

BUILDING DECENTRALIZED APPLICATIONS ON THE INTERNET COMPUTER AN OVERVIEW AND IMPLEMENTATION GUIDE

S VENKATA MOHITH REDDY¹
YANAMALA SAI KEERTHANA²
MANYAM KIRAN KUMAR REDDY³
RASINENI HAREESH⁴
Computer Science Engineering^{1,2}
Electronics and Communications Engineering^{3,4}
Sri Venkateswara College of Engineering
and Technology

A B MANJU⁵
Assistant Professor
School of Technology
The Apollo University

ABSTRACT

Internet computing is a decentralized computing platform intended to enhance the capability of the web beyond traditional client-server architectures. The Internet computer enables developers to develop and share decentralized applications (dApps) by using a public blockchain that is available to everyone with an Internet connection. The process of establishing a dApp on an internet computer is presented. The installation of the Canister SDK, the development and deployment of the dApp on a local server, the purchase of cycles, and the public distribution of the dApp on the blockchain are explored and presented. The document also provides a command-line guide for using dfx, a versatile tool included in DFINITY that can be used to build and maintain dApps on the Internet computer platform.

Keywords—Blockchain; Internet Computer; Decentralized Applications; Canister SDK; Motoko programming language

I. INTRODUCTION

Blockchain technology has transformed the way we view and use digital currencies. One of the most exciting new entrants in the blockchain space is the DFINITY blockchain. DFINITY is a decentralized computing network that aims to revolutionize the way we build and run applications. The network uses a unique consensus mechanism called "Threshold Relay" that ensures fast finality and high security. DFINITY is built on the concept of "Internet Computer" which allows developers to build and deploy decentralized applications without relying on centralized infrastructure. The network is powered by a revolutionary new programming language called "Motoko," which simplifies the development of smart contracts and decentralized applications. In addition, DFINITY uses a unique canister-based architecture that allows developers to deploy their code as standalone units that can interact with other canisters on the network. One of the most important features of DFINITY is its consensus mechanism, the "Threshold Relay." This consensus mechanism allows DFINITY to achieve fast finality and high security without sacrificing scalability. Unlike other consensus mechanisms like proof-of-work and proof-of-stake, threshold relay does not rely on a fixed set of validators. Instead, validators are selected randomly from a large pool of potential validators. This ensures that the network is not vulnerable to attacks from a small group of validators. Another

important aspect of DFINITY is its focus on building a truly decentralized network. Unlike other blockchain networks that rely on a small group of node operators to secure the network, DFINITY is designed to be fully decentralized. This means that anyone can run a node and participate in the consensus process. In addition, DFINITY has a unique governance model that allows token holders to vote on network upgrades and changes.

The DFINITY blockchain is an exciting new entrant in the blockchain space that has the potential to revolutionize the way we build and run decentralized applications. The network's unique consensus mechanism, canister-based architecture, and focus on decentralization make it a compelling option for developers looking to build on a truly decentralized network. As the network continues to grow and mature, it will be interesting to see how it stacks up against other blockchain networks and whether it can achieve its goal of becoming the "Internet Computer" of the future.

II. LITERATURE

Blockchain is a distributed ledger technology that has gained a lot of attention due to its ability to ensure transparency and reduce fraud in various sectors. It stores transactions in a secure and transparent manner, making it difficult to manipulate or corrupt. Immutability, transparency, and decentralization features of blockchain have made it a promising technology for reducing fraud and enhancing transparency in various industries.

A. Blockchain for Transparency and Fraud Reduction

Blockchain technology has the potential to significantly reduce fraud and enhance transparency in various industries. In the financial industry, blockchain technology can reduce fraud by providing a tamper-proof record of all financial transactions. This can reduce the need for intermediaries, such as banks, and enable faster and more secure transactions. Several studies have shown that blockchain technology can improve financial systems' efficiency, security, and transparency (Kshetri, 2018 [13]; Crosby et al., 2016 [12]).

In the healthcare industry, blockchain technology can ensure the transparency and security of patient data. Blockchain can ensure that patient data is secure and can only be accessed by authorized personnel. Blockchain can also ensure the accuracy of medical records by providing an immutable record of all medical transactions. Several studies have shown that blockchain technology can improve healthcare systems' efficiency, security, and transparency (Pilkington, 2016 [26]; Radanovic et al., 2018 [27]).

Blockchain technology in the supply chain industry ensures all transactions in the supply chain are transparent and secure, reducing the possibility of fraud. This can help to ensure that products are authentic and of high quality. Several studies have shown that blockchain technology can improve the efficiency, security, and transparency of supply chain systems (Zheng et al., 2017 [14]; Xu et al., 2018 [11]).

Smart contracts, which are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code, can further enhance transparency and reduce fraud using blockchain. Smart contracts can help to ensure that all parties involved in a transaction fulfill their obligations, reducing the possibility of fraud. Several studies have shown that smart contracts can improve various blockchain systems' efficiency, security, and transparency (Buterin, 2014 [23]; Domingo-Ferrer et al., 2018 [24]).

The use of blockchain technology has the potential to greatly reduce fraud and increase transparency across a range of businesses such as financial, healthcare, smart contracts and supply chain sectors due to its fundamental characteristics of immutability, transparency, and decentralization.

III. PROPOSED WORK

A. Architecture

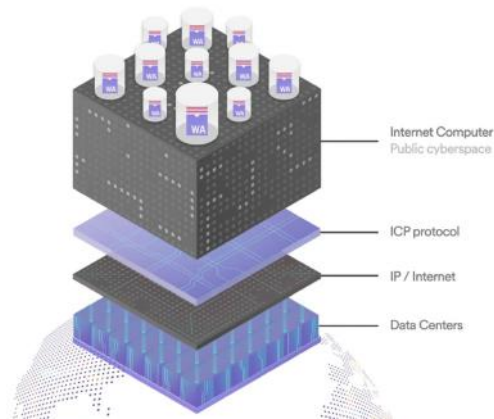


Figure 1: Internet Computer Architecture

Creating a dApp on the Internet Computer, a canister smart contract containing Web Assembly code and configuration is required.

Canister modules are created using Motoko language and SDK. Deployed using HTTPS interface. Users interact with the canister via System API. System API allows them to read from and write to the canister's state, among other things. In addition, users can issue read-only queries for faster results, but they won't be able to modify the canister's state.

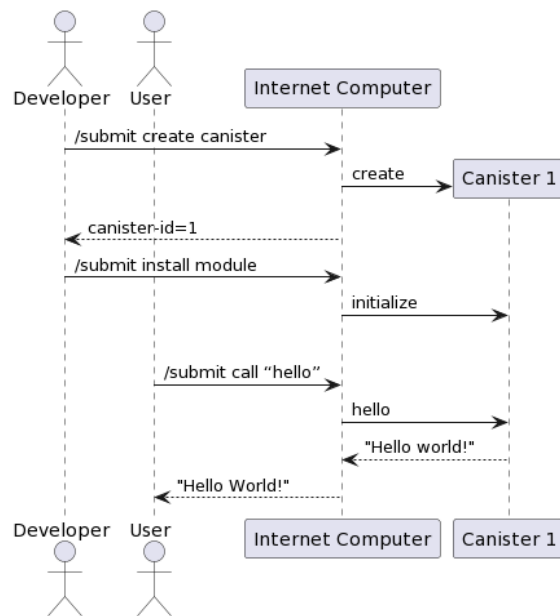


Figure 2 : A Typical use of Internet Computer

B. Peer to peer layer

The peer-to-peer layer (P2P) of the Internet Computer enables secure and reliable communication of network messages (artifacts) between the nodes of a subnet. Artifacts include network messages that need to be broadcasted in the subnet, such as input to canister smart contracts submitted by users or protocol-originating messages like blocks produced by the consensus layer. P2P guarantees the eventual broadcast delivery of an artifact to all nodes requiring it to progress, making it the communication fabric for the IC protocol stack. Gossip protocol is the basic principle behind P2P communication networks. Every node in the subnet is connected to a subset of other nodes, and whenever a node receives or generates an artifact, it gossips it to all its peers. This ensures that every artifact eventually propagates through the whole subnet. Figure 3 presents peer to peer layer

Peer-to-Peer Layer

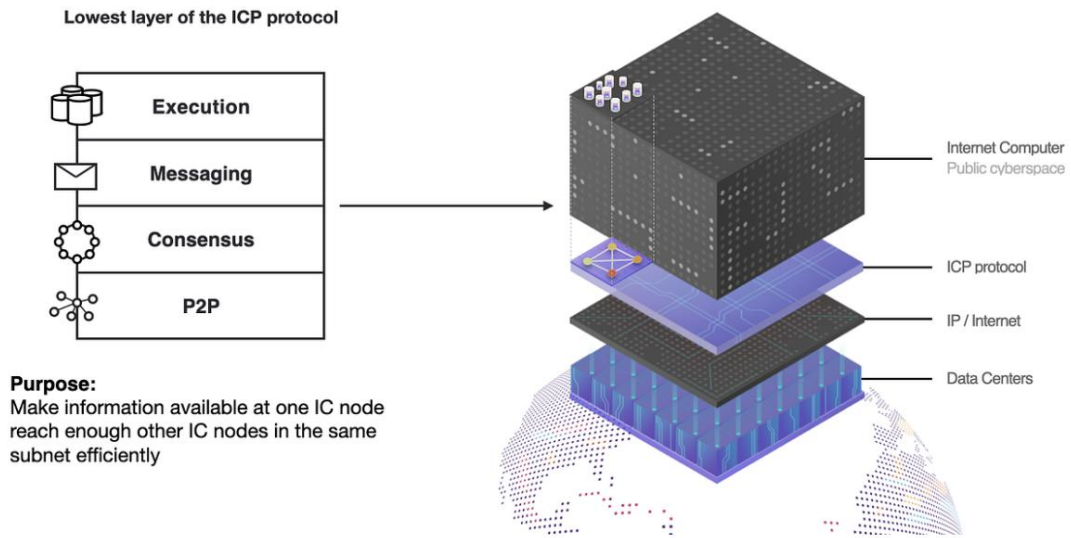


Figure 3: Peer-to-peer layer

Adverts are used to mitigate duplication of delivered artifacts. Instead of sending the artifact to all peers, nodes send adverts to artifacts containing the hash of the artifact and some additional metadata. After receiving an advert, a node may request the corresponding artifact from one or more of its peers who sent it an advert for that artifact.

Prioritization of artifacts is important to ensure that the protocol can always make progress and not be starved of network bandwidth by "less important" traffic. This principle is well known from traditional networking and applies equally well to a blockchain system.

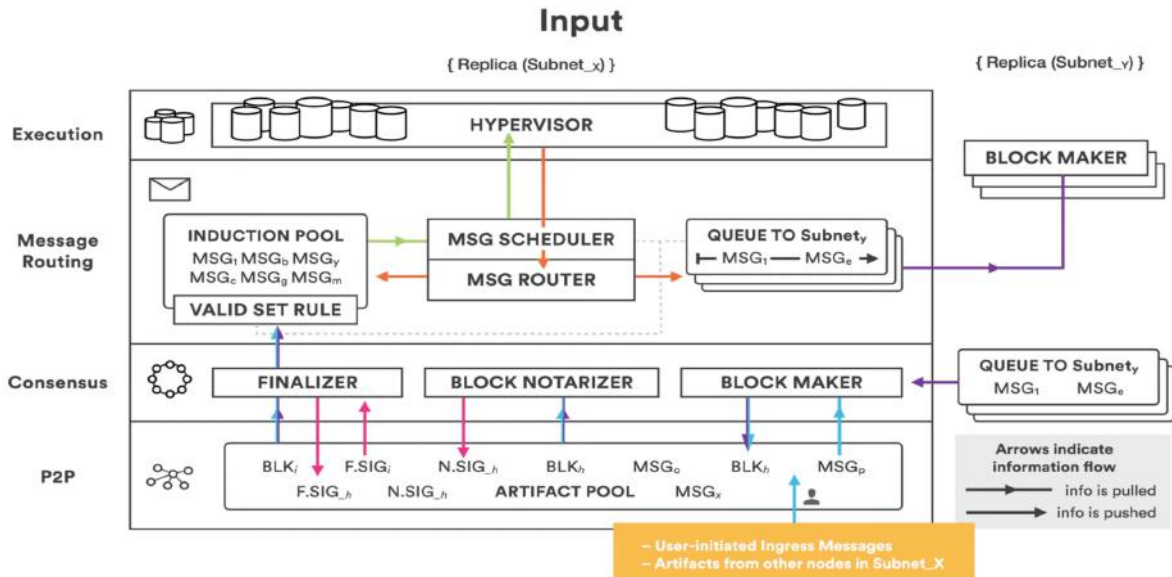


Figure 4: P2P Architecture

Figure 4 presents P2P architecture. To prevent DOS attacks, nodes only request and accept connections with nodes in the same subnet. Subnet membership is managed by the Network Nervous System (NNS). P2P guarantees that all the communication between two nodes is encrypted and authenticated using TLS thanks to the information stored in the NNS canisters.

Consensus Handles Artifact Pool Changes

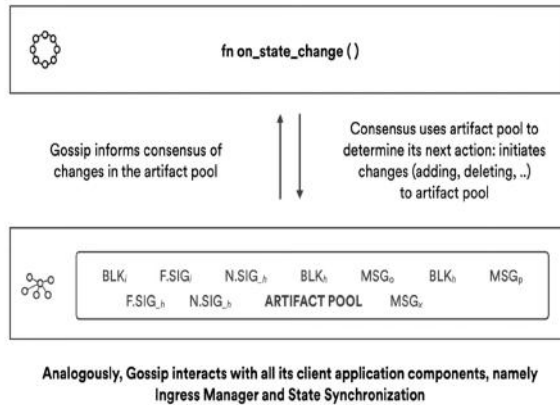


Figure 5: Consensus handles Artifact pool changes

Figure 5, presents consensus handles artifacts pool changes. Asynchronous communication network assumption is used for the IC's communication and consensus layers as it reflects the properties of real-world networks.

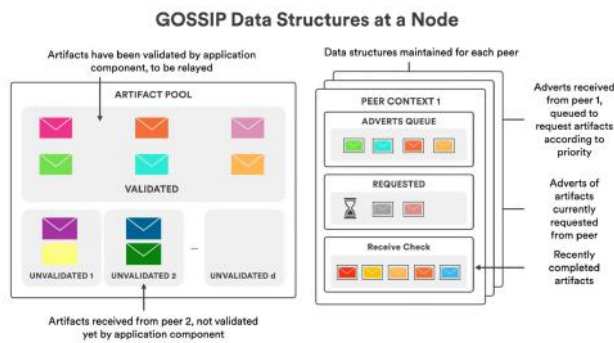


Figure 6: GOSSIP Data Structure at a Node

Figure 6 presents GOSSIP data structure at a Node. The P2P layer is used by the consensus layer to broadcast artifacts to the nodes in the subnet, and it is the communication fabric for the IC protocol stack.

C. Canister Architecture

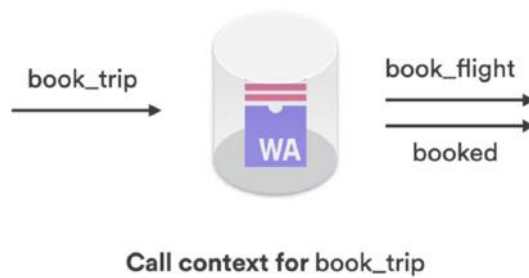


Figure 7: Canister

Figure 7 presents a canister. In the context of the Internet Computer Protocol (ICP), a canister is a fundamental building block that encapsulates an application or a smart contract. It is a secure and isolated environment where code can be deployed and executed on the ICP network.

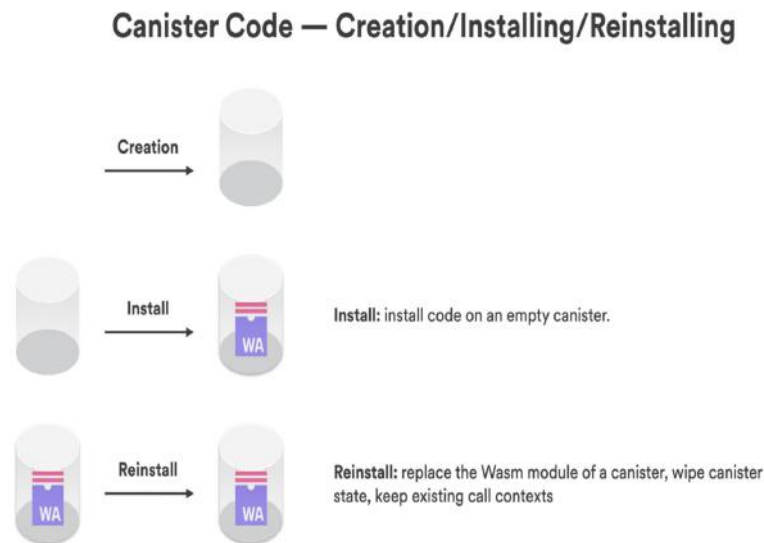


Figure 8: Creation, Installing, and Reinstalling

Creation, Installation, and Reinstalling of canister code is shown in figure 8. A canister is essentially a container that holds the code and data of an application or a smart contract. It is similar to a container in a traditional computing environment but with some important differences. For example, canisters are designed to be completely isolated from each other and from the underlying operating system. This means that even if one canister is compromised, it cannot affect the security or stability of other canisters on the network.

Canisters are a key component of the ICP network because they enable developers to deploy and execute code without having to worry about the underlying infrastructure. They also provide a high degree of security and isolation, which is critical for applications that deal with sensitive data or transactions.

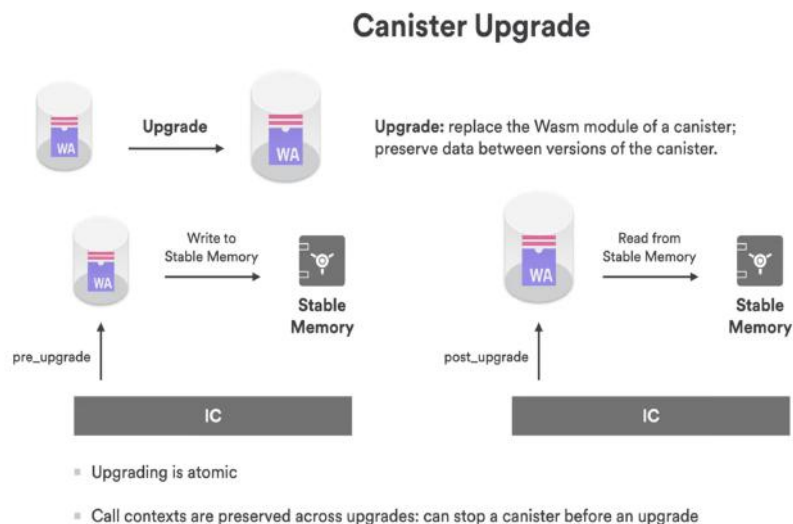


Figure 9: Canister Upgrade

Figure 9 presents the upgradation of the canister. One of the main advantages of using canisters on the ICP network is scalability. Because canisters are designed to be completely isolated from each other, they can

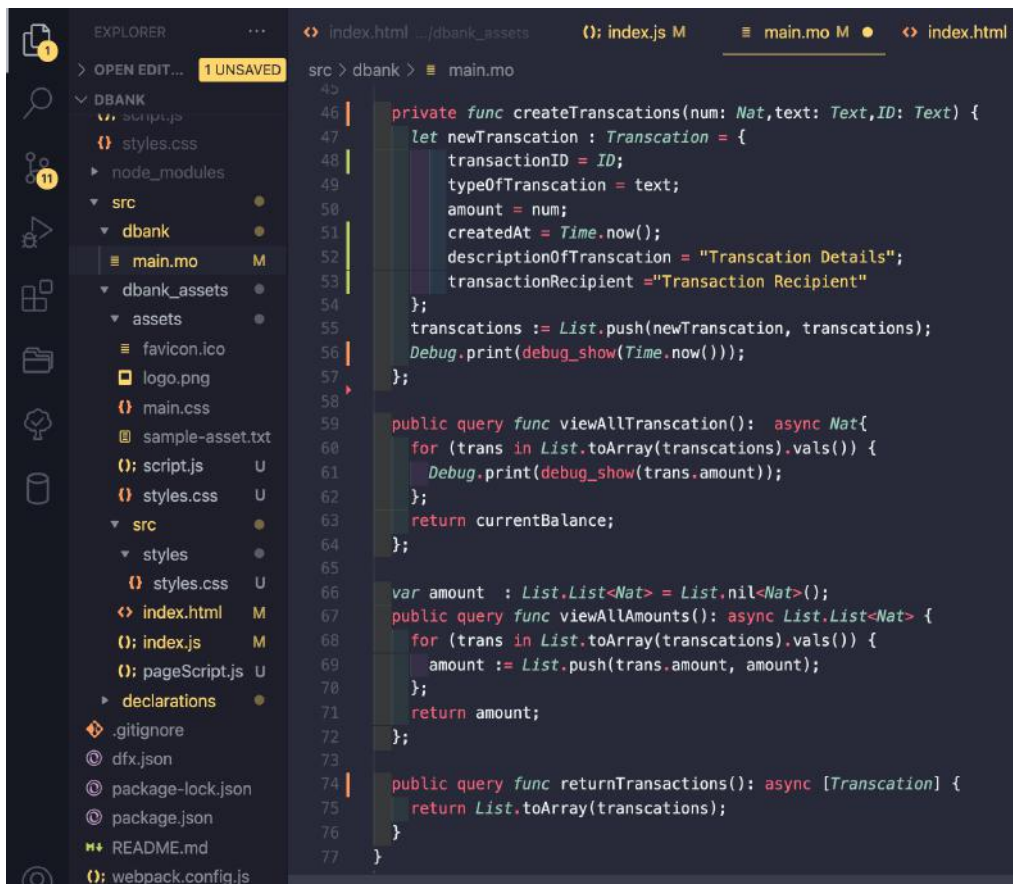
be deployed and scaled independently. This means that applications can be easily scaled up or down as needed, without having to worry about the underlying infrastructure. Another advantage of using canisters on the ICP network is interoperability. Canisters can interact with each other and with other components of the ICP network, such as the Internet Identity (II) service, using open standards and protocols. This enables developers to create complex applications that can leverage the full power of the ICP network.

Overall, canisters are a key component of the ICP network and provide a secure and scalable environment for deploying and executing applications and smart contracts. They enable developers to focus on building their applications, without having to worry about the underlying infrastructure or security concerns.

D. Working Model:

The proposed work involves the development of a transaction history management system using blockchain technology to reduce the misuse of funds with an immutable ledger without using traditional databases. The system is developed using Motoko programming language and ICP Dfinity blockchain platform.

The developed transaction history management system is based on a decentralized blockchain network, where all transactions will be recorded and stored in an immutable ledger. The system developed using Motoko programming language and deployed on the ICP Dfinity blockchain platform.



```
45
46 | private func createTransacions(num: Nat, text: Text, ID: Text) {
47 |     let newTransacion : Transacion = {
48 |         transactionID = ID;
49 |         typeOfTransacion = text;
50 |         amount = num;
51 |         createdAt = Time.now();
52 |         descriptionOfTransacion = "Transacion Details";
53 |         transactionRecipient = "Transaction Recipient"
54 |     };
55 |     transacions := List.push(newTransacion, transacions);
56 |     Debug.print(debug_show(Time.now()));
57 | };
58
59 | public query func viewAllTransacion(): async Nat{
60 |     for (trans in List.toArray(transacions).vals()) {
61 |         Debug.print(debug_show(trans.amount));
62 |     };
63 |     return currentBalance;
64 | };
65
66 | var amount : List, List<Nat> = List.nil<Nat>();
67 | public query func viewAllAmounts(): async List, List<Nat> {
68 |     for (trans in List.toArray(transacions).vals()) {
69 |         amount := List.push(trans.amount, amount);
70 |     };
71 |     return amount;
72 | };
73
74 | public query func returnTransacions(): async [Transacion] {
75 |     return List.toArray(transacions);
76 | }
77 }
```

Figure 10: Backend Code in Motoko

Figure 10 presents backend code in Motoko. The system consists of a smart contract responsible for managing all transactions and maintaining an up-to-date record of the account balance. The smart contract is developed using the Motoko programming language and deployed on the ICP Dfinity blockchain platform. When an account holder performs a transaction, the smart contract will verify the transaction and record it in the blockchain ledger. This will ensure that all transactions are transparent and tamper-proof, reducing the risk of fraud and misuse of funds.

```

src > dbank_assets > src > (); index.js > convertTime
1  import { dbank } from "../../declarations/dbank";
2
3  const records=[]
4
5  function convertTime(intTime){
6      const unixTimeNano = intTime;
7
8      // Convert nanoseconds to milliseconds
9      const unixTimeMs = unixTimeNano / 1000000;
10
11     // Create a new Date object with the Unix time in milliseconds
12     const date = new Date(unixTimeMs);
13
14     // Get the components of the date and time in UTC
15     const year = date.getUTCFullYear();
16     const month = date.getUTCMonth() + 1; // Months are 0-indexed
17     const day = date.getUTCDate();
18     const hours = date.getUTCHours();
19     const minutes = date.getUTCMinutes();
20     const seconds = date.getUTCSeconds();
21
22     // Add 5 hours and 30 minutes to convert to IST
23     const hoursIST = hours + 5;
24     const minutesIST = minutes + 30;
25
26     // Adjust for overflow in minutes
27     if (minutesIST >= 60) {
28         hoursIST += 1;
29         minutesIST -= 60;
30     }

```

Figure 11: Frontend Code in JavaScript

Figure 11 presents the frontend code in javascript. To view the transaction history, account holders and administrators can access the ledger through a user interface that displays all transactions, including the date, time, type of transaction, and amount. The ledger is stored in a decentralized manner, ensuring that it is available to all users at all times and cannot be manipulated.

```

src > declarations > dbank > (); dbank.did.js > ...
1  export const idlFactory = ({ IDL }) => {
2      const List = IDL.Rec();
3      const Transaction = IDL.Record({
4          'typeOfTransaction' : IDL.Text,
5          'transactionRecipient' : IDL.Text,
6          'createdAt' : IDL.Int,
7          'descriptionOfTransaction' : IDL.Text,
8          'amount' : IDL.Nat,
9      });
10     List.fill(IDL.Opt(IDL.Tuple(IDL.Nat, List)));
11     return IDL.Service({
12         'checkBalance' : IDL.Func([], [IDL.Nat], ['query']),
13         'returnTransactions' : IDL.Func([], [IDL.Vec(Transaction)], ['query']),
14         'topUp' : IDL.Func([IDL.Nat], [], ['oneway']),
15         'viewAllAmounts' : IDL.Func([], [List], ['query']),
16         'viewAllTransaction' : IDL.Func([], [IDL.Nat], ['query']),
17         'withdraw' : IDL.Func([IDL.Nat], [], ['oneway']),
18     });
19 };
20 export const init = ({ IDL }) => { return []; };
21

```

Figure 12: Motoko function declarations.

Figure 12 shows the function declaration part in Motoko. The proposed transaction history management system using blockchain technology provides an immutable, transparent, and tamper-proof ledger to reduce the risk of fraud and misuse of funds. The system was developed using the Motoko programming language and deployed on the ICP Dfinity blockchain platform. The smart contract manages all transactions and maintains an up-to-date record of the account balance, while the ledger will be stored in a decentralized manner to ensure availability and security.

The impact of using blockchain technology to maintain transaction records can be significant compared to traditional databases.

E. Results and Discussions

Canister SDK is installed. Build and deploy dApp on a local server. Free cycles are acquired to power the dApp. “Cycle wallets” are established to enable transfer cycles to other dApps that require them. Deploy the dApp on the blockchain for public usage. The following are the steps to develop and deploy a decentralized application (dApp).

IV. STEPS FOR IMPLEMENTATION

A. Installing Tools

- **SDK installation:**

Download and install the latest version of the DFINITY Canister smart contract SDK, called dfx, by running the command below. dfx is natively supported on Linux or macOS 12.* Monterey or later.

- **Install on Mac/Linux:**

To install dfx, Table 1 presents the SDK installation.

Table 1: SDK installation

```
``bash
sh -ci "$(curl -fsSL https://internetcomputer.org/install.sh)"
...

:::tip
If you are using a machine running Apple silicon, you will need to have
[Rosetta](https://support.apple.com/en-us/HT211861) installed. You can
install Rosetta by running
`softwareupdate --install-rosetta` in your terminal.
...

```

- **Install on Windows:**

To install dfx, there is no native support for dfx on Windows. However, by installing Windows Subsystem for Linux (WSL), Follow Microsoft's instructions for installing the [Windows Subsystem for Linux](https://docs.microsoft.com/en-us/windows/wsl/install). Make sure Windows 10 (version 2004 or higher) or Windows 11 is running.

The SDK installation script installs several components in default locations on the local computer. The following table describes the development environment components that the installation script installs:

Table 2 presents the different components that get installed.

Table 2: Components that get installed

Component	Description	Default location
dfx	DFINITY execution command-line interface (CLI)	/usr/local/bin/dfx
moc	Motoko runtime compiler	~/.cache/dfinity/versions/<VERSION>/moc
replica	Internet computer local network Binary	~/.cache/dfinity/versions/<VERSION>/replica
uninstall.sh	Script to remove the SDK and all of its components	~/.cache/dfinity/uninstall.sh
versions	Cache directory that contains a subdirectory for each version of the SDK you install.	~/.cache/dfinity/versions

B. Core components in a versioned directory

The ~/.cache/dfinity/versions directory is used to store cached copies of the DFINITY SDK, but the specific directory structure and contents can vary based on the system and installed components. Table 3 presents the contents of the directory.

Table 3 contents of the ~/.cache/dfinity/versions/0.9.3 directory

total 349192			
drwxr-xr-x	17 pubs	staff	544 Mar 15 11:55 .
drwxr-xr-x	4 pubs	staff	128 Mar 25 14:36 ..
drwxr-xr-x	49 pubs	staff	1568 Mar 15 11:55 base
drwxr-xr-x	20 pubs	staff	640 Mar 15 11:55 bootstrap
-r-x-----	1 pubs	staff	66253292 Mar 15 11:55 dfx
-r-x-----	1 pubs	staff	10496256 Dec 31 1969 ic-ref
-r-x-----	1 pubs	staff	5663644 Dec 31 1969 ic-starter
-r-x-----	1 pubs	staff	9604 Dec 31 1969 libcharset.1.0.0.dylib
-r-x-----	1 pubs	staff	38220 Dec 31 1969 libffi.7.dylib
-r-x-----	1 pubs	staff	668300 Dec 31 1969 libgmp.10.dylib
-r-x-----	1 pubs	staff	958248 Dec 31 1969 libiconv.2.4.0.dylib

```
-r-x----- 1 pubs staff 4200 Dec 31 1969 libiconv.dylib
-r-x----- 1 pubs staff 96900 Dec 31 1969 libz.1.2.11.dylib
-r-x----- 1 pubs staff 15417684 Dec 31 1969 mo-doc
-r-x----- 1 pubs staff 14634020 Dec 31 1969 mo-ide
-r-x----- 1 pubs staff 15111508 Dec 31 1969 moc
-r-x----- 1 pubs staff 49404128 Dec 31 1969 replica
```

C. Command-Line Reference

DFINITY's command-line execution environment, `dfx`, is a versatile tool for developing and managing dApps on the Internet Computer platform. With `dfx`, allows to create, deploy, and manage dApps using various flags and subcommands. To use `dfx`, simply need to enter the appropriate flags and subcommands for the operation that you want to perform. For instance, use `dfx deploy` to deploy a dApp, or `dfx canister` to manage a canister. The syntax for running `dfx` commands is straightforward and easy to use.

D. Basic Syntax

`dfx [subcommand] [flag]`

Depending on the subcommand, the options and flags that specify might apply to the parent command or to a specific subcommand.

The `dfx upgrade` command allows you to check for and download any available updates to the DFINITY SDK. Run this command, it compares the version of `dfx` currently installed with the latest version available for download, and if a newer version is available, it automatically downloads and installs it. Keeping DFINITY SDK up-to-date is important because it ensures access to the latest fixes and enhancements.

Syntax

`dfx upgrade`

E. Installing the Node.js

Node.js is necessary for rendering the frontend assets and so is necessary to complete this tutorial. Note however that Node.js is not needed for canister development in general. It supports all stable versions of Node.js starting with 12. Install 12, 14, or 16. Please note that Node 17 does not support Webpack's api proxy tool, so `npm start` may not work correctly.

Download the Node.js source code or a pre-built installer for the platform.

<https://nodejs.org/en/download/>

F. Motoko Programming Language

Motoko is a modern, general-purpose programming language use specifically to author [Internet Computer](#) canister smart contracts.

The base directory in the versioned subdirectory of the SDK contains the Motoko base library modules that are compatible with that version of the SDK.

G. Create a New Project

A dfx project is a set of artifacts, including source code and configuration files, that can be compiled to a canister. We Use Command Line Interface using command

```
dfx new [name]
```

The terminal output should look similar to this:

```
→ icp-projects dfx new dapp
Fetching manifest https://sdk.dfinity.org/manifest.json
WARN: You seem to be running an outdated version of dfx.
WARN:
You are strongly encouraged to upgrade by running 'dfx upgrade!'
Creating new project "dapp"...
CREATE      dapp/src/dapp_backend/main.mo (105B)...
CREATE      dapp/src/dapp_frontend/assets/sample-asset.txt (24B)...
CREATE      dapp/dfx.json (385B)...
CREATE      dapp/.gitignore (202B)...
CREATE      dapp/README.md (2.32KiB)...
CREATE      dapp/src/dapp_frontend/src/index.js (547B)...
CREATE      dapp/src/dapp_frontend/src/index.html (651B)...
CREATE      dapp/src/dapp_frontend/assets/logo2.svg (14.78KiB)...
· Installing node dependencies...
CREATE      dapp/src/dapp_frontend/assets/main.css (537B)...
CREATE      dapp/src/dapp_frontend/assets/favicon.ico (15.04KiB)...
CREATE      dapp/package.json (1.07KiB)...
· Installing node dependencies...

added 400 packages, and audited 401 packages in 1m

68 packages are looking for funding
  run `npm fund` for details

Done.
Creating git repository...

=====
Welcome to the internet computer developer community!
You're using dfx 0.12.1
```

Figure 13: Terminal output when we create a new dfx project.

Figure 13, presents terminal output when we create a new dfx project.

H. Run dApp Locally

dfx has the ability to start a local execution environment for deploying dApps locally. This environment is optimized for deployment purposes and is not a complete replica of the IC network. As a lightweight environment, it is solely intended for efficient deployment of dApps.

Syntax

```
dfx start
```

```
Feb 18 20:38:38.542 WARN s:rq6jw-w6asl-4fzor-aujbz-4bwjc-wigej-7bwdv-6m
ivh-pao4q-ylrmf-mae/n:xmnc-atnm6-xiahs-cklpy-jymfl-dgf7l-gwi5s-reyjt-z
ats4-jcqav-wae/ic_consensus/block_maker Cannot propose block as the loc
ally available validation context is smaller than the parent validation
context (locally available=ValidationContext { registry_version: 1, ce
rtified_height: 152000, time: Time(1676752718541625000) }, parent conte
xt=ValidationContext { registry_version: 1, certified_height: 152044, t
ime: Time(1676621767444354000) })
```

Figure 14: Terminal output

Figure 14 presents terminal output. Register, build, and deploy the hello canister to the local execution environment by running

Syntax

`dfx deploy`

```
Committing batch.  
Deployed canisters.  
URLs:  
Frontend:  
  dbank_assets: http://127.0.0.1:8000/?canisterId=ryjl3-tyaaa-aaaaa-a  
aaba-cai  
Candid:  
  dbank: http://127.0.0.1:8000/?canisterId=r7inp-6aaaa-aaaaa-aaabq-ca  
i&id=rrkah-fqaaa-aaaaa-aaaaq-cai
```

Figure 15: Terminal output

Figure 15 presents terminal output. Use npm start to start the front-end.

Syntax

`npm start`

```
sent 752 bytes received 48 bytes 1600.00 bytes/sec  
total size is 20693 speedup is 25.87  
11 29d 16h 26m 1 [tmux]
```

Figure 16: Terminal output

Figure 16 presents terminal output. Deploying to Internet Computer Mainnet Deployment

Step 1 - Check the connection

To check the current status of the IC and the ability to connect to it, run the following command

`dfx ping ic`

The expected output is presented in table 4

Table 4: Expected Output

```
{  
  "certified_height": 58554739 "ic_api_version": "0.18.0" "impl_hash":  
"4d21914baeebfd4f510746d571b70646d2dfe6cfcf06457d726d6eb66e7d696c"  
"impl_version": "a2fd44dafbf4ae6c41b26575077f99f847bc924c"  
"replica_health_status": "healthy" "root_key": [48, 129, 130, 48, 29, 6, 13, 43, 6, 1, 4, 1,  
130, 220, 124, 5, 3, 1, 2, 1, 6, 12, 43, 6, 1, 4, 1, 130, 220, 124, 5, 3, 2, 1, 3, 97, 0, 129, 76,  
14, 110, 199, 31, 171, 88, 59, 8, 189, 129, 55, 60, 37, 92, 60, 55, 27, 46, 132, 134, 60,  
152, 164, 241, 224, 139, 116, 35, 93, 20, 251, 93, 156, 12, 213, 70, 217, 104, 95, 145, 58,  
12, 11, 44, 197, 52, 21, 131, 191, 75, 67, 146, 228, 103, 219, 150, 214, 91, 155, 180, 203,  
113, 113, 18, 248, 71, 46, 13, 90, 77, 20, 80, 95, 253, 116, 132, 176, 18, 145, 9, 28, 95,
```

```
135, 185, 136, 131, 70, 63, 152, 9, 26, 11, 170, 174]
}
```

Step 2: Connecting to the ledger to get account information

- Use the following command to confirm which developer identity that are currently using:
dfx identity whoami
- Use the following command to view the principal associated with the current identity:
dfx identity get-principal
- To get the account identifier associated with the developer identity, use this command:
dfx ledger account-id
- Finally, check the account balance using the following command:
dfx ledger --network ic balance

Step 3: Register, Build and Deploy

- Make sure to be in the root directory of the project.
- If necessary, install the node modules by running "npm install" in the project directory.
- Run "**dfx deploy --network ic**" to register, build, and deploy the application. This command installs dApp on the Internet Computer blockchain mainnet.
- Check the command output to ensure that canisters are created, built, installed, and authorized correctly.
- If enough cycles are not available, add more by running
"dfx ledger --network ic top-up gastn-uqaaa-aaaae-aaafq-cai --amount 1.005"
- To get canister ID run command
dfx canister --network ic id [project_name]_assets

This gives the canister ID that looks like this

ryjl3-tyaaa-aaaaa-ababa-cai

- To see the project which is live in blockchain, use this canister ID
https://[canister_ID].raw.ic0.app

V. CONCLUSION

Blockchain technology has become a necessity in today's digital age due to its ability to provide secure, decentralized, and transparent systems for various applications. As traditional centralized systems become more vulnerable to hacking, fraud, and data breaches, blockchain offers a solution by providing a distributed ledger system that is immutable and transparent. This technology has already been implemented in various industries such as finance, supply chain management, healthcare, and voting systems.

In addition to providing secure and efficient transactions, blockchain technology promotes decentralization by removing intermediaries such as banks and financial institutions in financial transactions. This creates more opportunities for financial inclusion and empowerment, particularly in developing countries where traditional banking services may not be readily available. The use of blockchain technology is becoming increasingly necessary as society becomes more dependent on digital systems and transactions. Its ability to provide security, transparency, and decentralization makes it a valuable tool in various industries and applications.

The DFINITY Foundation, a Swiss-based not-for-profit organization, recognizes the potential of blockchain technology and has a vision to create a blockchain singularity. The foundation aims to rebuild every system and service using smart contracts that run entirely on the Internet Computer blockchain. As a major contributor to the development of the Internet Computer blockchain, the foundation has the largest R&D team in the blockchain industry. Furthermore, the Internet Computer blockchain is an open-source tool that allows developers to build decentralized applications and smart contracts. With the development of innovative technologies like the Internet Computer blockchain, the potential for future innovation in blockchain technology is vast.

VI. CODE REFERENCE

<https://github.com/mohith-venkata/ic-projects>

REFERENCES

- [1] The Internet Computer for Geeks (v1.3), The DFINITY Team* April 19, 2022 <https://internetcomputer.org/whitepaper.pdf>
- [2] J. Camenisch, M. Drijvers, T. Hanke, Y.-A. Pignolet, V. Shoup, and D. Williams. Internet Computer Consensus. Cryptology ePrint Archive, Report 2021/632, 2021. <https://ia.cr/2021/632>.
- [3] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Retrieved from <https://bitcoin.org/bitcoin.pdf>
- [4] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In E. Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques*, Warsaw, Poland, May 4-8, 2003, Proceedings, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.
- [5] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In C. Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security*, Gold Coast, Australia, December 9-13, 2001, Proceedings, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.
- [6] Y. Desmedt. Society and Group Oriented Cryptography: A New Concept. In C. Pomerance, editor, *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques*, Santa Barbara, California, USA, August 16-20, 1987, Proceedings, volume 293 of *Lecture Notes in Computer Science*, pages 120–127. Springer, 1987.
- [7] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. Cryptology ePrint Archive, Report 2017/454, 2017. <https://eprint.iacr.org/2017/454>.
- [8] R. Pass and E. Shi. Thunderella: Blockchains with Optimistic Instant Confirmation. In J. B. Nielsen and V. Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II, volume 10821 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2018.
- [9] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham. HotStuff: BFT Consensus in the Lens of Blockchain, 2018. arXiv:1803.05069, <http://arxiv.org/abs/1803.05069>.
- [10] R. C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques*, Santa Barbara, California, USA, August 16-20, 1987, Proceedings, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987.
- [11] Xu et al., 2018 is a paper titled "A Survey on Smart Contracts: Challenges, Solutions, and Open Issues."
- [12] Crosby, M., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2(6-10), 71-81.
- [13] Kshetri, N. (2018). Blockchain's roles in meeting key supply chain management objectives. *International Journal of Information Management*, 39, 80-89.
- [14] Zheng, Z., Xie, S., Dai, H.-N., Chen, W., & Wang, H. (2018). Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4), 352-375.
- [15] Zhang, J., Wen, X., & Zhang, H. (2018). Blockchain based transparent and secure logistics. In *2018 3rd International Conference on Automation, Control and Robotics Engineering (CACRE)* (pp. 21-24). IEEE.
- [16] Huang, X., Zeng, X., Wu, Q., & Li, Y. (2017). An e-commerce logistics model based on blockchain technology. In *Proceedings of the 2017 2nd International Conference on Logistics and Intelligent Transportation Systems* (pp. 177-182). ACM.
- [17] Li, S., Li, J., & Li, X. (2019). Blockchain in logistics and supply chain: A review. *International Journal of Information Management*, 49, 264-272.
- [18] Lee, J. (2019). Blockchain-based smart contract in supply chain management. *Sustainability*, 11(16), 4407.

- [19] Fanning, K., & Centers, D. P. (2016). Blockchain and its coming impact on financial services. *The Journal of Corporate Accounting & Finance*, 27(5), 53-57.
- [20] Tapscott, D., & Tapscott, A. (2016). *Blockchain revolution: how the technology behind bitcoin is changing money, business, and the world*. Penguin.
- [21] Antonopoulos, A. M. (2014). *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*.
- [22] Domingo-Ferrer, J., Martínez-Ballesté, A., & Soria-Comas, J. (2018). The rise of smart contract-based decentralized autonomous organizations. *IEEE Internet Computing*, 22(3), 26-34.
- [23] Buterin, V. (2014). A next-generation smart contract and decentralized application platform. *Ethereum Project Yellow Paper*, 151(3)
- [24] Domingo-Ferrer, J., de la Prieta, F., & Prieto, J. M. (2018). A systematic review of smart contracts: Challenges, solutions, and open issues. *ACM Computing Surveys (CSUR)*, 51(5), 1-39.
- [25] Swan, M. (2015). *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc.
- [26] Pilkington, M. (2016). Smart contracts: State of the art, challenges, and opportunities. *International Journal of Electronic Commerce*, 21(1), 22-39.
- [27] Radanovic, M., Tasca, P., & Tessone, C. J. (2018). Blockchain-empowered smart contracts: A systematic mapping study. *ACM Computing Surveys (CSUR)*, 51(4), 1-42.