 **Dr. D. Y. Patil Pratishthan’s**

**DR. D. Y. PATIL INSTITUTE OF ENGINEERING, MANAGEMENT & RESEARCH**

**Approved by A.I.C.T.E, New Delhi , Maharashtra State Government, Affiliated to Savitribai Phule Pune University**

Sector No. 29, PCNTDA , Nigidi Pradhikaran, Akurdi, Pune 411044. Phone: 020–27654470, Fax: 020-27656566  
Website : [www.dypiemr.ac.in](http://www.dypiemr.ac.in) Email : [principal.dypiemr@gmail.com](mailto:principal.dypiemr@gmail.com)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DEPARTMENT**

**OF**

**COMPUTER ENGINEERING**

**LAB MANUAL**

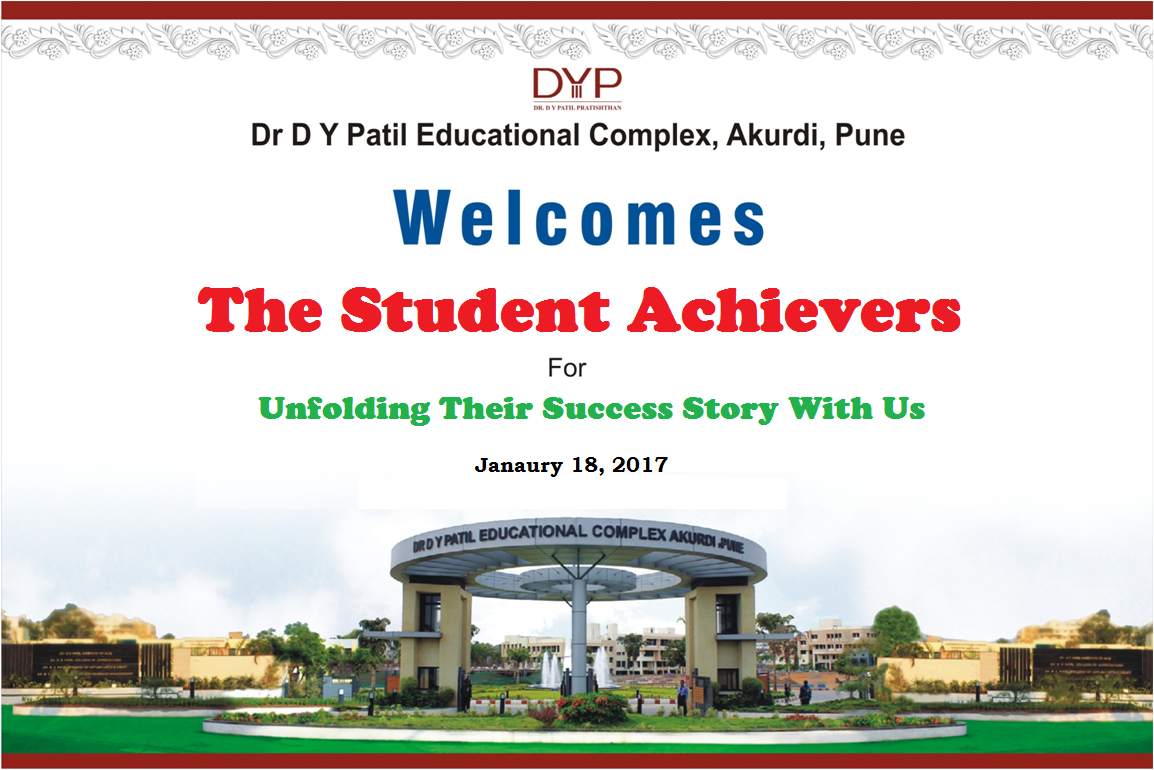
**Lab Practices III**

**(Final Year Engineering)**

**Semester – I**

**Prepared by :** Mrs. Pooja Mishra

Ms. Deepali Jawale



**Institute Vision**

To strive for excellence by providing quality technical education and facilitate research for the welfare of society

**Institute Mission**

1. To educate students with strong fundamentals by providing conducive environment
2. To inculcate research with creativity & innovation
3. To strengthen leadership, team work, professional & communication skills and ethical standards
4. To promote Industry Institute collaboration & prepare students for life long learning in context of technological change.

**Department Vision**

To produce quality computer professionals and fostering research aptitude for dispensing service to society

**Department Mission**

1. To promote growth of an individual by imparting comprehensive knowledge of tools and technologies.
2. To facilitate research and innovation by engaging faculty and students in research activities.
3. To enrich industry-institute interaction in order to provide a platform to know industry demands and motivation for self-employment.
4. To bring forth a conducive environment to enhance soft skills and professional skills to cater needs of society

**Program Specific Outcomes**

1. Professional Skills: The ability to comprehend, analyze and develop software and hardware systems and amp; applications through research, in varying domains.
2. Problem-Solving Skills: The ability to apply standard paradigms and strategies in software project development using open-ended programming environments to deliver a quality product.
3. Successful Career and Entrepreneurship: Adaptation of modern practical and systematic approaches in creating innovative solutions for a successful career, entrepreneurship, and a zest for higher studies.

**Course Objective**

1. Learn effect of data preprocessing on the performance of machine learning algorithms
2. Develop in depth understanding for implementation of the regression models.
3. Implement and evaluate supervised and unsupervised machine learning algorithms.
4. Analyze performance of an algorithm.
5. Learn how to implement algorithms that follow algorithm design strategies namely divide and conquer, greedy, dynamic programming, backtracking, branch and bound.
6. Understand and explore the working of Blockchain technology and its applications.

**Course Outcome**

After completion of the course, students will be able to

CO1: Apply preprocessing techniques on datasets.

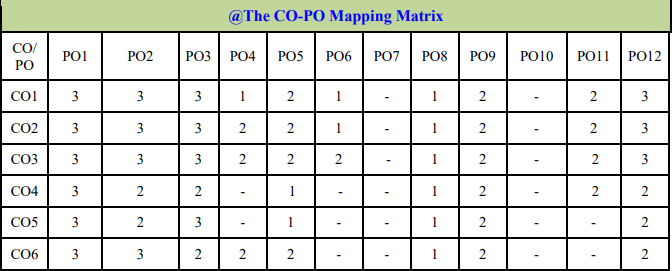
CO2: Implement and evaluate linear regression and random forest regression models.

CO3: Apply and evaluate classification and clustering techniques.

CO4: Analyze performance of an algorithm.

CO5: Implement an algorithm that follows one of the following algorithm design strategies: divide and conquer, greedy, dynamic programming, backtracking, branch and bound.

CO6: Interpret the basic concepts in Blockchain technology and its applications



**Guidelines for Students**

Guidelines for Student's Laboratory Journal The laboratory assignments are to be submitted by students in the form of a journal. Journal consists of Certificate, table of contents, and handwritten write-up of each assignment (Title, Date of Completion, Objectives, Problem Statement, Software and Hardware requirements, Assessment grade/marks and assessor's sign, Theory- Concept in brief, algorithm, flowchart, test cases, Test Data Set(if applicable), mathematical model (if applicable), conclusion/analysis. Program codes with sample output of all performed assignments are to be submitted as a softcopy. As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to a journal must be avoided. Use of DVD containing student programs maintained by Laboratory In-charge is highly encouraged. For reference one or two journals may be maintained with program prints in the Laboratory.

**Guidelines for Laboratory /Term Work Assessment**

Continuous assessment of laboratory work should be based on overall performance of Laboratory assignments by a student. Assessment of each Laboratory assignment will assign grade/marks based on parameters, such as timely completion, performance, innovation, efficient codes, punctuality, documentation and neatness

**Virtual Laboratory:**

1. <http://cse01-iiith.vlabs.ac.in/>
2. <http://vlabs.iitb.ac.in/vlabs-dev/labs/blockchain/labs/index.php>
3. <http://vlabs.iitb.ac.in/vlabs-dev/labs/machine_learning/labs/index.php>

# Table of Contents

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.No |  | Title of the Experiment | Page No |
| **Group A: Design and Analysis of Algorithms** | | | |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| **Group B : Machine Learning** | | | |
| 9 |  |  |  |
| 10 |  |  |  |
| 11 |  |  |  |
| 12 |  |  |  |
| 13 |  |  |  |
| **Group C : Blockchain Technology** | | | |
| 14 |  |  |  |
| 15 |  |  |  |

**Assignment No. C1**

* **Title:** Installation of MetaMask and study spending Ether per transaction.
* **Objective:** Study and understand Open Source metamask tool to get understanding of ether per transaction.
* **Theory:**

MetaMask is a [software](https://en.wikipedia.org/wiki/Software) [cryptocurrency wallet](https://en.wikipedia.org/wiki/Cryptocurrency_wallet) used to interact with the [Ethereum](https://en.wikipedia.org/wiki/Ethereum) [blockchain](https://en.wikipedia.org/wiki/Blockchain). It allows users to access their Ethereum wallet through a [browser extension](https://en.wikipedia.org/wiki/Browser_extension) or [mobile app](https://en.wikipedia.org/wiki/Mobile_app), which can then be used to interact with [decentralized applications](https://en.wikipedia.org/wiki/Decentralized_application). MetaMask is developed by [ConsenSys Software Inc.](https://en.wikipedia.org/wiki/ConsenSys), a [blockchain](https://en.wikipedia.org/wiki/Blockchain) [software company](https://en.wikipedia.org/wiki/Software_company) focusing on [Ethereum](https://en.wikipedia.org/wiki/Ethereum)-based tools and infrastructure.

MetaMask allows users to store and manage account [keys](https://en.wikipedia.org/wiki/Key_(cryptography)), broadcast transactions, send and receive [Ethereum](https://en.wikipedia.org/wiki/Ethereum)-based [cryptocurrencies](https://en.wikipedia.org/wiki/Cryptocurrency) and tokens, and securely connect to [decentralized applications](https://en.wikipedia.org/wiki/Decentralized_application) through a compatible [web browser](https://en.wikipedia.org/wiki/Web_browser) or the [mobile app](https://en.wikipedia.org/wiki/Mobile_app)'s built-in browser.

Websites or other [decentralized applications](https://en.wikipedia.org/wiki/Decentralized_application) are able to connect, authenticate, and/or integrate other [smart contract](https://en.wikipedia.org/wiki/Smart_contract) functionality with a user's MetaMask wallet (and any other similar blockchain wallet browser extensions) via [JavaScript](https://en.wikipedia.org/wiki/JavaScript) code that allows the website to send action prompts, signature requests, or transaction requests to the user through MetaMask as an intermediary.

The application includes an integrated service for exchanging [Ethereum](https://en.wikipedia.org/wiki/ERC-20) tokens by aggregating several [decentralized exchanges](https://en.wikipedia.org/wiki/Decentralized_exchange) (DEXs) to find the best [exchange rate](https://en.wikipedia.org/wiki/Exchange_rate). This feature, branded as MetaMask Swaps, charges a service fee of 0.875% of the transaction amount.

* **Procedure /Method :**

Here is how to set up and install MetaMask on a desktop browser Download and install the MetaMask extension for your browser

1. Click on the “Get Started” button
2. Select “Create a Wallet” and create a password
3. Write down, store, or memorize your Secret Backup Phrase
4. Confirm your Secret Backup Phrase to ensure you have it correct

Here is how to set up and install MetaMask on mobile devices:

1. Download and launch the MetaMask app from Google Play or the App Store
2. Click on the “Get Started” button
3. Select “Create a Wallet” and create a password
4. Write down, store, or memorize your Secret Backup Phrase
5. Confirm your Secret Backup Phrase to ensure you have it correct

### Download and install MetaMask

The first step is to download the official MetaMask extension, add-on, or application. To set up MetaMask on a mobile device, you will need to visit either Google Play or the App Store. Desktop computer users will need to download the extension or add-on for their browser of choice. MetaMask is currently available for iOS, Android, Chrome (or Brave), Firefox, and Microsoft Edge. Here are the official download links for [MetaMask](https://metamask.io/):

* [App Store (iOS)](https://apps.apple.com/us/app/metamask/id1438144202)
* [Google Play (Android)](https://play.google.com/store/apps/details?id=io.metamask)
* [Chrome (or Brave)](https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn)
* [Firefox](https://addons.mozilla.org/en-US/firefox/addon/ether-metamask/)
* [Microsoft Edge](https://microsoftedge.microsoft.com/addons/detail/metamask/ejbalbakoplchlghecdalmeeeajnimhm?hl=en-US)

### Getting Started

MetaMask has an on-screen setup tutorial for all users on every supported browser. Follow the instructions, read through the terms, and click “I Agree” to continue the setup process.

### Create Your Wallet

The first step in getting set up with MetaMask is creating a wallet. Click on the “Create a Wallet” button. You will then need to create a password with a minimum of eight characters that you will use to log in to the browser extensions or add-on.

### Secret Backup Phrase

Your Secret Backup Phrase is a way to easily backup and restore your account at any time. It consists of twelve words in a particular sequence. It is essential to memorize this, write it down on a piece of paper and place it in a safe place, or store it in a password manager like [1Password](https://1password.com/) or [LastPass](https://www.lastpass.com/).

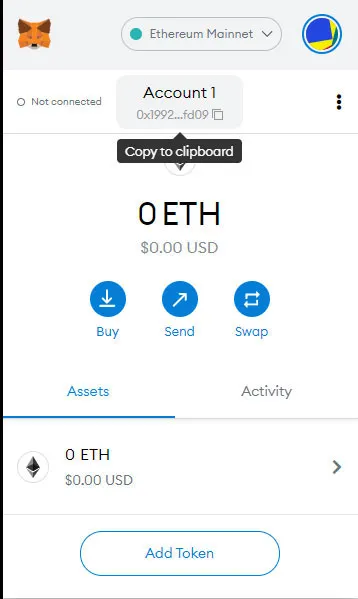
Without your Secret Backup Phrase, it will be impossible to restore your account. You can download the phrase and store it on an external encrypted hard drive or storage device for added security. We do not recommend copy and pasting it into a notepad and leaving it on your desktop. If your computer gets stolen, hacked, or someone finds the file, someone can easily log in to your MetaMask wallet.

Confirm your Secret Backup Phrase on the next on-screen prompt when you have it safely stored. This step ensures you have the correct phrase on hand in case you need to restore your account. After confirming your Secret Backup Phrase, click on the “All Done” button. You have now created a MetaMask wallet and are connected to the Ethereum Mainnet.

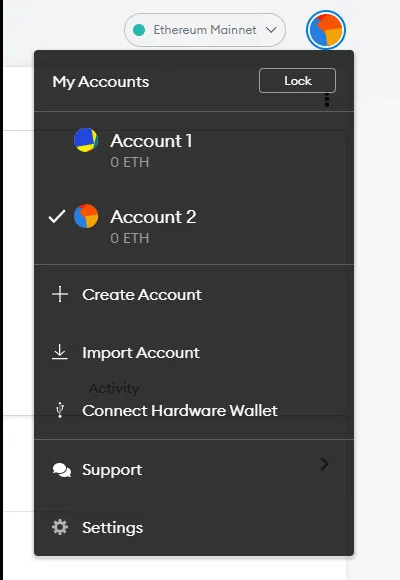
Those are the complete steps for how you install and set up MetaMask on PC and Mobile for Chrome, Firefox, Microsoft Edge, iOS, and Android.

### How to find your Ethereum address in MetaMask

After installing MetaMask in your browser and the initial setup process is complete, you can click the MetaMask extension or add-on icon from the toolbar. If you are on mobile, just launch the application. The MetaMask interface should now be visible to you, and you will see a string of numbers and letters under “Account 1” near the top of the window. That is your Ethereum address, and you can click on it to copy it to your clipboard.



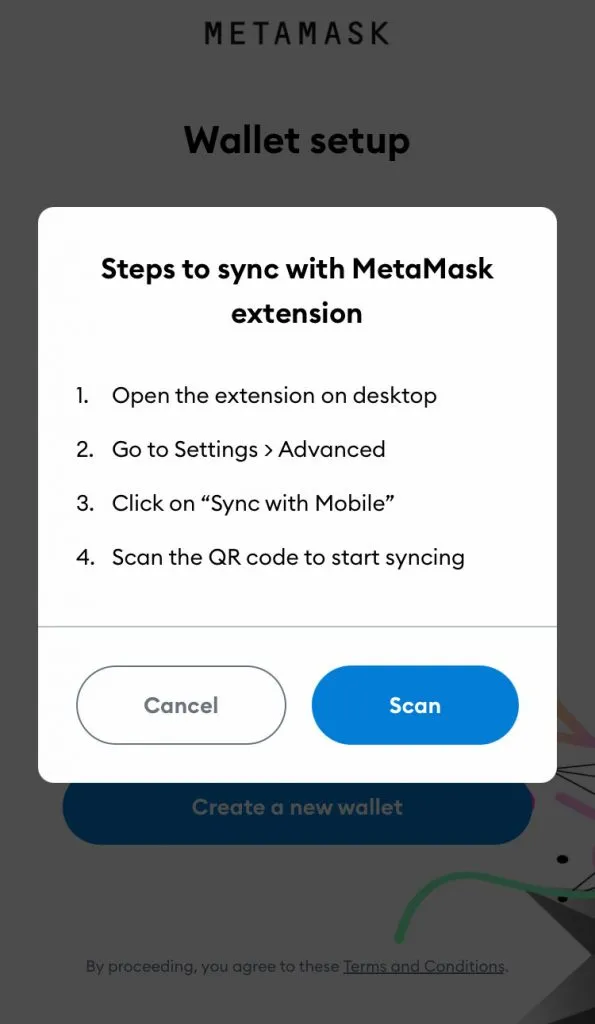
You can create an unlimited amount of Ethereum addresses under the same account. To achieve this, click on the circle icon on the top right of the interface (or the top middle on mobile). Select the Create Account option, pick an Account Name, and click on the Create button. A new account has a new unique Ethereum address. You can toggle between these by clicking on the same circle icon and selecting your account under My Accounts.



## **How to Sync Metamask Across Devices :**

MetaMask allows you to sync your wallets across multiple devices. To sync your MetaMask mobile app to your MetaMask browser extension or add-on, follow these steps:

1. Open your preferred browser and log in to **MetaMask**.
2. Tap on the **MetaMask icon** from the browser toolbar to open the interface, and click on your icon on the top right.
3. Select the “Settings” option, go to “**Advanced**,” and select “**Sync with Mobile**.” You should see a QR code.
4. Open the **MetaMask mobile application**.
5. Select the “Sync or Import” option and click on “**Scan QR code**.”
6. Scan the QR code with your MetaMask mobile app.



* **Refrances :**
* **Questions and answer :**
* **Conclusion :**

The students got the knowledge of Installation of MetaMask and study spending Ether per transaction.

**Assignment No. C2**

* **Title:** Create your own wallet using Metamask for crypto transactions.
* **Objective:** Is to understand Open Source metamask tool and create own wallet for crypto transactions
* **Theory :**

MetaMask is one of the most popular cryptocurrency wallets. Despite the ups and downs of the market, there are more and more users each year. MetaMask has about 21 million active users each month, 80 times more than in 2019 and significantly more than any noncustodial wallet. This article will explain why this wallet is so popular and its functions. Also, we will describe the development of a cryptocurrency wallet like MetaMask.

## **A Brief introduction to MetaMask**

MetaMask is an open-source, straightforward, and easy-to-use cryptocurrency wallet. It functions as a web browser extension available for Chrome, Firefox, Brave, or a mobile application for iOS or Android. Initially, this wallet supported only Ether and ERC-20 tokens, and now it is compatible with ERC-721 and ERC-1155 token standards. Furthermore, MetaMask benefits include interaction with websites; hence, it can function as a connection node for various DApps on Ethereum.

[Adrian Devis](https://twitter.com/kumavis_) and [Dan Finlay](https://twitter.com/danfinlay) are the MetaMask developers. Their idea was revolutionary and straightforward; they intended to create a web browser extension that would allow managing cryptocurrency and using the browser for fast and secure access with DApps. ConsenSys Software Inc. — a development company, focusing on applications that use Ethereum’s blockchain, implemented the idea in 2016.

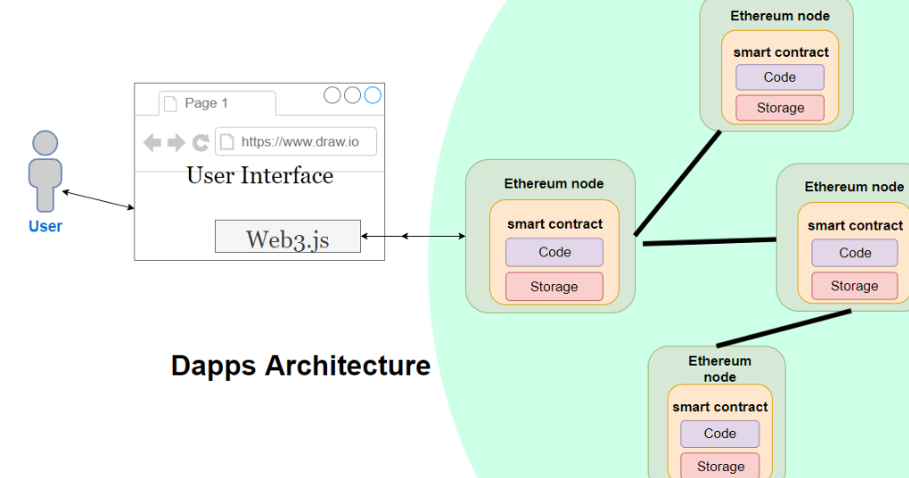
The solution used Ethereum’s interface and a web API called [web3.js](https://web3js.readthedocs.io/en/1.0/). This Ethereum library is the fundament of MetaMask since it allows the browser to interact with the local or remote blockchain nodes via HTTP, IPC, and WebSocket; also, it gained the ability to record and read data from smart contracts, transfer tokens, etc. In another way, web3.js allowed the blockchain developers to create proxy and communication bridges between MetaMask, DApps, and the user.

Adrian Devis and Dan Finlay admit that their idea was great. Yet, the technical implementation was super complicated, especially in providing security for the users (web wallets are considered the most vulnerable to hacker attacks). Nonetheless, ConsenSys succeeded, and on the 14th of July in 2016, they offered the first version of MetaMask web browser cryptocurrency wallet for Chrome. Later, they presented a version for Firefox, Brave, and other popular browsers. In 2019 they also launched the mobile version of the MetaMask cryptocurrency wallet.

## **How does the MetaMask wallet function?**

As we mentioned above, the MetaMask cryptocurrency wallet employs the web3.js library to function. This library is a part of the official Ethereum product. The library was developed focusing on the requirements of web applications that could interact with the Ethereum blockchain and take advantage of all blockchain’s benefits and functions.

MetaMask is a cryptocurrency wallet for Ethereum and an instrument that helps to interact with DApps. MetaMask connects the extension to the DApp so that to fulfill both tasks. When the application identifies the MetaMask, it creates a connection, and the user can start using all the features of a specific application.



For instance, it can assets trading, access to resources or services, or any other task within the capability of a DApp. Each action has its cost (transaction fee) that must be paid in Ethereum or any specified token. MetaMask wallet has all instruments and protocols for this purpose.

Hence, we can state that Metamusk also controls the interaction of the user and DApp, and processes the operations required for specific actions, besides the function of a wallet. Reliable and secure cryptography and safe internet connection are the environments for these operations.

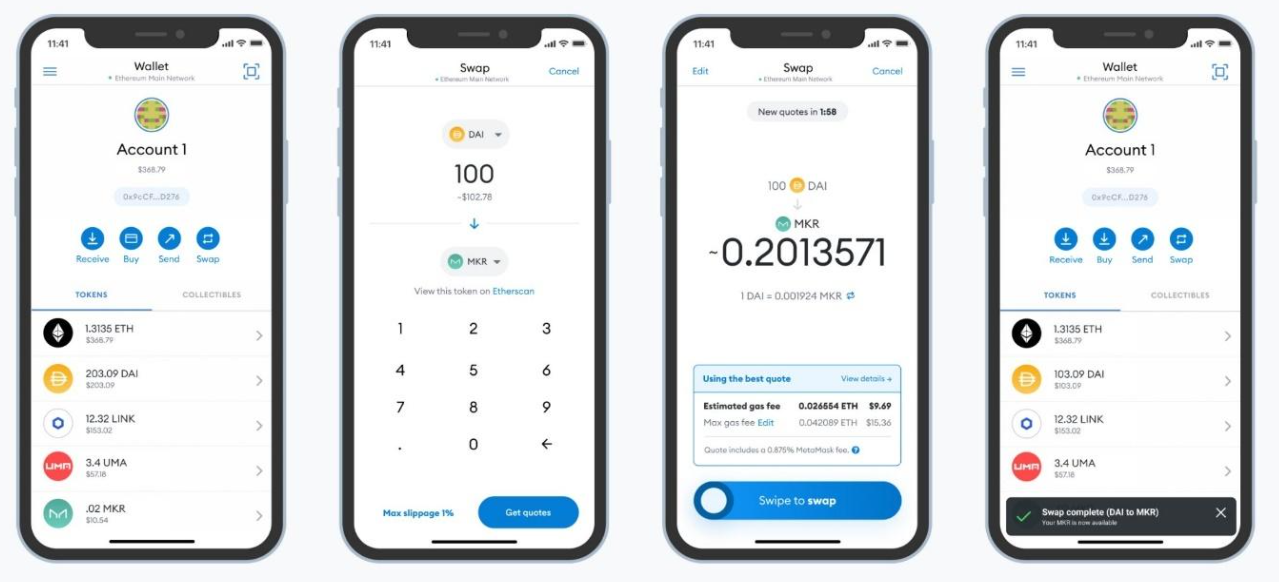
Furthermore, MetaMask can generate asymmetric keys, store them on a local device, and manage access to the keys. To sum up, MetaMask is a super-safe extension.

## **MetaMask wallet key features**

Easy to use. The first and most apparent benefit of the MetaMask wallet is the ease of use. The wallet offers an intuitive and straightforward user interface that makes the management of cryptocurrencies

and interaction with DApps easier than ever before.

Furthermore, it allows the creation of several wallets. When users create a new wallet, they generate new public and private keys that MetaMask users have fast access.



* **Integration with various DApps :**

MetaMask’s users can connect with numerous decentralized applications with a single tap on the screen. It allows them to exchange their tokens swiftly on Uniswap or PancakeSwap exchangers, launch blockchain games like Gods Unchained or Decentraland, buy NFTs on OpenSea or Rarible. All these actions do not require complicated processes. The only movement is connecting the MetaMask cryptocurrency wallet.

* **Integration with other blockchains :**

The additional benefit of MetaMask is that you can join it to other blockchains, e.g., Avax, Polygon, Fantom, Binance Smart Chain (BSC).

* **Hardware wallets support :**

MetaMask service is compatible with hardware cryptocurrency wallets like Ledger, Nano, or Trezor. All you need is to tap the ‘connect a hardware wallet’ button. This is a significant benefit that helps to interact with applications that do not support Trezor and Ledger hardware wallets.

* **Swaps support :**

In 2021 MetaMask offered its users a relatively cheap and fast opportunity to exchange their tokens within the wallet with Swaps. This function supports several automatic marker makers (MM) that help find the best token exchange rate.

* **NFT support :**

MetaMask wallet also allows storing NFTs. When a user buys an NFT on a platform, the token is automatically displayed in the tab of collectible items in the wallet.

## **Main functions of a cryptocurrency wallet**

To develop a cryptocurrency wallet like MetaMask, one should consider two roles of users: regular users and administrators. The average user apparently will use the wallet to store, trade, and exchange tokens. The administrators are the employees responsible for the proper functioning and management of the wallets. Often, they solve the issues of the regular users.

* **Here is a list of functions for regular users:**
  1. **Registration :**It must be easy to create a new profile. The essentials are opening an internet browser, downloading the MetaMask extension from the official website, and coming up with a complicated password. Next, the user will need to put down the seed phrase, and they are ready to go. If one already has a cryptocurrency address, one can easily connect it to the cryptocurrency wallet.
  2. **Applicable exchange rate.** MetaMask displays only the amount of tokens on the account, while other cryptocurrency wallets show the exchange rate for tokens in the investment profile. If you plan to develop a MetaMask crypto wallet clone, it is a sound idea to add these features, which will enhance the user experience of your upcoming project.
  3. **Operations with the cryptocurrency assets.** The main functions of any wallet are transferring, receiving, and exchanging cryptocurrencies; hence, it is vital that these functions are straightforward so that any user is confident in using the wallet. Furthermore, it is an advantage if there is a chance to check the current balance and transaction history. If you have experience trading on an exchange, you will expect MetaMask to display the history of operations and the current balance; yet MetaMask does not offer it. Pay special attention to it if you intend to develop a wallet.
  4. **Favorite addresses.** This is another feature that MetaMask is not showing, while the other wallets do. It allows the users to make a list of addresses for fast token transfers. It is super convenient for minor regular operation fees.
  5. **Employing other blockchains.** After installing the MetaMask extension or application, the user can use only Ether and ERC-20 tokens. However, the users can connect to other blockchains. What is more, they will have a single address for all blockchains.
  6. **NFT support.** Recently, MetaMask wallet also started supporting non-fungible tokens.

Here is a list of necessary functions for platform administrators.

1. T**ools panel :** The administrators can use this tab to trace the current state of the platform, current issues, the number of users, transaction volume, etc.
2. **User management** : This tab helps the administrators track the users' status and manage them: add, delete, block, communicate, verify, etc.
3. **Managing the fees.** It helps to add, delete, or change the transaction fees.
4. **Payment system management**. MetaMask wallet does not offer to link the credit or debit card, PayPal, or another payment system. However, if the wallet offers these functions, the administrators require instruments to manage it.
5. **Content management**. These are the user interface management tools (changing texts, icons, links, etc.).
6. **Connecting Facebook**, Twitter, and Instagram accounts.
7. **Reports and analytics**.
8. **API documentation**

## **Extended functions set for MetaMask clone**

To help your MetaMask wallet clone become famous, you should add some advantages that highlight it from the competitors and improve the user experience.

These can be the following:

* **Linking an account.** Your users will find it useful to be able to buy a cryptocurrency and exchange it for fiat within the wallet. This will be possible if you develop a wallet like MetaMask and add the feature of linking bank accounts, credit/debit cards, PayPal, or other online payment systems.
* **Ecommerce Integrations :** We mean integrating the wallet with exchanges, NFT marketplaces, decentralized applications, shops, and other services that the users might find useful.
* **Multilingual interface :** If you focus on a market where all people speak the same language, you might neglect this aspect. However, your intentions are global, and you should add as many languages as possible to increase the target audience.
* **Push notifications** **:** The notifications will inform the users of receiving payments, ending transactions, rapid exchange rate changes in the investment account, system updates,  suspicious activity, etc.
* **VIP support** **:**Numerous cryptocurrency trading platforms offer support for an additional fee. This may include 24/7 support, communication with a personal specialist, etc.
* **QR scanner** **:**This is another useful feature that allows your users to make payments even faster. Moreover, it will decrease the number of transfers done by mistake.

**Assignment No. C3**

* **Title:** Write a smart contract on a test network, for Bank account of a customer for following operations:
* Deposit money
* Withdraw Money
* Show balance
* **Objective:**
* **Theory :**

Solidity is a high-level language. The structure of smart contracts in solidity is very similar to the structure of classes in object-oriented languages. The solidity file has an extension .sol.

What are Smart Contracts?

Solidity’s code is encapsulated in contracts which means a contract in Solidity is a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain. A contract is a fundamental block of building an application on Ethereum.

Example: In the below example, the aim is to deploy a Smart Contract for Marks Management System by using Solidity. In this contract, the details of every student like student ID, Name, Marks, etc can be added and if one wants to give some bonus marks to students then they can also be added. After building the contract all the details of every student can be retrieved.

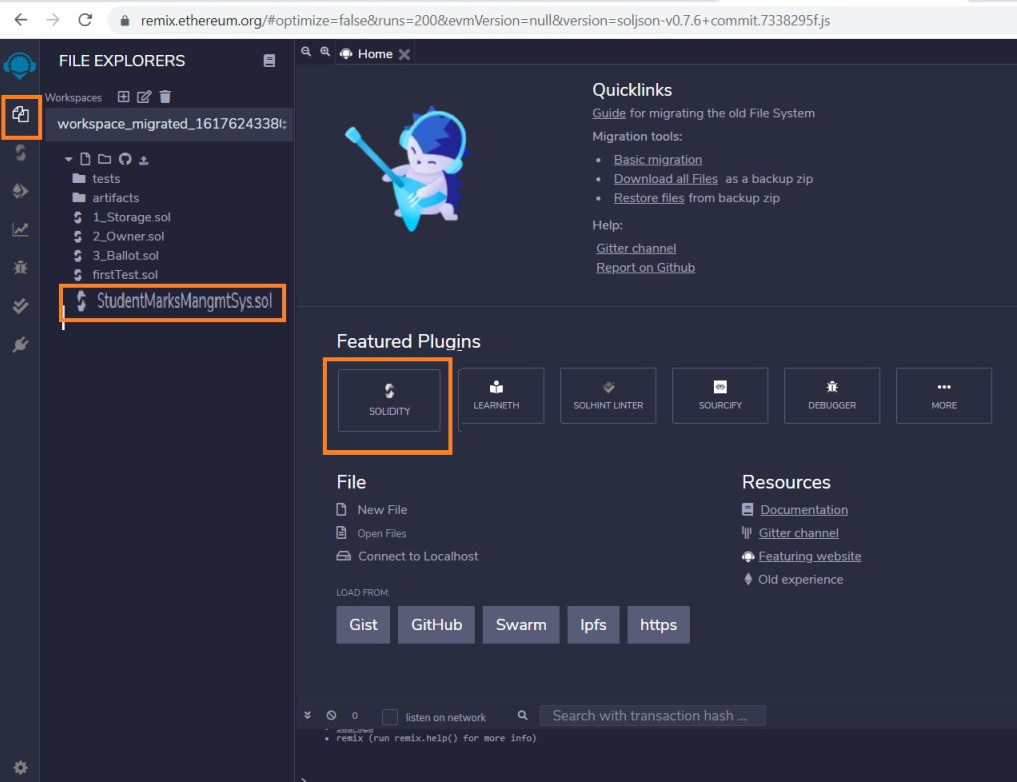
Approach:

1. The first step is to deploy the smart contract using the Remix IDE. After writing the code compile the code. When it is successfully compiled then deploy it. After deploying the contract a deployed Contract is obtained and then add the student details one by one.
2. If bonus marks need to be added then add in the bonusMarks section after that click on stdCount and fetch the student details to call the stdRecords.
3. Add one or more new student details in this Smart Contract by the increment of stdCount.

* **Implementation:**

Step 1: Open Remix IDE.

Step 2: Click on File Explorers and select Solidity in the environment and create a new file StudentMarksMangmtSys.sol by clicking on New File section.



Step 3: Build a smart contract that contains all the details of the student with the help of Remix IDE by clicking on the file name.

// Solidity program to implement

// the above approach

pragma solidity >= 0.7.0<0.8.0;

// Build the Contract

contract MarksManagmtSys

{

// Create a structure for

// student details

struct Student

{

int ID;

string fName;

string lName;

int marks;

}

address owner;

int public stdCount = 0;

mapping(int => Student) public stdRecords;

modifier onlyOwner

{

require(owner == msg.sender);

\_;

}

constructor()

{

owner=msg.sender;

}

// Create a function to add

// the new records

function addNewRecords(int \_ID,

string memory \_fName,

string memory \_lName,

int \_marks) public onlyOwner

{

// Increase the count by 1

stdCount = stdCount + 1;

// Fetch the student details

// with the help of stdCount

stdRecords[stdCount] = Student(\_ID, \_fName,

\_lName, \_marks);

}

// Create a function to add bonus marks

function bonusMarks(int \_bonus) public onlyOwner

{

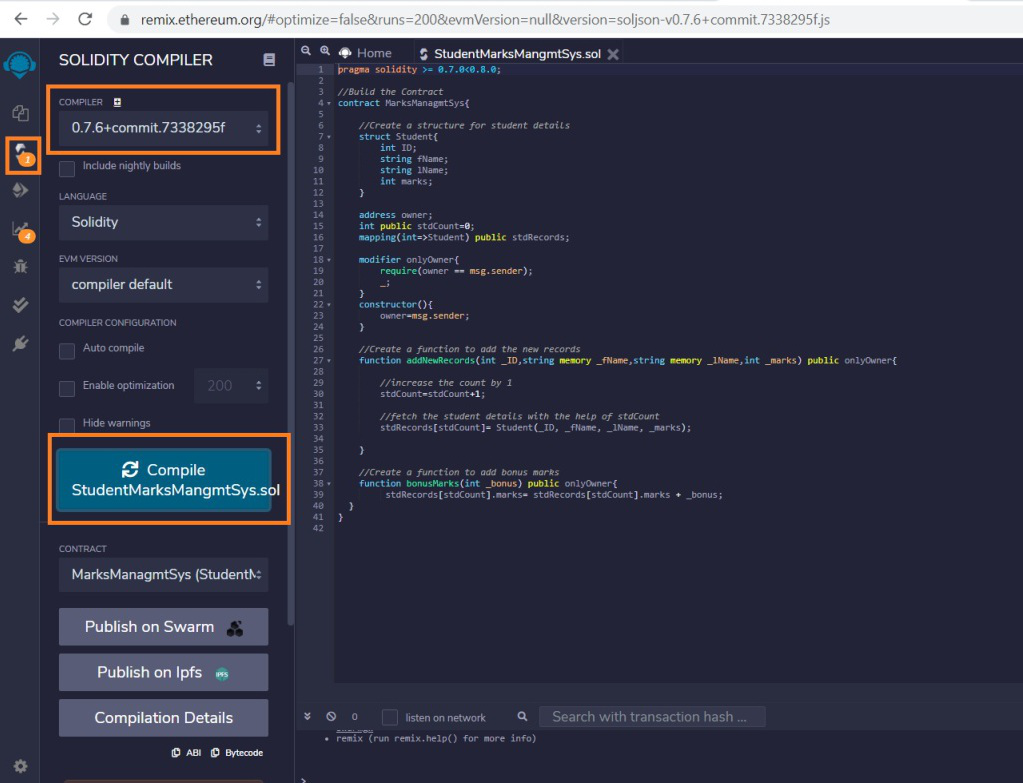
stdRecords[stdCount].marks =

stdRecords[stdCount].marks + \_bonus;

}

}

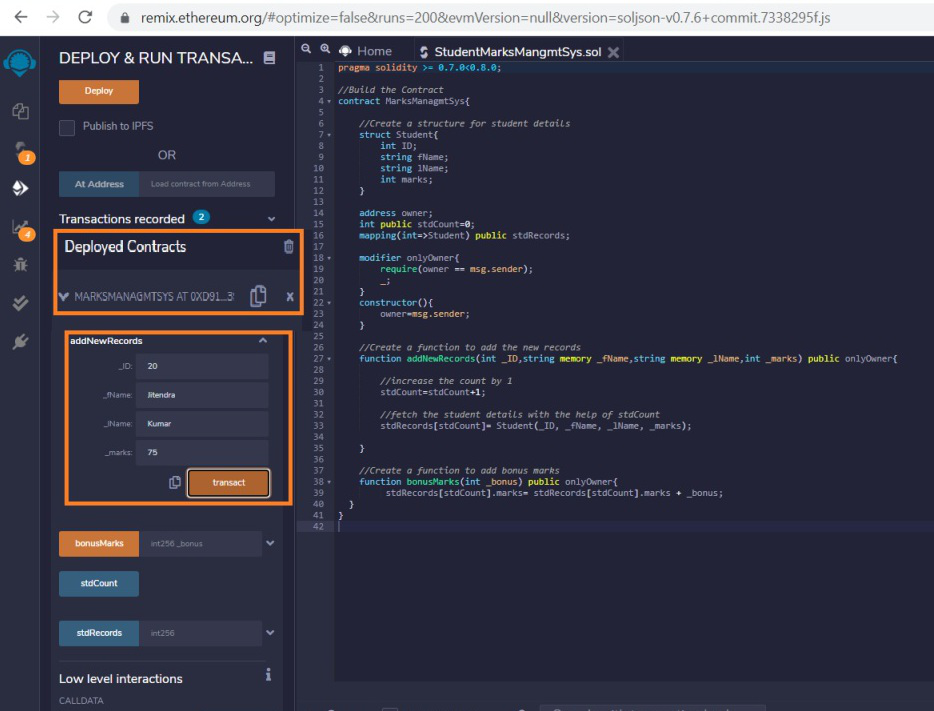
Step 4: After building the contract compile it. Select the compiler version before clicking on Compile button.



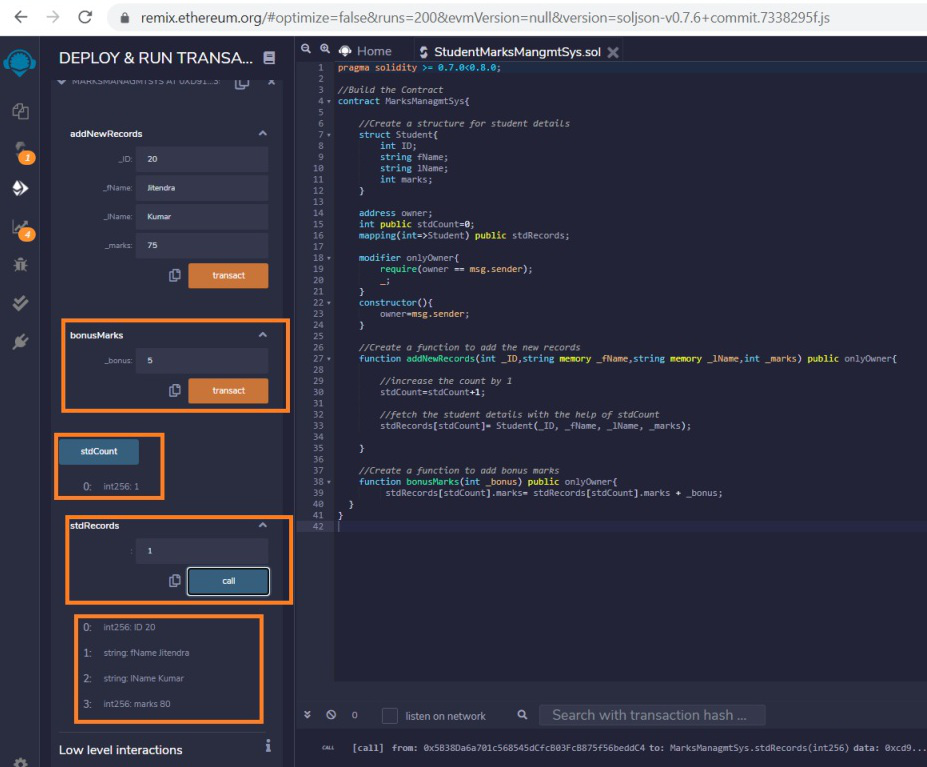
Step 5: After successful compilation, to deploy the contract, select the Environment JavaScript VM (Berlin) before clicking on the Deploy button.



Step 6: If the contract is successfully deployed then deployed contract is obtained. Open the deployed contract and add the student details and transact it.



Step 7: Add the bonus marks if you want to give them to the student and transact it after that click on the stdCount. One can see the student details after calling the stdRecords by entering the stdCount.



**Assignment No. C4**

* **Title:** Write a program in solidity to create Student data. Use the following constructs:
* Structures
* Arrays
* Fallback

Deploy this as smart contract on Ethereum and Observe the transaction fee and Gas values.

* **Objective:**
* **Theory :**

**What is Ethereum?**

Ethereum is a decentralized blockchain designed to be highly secure, fault-tolerant, and programmable.

Ethereum blockchain is a choice for many developers and businesses. As said programmable, the main

task of Ethereum is to securely execute and verify the application code known as smart contracts.

Ethereum helps to build native scripting language(solidity) and EVM. Ethereum consensus mechanism

is proof of work to operate to verify the new transaction. Now we will learn about smart contracts and

how it runs on the Ethereum platform.

**Overview of Smart Contracts**

A smart contract is a small program that runs on an Ethereum blockchain. Once the smart contract

is deployed on the Ethereum blockchain, it cannot be changed. To deploy the smart contract to Ethereum,

you must pay the ether (ETH) cost. Understand it as a digital agreement that builds trust and allows both

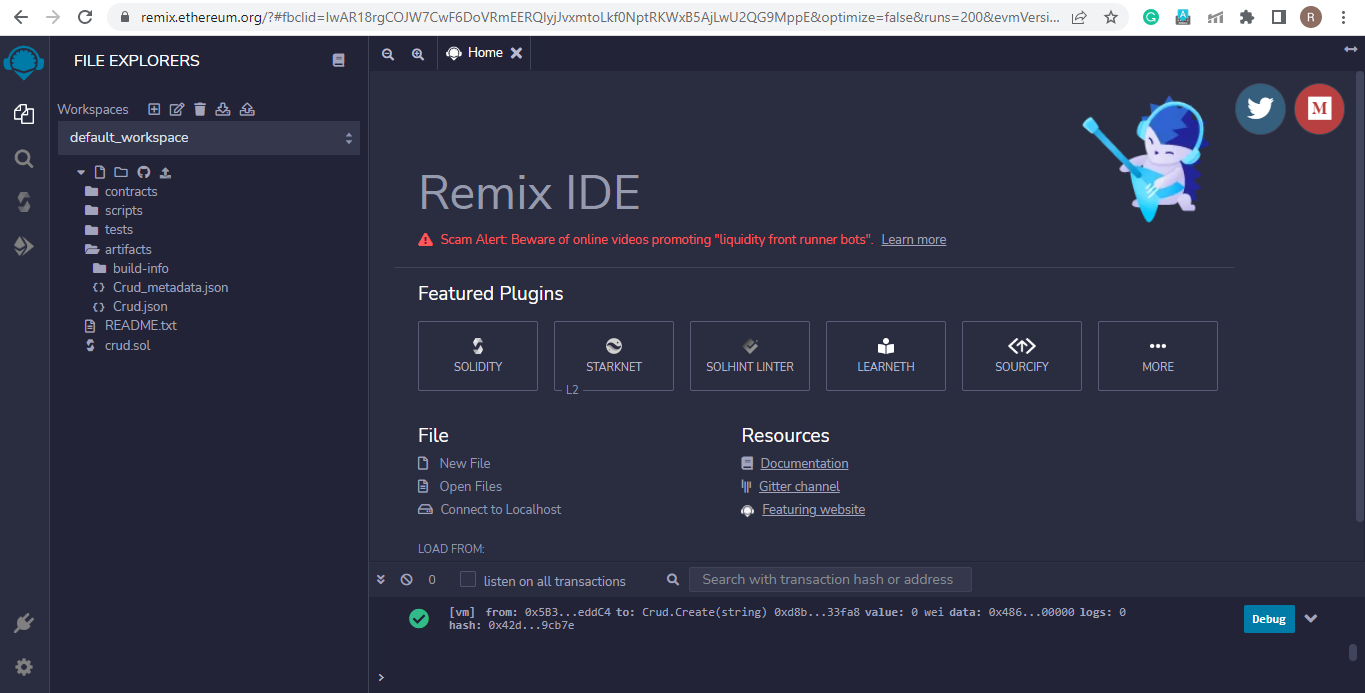
parties to agree on a particular set of conditions that cannot be tampered with.

## **Introduction to Solidity Programming**

Solidity is object-oriented, high-level statically-typed programing language used to create smart contracts. Solidity programming looks similar to Javascript, but there are a lot of differences between both languages. In solidity, you need to compile the program first, while in Javascript, you can run the program directly in your browser or by using Node JS. With solidity, you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets. It is also a case-sensitive programming language. Visit the official solidity documentation to read more and be updated about any new functionality release.

## **What is Remix IDE?**

It is an online IDE for creating solid, smart contracts, so you do not need to install or download anything to do any setup. You can develop, deploy, and Administer your solidity smart contract using Remix IDE. Visit [this](https://remix.ethereum.org/?#fbclid=IwAR18rgCOJW7CwF6DoVRmEERQIyjJvxmtoLkf0NptRKWxB5AjLwU2QG9MppE) link to access the Remix IDE where you will find multiple options and a window shown below. The window is a little bit similar to VS code where on the left-hand side, you will find some icons to terminate to other options like a compiler, file explorer, search files, deploy, etc.



Using the file explorer, you can create and open any file. We can set the compiler version, run our smart contract, and observe the output using the compiler. Each compiler type provides a different amount of fake ethers used for practicing purposes.

**Solidity Compilation Process**

Smart contract compilation is a critical process to understand how a smart contract runs when created using solidity. We will understand the process using the below flow chart.

We can see that the smart contract written in solidity with sol extension first gets the compiler version. After it goes under the compiler, It gets split into two parts where one is Byte code, and the other is ABI (Abstract Binary Interface) key. Byte code is only executed and deployed on the Ethereum blockchain, not the complete smart contract. Whenever any smart contract wants to communicate with this smart contract, they need the ABI key to call functions and variables.

To observe how ABI and Byte code is generated on Remix IDE, visit IDE, open any contract in the contracts folder, and compile and run it. Scroll down, and you will find two options: ABI and Byte code, where you can copy and paste them into any notepad and observe how your code gets converted to Byte code.

To create a smart contract, the first thing is to define the compiler version to use using the Pragma keyword (you can also determine whether the program supports multiple versions or the version in a particular range); after this, you define the contract using the contract keyword which is same as creating a class in object-oriented programming.

**Important points related to smart contract Compilation**

1. Contract Bytecode is public in readable form – It means It does not get encrypted because It will run on different nodes of Ethereum. For then, It needs to decrypt again and again not to increase computation time. It is kept in a readable form.
2. The contract doesn’t have to be public – It does not need to keep contracts public, but most organizations keep them public to maintain the trust.
3. Bytecode is immutable
4. ABI act as a bridge between application and smart contract
5. ABI and bytecode cannot be generated without source code

**State and Local Variables in Solidity**

Any variable declared on the contract level is known as a state variable. The critical property of the state variable is that it is permanently stored in the blockchain, so you have to pay some amount of gas and use the state variable with care. Solidity does not have a concept of Null or None; indeed, each data type has one default value which on declaration is assigned to that variable. To define Public before any variable or function, automatically, one get function is set with that variable, and you can access its value. Storage to state variable is not dynamically allocated (To initialize state variable with the value, you need to assign a value at declaration time, use constructor, use getter and setter functions). An instance of a contract variable cannot have another state variable besides those already declared.

Local variables are those variables that are declared in the function body and are stored in a stack, not in contract storage. Local variables don’t cost gas; some types reference the storage by default. Memory keywords cannot be used at the contract level.

**Functions in Solidity**

Functions are an essential part of any programming language for the reusability of a particular code. We will see the getter and setter function in solidity to learn how to create a function in solidity. The getter function is a function from which we can access the value of our variables. It is a view-only function, so we can define it as a view or Pure, which states that the value of the state variable cannot be changed it returns the variable’s value, so we define the return type of value. On the other side, the setter function changes the value, so it is a simple public function.

pragma solidity >= 0.5.0 < 0.9.0;

contract local {

uint age = 10;

function getter() public view returns(uint) {

return age;

}

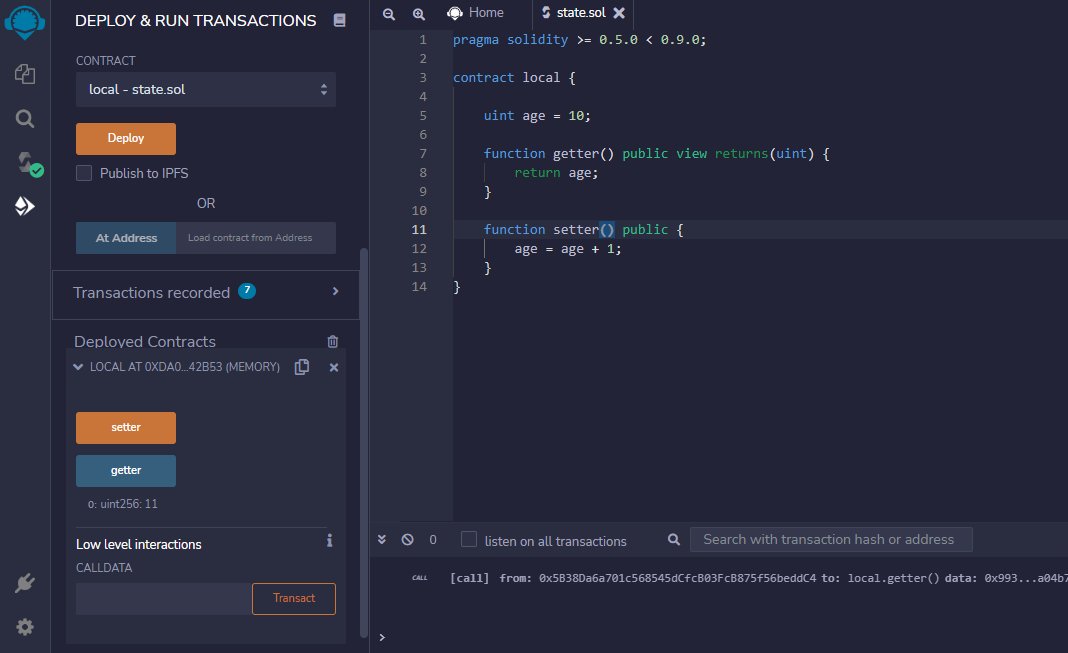
function setter() public {

age = age + 1;

}

}

After writing the above code on Remix IDE in a new file with sol extension, you can compile the code, visit the deploy section, and deploy the code to observe the deploy section output as shown below. The value will increase as you click the setter and getter function buttons.



Suppose you want to implement the setter function to set the new value of age so we can pass the parameter in the setter function and set the value of age. Thus in the setter function, we change the matter, so we need to pay a certain amount of gas in the setter function, but the getter function is view-only and does not require any amount of gas to be paid. By default, the visibility of a function is private, so to make it public, we define a function as Public.

**Pure and View in Solidity**

We have seen that we use to view and pure where we are not updating the state variable. But pure, you cannot use where it is also reading the state variable. Pure is used where both reading and writing are not performed. Indeed in View, reading is allowed, but writing is not permitted. When we do not define any one of the following to a function, it simply warns that we can provide one restriction of pure or View to function.

Constructor in Solidity

A constructor is a particular type of function which executes only once when you create your contract. Constructor is used to work with state variables and define smart contracts’ owners. You can create only one constructor, and it is optional to create. The compiler creates a default constructor if there is no explicitly defined constructor.

pragma solidity >= 0.5.0 < 0.9.0;

contract local {

uint public count;

constructor(uint new\_count) {

count = new\_count;

}

}

In the above code, we have created a constructor, and before clicking on deploy, you need to define the value of the constructor before it is called only once, so enter the value and click on deploy, and on scrolling, you can observe the value of count.

**Control Statements in Solidity**

All programming languages have control statements that help us check multiple conditions using loops and an if-else ladder, and solidity also supports loops and if-else statements.

**Loops in Solidity**

Solidity also supports three loops: a while loop, a Do while loop, and a loop. If you are familiar with any other programming language, you must know about control statements and using loops to run particular code multiple times with different values. In solidity, you cannot write the loops directly in contract storage; instead, you must declare them in any function.

* While the loop runs a code snippet multiple times until the condition is proper, the loop terminates when the condition is false. In contrast, the loop runs 0 or multiple times.
* Do while loop is a loop that runs even one time when the condition in the while loop is false. So it is used when you need to run a particular code at least once and if certain conditions meet, then run it multiple times.
* For loop is a loop that is used when you know the start and end time of the loop and how many intervals you need to take. For loop, the initialization and iterator updating are part of loop syntax.

So let us look after the syntax of each type of loop using a sample program.

pragma solidity >= 0.5.0 < 0.9.0;

contract Loops {

uint [3] public arr;

uint public count;

function Whileloop() public {

while(count < arr.length) {

arr[count] = count;

count++;

}

}

function Forloop() public {

for(uint i=count; i<arr.length; i++) {

arr[count] = count;

count++;

}

}

function doWhileLoop() public {

do {

arr[count] = count;

count++;

}while(count < arr.length);

}

}

I

**If-else Statements in Solidity**

If-else statements are an essential part of any programming language that helps compare two or more two types of values to make a particular decision. Below is the sample code snippet denoting the use of if-else in the solidity that you should try and deploy the contract. After deploying, check by entering the different values.

pragma solidity >= 0.5.0 < 0.9.0;

contract Array {

function check(int a) public pure returns(string memory) {

string memory value;

if(a > 0) {

value = "Greater Than zero";

}

else if(a == 0) {

value = "Equal to zero";

}

else {

value = "Less than zero";

}

return value;

}

}

**Arrays in Solidity**

The array is a special data structure used to create a list of similar type values. The array can be of fixed size and dynamic-sized. With the help of index elements can be accessed easily. below is a sample code to create, and access a fixed-sized array in solidity.

pragma solidity >= 0.5.0 < 0.9.0;

contract Array {

uint [4] public arr = [10, 20, 30, 40];

function setter(uint index, uint value) public {

arr[index] = value;

}

function length() public view returns(uint) {

return arr.length;

}

}

You can compile and deploy the code to try changing the array elements with an index and printing the array length.

**Creating Dynamic Array**

A dynamic array is an array where we can insert any number of elements and delete the details easily using an index. So solidity has functions like push and pops like python, making it easy to create a dynamic array. Below is a code using which you can create a dynamic array. After writing code, compiles and deploy the code by visiting the deploy section in the left-side navigation bar. After that, try inserting and deleting some elements from an array.

pragma solidity >= 0.5.0 < 0.9.0;

contract Array {

uint [] public arr;

function PushElement(uint item) public {

arr.push(item);

}

function Length() public view returns(uint) {

return arr.length;

}

function PopElement() public {

arr.pop();

}

}

**Structure in Solidity**

The structure is a user-defined data type that stores more than one data member of different data types. As in array, we can only store elements of the same data type, but in structure, you can keep elements of different data types used to create multiple collections. The structure can be made outside and inside the contract storage, and the Structure keyword can be used to declare the form. The structure is storage type, meaning we use it in-store only, and if we want to use it in function, then we need to use the memory keyword as we do in the case of a string.

pragma solidity >= 0.5.0 < 0.9.0;

struct Student {

uint rollNo;

string name;

}

contract Demo {

Student public s1;

constructor(uint \_rollNo, string memory \_name) {

s1.rollNo = \_rollNo;

s1.name = \_name;

}

// to change the value we have to implement a setter function

function changeValue(uint \_rollNo, string memory \_name) public {

Student memory new\_student = Student( {

rollNo : \_rollNo,

name : \_name

});

s1 = new\_student;

}

}

**Create a Smart Contract with CRUD Functionality**

We have excellent theoretical and hands-on practical knowledge about solidity, and now you can create a primary smart contract like hello world, getter, and setter contracts. So it’s a great time to try making some functional smart contracts, and the best way to try all the things in one code is to create one program that performs all CRUD operations.

pragma solidity ^0.5.0;

contract Crud {

struct User {

uint id;

string name;

}

User[] public users;

uint public nextId = 0;

function Create(string memory name) public {

users.push(User(nextId, name));

nextId++;

}

function Read(uint id) view public returns(uint, string memory) {

for(uint i=0; i<users.length; i++) {

if(users[i].id == id) {

return(users[i].id, users[i].name);

}

}

}

function Update(uint id, string memory name) public {

for(uint i=0; i<users.length; i++) {

if(users[i].id == id) {

users[i].name =name;

}

}

}

function Delete(uint id) public {

delete users[id];

}

function find(uint id) view internal returns(uint) {

for(uint i=0; i< users.length; i++) {

if(users[i].id == id) {

return i;

}

}

// if user does not exist then revert back

revert("User does not exist");

}

}

**Conclusion:**

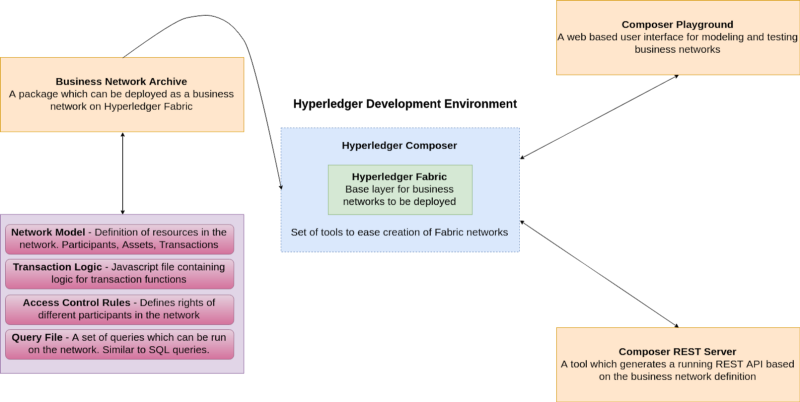
Solidity is an object-oriented high-level programming language for creating a smart contract that runs on the Ethereum blockchain. We have learned about the smart contract and its creation using solidity programming. Let us conclude the article with the main critical points through learning from an article.

1. The smart contract is a digital agreement containing some condition on which both signing parties agree, and when the state meets, the smart contract automatically gets terminated.
2. The smart contract is created using some programming language in which solidity is the leading choice and allows different people to collaborate, coordinate, and cooperate to make transactions that do not trust each other.
3. Solidity is the same as another programming language where creating a class is the same as creating a contract and differs in some exceptions like string should always be given with memory in function.
4. The syntax of solidity programming is 90 percent the same as javascript. Still, javascript directly runs on the browser while solidity runs on the Ethereum blockchain platform, which needs a certain amount of Ether (ETH) to deploy the contract and gas to create a transaction.
5. Blockchain technology today is booming, and many businesses across the globe are trying to acquire blockchain technology in many domains of finance, insurance, security, etc.

**Assignment No. C6**

* **Title:** Write a program to create a Business Network using Hyperledger
* **Objective:**
* **Theory :**

### Introduction to Hyperledger Fabric and related applications

Development environment overview for Hyperledger

**Hyperledger Fabric** is an open source framework for making private (permissioned) blockchain business networks, where identities and roles of members are known to other members. The network built on fabric serves as the back-end, with a client-side application front-end. SDK’s are available for Nodejs and Java to build client applications, with Python and Golang support coming soon.

**Hyperledger Composer** is a set of Javascript based tools and scripts which simplify the creation of Hyperledger Fabric networks. Using these tools, we can generate a **business network archive (BNA)** for our network. Composer broadly covers these components:

* Business Network Archive (BNA)
* Composer Playground
* Composer REST Server

**Business Network Archive** — Composer allows us to package a few different files and generate an archive which can then be deployed onto a Fabric network. To generate this archive, we need:

* **Network Model** — A definition of the resources present in the network. These resources include Assets, Participants, and Transactions. We will come back to these later.
* **Business Logic** — Logic for the transaction functions
* **Access Control Limitations —** Contains various rules which define the rights of different participants in the network. This includes, but is not limited to, defining what Assets the Participants can control.
* **Query File (optional) —** A set of queries which can be run on the network. These can be thought of as similar to SQL queries. You can read more on queries [here](https://hyperledger.github.io/composer/latest/reference/query-language).

**Composer Playground** is a web based user interface that we can use to model and test our business network. Playground is good for modelling simple Proofs of Concept, as it uses the browser’s local storage to simulate the blockchain network. However, if we are running a local Fabric runtime and have deployed a network to it, we can also access that using Playground. In this case, Playground isn’t simulating the network, it’s communicating with the local Fabric runtime directly.

**Composer REST Server** is a tool which allows us to generate a REST API server based on our business network definition. This API can be used by client applications and allows us to integrate non-blockchain applications in the network.

### Installing the prerequisites, tools, and a Fabric runtime

#### **1. Installing Prereqs**

Now that we have a high level understanding of what is needed to build these networks, we can start developing. Before we do that, though, we need to make sure we have the prerequisites installed on our system. An updated list can be found [here](https://hyperledger.github.io/composer/latest/installing/installing-prereqs.html).

* Docker Engine and Docker Compose
* Nodejs and NPM
* Git
* Python 2.7.x

For Ubuntu users, Hyperledger has a bash script available to make this process extremely easy. Run the following commands in your terminal:

Unfortunately, Mac users have to manually install the aforementioned tools and make sure they have all the prerequisites on their system. [This page](https://hyperledger.github.io/composer/latest/installing/installing-prereqs.html) is kept up to date with installation instructions.

#### 2. **Installing tools to ease development**

Run the following commands in your Terminal, and make sure you’re **NOT** using sudo when running npm commands.

**composer-cli** is the only essential package. The rest aren’t core components but will turn out to be extremely useful over time. We will learn more about what each of these do as we come across them.

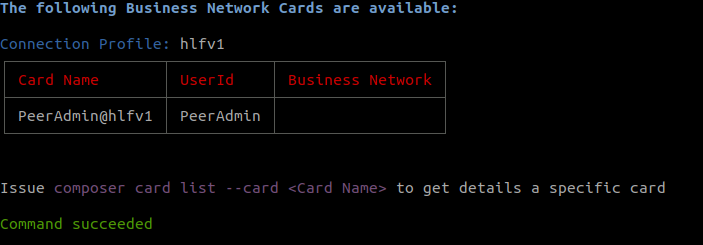
#### 3. **Installing a local Hyperledger Fabric runtime**

Let’s go through the commands and see what they mean. First, we make and enter a new directory. Then, we download and extract the tools required to install Hyperledger Fabric.

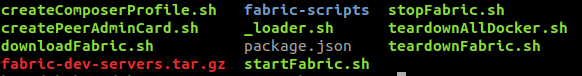
We then specify the version of Fabric we want, at the time of writing we need 1.2, hence **hlfv12**. Then, we download the fabric runtime and start it up.

Finally, we generate a PeerAdmin card. Participants in a Fabric network can have business network cards, analogous to real life business cards. As we mentioned before, Fabric is a base layer for private blockchains to build upon. The holder of the PeerAdmin business card has the authority to deploy, delete, and manage business networks on this Fabric runtime (aka YOU!)

If everything went well, you should see an output like this:



Also, if you type ls you’ll see this:



Basically what we did here was just download and start a local Fabric network. We can stop is using ./stopFabric.sh if we want to. At the end of our development session, we should run ./teardownFabric.sh

**NOTE:** This local runtime is **meant to be frequently started, stopped, and torn down** for development use. For a runtime with more persistent state, you’ll want to deploy the network outside the dev environment. You can do this by running the network on Kubernetes or on managed platforms like IBM Blockchain. Still, you should go through this tutorial first to get an idea.

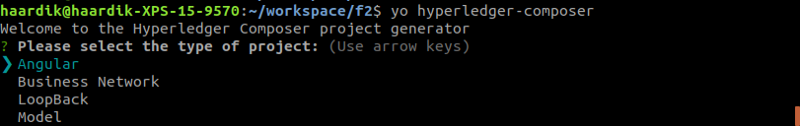
### Creating and deploying our business network

Remember the packages yo and generator-hyperledger-composer we installed earlier?

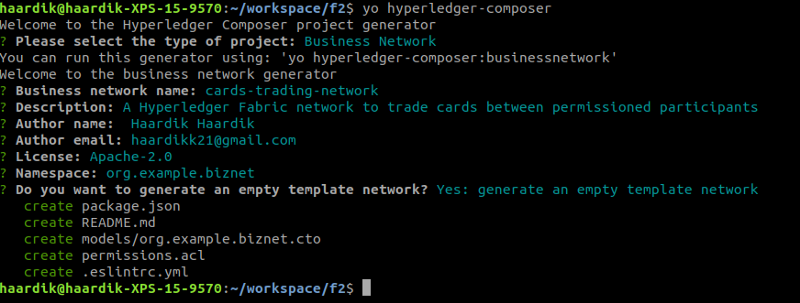
yo provides us a generator ecosystem where generators are plugins which can be run with the yo command. This is used to set up boilerplate sample applications for various projects. generator-hyperledger-composer is the Yo generator we will be using as it contains specs to generate boilerplate business networks among other things.

#### **1. Generating a business network**

Open terminal in a directory of choice and type yo hyperledger-composer



You’ll be greeted with something similar to the above. Select Business Network and name it cards-trading-network as shown below:



#### 2. **Modeling our business network**

The first and most important step towards making a business network is identifying the resources present. We have four resource types in the modeling language:

* Assets
* Participants
* Transactions
* Events

For our cards-trading-network , we will define an asset typeTradingCard , a participant type Trader , a transaction TradeCard and an event TradeNotification.

Go ahead and open the generated files in a code editor of choice. Open up org.example.biznet.cto which is the modeling file. Delete all the code present in it as we’re gonna rewrite it (except for the namespace declaration).

This contains the specification for our asset TradingCard . All assets and participants need to have a unique identifier for them which we specify in the code, and in our case, it’s cardId

Also, our asset has a GameType cardType property which is based off the enumerator defined below. Enums are used to specify a type which can have up to N possible values, but nothing else. In our example, no TradingCard can have a cardType other than Baseball, Football, or Cricket

Now, to specify our Trader participant resource type, add the following code in the modeling file

This is relatively simpler and quite easy to understand. We have a participant type Trader and they’re uniquely identified by their traderIds.

Now, we need to add a reference to our TradingCards to have a reference pointing to their owner so we know who the card belongs to. To do this, add the following line inside your TradingCard asset:

--> Trader owner

so that the code looks like this:

This is the first time we’ve used --&gt; and you must be wondering what this is. This is a relationship pointer. o and --> are how we differentiate between a resource’s own properties vs a relationship to another resource type. Since the owner is a Trader which is a participant in the network, we want a reference to that Trader directly, and that’s exactly what --> does.

Finally, go ahead and add this code in the modeling file which specifies what parameters will be required to make a transaction and emitting an event.

#### 3. **Adding logic for our transactions**

To add logic behind the TradeCard function, we need a Javascript logic file. Create a new directory named lib in your project’s folder and create a new file named logic.js with the following code:

**NOTE:** The decorator in the comments above the function is very important. Without the @param {org.example.biznet.TradingCard} trade , the function has no idea which Transaction the code refers to from the modeling language. Also, make sure the parameter name being passed (i.e. trade) is the one you’re passing along in the function definition right after.

This code basically checks if the specified card has forTrade == true and updates the card’s owner in that case. Then, it fires off the TradeNotification event for that card.

#### 4. **Defining permissions and access rules**

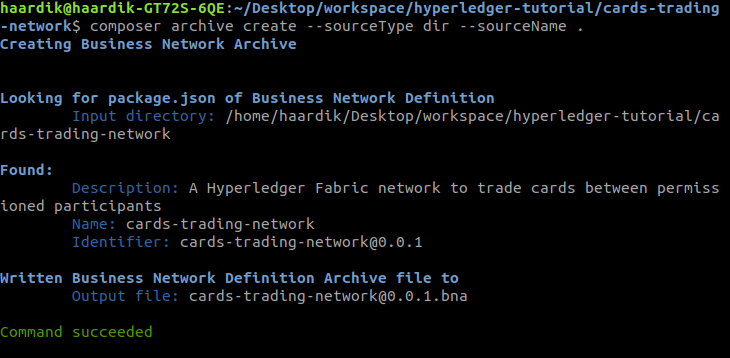
Add a new rule in permissions.acl to give participants access to their resources. In production, you would want to be more strict with these access rules. You can read more about them [here](https://hyperledger.github.io/composer/latest/reference/acl_language).

#### 5. **Generating a Business Network Archive (BNA)**

Now that all the coding is done, it’s time to make an archive file for our business network so we can deploy it on our local Fabric runtime. To do this, open Terminal in your project directory and type this:

composer archive create --sourceType dir --sourceName .

This command tells Hyperledger Composer we want to build a BNA from a directory which is our current root folder.

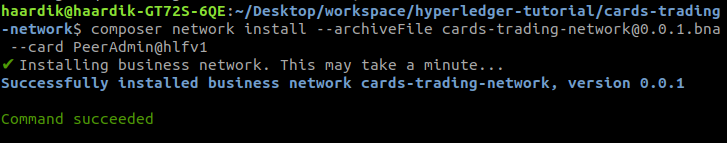


**NOTE:** The BNA name and version come from the package.json file. When you add more code, you should change the version number there to deploy unique archives capable of upgrading existing business networks.

#### 6. **Install and Deploy the BNA file**

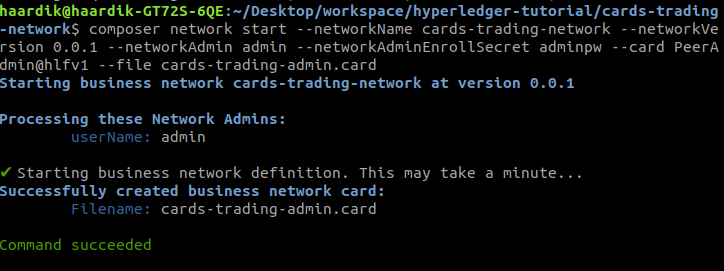
We can install and deploy the network to our local Fabric runtime using the PeerAdmin user. To install the business network, type

composer network install --archiveFile cards-trading-network@0.0.1.bna --card PeerAdmin@hlfv1



To deploy the business network, type

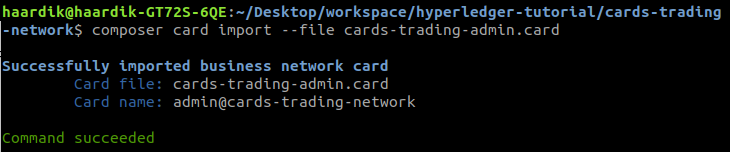
composer network start --networkName cards-trading-network --networkVersion 0.0.1 --networkAdmin admin --networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file cards-trading-admin.card



The networkName and networkVersion must be the same as specified in your package.json otherwise it won’t work.

--file takes the name of the file to be created for THIS network’s business card. This card then needs to be imported to be usable by typing

composer card import --file cards-trading-admin.card

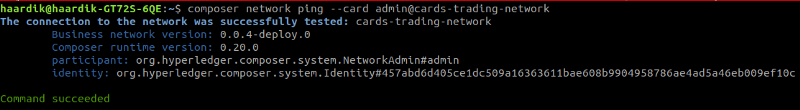


Amazing. We can now confirm that our network is up and running by typing

composer network ping --card admin@cards-trading-network

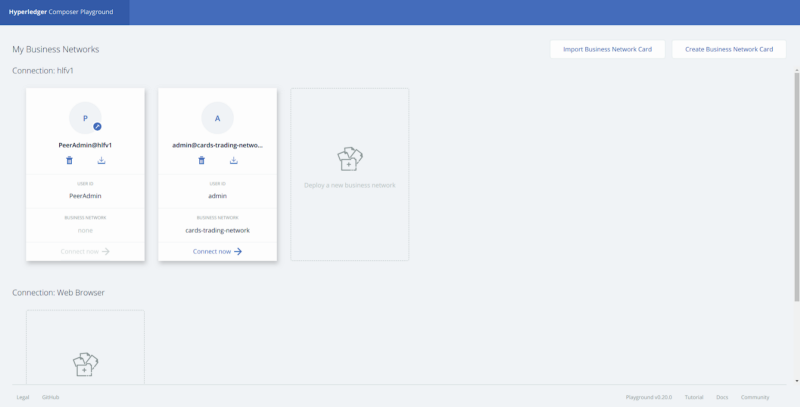
--card this time takes the admin card of the network we want to ping.

If everything went well, you should see something similar to this:

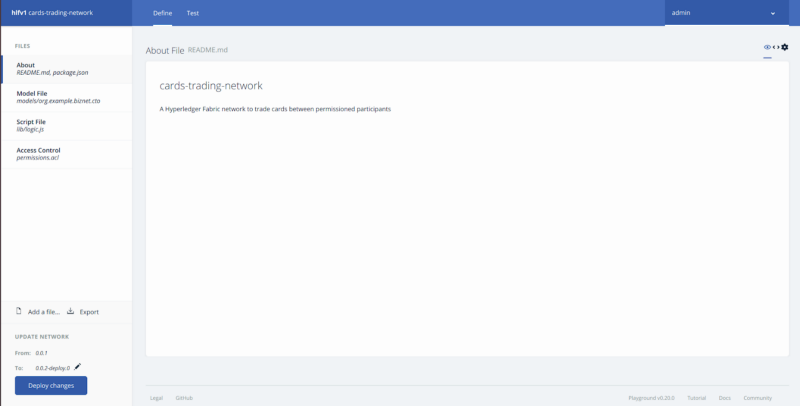
**Your network version will be 0.0.1 or whatever your package.json specifies —** I actually forgot to take this screenshot and uploaded it after I was done writing the tutorial and making edits

### Testing our Business Network

Now that our network is up and running on Fabric, we can start Composer Playground to interact with it. To do this, type composer-playground in Terminal and open up <http://localhost:8080/> in your browser and you should see something similar to this:

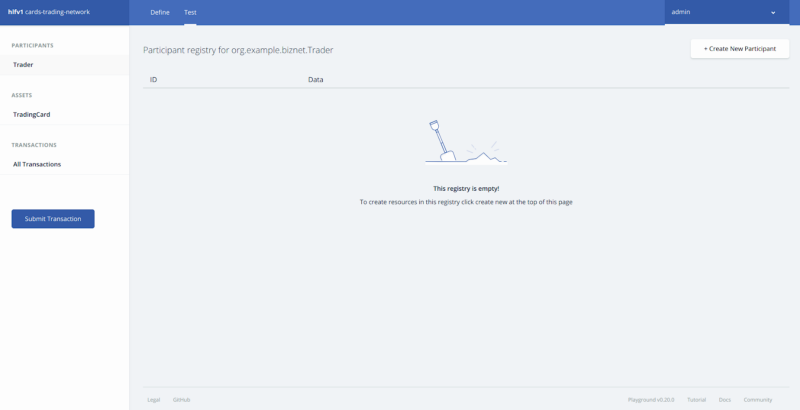


Press Connect Now for admin@cards-trading-network and you’ll be greeted with this screen:

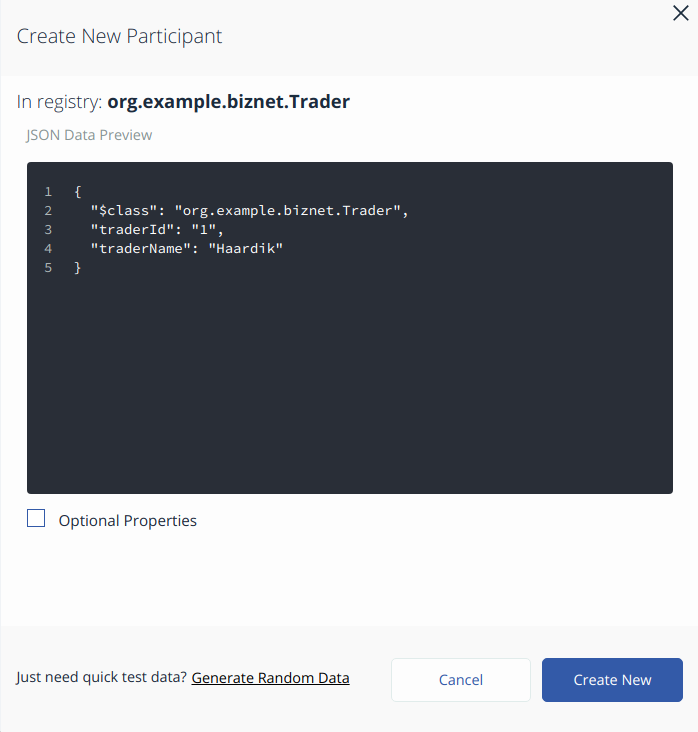


The **Define** page is where we can make changes to our code, deploy those changes to upgrade our network, and export business network archives.

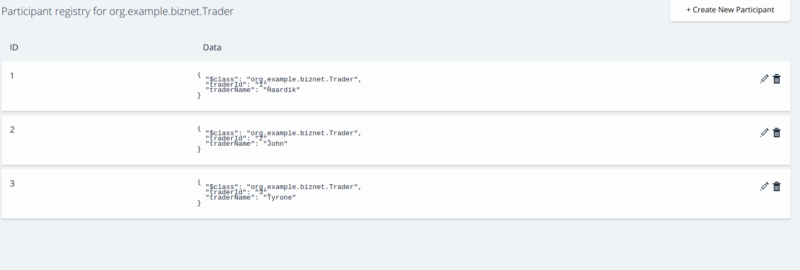
Head over to the **Test** page from the top menu, and you’ll see this:



Select Trader from Participants, click on **Create New Participant** near the top right, and make a new Trader similar to this:



Go ahead and make a couple more Traders. Here are what my three traders look like with the names Haardik, John, and Tyrone.

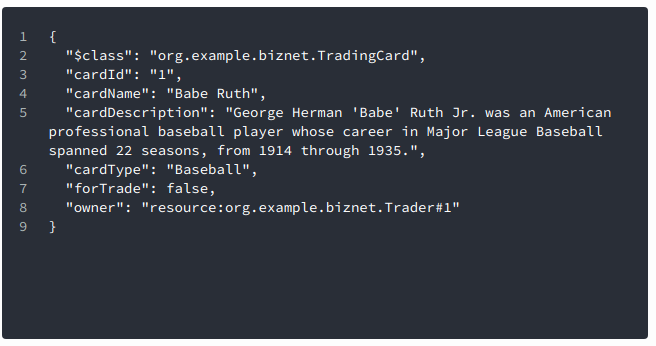


Now, let’s make some Assets. Click on TradingCard from the left menu and press **Create New Asset**. Notice how the owner field is particularly interesting here, looking something like this:

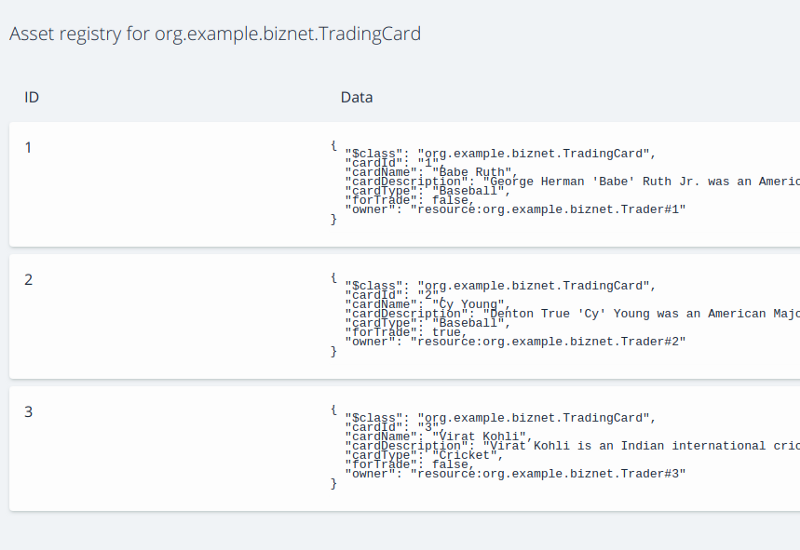
0*Q_ZmrJorW6RcUxUe

This is a relationship. This is what the --&gt; means. We specify the exact resource type followed by their unique identifier and voila, we have a relationship pointer.

Go ahead and finish making a TradingCard something similar to this:



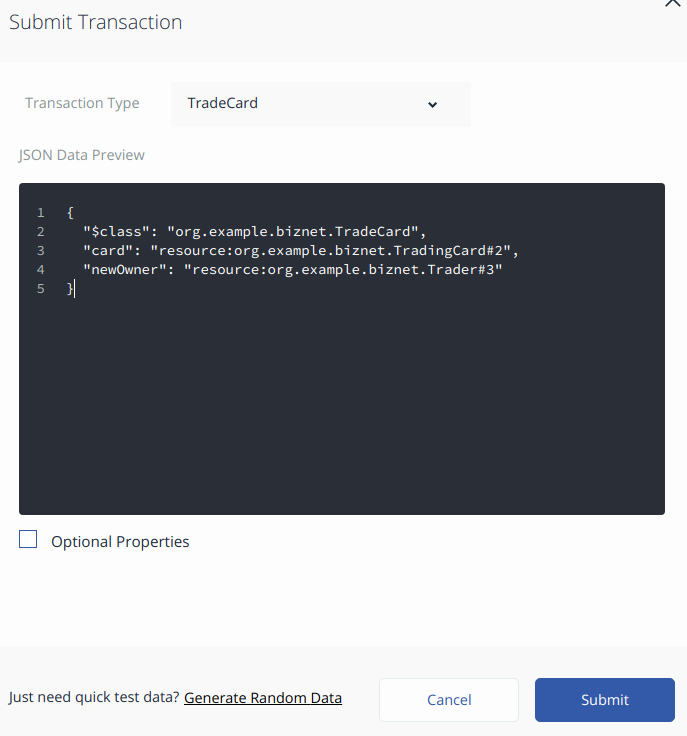
Notice how the owner fields points to Trader#1 aka Haardik for me. Go ahead and make a couple more cards, and enable a couple to have forTrade set to true.



Notice how my Card#2 has forTrade == true?

Now for the fun stuff, let’s try trading cards :D

Click on **Submit Transaction** in the left and make card point to TradingCard#2 and newOwner point to Trader#3 like this:



Press **Submit** and take a look at your TradingCards, you’ll see that Card#2 now has owner Trader#3 :D

### Generating a REST API Server

Doing transactions with Playground is nice, but not optimal. We have to make client-side software for users to provide them a seamless experience, they don’t even have to necessarily know about the underlying blockchain technology. To do so, we need a better way of interacting with our business network. Thankfully, we have the composer-rest-server module to help us with just that.

Type composer-rest-server in your terminal, specify admin@cards-trading-network , select **never use namespaces**, and continue with the default options for the rest as follows:

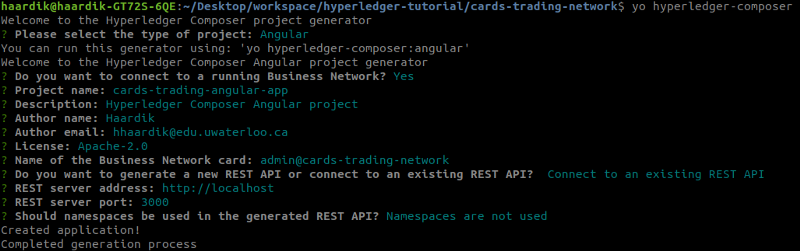


Open <http://localhost:3000/explorer/> and you’ll be greeted with a documented version of an automatically generated REST API :D

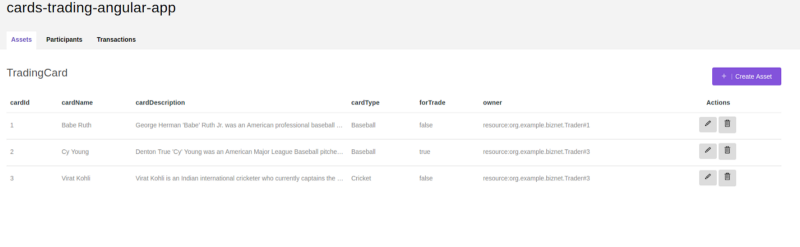
### Generating an Angular application which uses the REST API

Remember the yo hyperledger-composer generator? It can do more than generating a business network. It can also create an Angular 4 application running against the REST API we created above.

To create the Angular web application, type yo hyperledger-composer in your Terminal, select Angular, choose to connect to an existing business network with the card admin@cards-trading-network, and connect to an existing REST API as well. (**Edit:** Newer versions of the software may ask for the card file instead of just the name of the card)



This will go on to run npm install , give it a minute, and once it’s all done you’ll be able to load up <http://localhost:4200/> and be greeted with a page similar to this:  
**Edit:** Newer versions of the software may require you to run npm install yourself and then run npm start



You can now play with your network from this application directly, which communicates with the network through the REST server running on port 3000.

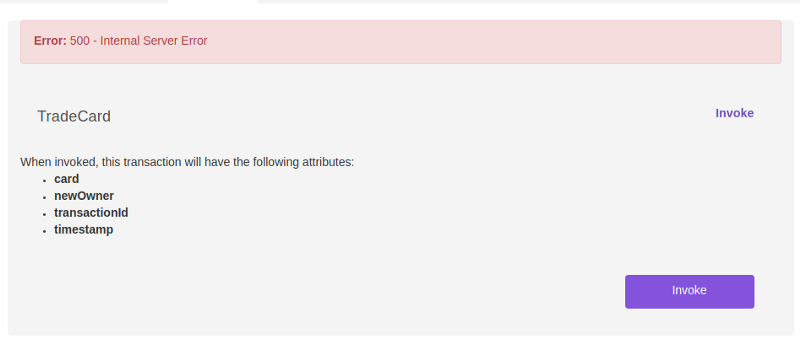
Congratulations! You just set up your first blockchain business network using Hyperledger Fabric and Hyperledger Composer :D

You can add more features to the cards trading network, setting prices on the cards and giving a balance to all Trader. You can also have more transactions which allow the Traders to toggle the value of forTrade . You can integrate this with non blockchain applications and allow users to buy new cards which get added to their account, which they can then further trade on the network.

The possibilities are endless, what will you make of them? Let me know in the comments :D

### KNOWN BUG: Does your Angular web app not handle Transactions properly?

At the time of writing, the angular generator has an issue where the purple Invoke button on the Transactions page doesn’t do anything. To fix this, we need to make a few changes to the generated angular app.



#### **1. Get a modal to open when you press the button**

The first change we need to make is have the button open the modal window. The code already contains the required modal window, the button is just missing the (click) and data-target attributes.

To resolve this, open up /cards-trading-angular-app/src/app/**TradeCard/TradeCard.component.html**

The file name can vary based on your transaction name. If you have multiple transactions in your business network, you’ll have to do this change across all the transaction resource type HTML files.

Scroll down till the very end and you shall see a <button> tag. Go ahead and add these two attributes to that tag:

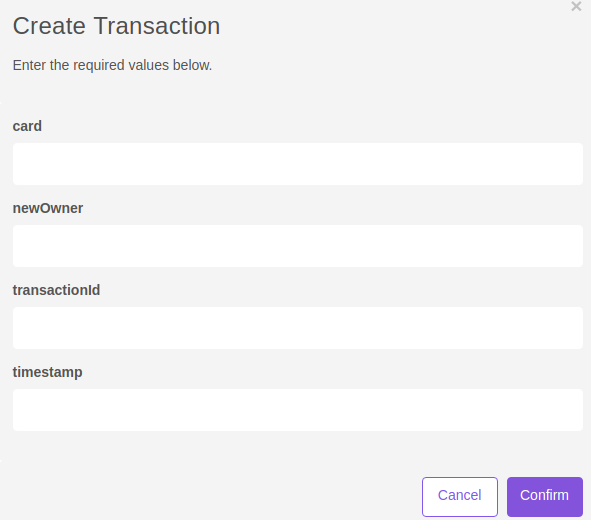
(click)="resetForm();" data-target="#addTransactionModal"

so the line looks like this:

<button type=”button” class=”btn btn-primary invokeTransactionBtn” data-toggle=”modal” (click)=”resetForm();” data-target=”#addTransactionModal”>Invoke<;/button>

The (click) attribute calls resetForm(); which sets all the input fields to empty, and data-target specifies the modal window to be opened upon click.

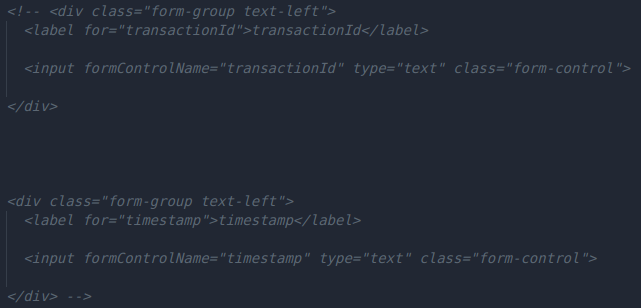
Save the file, open your browser, and try pressing the invoke button. It should open this modal:



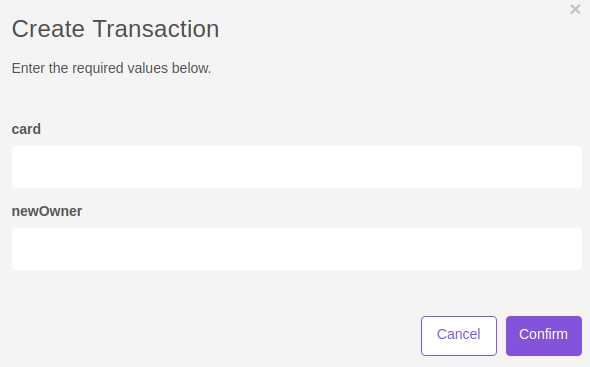
#### 2. **Removing unnecessary fields**

Just getting the modal to open isn’t enough. We can see it requests transactionId and timestamp from us even though we didn’t add those fields in our modeling file. Our network stores these values which are intrinsic to all transactions. So, it should be able to figure out these values on it’s own. And as it turns out, it actually does. These are spare fields and we can just comment them out, the REST API will handle the rest for us.

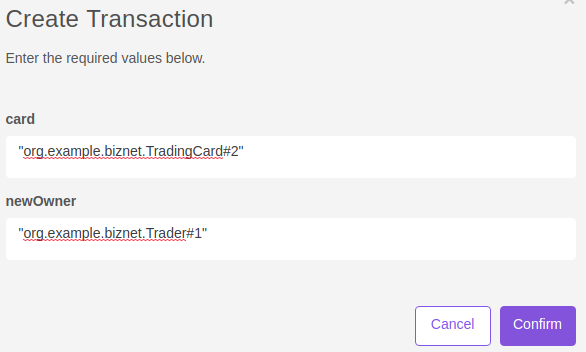
In the same file, scroll up to find the input fields and comment out the divs responsible for those input fields inside addTransactionModal



Save your file, open your browser, and press Invoke. You should see this:



You can now create transactions here by passing data in these fields. Since card and newOwner are relationships to other resources, we can do a transaction like this:



Press **Confirm**, go back to the **Assets** page, and you will see that TradingCard#2 now belongs to Trader#1:

0*teBdk1zFVR2hj2Tx