our system is also tied to the scalability of the implementation system as well.

3.3.3. Stakeholders management

For any system to run well, stakeholders must be involved in the system so that we can conduct business analysis and requirement gathering.

3.3.4. Take away

There should be a journey into learning Blockchain little by little when the consumer wants to, in order to make sure they don't wander blindly into a cryptocurrency system.

3.4. Development Approach

We have implemented the blockchain technology on the Ethereum platform to build a smart contract which is a piece of code that dictates the cryptocurrency system. The smart contract was written on the solidity programming language. ERC20 standard was maintained while making the contract. Test-Driven Development (TDD) approach was used while making smart contracts. Chai.js library was chosen for this process. The system thus far can be used using the command line interface from any operating system.

In order to perform any transaction, the request has to be sent to the blockchain network and be verified. This process is called mining. The mining is done through geth software in our own private network through the JSON RPC server.

Web3.js and truffle.js is used to connect with the backend and frontend. Web3.js helps using smart contracts from solidity in javascript through the JSON RPC server, thus acting as a bridge between our frontend and backend. Truffle converts smart contracts in JSON RPC format.

The frontend design was made using react.js. In order to perform any transaction from this frontend design, any browser must contain web3. Metamask is an extension and easy to use tool that injects a global web3 into browsers. Another alternative is using a mist browser. The web3 contained browser is connected to our private server that provides an initial coin on every account for development purposes. This coin has no real market value. It is important to use services like metamask for development since we cannot afford real coins (i.e. ethers).

Since the value in the cryptocurrency is collected through crowd fundraising, the decentralized crowdfunding contract is also ready to be implemented with the front-end design. This process is also called an initial coin offering (ICO). The potential investors are brought together through the white paper (includes the system operation details and plan in the coming years) over the internet. The deployed system can be integrated into general applications as a payment gateway.

4. METHODOLOGY

4.1. System Design

4.1.1. System Architecture

The Architecture of the system is as follows:

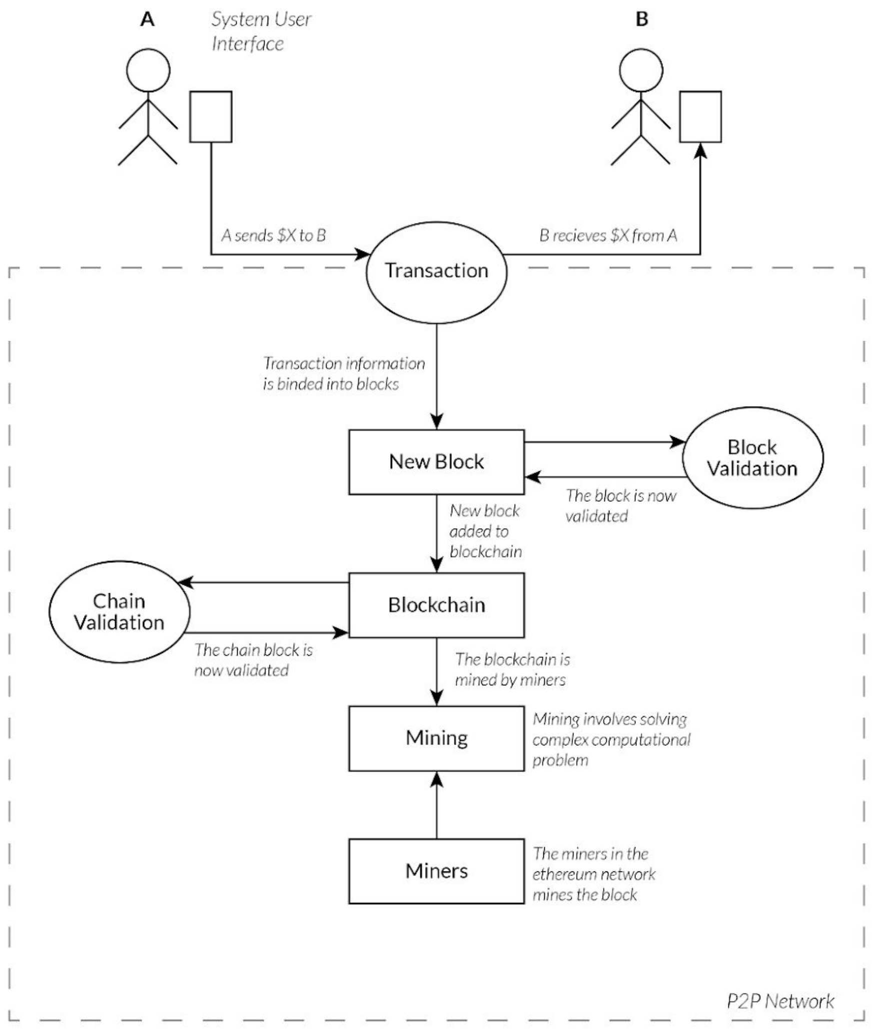

A
System User Interface
B
A sends $X to B
Transaction
B recieves $X from A
Transaction information is binded into blocks
New Block
Block Validation
The block is now validated
New block added to blockchain
Chain Validation
Blockchain
The chain block is now validated
The blockchain is mined by miners
Mining
Mining involves solving complex computational problem
Miners
The miners in the ethereum network mines the block
P2P Network

Fig. 4.1.1: System Architecture

4.1.1.1. User

There are two types of users: senders and receivers. The users use the wallet app to perform various transactions such as sending currency. Since there are no third parties like banks the sending process is very fast and the transaction fee is also minimal.

4.1.1.2. P2P nodes

P2P network is the type of network that doesn't have any central system. Peer-to-peer (P2P) computing or networking is a distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.

In a P2P network, the user utilizes and provides the foundation of the network at the same time, although providing the resources is entirely voluntary. Each peer (a "peer" being a computer system on the network) is considered equal and are commonly referred to as nodes. A peer makes a portion of computing resources such as disk storage, processing power or network bandwidth, directly available to other participants without the need for any central coordination by servers or stable hosts. Despite all nodes being equal, they can take on different roles within the blockchain ecosystem, such as that of a miner or a "full node". In the case of a full node, the whole blockchain is copied onto a single device, while the device is connected to the network. What this means is that the information stored on a blockchain cannot be lost or destroyed because to do so would mean having to destroy every single full node on the network. Therefore, as long as a single node with a copy of a blockchain exists, all the records will remain intact, providing the possibility to rebuild that network.

4.1.1.3. Blockchain

Blockchain is a growing list of records, called *blocks*, which are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data (generally represented as a Merkle tree root hash). By design, a blockchain is resistant to modification of the data. It is an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way. The security is built into the blockchain technology and that rigid compliance has now spawned smart contracts that have almost limitless potential. Smart contracts can replace everything from company payment systems to selling a house. In each contract, the next step is only triggered when conditions are met, so the contract itself manages complex financial transactions. Private blockchains that can be verified by anybody on the system are finally viable with this new type of source code.

The clue to the nature of blockchains is in the name: every block is essentially chained to the previous block, and every block that preceded it – and because the blocks are 'distributed', among several, (or millions) of computers and servers, no one entity can go back and change the history of those blocks (except in an extraordinary hypothetical circumstance involving taking over the majority of the network).

4.1.1.4. Miners

The purpose of mining is probably a little confusing at first. Mining is not about creating a new cryptocurrency. Mining is the mechanism that allows the blockchain to be decentralized security. It secures the bitcoin system and enables a system without a central authority and miners are the actual people behind the mining operations. Mining, in the context of blockchain technology, is the process of adding transactions to the large distributed public ledger of existing transactions, known as the blockchain. Mining is typically done on a dedicated computer, as it requires a fast CPU, as well as higher electricity usage and more heat generated than typical computer operations. The main incentive for mining is that users who choose to use a computer for mining are rewarded for doing so.

As there is no central authority or central bank, there has to be a way of gathering every transaction carried out with a cryptocurrency in order to create a new block. Network nodes that carry out this task called 'miners'. Every time a slew of transactions is amassed into a block, this is appended to the blockchain. Whoever appends the block gets rewarded with some of that cryptocurrency. To prevent the devaluation of the currency by miners building lots of blocks, the task is made harder to conduct. This is achieved by making miners solve complicated mathematical problems called proof of work'.

4.1.2. Activity Diagram

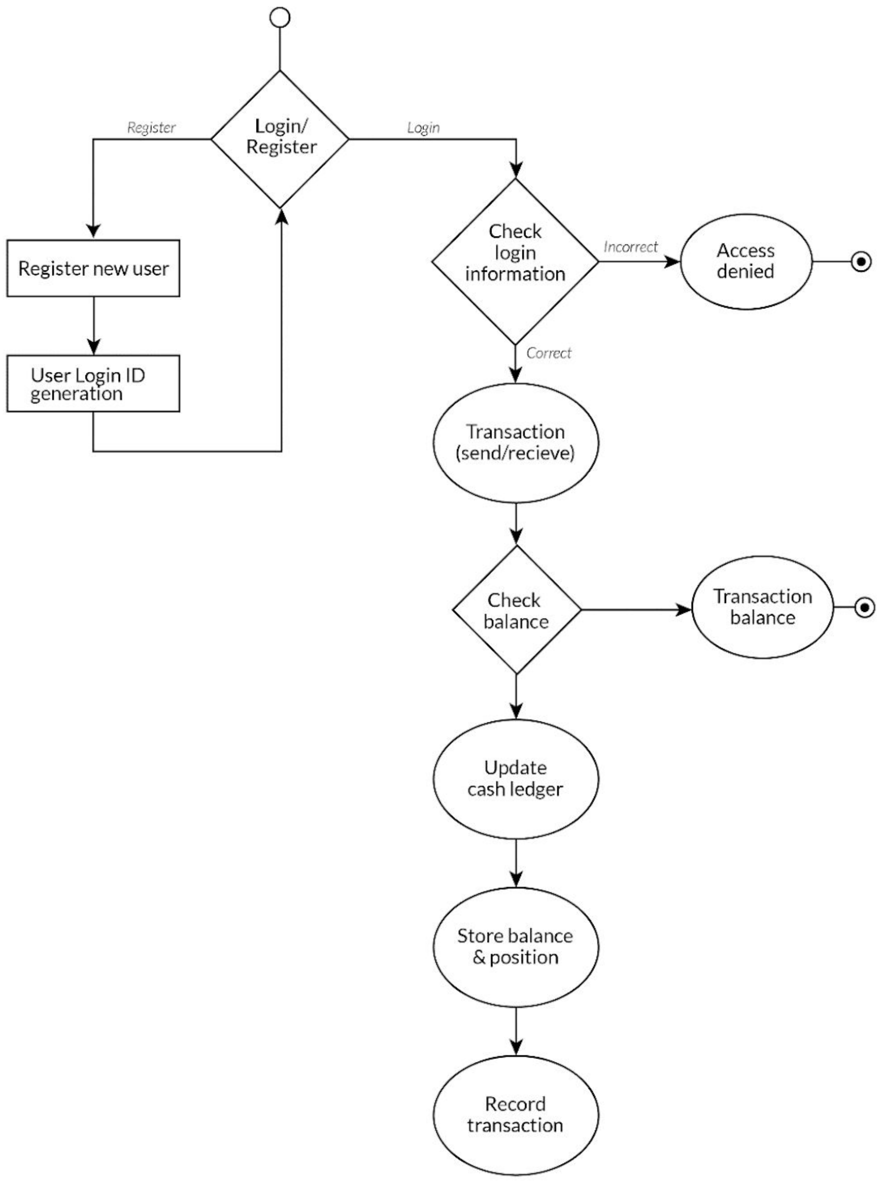


Fig 4.1.2: Activity Diagram

4.2. Tools and Techniques

Following are the tools used while building the system:

4.2.1. NPM

Node.js makes it possible to write applications in JavaScript on the server. It's built on the V8 JavaScript runtime and written in C++ — so it's fast. Originally, it was intended as a server environment for applications, but developers started using it to create tools to aid them in local task automation. Since then, a whole new ecosystem of Node-based tools (such as Grunt, Gulp, and Webpack) has evolved to transform the face of front-end development.

To make use of these tools (or packages) in Node.js we need to be able to install and manage them in a useful way. This is where npm, the Node package manager, comes in. It installs the packages you want to use and provides a useful interface to work with them.

NPM is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It helps in discovering packages of reusable code and assemble them in an effective way [12].

Main Dependencies installed from NPM are:

- React
- React Dom
- React Scripts
- Rsuite
- Truffle
- Web3
- Babel Core
- Eslint
- Chai
- Mocha
- Solidity
- Webpack
- Prettier
- Husky
- React Hot Loader

4.2.2. Truffle

Truffle is a world-class development environment, testing framework and asset pipeline for Ethereum, aiming to make life as an Ethereum developer easier [13].

- Built-in smart contract compilation, linking, deployment and binary management.
- Automated contract testing for rapid development.
- Scriptable, extensible deployment & migrations framework.
- Network management for deploying to any number of public & private networks.
- Package management with EthPM & NPM, using the ERC190 standard.
- Interactive console for direct contract communication.

- Configurable build pipeline with support for tight integration.
- External script runner that executes scripts within a Truffle environment.

4.2.3. Solidity tools

Solidity is a contract-oriented programming language for writing smart contracts. It is used for implementing smart contracts on various blockchain [14].

Remix IDE is used as the primary tool for solidity. Remix is a powerful, open-source tool that helps write Solidity contracts straight from the browser. Written in Javascript, Remix supports both usages in the browser or locally. Remix also supports testing, debugging and deploying smart contracts and much more. Other features include:

- File Explorer
    - Create new File
    - Add Local File
    - Publish to Gist
    - Copy to another Remix instance
    - Connect your filesystem to Remix
- Debugging
- Analysis
- Terminal

Remix IDE is used during the development phase of smart contracts, which allows us to validate the smart contract and input transaction data through a web browser.

4.2.4. Ethereum mist browser

A standard web browser like Chrome, Firefox or Internet Explorer allows users to access websites such as Yahoo, Facebook, and Google. A similar comparison can be made to various mobile apps available through Google Play. Similarly, the Mist browser also allows users to access decentralized apps available on the Ethereum network [15].

However, the word “browser” doesn't fully capture the Mist browser's capabilities. It doesn't only display existing apps, it is a workbench, a collection of decentralized blockchain tools for the Ethereum network. It is a tool specifically designed for non-technical users who can build, copy, utilize, configure and run any existing or new decentralized apps right from the browser. Some tasks that can be performed through the Mist browser include:

- Generating user-selected smart contracts that aim to make business transactions safer and more reliable. By using the on-screen configuration screens, even non-technical users can easily create and run various contracts and Dapps through Mist browser
- Enabling people to pool their money, like in a trustless, decentralized crowdfunding solution, which allows a variety of people to contribute, invest and benefit from new and promising business propositions
- Sharing important information in a reliable manner, which can be contributed by a select group of participants. For instance, a majority of individuals check weather updates on a daily basis, but such key information needs to be contributed only by designated and qualified individuals. The mist browser allows a user to create such a configuration through which a database can be created to allow a public view.

Mist Browser is Ethereum Client Application that can connect to any decentralized applications network. Users can connect to our system network though Mist Browser, While

We have also a web app interface for ease of use which eliminates the learning curve of the mist browser and offers more user-friendly features.

4.2.5. Metamask

MetaMask is a browser plugin that allows users to make Ethereum transactions through regular websites. It facilitates the adoption of Ethereum because it bridges the gap between the user interfaces for Ethereum (e.g. Mist browsers, Dapps) and the regular web (e.g. Chrome, Firefox, websites). Without this, Ethereum cannot become mainstream because the regular web has a strong network effect which inhibits the average user from switching [16].

MetaMask injects a javascript library called web3.js into the namespace of each page your browser loads. web3.js is written by the Ethereum core team and has functions that regular web pages can use to make read and write requests on the blockchain that are consistent with the existing protocol.

Furthermore, MetaMask allows users to specify which Ethereum node to send these requests to. The ability to send requests to nodes outside of the user's computers is important because it means that people can use Ethereum without having to download a node consisting of the 10+GB blockchain on to their computers.

4.2.6. Geth

Geth is a command-line interface for running a full ethereum node implemented in the programming language Go. Gethis the core application on your computer that will connect you to a blockchain. It can also start a new one (in our case we will create a local test blockchain), create a contract, mine ether, etc.

Features:

- Mine real ether
- Transfer funds between addresses
- Create smart contracts
- Send transactions

4.3. Libraries used

4.3.1. Web3

The web3.js library is a collection of modules that contain specific functionality for the Ethereum ecosystem. Ethereum is made up of nodes that all share a copy of the same data. Setting a Web3 Provider in Web3.js tells our code which node we should be talking to handle our reads and writes. It's kind of like setting the URL of the remote web server for your API calls in a traditional web app.

4.3.2. ReactJS

ReactJS basically is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It's used for handling a view layer for web and mobile apps. React also allows us to create reusable UI components. React allows developers to create large web applications that can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple. It works only on user interfaces in the application. This corresponds to the view in the MVC template. It can be used with a combination of other JavaScript libraries or frameworks, such as Angular JS in MVC.

Notable features of ReactJS include:

- One-way data binding with props
- Stateful components
- Virtual DOM
- Lifecycle Methods
- JSX
- Architecture beyond HTML
- Use of the flux architecture

4.3.3. Webpack

Webpack is a module bundler for modern JavaScript applications. When webpack processes an application, it recursively builds a dependency graph that includes every module the application needs, then packages all of those modules into a small number of bundles - often only one - to be loaded by the browser. As we're evolving towards web apps and are relying more and more on JavaScript it becomes mandatory to find ways to organize all that code on the frontend.

Webpack begins its work from entries. Often these are JavaScript modules where webpack begins its traversal process. During this process, the webpack evaluates entry matches against loader configurations that tell webpack how to transform each match.

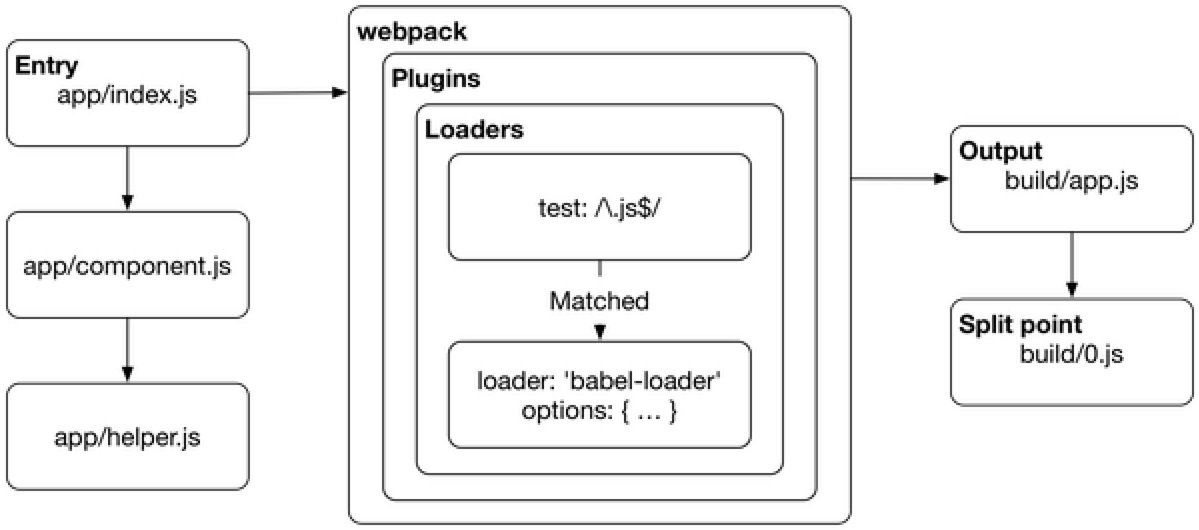


Fig 4.3.3: Webpack's execution process

4.3.4. Mocha and Chai

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting while mapping uncaught exceptions to the correct test cases. Chai is an assertion library for node and the browser that can be delightfully paired with any javascript testing framework.

4.4. Tools and Libraries work flow

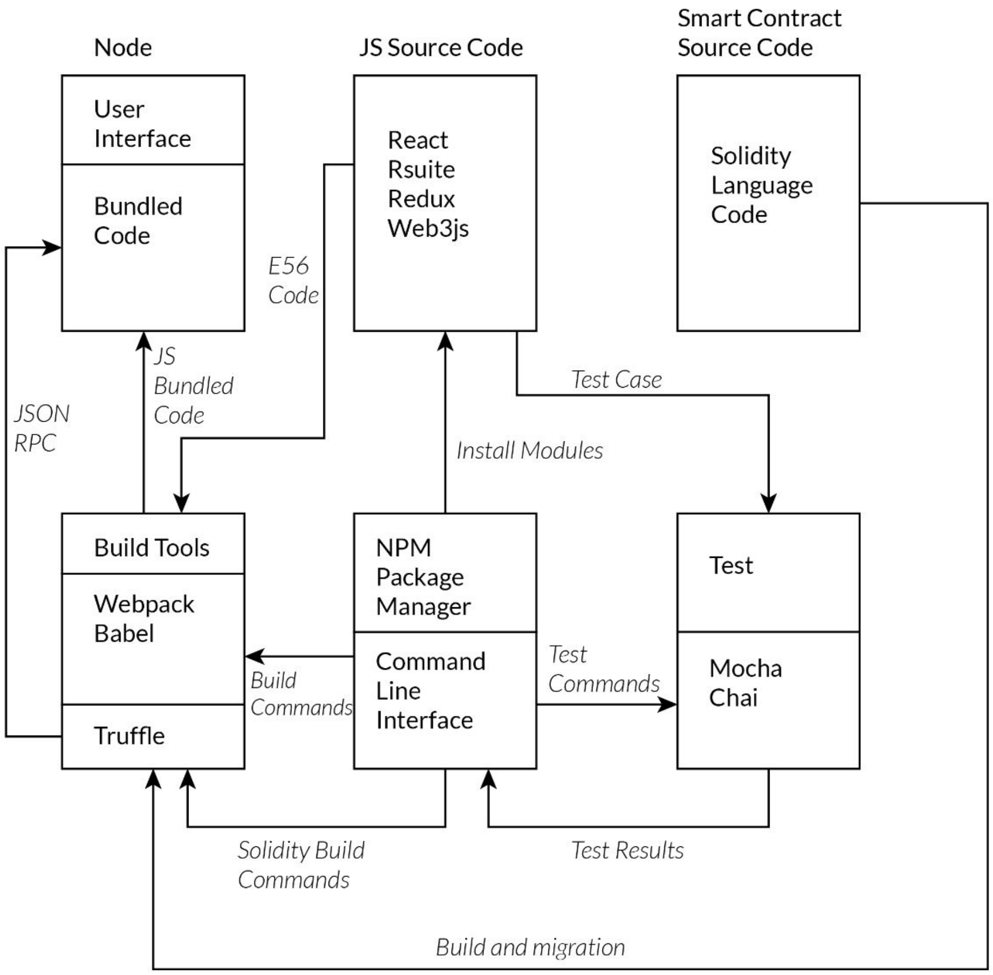


Fig 4.4. How these Tools and Libraries fit together in the System

4.5. Theoretical Background

4.5.1. How blockchain works

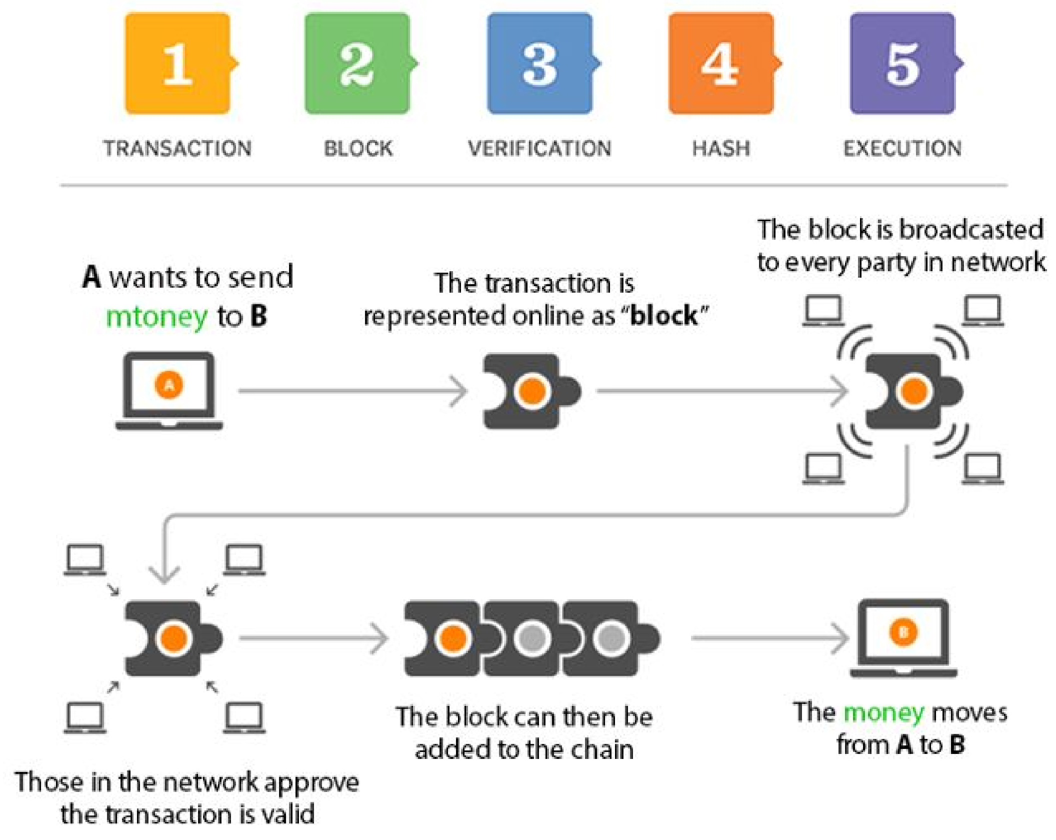

1
TRANSACTION
2
BLOCK
3
VERIFICATION
4
HASH
5
EXECUTION
A wants to send mtoney to B
A
The transaction is represented online as "block"
The block is broadcasted to every party in network
Those in the network approve the transaction is valid
The block can then be added to the chain
B
The money moves from A to B

Fig. 4.4.1.1: How Blockchain Works

When a digital transaction is carried out, it is grouped together in a cryptographically protected block with other transactions that have occurred in the last 10 minutes and sent out to the entire network. A block is an aggregated set of data. Data are collected and processed to fit in a block through a process called mining. Each block could be identified using a cryptographic hash (also known as a digital fingerprint). The block formed will contain a hash of the previous block, so that blocks can form a chain from the first block ever (known as the Genesis Block) to the formed block. In this way, all the data could be connected via a linked list structure. Miners (members in the network with high levels of computing power) than compete to validate the transactions by solving complex coded problems. The first miner to solve the problems and validate the block receives a reward. (In the Bitcoin Blockchain network, for example, a miner would receive Bitcoins). The validated block of transactions is then time-stamped and added to a chain in a linear, chronological order. Each block header has a field for the previous block hash. Hence, all blocks will contain a reference of its previous block, and this could build up a chain of blocks. New blocks of validated transactions are linked to older blocks, making a chain of blocks that show every transaction made in the history of that blockchain. The entire chain is continually updated so that every ledger in the network is the same, giving each member the ability to prove who owns what at any given time.

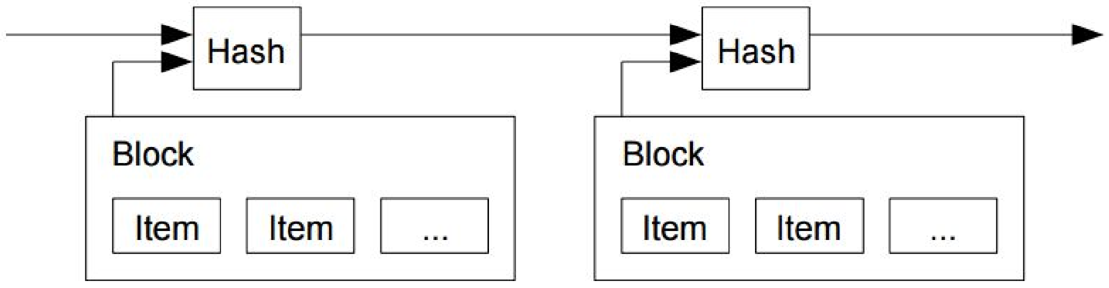


Fig. 4.4.1.2: A visual illustration of blockchain

Blockchain is decentralized, open & cryptographic nature allows people to trust each other and transact peer to peer, making the need for intermediaries obsolete. This also brings unprecedented security benefits. Hacking attacks that commonly impact large centralized intermediaries like banks would be virtually impossible to pull off on the blockchain. For example — if someone wanted to hack into a particular block in a blockchain, a hacker would not only need to hack into that specific block, but all of the proceeding blocks going back the entire history of that blockchain. And they would need to do it on every ledger in the network, which could be millions, simultaneously.

4.5.2. Consensus Mechanisms

Both Bitcoin and Ethereum use a decentralized system to confirm the transactions without relying on a trusted third party. Therefore, consensus, or coming to a uniform agreement, helps a network of autonomous programs and computers come to an agreed state of the blockchain without conflict. As a matter of fact, the consensus is the backbone of the Blockchain and any other decentralized and distributed technology.

The proof of work, proof of stake and closed consensus are the most common mechanisms used in Blockchain technologies.

4.5.3. Proof of work

The most common consensus mechanism that's used for Blockchain technology is what's called "proof of work". It is the system used in Bitcoin.

When a transaction is initiated, the information is stored in a candidate block which is filled with the transaction's information. A cryptographic beacon is sent out to the mining network that the candidate block has been created, and the miners get to work on solving a

cryptographic puzzle that has a prize for whoever solves it, in the form of newly minted coins/currency.
Miners have what some would think of as supercomputers that are much more powerful than the average Person's MacBook pro. These machines have a "hash rate" or computing power that gives them an advantage when competing to solve consensus problems for a reward.

I know what all your climate control advocates are saying: Doesn't that demand a lot of electricity and processing power?

The short answer is yes, the cost of mining is based primarily, on hardware, electricity costs, and to some degree temperature.

The problem with the Proof of work consensus is that it requires the miner to use their supercomputer to try out millions of computations per second, in competition with other supercomputers around the world, to determine if the Blockchain can be updated or not.

4.5.4. Cryptocurrency

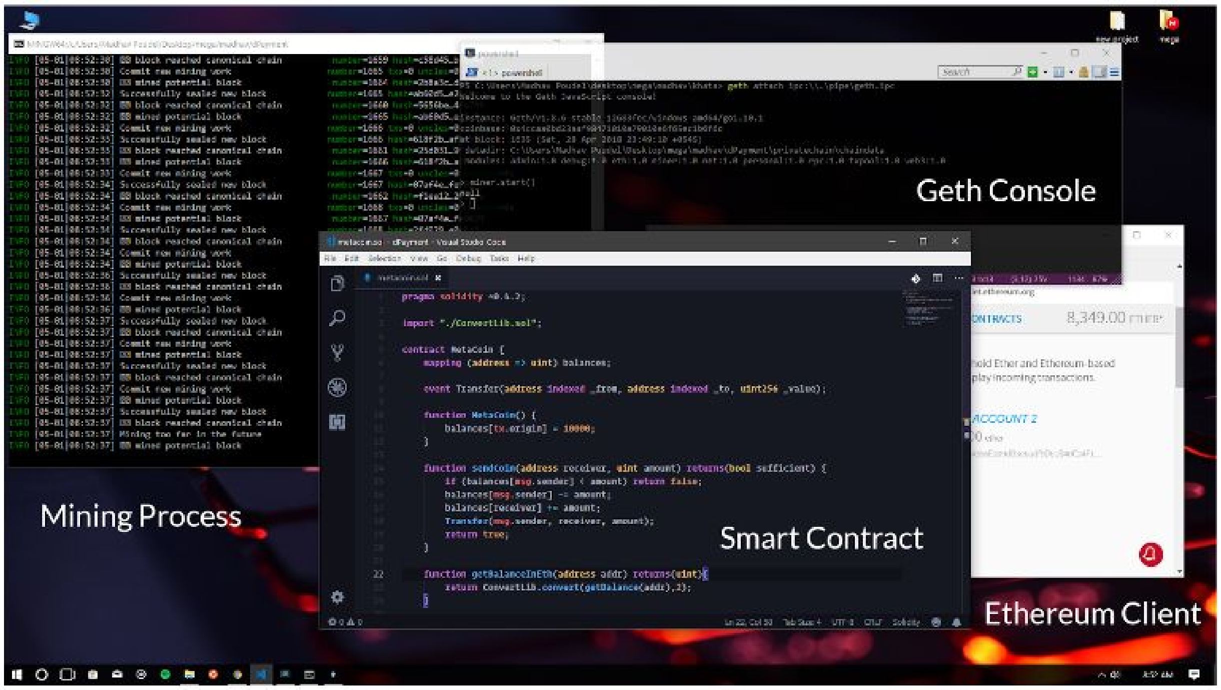
Cryptocurrencies make it easier to transfer funds between two parties in a transaction, these transfers are facilitated through the use of public and private keys for security purposes. These fund transfers are done with minimal processing fees, allowing users to avoid the steep fees charged by most banks and financial institutions for wire transfers. The anonymous nature of cryptocurrency transactions makes them well-suited for a host of nefarious activities, such as money laundering and tax evasion. However, cryptocurrency advocates often value anonymity highly. Cryptocurrencies are also considered by some economists to be a short-lived fad or speculative bubble - concerned especially that the currency units, such as Bitcoins, are not rooted in any material goods. Bitcoin has indeed experienced some rapid surges and collapses in value. Cryptocurrencies are not immune to the threat of hacking. In Bitcoin's short history, the company has been subject to over 40 thefts, including a few that exceeded $1 million in value. Still, many observers look at cryptocurrencies as hope that a currency can exist that preserves value, facilitates exchange, is more transportable than hard metals, and is outside the influence of central banks and governments.

Today cryptocurrencies have become a global phenomenon known to most people. While still somehow geeky and not understood by most people, banks, governments and many companies are aware of its importance. In 2016, you ‘ll have a hard time finding a major bank, a big accounting firm, a prominent software company or a government that did not research cryptocurrencies, publish a paper about it or start a so-called blockchain-project. To realize digital cash, you need a payment network with accounts, balances, and transactions. That’s easy to understand. One major problem every payment network has to solve is to prevent the so-called double spending: to prevent that one entity spends the same amount twice. Usually, this is done by a central server that keeps records about the balances.

5. EPILOGUE

5.1. Output

5.1.1. Main Operations



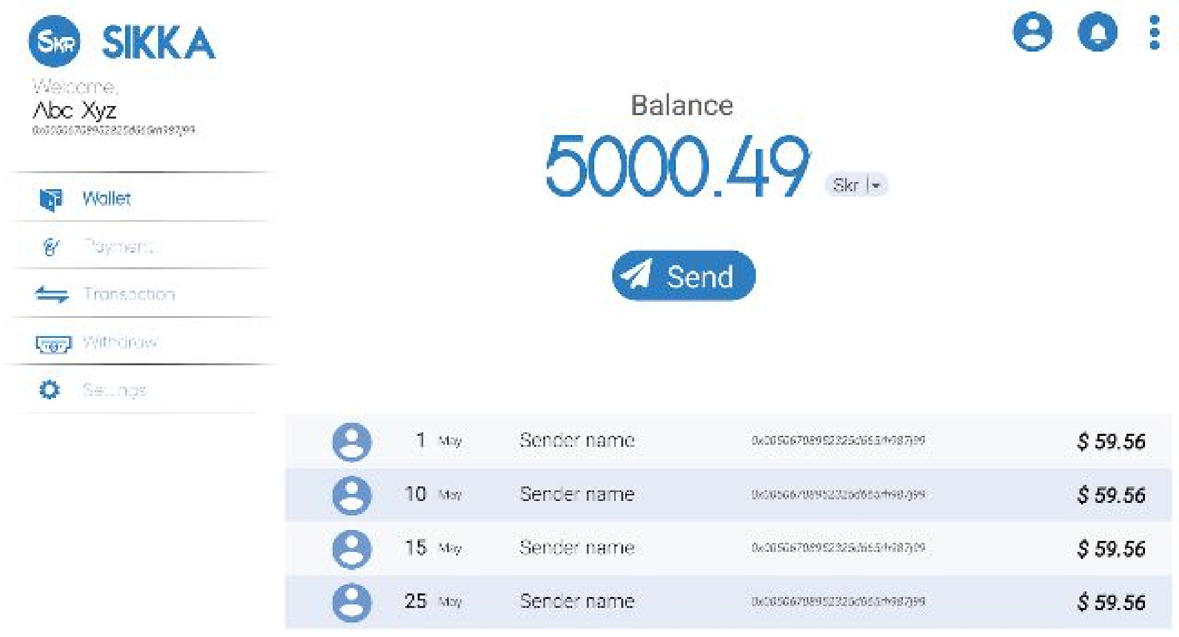
```
INFO [05-01|08:52:30] 🔗 block reached canonical chain
INFO [05-01|08:52:30] Commit new mining work
INFO [05-01|08:52:30] 🔨 mined potential block
INFO [05-01|08:52:32] Successfully sealed new block
INFO [05-01|08:52:32] 🔗 block reached canonical chain
INFO [05-01|08:52:32] 🔨 mined potential block
INFO [05-01|08:52:32] Commit new mining work
INFO [05-01|08:52:33] Successfully sealed new block
INFO [05-01|08:52:33] 🔗 block reached canonical chain
INFO [05-01|08:52:33] 🔨 mined potential block
INFO [05-01|08:52:33] Commit new mining work
INFO [05-01|08:52:34] Successfully sealed new block
INFO [05-01|08:52:34] 🔗 block reached canonical chain
INFO [05-01|08:52:34] Commit new mining work
INFO [05-01|08:52:34] 🔨 mined potential block
INFO [05-01|08:52:34] Successfully sealed new block
INFO [05-01|08:52:34] 🔗 block reached canonical chain
INFO [05-01|08:52:34] Commit new mining work
INFO [05-01|08:52:34] 🔨 mined potential block
INFO [05-01|08:52:36] Successfully sealed new block
INFO [05-01|08:52:36] 🔗 block reached canonical chain
INFO [05-01|08:52:36] Commit new mining work
INFO [05-01|08:52:36] 🔨 mined potential block
INFO [05-01|08:52:37] Successfully sealed new block
INFO [05-01|08:52:37] 🔗 block reached canonical chain
INFO [05-01|08:52:37] Commit new mining work
INFO [05-01|08:52:37] 🔨 mined potential block
INFO [05-01|08:52:37] Successfully sealed new block
INFO [05-01|08:52:37] 🔗 block reached canonical chain
INFO [05-01|08:52:37] Commit new mining work
INFO [05-01|08:52:37] 🔨 mined potential block
INFO [05-01|08:52:37] Successfully sealed new block
INFO [05-01|08:52:37] 🔗 block reached canonical chain
INFO [05-01|08:52:37] Mining too far in the future
INFO [05-01|08:52:37] 🔨 mined potential block
```

```
PS C:\Users\Madhav Poudel\desktop\mega\madhav\khata> geth attach ipc:\\.\pipe\geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.6-stable-12683fec/windows-amd64/go1.10.1
coinbase: 0x1ccaa8bd23ae f90471010a79010e6f65e1b6f6c
at block: 1635 (Sat, 28 Apr 2018 23:49:10 +0545)
 datadir: C:\Users\Madhav Poudel\Desktop\mega\madhav\dPayment\privatechain\chaindata
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> miner.start()
null
> 
```

```
pragma solidity ^0.4.2;

import "./ConvertLib.sol";

contract MetaCoin {
    mapping (address => uint) balances;

    event Transfer(address indexed _from, address indexed _to, uint256 _value);

    function MetaCoin() {
        balances[tx.origin] = 10000;
    }

    function sendCoin(address receiver, uint amount) returns(bool sufficient) {
        if (balances[msg.sender] < amount) return false;
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        Transfer(msg.sender, receiver, amount);
        return true;
    }

    function getBalanceInEth(address addr) returns(uint){
        return ConvertLib.convert(getBalance(addr),2);
    }
```

Fig. 5.1.1.1: Main Operations

5.1.2. Main UI



1	May	Sender name	[illegible]	$ 59.56
10	May	Sender name	[illegible]	$ 59.56
15	May	Sender name	[illegible]	$ 59.56
25	May	Sender name	[illegible]	$ 59.56

Fig. 5.1.2.1: Main System UI

5.2. Limitations and Further Enhancements

Every small and big project have some aspects and area of improvements and so does our project. Here are some limitations in our system observed so far:

- Since the value of cryptocurrency is very high, this system might not be favorable for daily microtransactions.
- Remembering user tokens is a near tough thing to do.
- The resource demand for system running is high.
- Risk of ‘51% attack’.
- Political disagreement.

This system can be enhanced to include these following features:

- The system can be integrated to different e-commerce platforms.
- API could be provided to different trading platforms for their use.
- Different contactless technologies like MasterCard can be integrated with the system for e-shopping and contactless trading applications.
- Banks can adopt cryptocurrency to make their system more secure and reliable.

5.3. Conclusion

Sikka is a decentralized payment system based on blockchain technology. It has an Ethereum based system through which people can conduct transactions of any scale. With blockchain backing it up, all the transactions occurring thought it is secure and fault-proof. Hence the project entitled “Decentralized Payment System” conducted as a major project for partial fulfillment of a degree in Computer Engineering was successfully completed within the given time duration passing through multiple hindrances and difficulties.

6. REFERENCES

[1] John Kelleher, “Bitcoin,” *Investopedia*, December 10, 2017. [Online]. Available: https://www.investopedia.com/terms/b/bitcoin.asp

[2] Prableen Bajpai CFA, “Bitcoin Vs Ethereum: Driven by Different Purpose,” *Investopedia*, November 27, 2017. [Online]. Available:https://www.investopedia.com/articles/investing/031416/bitcoin-vs-ethereum-driven-different-purposes.asp

[3] “What is Cryptocurrency,” *blockgeeks,* September 13, 2018. [Online]. Available: https://blockgeeks.com/guides/what-is-cryptocurrency/

[4] Timothy B. Lee, “Want to Really Understand How Bitcoin Works? Here’s a Gentle Primer,” *Ars Technica*, December 15, 2017. [Online]. Available: https://www.arstechnica.com/tech-policy/2017/12/how-bitcoin-works/.

[5] Sijan Raval, Decentralized Applications: Harnessing Bitcoin's Blockchain Technology, Sebastopol: O’Reilly Media, July 13, 2016

[6]” Openbazaar Seller guide: what to expect in this decentralized marketplace,” *Openbazaar,* October 19, 2017. [Online]. Available:https://openbazaar.org/blog/openbazaar-seller-guide-what-to-expect-in-this-decentralized-marketplace

[7]” What is Storj,” *Storj Blog*, May 29 2014. [Online].
Available: https://storj.io/blog/2014/05/what-is-storj/

[8] “Weifund: crowdfunding campaigns on the Ethereum blockchain with a money back guarantee,” *Futuretechpodcast,* March 20, 2017. [Online].
Available:https://www.futuretechpodcast.com/podcasts/weifund-crowdfunding-campaigns-on-the-ethereum-blockchain-with-a-money-back-guarantee/

[9] “Augur | A Decentralized Oracle & Prediction Market Protocol,” *Augur.* [Online].
Available: https://www.augur.net/

[10] “Melonport - Asset Management Computer,” *Melonport.* [Online].
Available: https://melonport.com/

[11] “Golem: home,” *Golem.* [Online].
Available: https://golem.network/

[12] *NPM.* [Online]. Available: https://www.npmjs.com/

[13] *truffle*. [Online]. Available: https://truffleframework.com/docs

[14] *Remix Documentation*. [Online]. Available: https://remix.readthedocs.io/en/latest/

[15] “Mist browser.” *Investopedia.* [Online]. Available:
https://www.investopedia.com/terms/m/mist-browser.asp

[16] *Metamask*. [Online]. Available: https://metamask.io/

[17] “Ethereum wiki,” *GitHub*. [Online]. Available:
https://github.com/ethereum/wiki/wiki/JavaScript-API

Ethereum Contract source code

```
pragma solidity ^0.4.18;

// ----------------------------------------------------------------------------
// 'Sikka' token contract

// Deployed to      : 0x7B96f0cafD4ef8ba4f0344C138afcC84bd1ED345
// Symbol           : SIKKA
// Name             : Sikka Token
// Total supply     : 100000000
// Decimals         : 18

// Decentralize Payment System - BCT 071, WRC, IOE
// ----------------------------------------------------------------------------


// ----------------------------------------------------------------------------
// Safe maths
// ----------------------------------------------------------------------------
contract SafeMath {
    function safeAdd(uint a, uint b) public pure returns (uint c) {
        c = a + b;
        require(c >= a);
    }
    function safeSub(uint a, uint b) public pure returns (uint c) {
        require(b <= a);
        c = a - b;
    }
    function safeMul(uint a, uint b) public pure returns (uint c) {
        c = a * b;
        require(a == 0 || c / a == b);
    }
    function safeDiv(uint a, uint b) public pure returns (uint c) {
        require(b > 0);
        c = a / b;
    }
}

// ----------------------------------------------------------------------------
// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
// ----------------------------------------------------------------------------

contract ERC20Interface {
    function totalSupply() public constant returns (uint);
    function balanceOf(address tokenOwner) public constant returns (uint balance);
    function allowance(address tokenOwner, address spender) public constant returns (uint
remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool success);
    function transferFrom(address from, address to, uint tokens) public returns (bool
success);

    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}
// ----------------------------------------------------------------------------
// Contract function to receive approval and execute function in one call
// Borrowed from MiniMeToken
// ----------------------------------------------------------------------------
contract ApproveAndCallFallBack {
    function receiveApproval(address from, uint256 tokens, address token, bytes data)
public;
}


// ----------------------------------------------------------------------------
// Owned contract
// ----------------------------------------------------------------------------
contract Owned {
    address public owner;
    address public newOwner;

    event OwnershipTransferred(address indexed _from, address indexed _to);

    function Owned() public {
        owner = msg.sender;
    }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }


    function transferOwnership(address _newOwner) public onlyOwner {
        newOwner = _newOwner;
    }




    function acceptOwnership() public {
        require(msg.sender == newOwner);
        OwnershipTransferred(owner, newOwner);
        owner = newOwner;
        newOwner = address(0);
    }
}
```

```
// ----------------------------------------------------------------------------
// ERC20 Token, with the addition of symbol, name and decimals and assisted
// token transfers
// ----------------------------------------------------------------------------

contract SikkaToken is ERC20Interface, Owned, SafeMath {
    string public symbol;
    string public  name;
    uint8 public decimals;
    uint public _totalSupply;

    mapping(address => uint) balances;
    mapping(address => mapping(address => uint)) allowed;


    // ------------------------------------------------------------------------
    // Constructor
    // ------------------------------------------------------------------------
    function SikkaToken() public {
        symbol = "SIKKA";
        name = "Sikka Token";
        decimals = 18;
        _totalSupply = 100000000000000000000000000;
        balances[0x5A86f0cafD4ef3ba4f0344C138afcC84bd1ED222] = _totalSupply;
        Transfer(address(0), 0x5A86f0cafD4ef3ba4f0344C138afcC84bd1ED222, _totalSupply);
    }


    // ------------------------------------------------------------------------
    // Total supply
    // ------------------------------------------------------------------------
    function totalSupply() public constant returns (uint) {
        return _totalSupply  - balances[address(0)];
    }


    // ------------------------------------------------------------------------
    // Get the token balance for account tokenOwner
    // ------------------------------------------------------------------------
    function balanceOf(address tokenOwner) public constant returns (uint balance) {
        return balances[tokenOwner];
    }


    // ------------------------------------------------------------------------
    // Transfer the balance from token owner's account to to account
    // - Owner's account must have sufficient balance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------
    function transfer(address to, uint tokens) public returns (bool success) {
        balances[msg.sender] = safeSub(balances[msg.sender], tokens);
        balances[to] = safeAdd(balances[to], tokens);
        Transfer(msg.sender, to, tokens);
        return true;
    }
    // ------------------------------------------------------------------------
    // Token owner can approve for spender to transferFrom(...) tokens
    // from the token owner's account
    //
    // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
    // recommends that there are no checks for the approval double-spend attack
    // as this should be implemented in user interfaces
    // ------------------------------------------------------------------------
    function approve(address spender, uint tokens) public returns (bool success) {
        allowed[msg.sender][spender] = tokens;
        Approval(msg.sender, spender, tokens);
        return true;
    }


    // ------------------------------------------------------------------------
    // Transfer tokens from the from account to the to account
    //
    // The calling account must already have sufficient tokens approve(...)-d
    // for spending from the from account and
    // - From account must have sufficient balance to transfer
    // - Spender must have sufficient allowance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------

    function transferFrom(address from, address to, uint tokens) public returns (bool
success) {
        balances[from] = safeSub(balances[from], tokens);
        allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
        balances[to] = safeAdd(balances[to], tokens);
        Transfer(from, to, tokens);
        return true;
    }


    // ------------------------------------------------------------------------
    // Returns the amount of tokens approved by the owner that can be
    // transferred to the spender's account
    // ------------------------------------------------------------------------
    function allowance(address tokenOwner, address spender) public constant returns (uint
remaining) {
        return allowed[tokenOwner][spender];
    }


    // ------------------------------------------------------------------------
    // Token owner can approve for spender to transferFrom(...) tokens
    // from the token owner's account. The spender contract function
    // receiveApproval(...) is then executed
    // ------------------------------------------------------------------------
    function approveAndCall(address spender, uint tokens, bytes data) public returns (bool
success) {
        allowed[msg.sender][spender] = tokens;
        Approval(msg.sender, spender, tokens);
        ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, this, data);
        return true;
    }
    // ----------------------------------------------------------------------------
    // Don't accept ETH
    // ----------------------------------------------------------------------------
    function () public payable {
        revert();
    }


    // ----------------------------------------------------------------------------
    // Owner can transfer out any accidentally sent ERC20 tokens
    // ----------------------------------------------------------------------------
    function transferAnyERC20Token(address tokenAddress, uint tokens) public onlyOwner
returns (bool success) {
        return ERC20Interface(tokenAddress).transfer(owner, tokens);
    }
}
```