



HOW TO CREATE **Ethereum dApps**

MODUL PRAKTIK DAPPS

SMART CONTRACT WEB3JS IPFS

Abstract

Modul ini merupakan modul praktik pada workshop Blockchain yang dilakukan di Universitas Islam Indonesia

PURWONO, S.KOM., M.KOM.
purwono@uhb.ac.id

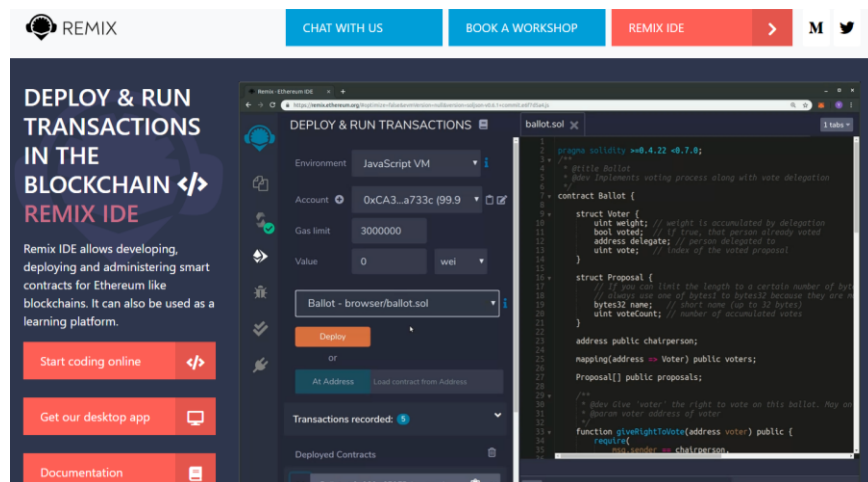
Contents

HELLO WORLD	1
LUAS PERSEGI PANJANG	6
INSTALASI METAMASK.....	7
Instal MetaMask	7
KONFIGURASI GANACE	14
MENGHUBUNGKAN METAMASK DAN GANACHE	16
MEMBUAT APLIKASI DAPPS EHR IPFS	20

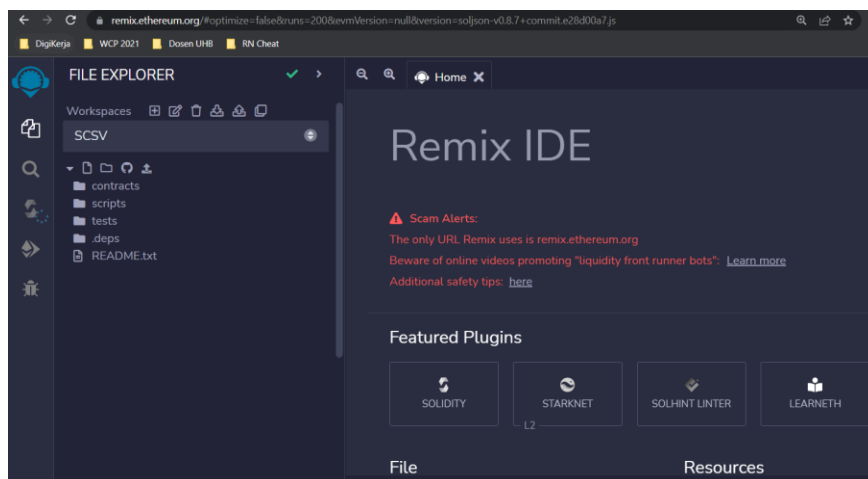
HELLO WORLD

Berikut ini adalah tahap dasar untuk membuat *smart contract HelloWorld* pada Remix IDE Ethereum. Terlebih dahulu kita mempersiapkan aplikasi browser yaitu Google Chrome. Silahkan ikuti beberapa tahap berikut ini:

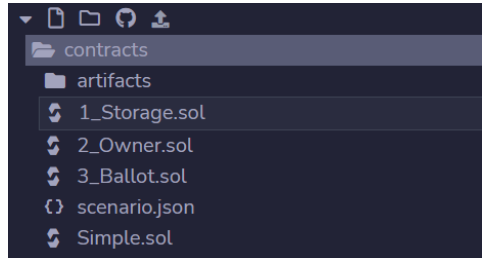
1. Bukalah aplikasi browser Chrome kita lalu ketikkan alamat berikut <https://remix-project.org/>
2. Anda saat ini masuk pada web remix, sebuah web yang dapat kita gunakan untuk membuat dan testing *smart contract*.
3. Smart contract dibuat dengan bahasa pemrograman *solidity*. Berikut ini adalah halaman awal remix IDE.



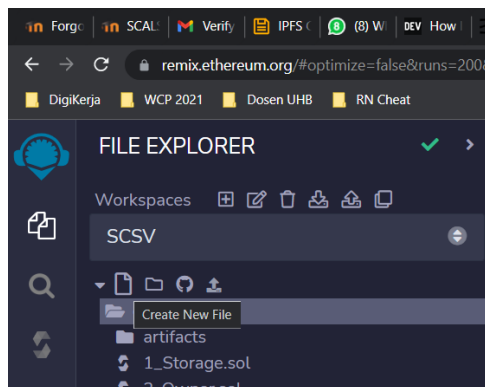
4. Pilih button berwarna merah pada pojok sebelah kanan, yaitu REMIX IDE kemudian klik untuk masuk pada editor remix.



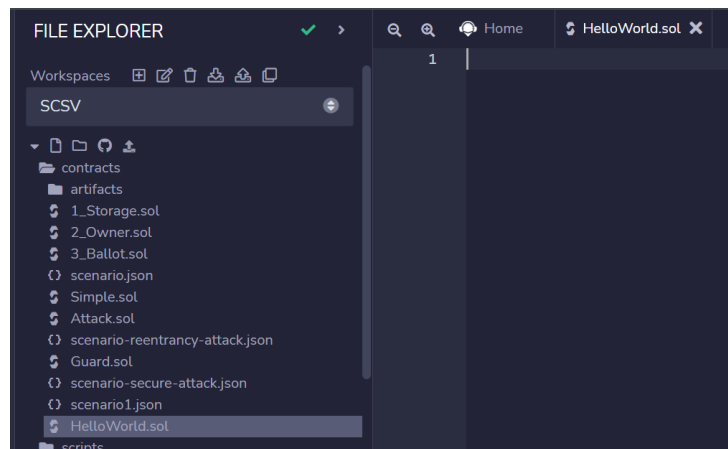
5. Dilihat dari struktur *file explorer*, kita bisa melihat beberapa folder yaitu *contracts*, *scripts*, *test*, *deps* dan file *Readme.txt*
6. Folder **contract** merupakan lokasi dimana kita akan membuat kontrak blockchain baru. Jika kita klik maka susuna folder kontrak adalah sebagai berikut.



7. Terlihat terdapat file-file ber ekstensi solidity yang bisa kita uji coba, namun kita hendak membuat file Hello World sebuah smart contract baru yang akan kita install.
8. Untuk membuat kontrak baru kita bisa pilih menu **create new file**



9. Buatlah nama file baru yaitu **HelloWorld.sol**
10. Secara otomatis, kode editor untuk kontrak tersebut terbuka.



11. Sekarang anda bisa memasukkan kode smart contract pada **HelloWorld.sol**.
12. Ketikan kode *smart contract* berikut

```
pragma solidity >=0.7.0 <0.9.0;

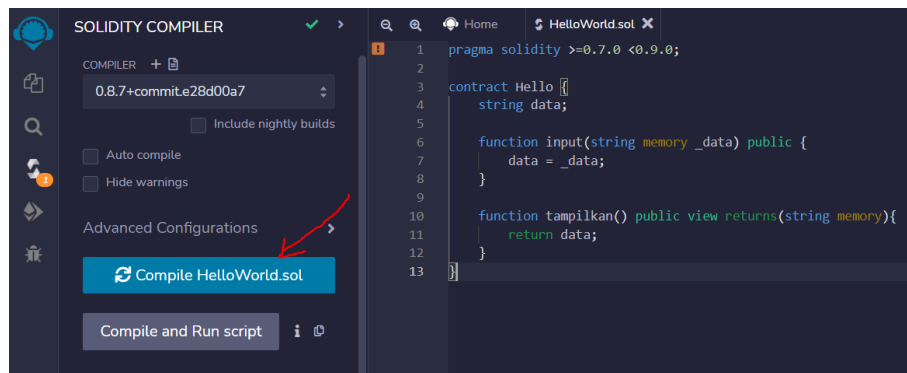
contract Hello {
    string data;

    function input(string memory _data) public {
        data = _data;
    }

    function tampilkan() public view returns(string memory){
        return data;
    }
}
```

Note: Source Budi Winarno

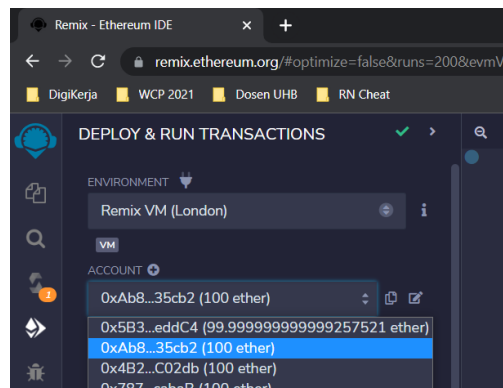
13. Selanjutnya pilih menu Compile **HelloWorld.sol**



14. Tunggu hingga proses *compile* selesai.

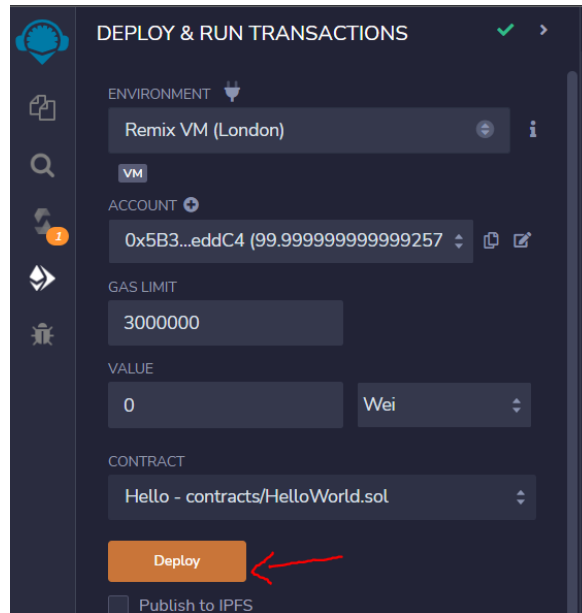
15. Pindah ke menu **Deploy & Run Transactions**

16. Pilih salah satu akun yang sudah diberikan saldo ether untuk testing pada **drop down account**.



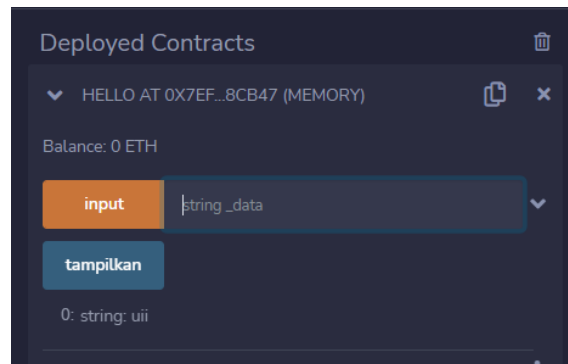
17. Akun test tersebut masing-masing diberikan saldo sebesar **100 ether** untuk kebutuhan uji coba atau simulasi.

18. Setelah dipilih salah satu akun ethereum tersebut, klik tombol **Deploy**



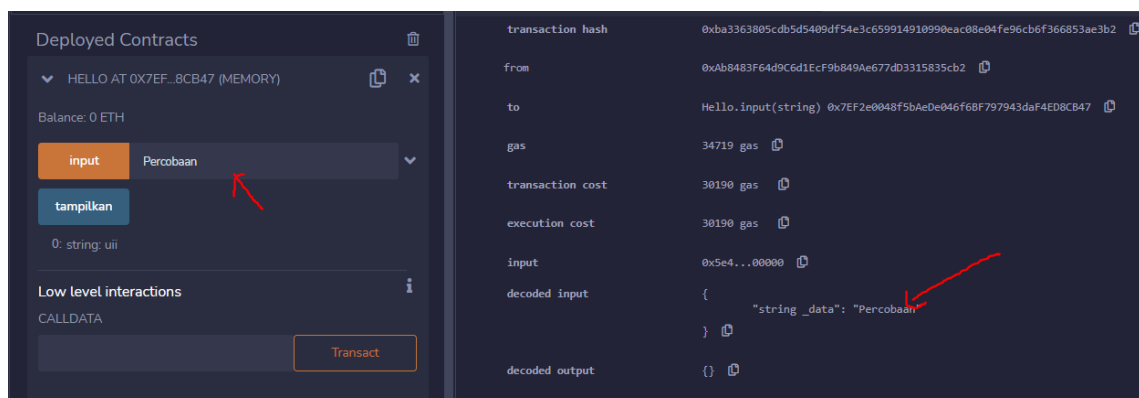
19. Tunggu hingga proses *deploy* selesai.

20. Jika *smart contract* berhasil di *deploy*, maka smart contract tersebut akan muncul di menu **Deployed Contract**

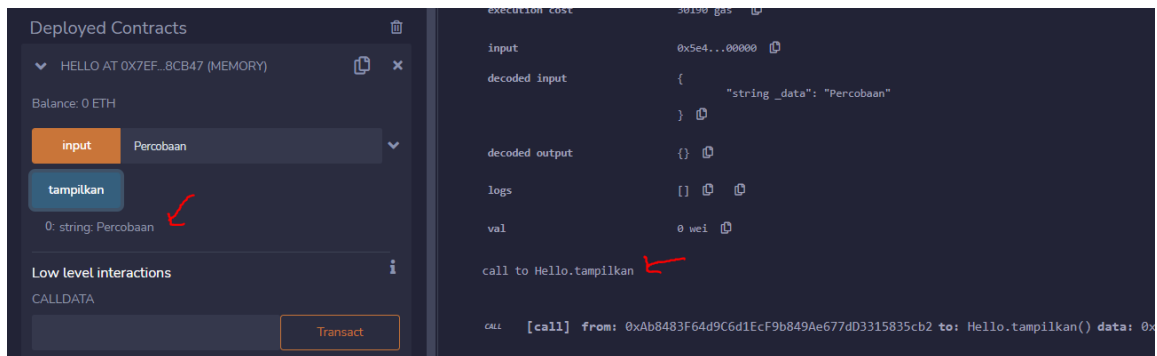


21. Kita bisa mencoba melakukan input pada kolom input yang berwarna orange, sebagai contoh kami masukan string **"Percobaan"**.

22. Kemudian jika dicek pada console maka akan terjadi proses transaksi.



23. Dapat dilihat bahwa ada proses decode input yang telah mengeluarkan *gas fee* pada blockchain ethereum.
24. Selanjutnya kita klik tombol “**tampilkan**” yang berwarna biru untuk menampilkan data yang sudah disimpan di blockchain.



25. Demikianlah percobaan sederhana menggunakan *smart contract* di **Remix**.

LUAS PERSEGI PANJANG

Setelah belajar membuat smart contract sederhana HelloWorld.sol kita akan mencoba membuat smart contract yang bisa menghitung luas segitiga. Pada kali ini berarti terdapat 3 variabel yang dapat digunakan yaitu variabel yang memuat panjang, lebar dan luas. Silahkan ketikkan kode solidity berikut.

```
pragma solidity >=0.7.0 <0.9.0;

contract Luas {

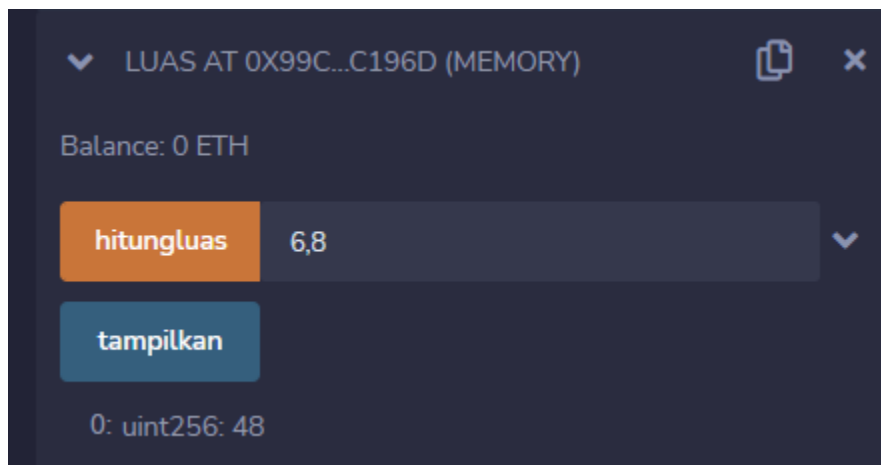
    uint256 luas;

    function hitungluas(uint256 panjang, uint256 lebar) public {
        luas = panjang * lebar;
    }

    function tampilkan() public view returns(uint256){
        return luas;
    }

}
```

Lakukan percobaan compile dan *deploy smart contract* seperti yang sudah dilakukan pada tahap sebelumnya. Jika berhasil maka anda akan melihat hasil sebagai berikut.



INSTALASI METAMASK

MetaMask adalah dompet berbasis ETH yang dapat dikelola dengan browser PC (Extension/Ads-on). Anda tidak hanya dapat mengelola token ETH yang sesuai dengan ERC20 – ERC721 – ERC223, tetapi banyak BCG (game blockchain) menggunakan MetaMask, bahkan termasuk yang akan kita bahas yakni BSC.

Mulai 13 Februari 2021, web browser berikut ini didukung oleh MetaMask.

- Google Chrome
- Firefox
- Brave
- Microsoft Edge
- KIWI BROWSER (Pengguna ANDROID)

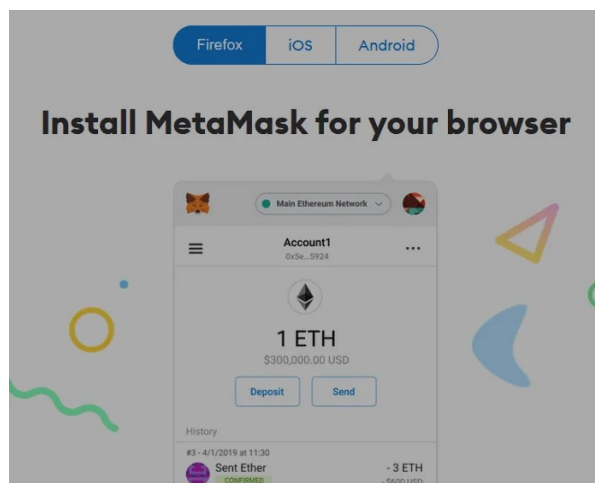
Anda dapat menggunakannya di “Edge” dan “Firefox” serta “Google Chrome” yang menempati sebagian besar pengguna internet di dunia.

Karena MetaMask memiliki dompet terpisah untuk setiap browser, dimungkinkan untuk membuat dompet yang berbeda untuk setiap browser. Anda dapat berbagi dompet yang sama dengan browser yang berbeda. maksudnya anda bisa menggunakannya secara multiple, yap yang anda butuhkan adalah MNEMONIC/PHRASE SEED atau bisa juga menggunakan PRIVATE KEY dari dompet anda, maka anda bisa meng-import nya dimana saja.

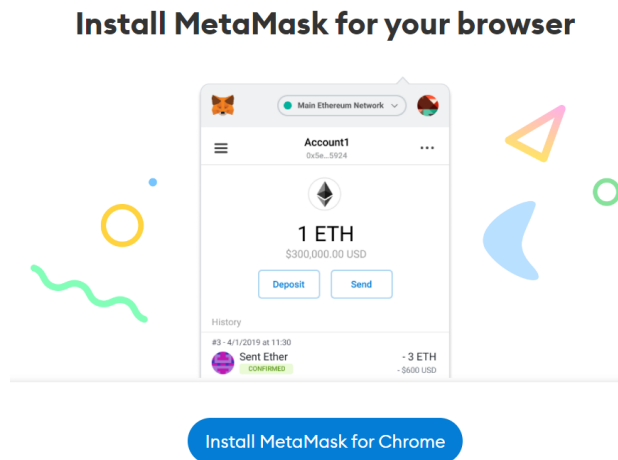
Misalnya, jika MetaMask di Chrome tidak berfungsi dengan baik, Anda dapat menggunakan MetaMask di Firefox untuk sementara. (Jika anda suka bermain game, AR, dan NFT saya sangat merekomendasikan menggunakan BRAVE browser).

Instal MetaMask

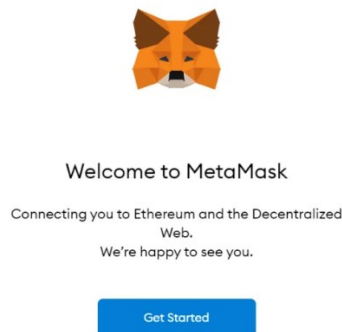
1. Pertama, buka web browser “Google Chrome” atau “Firefox” dan instal MetaMask. Link [Metamask](#)
2. Setelah mengakses halaman MetaMask dari link di atas, klik tombol **“Unduh sekarang”**. (Otomatis mendeteksi jenis browser dan jenis operasi system).



3. Kemudian ada tombol install yang sesuai dengan browser, jadi klik **“Install MetaMask for Chrome”**.
4. Halaman toko web akan terbuka, klik tombol **“Tambahkan ke Chrome”** untuk menambahkan ekstensi MetaMask ke browser Anda.




5. Setelah selesai anda akan dibawa ke extension manager *startup* seperti berikut, lalu klik **“Get Started”**.



6. Klik Mulai dan Anda dapat memilih untuk membuat dompet baru atau mengimpor dompet Anda yang sudah ada ke MetaMask. (menggunakan MNEMONIC).
7. Kami ingin membuat dompet baru di sini, jadi klik **“(Create Wallet) / Buat Dompet”**.




New to MetaMask?



No, I already have a seed phrase

Import your existing wallet using a 12 word seed phrase

Import wallet




Yes, let's get set up!

This will create a new wallet and seed phrase

Create a Wallet

8. Klik Create Wallet



Help Us Improve MetaMask

MetaMask would like to gather usage data to better understand how our users interact with the extension. This data will be used to continually improve the usability and user experience of our product and the Ethereum ecosystem.

MetaMask will..

- ✓ Always allow you to opt-out via Settings
- ✓ Send anonymized click & pageview events

- ✗ **Never** collect keys, addresses, transactions, balances, hashes, or any personal information
- ✗ **Never** collect your full IP address
- ✗ **Never** sell data for profit. Ever!

No Thanks

I Agree

This data is aggregated and is therefore anonymous for the purposes of General Data Protection Regulation (EU) 2016/679. For more information in relation to our privacy practices, please see our [Privacy Policy](#) here.

9. Lalu klik "I AGREE"

10. Kemudian Buatlah password / kata sandi



< Back

Create Password


New password (min 8 chars)

Confirm password

☐ I have read and agree to the [Terms of Use](#)

Create

11. Buatlah **Password / Kata sandi**, lalu konfirmasi atau ulangi. (Ingat dan Simpan Baik Baik).



< Back

Create Password

New password (min 8 chars)


Confirm password

☒ I have read and agree to the [Terms of Use](#)

Create

12. Lalu Ceklis Kotak **“I Have read and agree to the Terms of Use”** , lalu klik **“CREATE”**.

13. Jika anda baru membuat wallet, klik reveal words untuk menampilkan Mnemonic. (Dan Jangan Lupa Disimpan)



Secret Backup Phrase

Your secret backup phrase makes it easy to back up and restore your account.

WARNING: Never disclose your backup phrase. Anyone with this phrase can take your Ether forever.

CLICK HERE TO REVEAL SECRET WORDS

Remind me later Next

Tips:

Store this phrase in a password manager like 1Password.

Write this phrase on a piece of paper and store in a secure location. If you want even more security, write it down on multiple pieces of paper and store each in 2 - 3 different locations.

Memorize this phrase.

Download this Secret Backup Phrase and keep it stored safely on an external encrypted hard drive or storage medium.

14. Setelah itu anda akan melihat ini, lalu klik gambar GEMBOK **“click here to reveal secret words”**. lalu copy itu, tulis di BUKU Tulis dan simpan baik baik. lalu klik **“NEXT”**.



Secret Backup Phrase

Your secret backup phrase makes it easy to back up and restore your account.

WARNING: Never disclose your backup phrase. Anyone with this phrase can take your Ether forever.

hurdle path engage tree crater
screen cloud tomato corn tilt mix
debris

[Remind me later](#)

[Next](#)

Tips:

Store this phrase in a password manager like 1Password.

Write this phrase on a piece of paper and store in a secure location. If you want even more security, write it down on multiple pieces of paper and store each in 2 - 3 different locations.

Memorize this phrase.

[Download this Secret Backup Phrase and keep it stored safely on an external encrypted hard drive or storage medium.](#)

Confirm your Secret Backup Phrase

Please select each phrase in order to make sure it is correct.

cloud

corn

crater

debris

engage

hurdle

mix

path

screen

tilt

tomato

tree

[Confirm](#)

15. contoh hasilnya seperti berikut

Confirm your Secret Backup Phrase

Please select each phrase in order to make sure it is correct.

hurdle

path

engage

tree

crater

screen

cloud

tomato

corn

tilt

mix

debris

cloud

corn

crater

debris

engage

hurdle

mix

path

screen

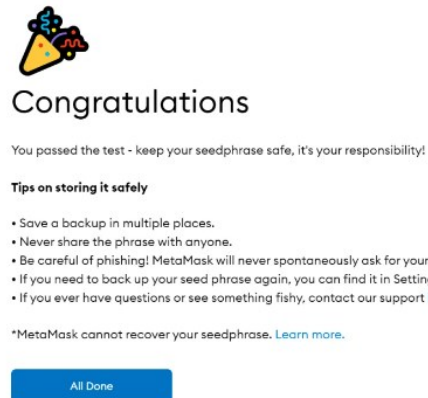
tilt

tomato

tree

[Confirm](#)

16. Urutkan dengan benar dan klik **“Confirm”**.



17. Lalu klik **“ALL DONE”**.

18. Jika ada notifikasi **“Token Swapping is Here”** silahkan klik tombol X / tutup saja notifikasinya.

Catatan: MNEMONIC , PHRASE , PRIVATE KEY , JSON , SEED, Dan SECRET KEY Adalah sesuatu yang sangat RAHASIA, sudah sepatutnya dijaga dengan baik karna itu seperti KODE BRANKAS DI BANK, saat anda kehilangan hal tersebut maka ASSET, TOKEN, ETH, BSC, NFT, Dan semua uang kalian akan hilang dan takan pernah bisa di recovery / dikembalikan.

oleh karna itu saran saya, buatlah salinan di banyak tempat. yang paling aman untuk menyimpannya adalah :

- BUKU TULIS (manual:offline)
- KEEP GOOGLE (aplikasi:online)
- FILE .TXT (dokumen di komputer/android:offline)
- CATATAN DRIVE (OneDrive/GoogleDrive:Online)
- PESAN TERSIMPAN (Cloud Telegram/Whatsapp:Online)

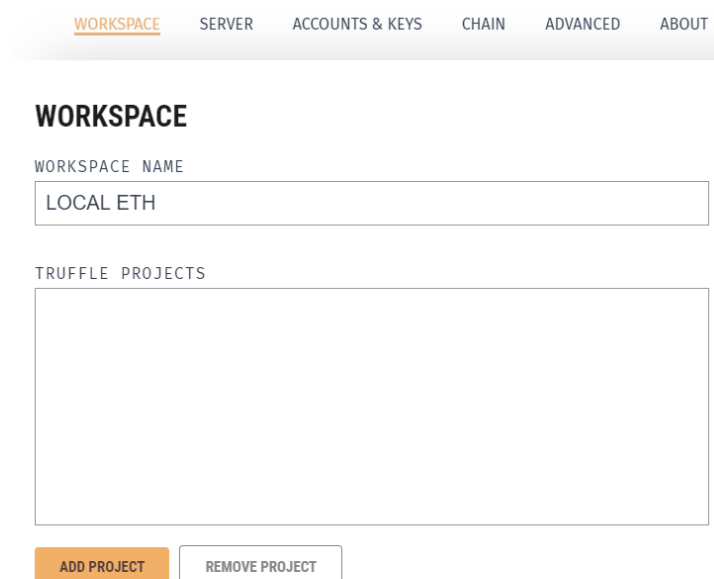
KONFIGURASI GANACHE

Ganache adalah database blockchain yang bisa kita gunakan sebagai alternatif akun yang memiliki saldo ethereum selain yang bisa digunakan di remix. Berikut ini tata cara untuk membuat workspace baru di ganache.

1. Download software **ganache** pada web berikut <https://trufflesuite.com/ganache/>
2. Lakukan instalasi **ganache** pada komputer anda
3. Setelah selesai melakukan instalasi, silahkan buat workspace baru yang akan kita hubungkan dengan **Metamask**



4. Pilih **New workspace** kemudian pilih kembali **ethereum**.
5. Sebagai contoh kita berikan nama **Local ETH**



6. Pilih **save** workspace

7. Tampilan setelah anda menyimpan adalah kumpulan akun-akun yang memiliki saldo **100 ETH**

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK0

GAS PRICE20000000000

GAS LIMIT6721975

HARDWAREMURGLACIER

NETWORK ID5777

RPC SERVERHTTP://127.0.0.1:7545

MINING STATUSAUTOMINING

WORKSPACELOCAL ETH

SWITCH

MNEMONIC

stove paper discover cluster original fitness park enhance acquire bid display system

HD PATHm/44'/60'/0'/0/account_index

ADDRESS0x48C9124E4ce1E9F0eb166876A2e2fe7aF0F25043

BALANCE100.00 ETH

TX COUNT0

INDEX0

ADDRESS0xf44c9da4B3cC9969c678602E1e13eB2fe6841EA9

BALANCE100.00 ETH

TX COUNT0

INDEX1

ADDRESS0x6e8b14362eFd0871785c8e9F61090E19aD021e18

BALANCE100.00 ETH

TX COUNT0

INDEX2

ADDRESS0xAF25dEb7a105cf7Bf8c9635D6Bf6A358694E88cC

BALANCE100.00 ETH

TX COUNT0

INDEX3

ADDRESS0x0065ae27760dD681d1F05387a21A8d05a918C876

BALANCE100.00 ETH

TX COUNT0

INDEX4

8. Disini dapat terlihat bahwa terdapat **Block, Transactions, Contracts, Events dan Logs**

9. Kita bisa mengambil **public key dan private key** dengan mengklik icon kunci.

10. Jika diklik maka anda akan diberikan informasi **key**

ACCOUNT INFORMATION

ACCOUNT ADDRESS

0x48C9124E4ce1E9F0eb166876A2e2fe7aF0F25043

PRIVATE KEY

fc220b03d1bc5e08ff127a6392356ca5986eaa719384d022892e03096eac9506

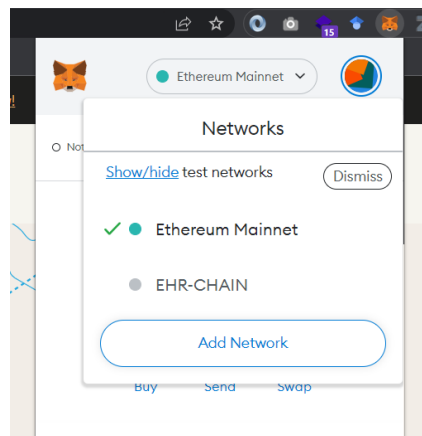
Do not use this private key on a public blockchain; use it for development purposes only!

DONE

MENGHUBUNGKAN METAMASK DAN GANACHE

Akun-akun dalam ganache dapat kita sambungkan dengan Wallet Metamask yang sudah kita instal pada Google Chrome. Sekarang kita akan mengimport dan membuat server lokal blockchain di komputer kita. Langkah-langkahnya adalah sebagai berikut:

1. Pastikan **Ganache** sudah running atau terbuka.
2. Klik ekstension **Metamask** pada Google Chrome



3. Pilih Add Network untuk membuat network baru yang akan terhubung ke **ganache**

Settings

Search in settings

General

Advanced

Contacts

Security & Privacy

Alerts

Networks

Experimental

About

Networks > Add a network > Add a network manually

A malicious network provider can lie about the state of the blockchain and record your network activity. Only add custom networks you trust.

Network Name

GANACHE

New RPC URL

HTTP://127.0.0.1:7545

Chain ID ⓘ

1337

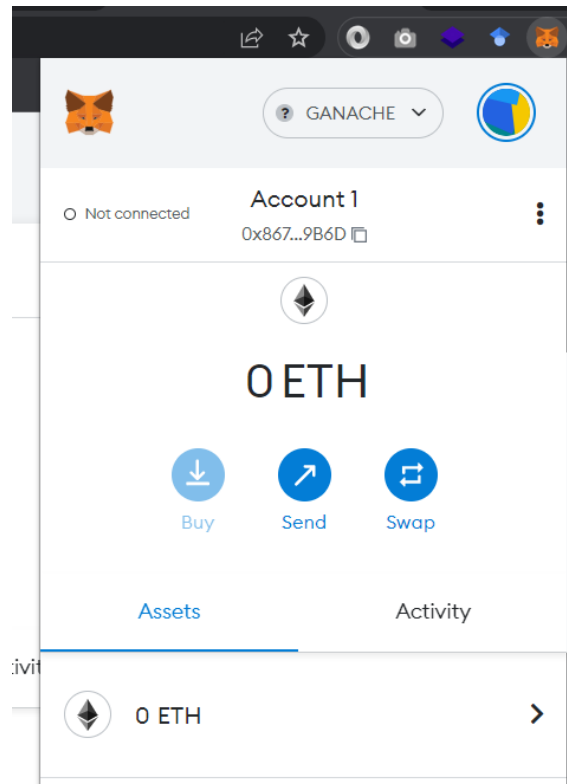
This Chain ID is currently used by the Localhost 8545 network.

Currency Symbol

ETH

The network with chain ID 1337 may use a different currency symbol (CPAY) than the one you have entered. Please verify before continuing.

4. Isikan data sesuai dengan gambar.
5. Klik **Save** dan kita sudah membuat network baru.



6. Dapat kita lihat saldo yang disediakan adalah 0 ETH.
7. Kita bisa melakukan **import** akun yang terdapat pada **Ganache** kita.
8. Kita tinggal mengimport **private key** yang ada.

ACCOUNT INFORMATION

ACCOUNT ADDRESS

0x48C9124E4ce1E9F0eb166876A2e2fe7aF0F25043

PRIVATE KEY

fc220b03d1bc5e08ff127a6392356ca5986eaa719384d022892e03096eac9506

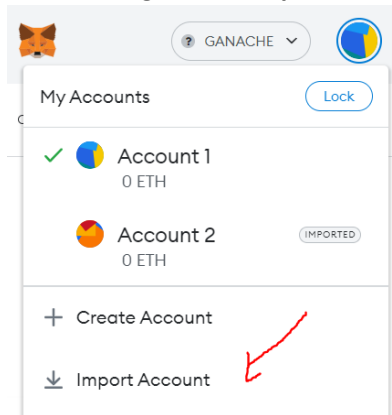
Do not use this private key on a public blockchain; use it for development purposes only!

DONE

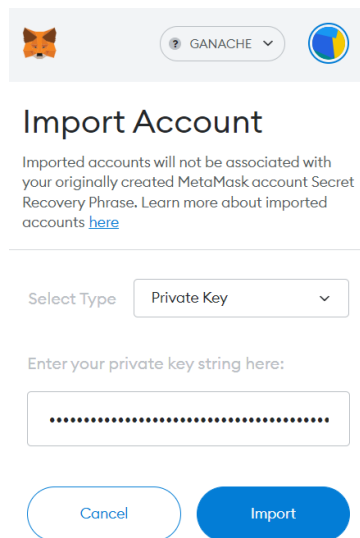
9. Sebagai contoh kita akan mengambil **private key** yang akan kita import.

fc220b03d1bc5e08ff127a6392356ca5986eaa719384d022892e03096eac9506

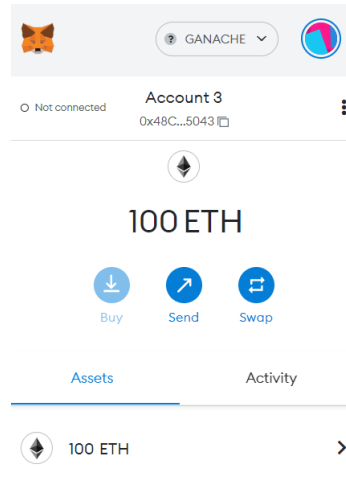
10. Kita akan mengimport pada metamask dengan cara **import** account.



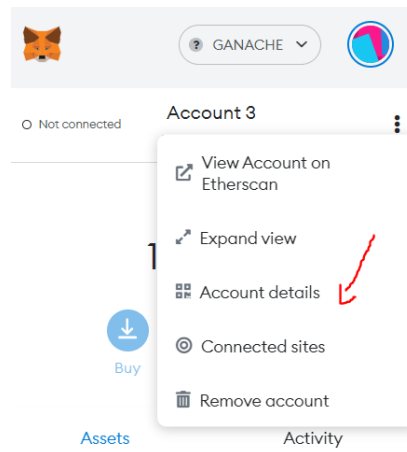
11. Klik import account lalu masukan **private key** yang sudah kita copy sebelumnya.



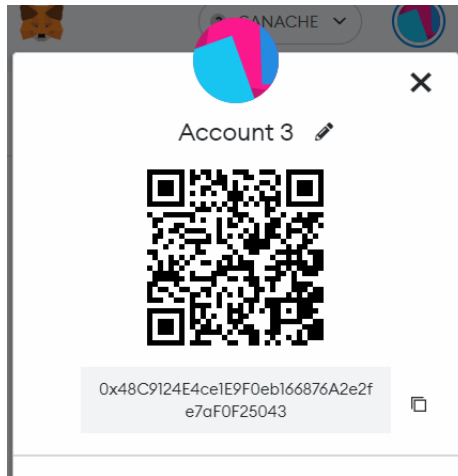
12. Klik import dan akun berisi saldo **100 ETH** masuk pada **Metamask**.



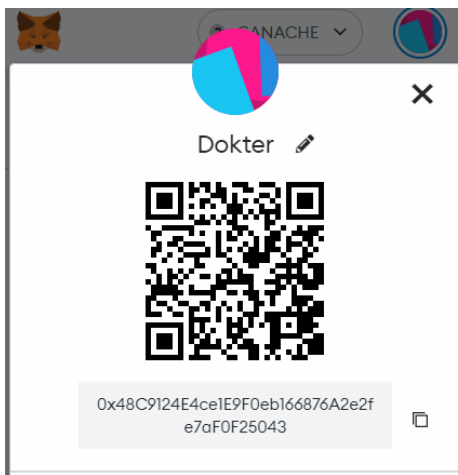
13. Sekarang, anda sudah memiliki akun berisi saldo Ethereum untuk **testing**.
14. Anda juga dapat mengimport akun lain jika membutuhkan banyak akun untuk testing dengan mengimport **private key**.
15. Sebagai informasi, kita bisa merubah nama Akun sesuai dengan yang kita inginkan.
16. Contohnya adalah kita akan merubah nama **Account3** menjadi **Dokter**.
17. Klik tanda titik tiga pada akun tersebut, lalu pilih account detail



18. Silahkan sesuaikan nama akun anda dengan cara klik icon **pencil**.



19. Ubah nama akun telah selesai.

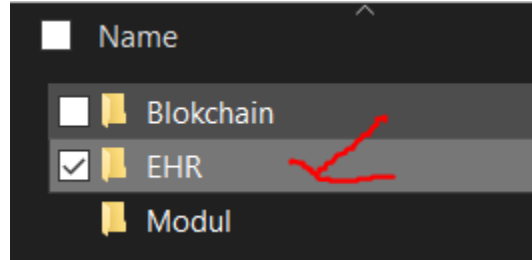


MEMBUAT APLIKASI DAPPS EHR IPFS

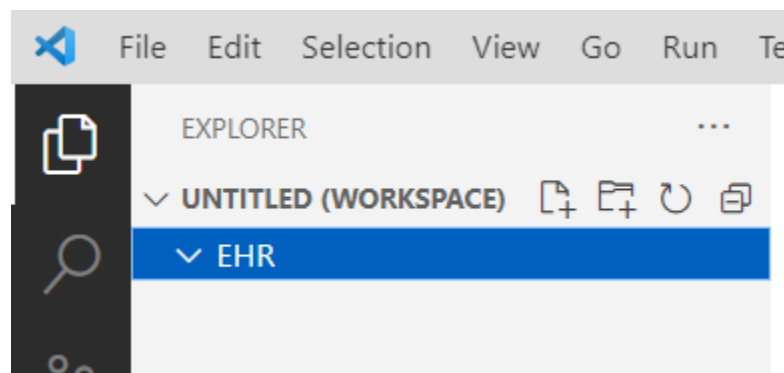
Dapps adalah program atau aplikasi digital yang aktif dan bekerja di blockchain atau jaringan komputer P2P. Jadi Dapps tidak bekerja di komputer single. Dapps berada di luar kontrol dan otoritas perorangan. Di dalam Dapps terdapat jaringan terpusat yang berisi gabungan antara tampilan antarmuka dan kontrak pintar. Pada latihan kali ini kita akan membuat aplikasi rekam medis sederhana yang memanfaatkan Smart Contract Ethereum, Metamask, Truffle, NodeJS, ReactJS, Web3Js dan IPFS.

Aplikasi ini akan menjadi portal antara dokter dan pasien dalam mengupload data rekam medis. Dokter dapat menambahkan data rekam medis pasien melalui DAPPS ethereum ini. Berikut ini adalah langkah-langkah dalam membangun Sistem Rekam Medis sederhana:

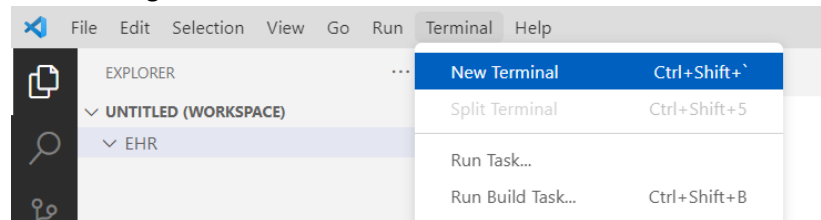
1. Silahkan download terlebih dahulu aplikasi **Node.JS**.
2. Aplikasi **Node.Js** akan kita gunakan untuk menginstal truffle dan react js.
3. Node.Js dapat kita download di <https://nodejs.org/en/>
4. Silahkan pilih **NodeJs** sesuai dengan sistem operasi yang digunakan.
5. Lakukan instalasi **Node.js** pada komputer anda.
6. Sekarang anda juga harus memiliki code editor yaitu **Visual Studio code** yang dapat didownload pada <https://code.visualstudio.com/download>
7. Lakukan instalasi **visual studio code** pada komputer anda.
8. Setelah visual studio code dan node.js terinstal, silahkan buat folder dengan nama EHR pada komputer anda, sebagai contoh kami membuatkan di dalam folder UII.



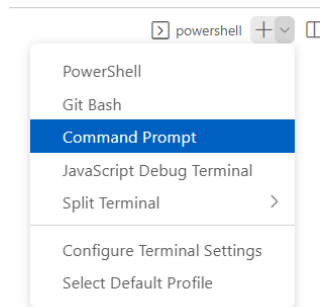
9. Buka aplikasi visual studio code anda, lalu pilih **File-> Add Folder to Workspace** dan pilih folder EHR.
10. Sekarang folder EHR sudah masuk pada **Visual Studio Code**.



11. Aktifkan terminal pada visual studio code, kita akan menginstal **truffle**.
12. Pengaktifan terminal dengan cara memilih menu **Terminal -> New Terminal**.



13. Sekarang anda dapat melihat pada bagian bawah VS Code anda, sebuah terminal **console** yang dapat kita gunakan untuk menginstal truffle.
14. Terlebih dahulu ubah **Power Shell menjadi CMD**



15. Sekarang terminal CMD anda sudah aktif, saatnya kita akan menginstal **truffle**.
16. Ketikkan perintah berikut pada terminal anda

```
npm install -g truffle
```

17. Tunggu hingga proses instalasi selesai

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

```
PS D:\UII\EHR> npm install -g truffle
npm WARN deprecated uuid@2.0.1: Please upgrade to version 7 or higher. Older versions may use
tic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated mkdirp-promise@5.0.1: This package is broken and no longer maintained. 'mkd:
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use
tic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated uuid@3.3.2: Please upgrade to version 7 or higher. Older versions may use
tic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated multicodec@0.5.7: stable api reached
npm WARN deprecated multibase@0.6.1: This module has been superseded by the multiformats module
npm WARN deprecated multibase@0.7.0: This module has been superseded by the multiformats module
```

18. Proses instalasi truffle membutuhkan beberapa waktu, pastikan koneksi internet anda stabil.
19. Jika berhasil maka anda akan menemukan informasi berikut.

```
+ truffle@5.5.29
```

```
added 29 packages from 21 contributors, removed 13 packages and updated  
13 packages in 104.505s
```

20. Sekarang kita akan cek versi **truffle** kita dengan mengetikan perintah.

```
npx truffle version
```

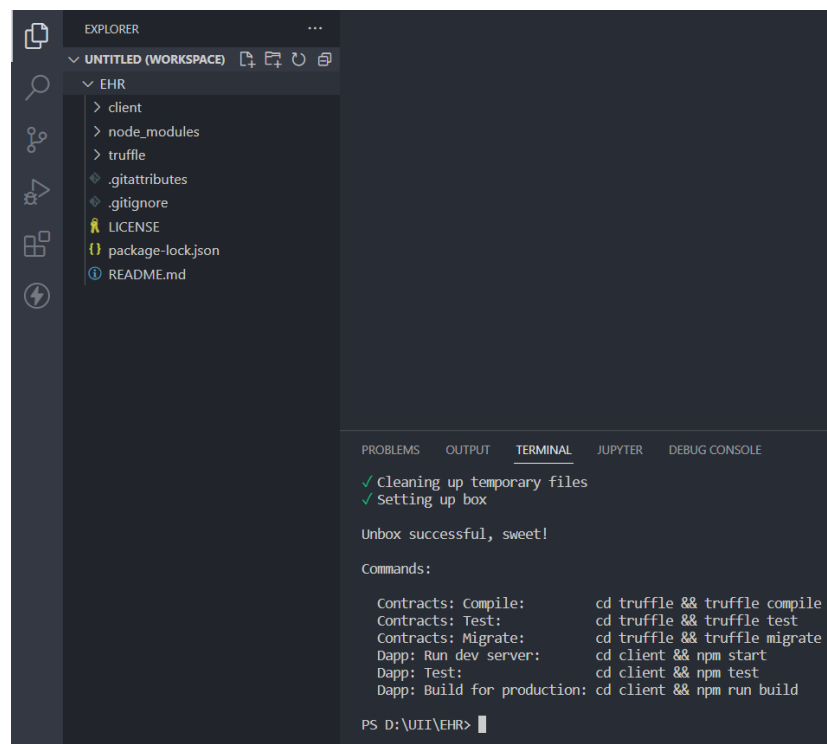
21. Hasilnya kita akan melihat versi dari **truffle** kita

```
Truffle v5.5.29 (core: 5.5.29)
```

22. Tunggu hingga proses instalasi selesai.
23. Setelah terinstal semua, kita akan menginstal **truffle react box**.
24. Pada terminal ketikan perintah berikut.

```
npx truffle unbox react
```

25. Tunggu proses instalasi **truffle react** selesai.
26. Jika berhasil maka anda akan melihat direktori instalasi baru ditambahkan.



27. Sekarang kita mendapatkan beberapa folder dari project truffle react kita.

28. Pada sub folder **truffle** -> **folder contract** buatlah sebuah file baru dengan nama **EHR.sol** dan isi dengan code berikut.

```
pragma solidity >=0.4.22 <0.9.0;

contract EHR {
    struct Record {
        string cid;
        string fileName;
        address patientId;
        address doctorId;
        uint256 timeAdded;
    }

    struct Patient {
        address id;
        Record[] records;
    }

    struct Doctor {
        address id;
    }

    mapping (address => Patient) public patients;
    mapping (address => Doctor) public doctors;

    event PatientAdded(address patientId);
    event DoctorAdded(address doctorId);
    event RecordAdded(string cid, address patientId, address doctorId);

    // modifiers

    modifier senderExists {
        require(doctors[msg.sender].id == msg.sender || patients[msg.sender].id ==
msg.sender, "Sender does not exist");
        _;
    }

    modifier patientExists(address patientId) {
        require(patients[patientId].id == patientId, "Patient does not exist");
        _;
    }

    modifier senderIsDoctor {
        require(doctors[msg.sender].id == msg.sender, "Sender is not a doctor");
        _;
    }

    // functions

    function addPatient(address _patientId) public senderIsDoctor {
        require(patients[_patientId].id != _patientId, "This patient already exists.");
        patients[_patientId].id = _patientId;

        emit PatientAdded(_patientId);
    }

    function addDoctor() public {
        require(doctors[msg.sender].id != msg.sender, "This doctor already exists.");
        doctors[msg.sender].id = msg.sender;
    }
}
```

```

        emit DoctorAdded(msg.sender);
    }

    function addRecord(string memory _cid, string memory _fileName, address _patientId)
    public senderIsDoctor patientExists(_patientId) {
        Record memory record = Record(_cid, _fileName, _patientId, msg.sender,
        block.timestamp);
        patients[_patientId].records.push(record);

        emit RecordAdded(_cid, _patientId, msg.sender);
    }

    function getRecords(address _patientId) public view senderExists
    patientExists(_patientId) returns (Record[] memory) {
        return patients[_patientId].records;
    }

    function getSenderRole() public view returns (string memory) {
        if (doctors[msg.sender].id == msg.sender) {
            return "doctor";
        } else if (patients[msg.sender].id == msg.sender) {
            return "patient";
        } else {
            return "unknown";
        }
    }

    function getPatientExists(address _patientId) public view senderIsDoctor returns
    (bool) {
        return patients[_patientId].id == _patientId;
    }
}

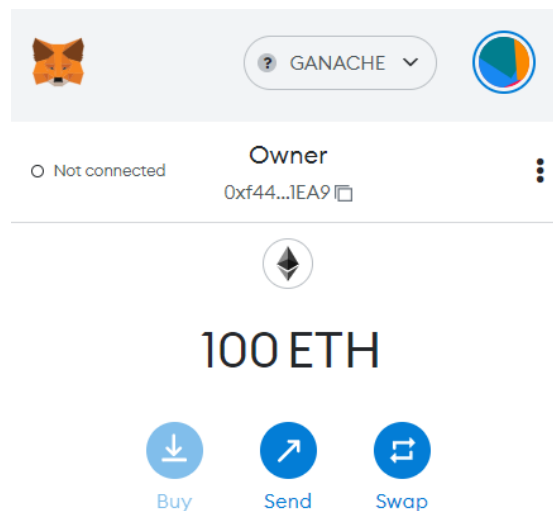
```

Keterangan:

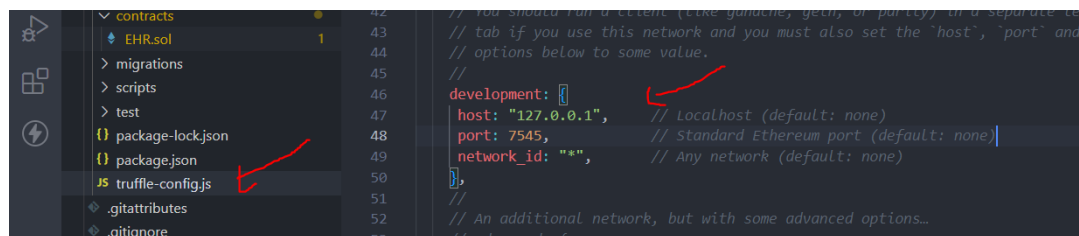
Kode	Keterangan
struct	Structs pada Solidity memungkinkan Anda membuat tipe data yang lebih rumit yang memiliki banyak properti. Anda dapat menentukan tipe Anda sendiri dengan membuat struct. Mereka berguna untuk mengelompokkan data terkait. Struct dapat dideklarasikan di luar kontrak dan diimpor dalam kontrak lain.
string	Mendukung tipe data string/character
address	address Ethereum berarti pengidentifikasi mata uang kripto kunci publik unik yang menunjuk ke dompet yang kompatibel dengan Ethereum tempat Eter, ONGX atau Token ERC20 atau Token Keamanan dapat dikirim atau disimpan. Contoh 1. address Ethereum berarti alamat 'dompet' di Blockchain Ethereum.
msg.sender	akan menjadi orang yang saat ini terhubung dengan kontrak.
mapping	tabel hash atau kamus dalam bahasa lain. Ini digunakan untuk menyimpan data dalam bentuk pasangan nilai kunci, kunci dapat berupa tipe data apa pun yang ada di dalamnya tetapi tipe referensi tidak diperbolehkan sedangkan nilainya dapat berupa tipe apa pun. Pemetaan sebagian besar digunakan untuk mengaitkan alamat Ethereum yang unik dengan jenis nilai terkait.

event	Event adalah cara untuk berkomunikasi dengan aplikasi klien atau situs web front-end bahwa sesuatu telah terjadi di blockchain.
modifier	Fungsi yang digunakan untuk mengubah perilaku suatu fungsi. Misalnya untuk menambahkan prasyarat ke suatu fungsi. Pertama kita buat modifier dengan atau tanpa parameter.
emit	Kata kunci Emit digunakan untuk memancarkan suatu peristiwa dalam soliditas, yang dapat dibaca oleh klien di Dapp.

29. Kode ini merupakan kode smart contract dari aplikasi EHR
30. Pastikan **Metamask** sudah terhubung dengan **ganache**.
31. Sekarang kita import sebuah akun ganache sebagai owner atau pemilik dari smart contract ini.
32. Tata cara import sudah dijelaskan pada bagian sebelumnya.
33. Sekarang kita sudah menambahkan Owner pada wallet Metamask dengan saldo 100 ETH



34. Pada file **truffle-config.js** aktifkan code berikut.



35. Kita sesuaikan dengan port **ganache** yaitu 7545
36. Kembali pada Ganache, kita setting **Ganache** -> **pilih icon setting**
37. **Add Project** yaitu **truffle-config.js** yang sudah kita ubah sebelumnya.

[WORKSPACE](#)
[SERVER](#)
[ACCOUNTS & KEYS](#)
[CHAIN](#)
[ADVANCED](#)
[ABOUT](#)

WORKSPACE

WORKSPACE NAME

TRUFFLE PROJECTS

38. Pilih **save and restart**.

39. Kembali pada VS code, kita akan beralih pada folder **truffle** pada terminal dengan cara mengetikkan perintah.

```
cd truffle
```

40. Perintah tersebut akan memindahkan posisi terminal ke direktori truffle.

41. Disini kita akan melakukan **compile** file solidity atau **EHR.sol**.

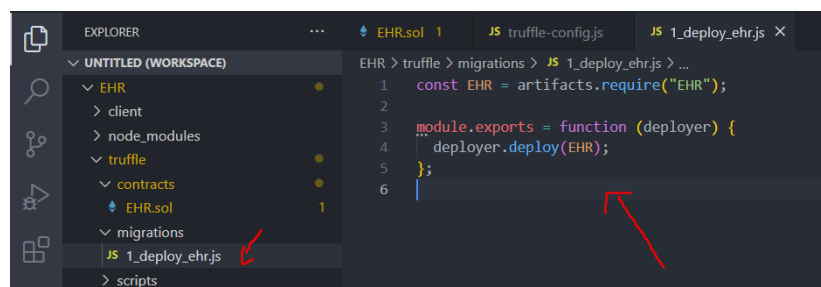
42. Namun terlebih dahulu kita harus **menghapus** file SimpleStorage.sol agar kita hanya menginstal **EHR.sol** saja.

43. Setelah terhapus, kita akan melakukan **compile**.

```
PS D:\UII\EHR> cd truffle
PS D:\UII\EHR\truffle> npx truffle compile
```

44. Tunggu hingga proses **compile** selesai

45. Pada folder **migrations** rename file default dan ubah isinya seperti ini.



46. Pada terminal kita akan lakukan compile **Smart contract** dengan cara mengetikan perintah berikut

```
- solc: 0.8.14+commit.80d49f37.Emscripten.clang
PS D:\UII\EHR\truffle> npx truffle migrate
[.....] / fetchMetadata: sill resolveWith
```

47. Tunggu hingga proses **migrate** selesai.

48. Jika berhasil maka ada informasi proses **compile** smart contract anda.

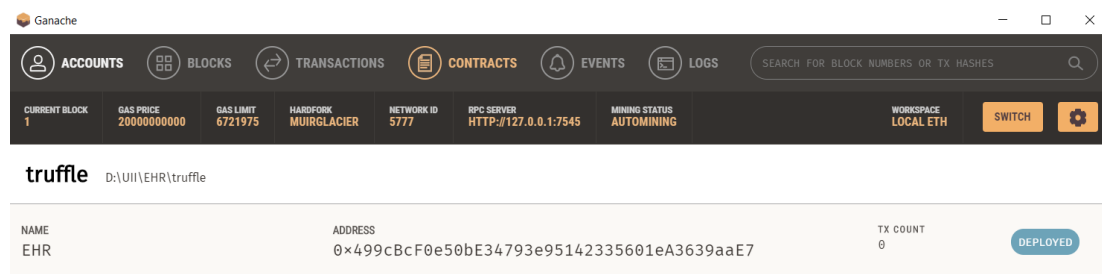
```
1_deploy_ehr.js
=====

Deploying 'EHR'
-----
> transaction hash: 0x1e84d22c485945d732e34b48417a9fd1f8974c650d3e931d17f6ebb04cedc6bf
> Blocks: 0
> contract address: 0x499cBcF0e50bE34793e95142335601eA3639aaE7
> block number: 1
> block timestamp: 1662718848
> account: 0x48C9124E4ce1E9F0eb166876A2e2fe7aF0F25043
> balance: 99.96988114
> gas used: 1505943 (0x16fa97)
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.03011886 ETH

> Saving artifacts
-----
> Total cost: 0.03011886 ETH

Summary
=====
> Total deployments: 1
> Final cost: 0.03011886 ETH
```


49. Kita juga bisa cek pada **Ganache** kita di tab **contract** sudah ada smart contract EHR.





The screenshot shows the Ganache application window. The 'CONTRACTS' tab is selected, displaying a table of deployed contracts. The table has columns for NAME, ADDRESS, and TX COUNT. One contract, 'EHR', is listed with the address '0x499cBcF0e50bE34793e95142335601eA3639aaE7' and a TX COUNT of 0. A 'DEPLOYED' button is visible next to the contract name.


NAME	ADDRESS	TX COUNT
EHR	0x499cBcF0e50bE34793e95142335601eA3639aaE7	0


50. Jika kita cek blokck juga sudah bertambah blok baru pada Blockchain kita


 ACCOUNTS

 BLOCKS

 TRANSACTIONS

 CONTRACTS

 EVENTS

 LOGS

CURRENT BLOCK 1	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK MUIRGLACIER	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:7545	MINING STATUS AUTOMINING
BLOCK 1	MINED ON 2022-09-09 17:20:48				GAS USED 1505943	
BLOCK 0	MINED ON 2022-09-09 16:04:27				GAS USED 0	

51. Saldo owner sekarang sudah berkurang jika kita cek di ganache karena proses deploy smartcontract.

MNEMONIC ?
stove paper discover cluster original fitness park enhance acquire bid display system

note: this mnemonic is not secure; don't use it on a public blockchain.

ADDRESS
0x48C9124E4ce1E9F0eb166876A2e2fe7aF0F25043

BALANCE
99.97 ETH

52. Sekarang kita akan pindah ke **folder client**.

53. Pada terminal ketikkan perintah berikut

```
cd ..
```

54. Ketikkan kembali

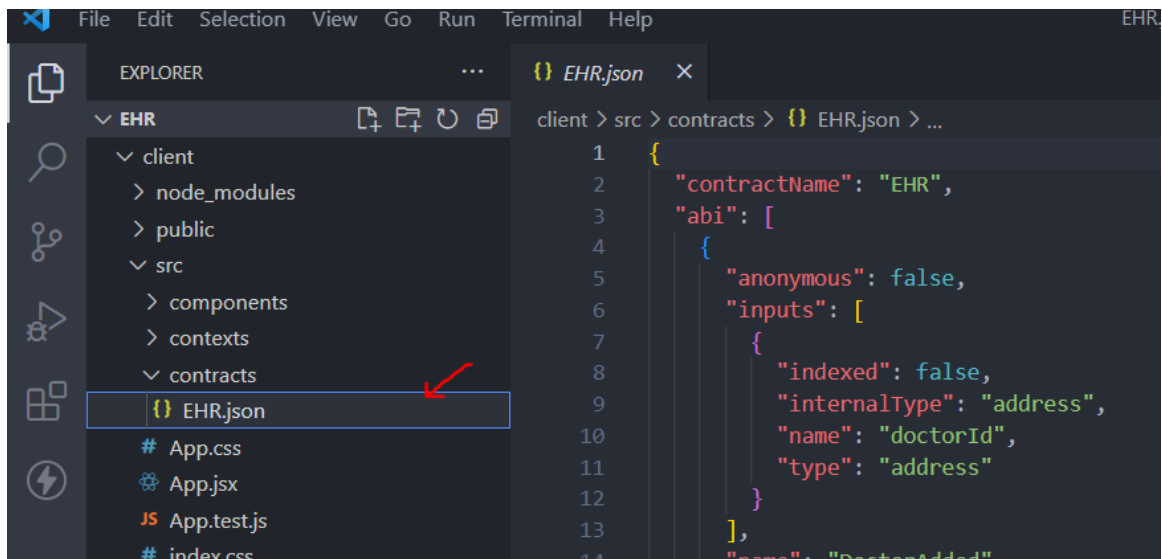
```
cd client
```

55. Sekarang posisi terminal sudah berubah pada terminal client.

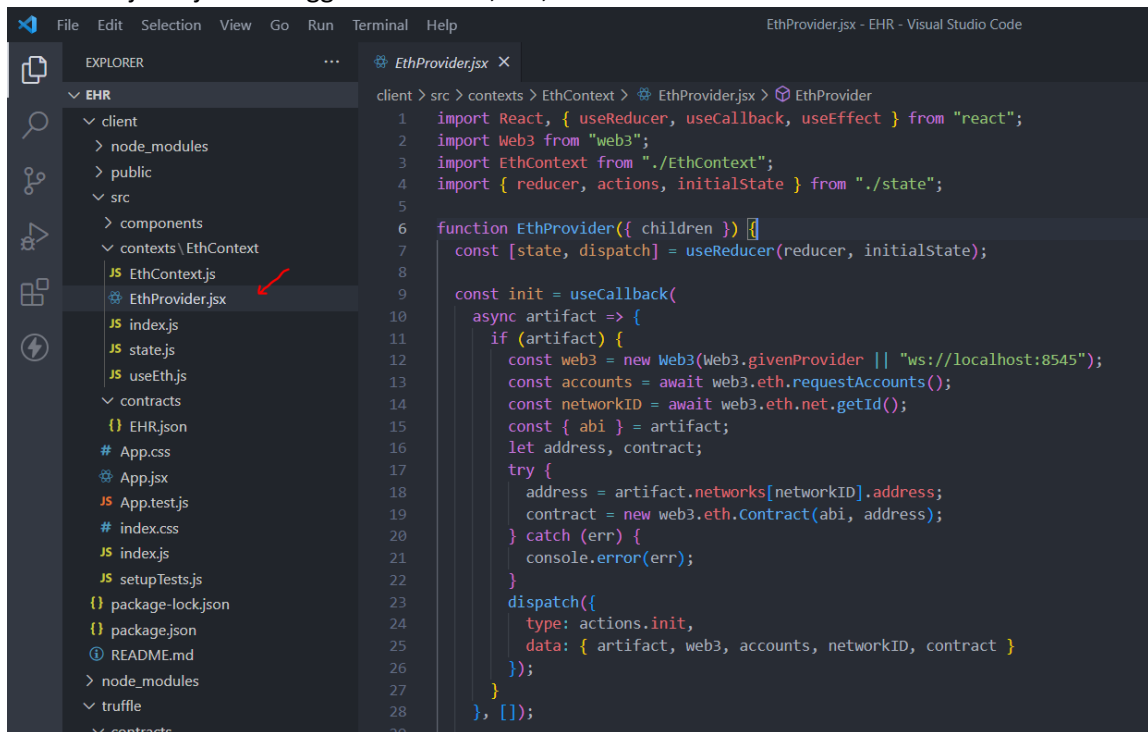
```
PROBLEMS 1 OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

PS D:\UII\EHR> cd client
PS D:\UII\EHR\client> |
```

56. Sebelum melanjutkan pada pembuatan tampilan di client, kita harus pahami bahwa setelah smart contract di migrate maka akan menciptakan file baru yaitu **EHR.json**. File ini dapat anda temukan pada folder **client->src->contracts->EHR.json**



57. File ini berbentuk **JSON** agar kita dapat menghubungkannya ke aplikasi client seperti website atau android.
58. Folder client dihasilkan secara otomatis oleh react truffle box. Beberapa pengaturan sudah dihasilkan mereka secara otomatis.
59. Kita tinggal memodifikasi sesuai kebutuhan. Sebagai contoh pada file **ETHProvider.jsx**
60. Pada file ini telah dihasilkan file otomatis yang menghubungkan ke **web3.js**
61. web3.js adalah kumpulan pustaka yang memungkinkan Anda berinteraksi dengan node ethereum lokal atau jarak jauh menggunakan HTTP, IPC, atau WebSocket.



62. Tahap selanjutnya adalah mengubah file **ETHProvider.jsx** yang terdapat pada folder context. Ubah baris ke 33 menjadi kode berikut.

```
const artifact = require("../contracts/EHR.json");
```

63. Kode tersebut menghubungkan dengan file JSON EHR yang telah kita hasilkan sebelumnya.

64. Masih pada file **ETHProvider.jsx** ubah isi dari fungsi async artifact menjadi berikut ini.

```
async artifact => {
  if (artifact) {
    const web3 = new Web3(Web3.givenProvider || "ws://localhost:7545");
    const accounts = await web3.eth.requestAccounts();
    const networkID = await web3.eth.net.getId();
    const { abi } = artifact;
    let address, contract;
    try {
      address = artifact.networks[networkID].address;
      contract = new web3.eth.Contract(abi, address);
    } catch (err) {
      console.error(err);
    }

    let role = 'unknown'
    if (contract && accounts) {
      role = await contract.methods.getSenderRole().call({ from: accounts[0] })
    }

    dispatch({
      type: actions.init,
      data: { artifact, web3, accounts, networkID, contract, role, loading: false
    }
  });
}, []);
```

65. Sehingga kode sepenuhnya untuk file **ETHProvider.jsx** adalah sebagai berikut.

```
ETHProvider.jsx
import React, { useReducer, useCallback, useEffect } from 'react'
import Web3 from 'web3'
import EthContext from './EthContext'
import { reducer, actions, initialState } from './state'

function EthProvider({ children }) {
  const [state, dispatch] = useReducer(reducer, initialState)

  const init = useCallback(async artifact => {
    if (artifact) {
      const web3 = new Web3(Web3.givenProvider || 'ws://localhost:7545')
      const accounts = await web3.eth.requestAccounts()
      const networkID = await web3.eth.net.getId()
      const { abi } = artifact
      let address, contract
      try {
        address = artifact.networks[networkID].address
        contract = new web3.eth.Contract(abi, address)
      } catch (err) {
        console.error(err)
      }
    }
  }, [])
```



```

        let role = 'unknown'
        if (contract && accounts) {
            role = await contract.methods.getSenderRole().call({ from: accounts[0] })
        }
        dispatch({
            type: actions.init,
            data: { artifact, web3, accounts, networkID, contract, role, loading: false
        },
    },
    })
    }
    }, [])

    useEffect(() => {
        const tryInit = async () => {
            try {
                const artifact = require('../../contracts/EHR.json')
                init(artifact)
            } catch (err) {
                console.error(err)
            }
        }

        tryInit()
    }, [init])

    useEffect(() => {
        const events = ['chainChanged', 'accountsChanged']
        const handleChange = () => {
            init(state.artifact)
        }

        events.forEach(e => window.ethereum.on(e, handleChange))
        return () => {
            events.forEach(e => window.ethereum.removeListener(e, handleChange))
        }
    }, [init, state.artifact])

    return (
        <EthContext.Provider
            value={{
                state,
                dispatch,
            }}
        >
            {children}
        </EthContext.Provider>
    )
}

export default EthProvider

```

66. Selanjutnya adalah merubah isi kode pada file **state.js** menjadi berikut ini.

```

const actions = {
    init: 'INIT',
    addDoctor: 'ADD_DOCTOR',
}

```

```

const initialState = {
  artifact: null,
  web3: null,
  accounts: null,
  networkID: null,
  contract: null,
  role: 'unknown',
  loading: true,
}

const reducer = (state, action) => {
  const { type, data } = action
  switch (type) {
    case actions.init:
      return { ...state, ...data }
    case actions.addDoctor:
      return { state: { ...state, role: 'doctor' } }
    default:
      throw new Error('Undefined reducer action type')
  }
}

export { actions, initialState, reducer }

```

67. Kode yang berwarna merah adalah kode yang kita tambahkan pada kode default.

68. Sekarang kita akan mengkonfigurasi dependensi project kita.

69. Pada file **package.json** kita ubah isinya sebagai berikut.

```

{
  "name": "truffle-client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.10.0",
    "@emotion/styled": "^11.10.0",
    "@material-ui/core": "^4.12.4",
    "@material-ui/icons": "^4.11.3",
    "@mui/icons-material": "^5.8.4",
    "@mui/material": "^5.10.0",
    "@testing-library/jest-dom": "^5.16.4",
    "@testing-library/react": "^13.2.0",
    "@testing-library/user-event": "^13.5.0",
    "ipfs-api": "^26.1.2",
    "ipfs-http-client": "^33.1.1",
    "material-ui-dropzone": "^3.5.0",
    "moment": "^2.29.4",
    "react": "^18.1.0",
    "react-dom": "^18.1.0",
    "react-router-dom": "^6.3.0",
    "react-scripts": "^4.0.3",
    "react-video-cover": "^1.2.2",
  }
}

```

```

    "web3": "^1.7.3"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}

```

70. Jalankan perintah **npm install** pada terminal client

```

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\UII\EHR> cd client
PS D:\UII\EHR\client> npm install
[.....] \ fetchMetadata: sill install loadAllDepsIntoIdealTree

```

71. Tunggu hingga proses instalasi selesai

72. Tahap ini merupakan tahap untuk mendownload dependensi dari projek dapps kita

```

added 257 packages from 338 contributors and audited 2327 packages in 112.54s

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

found 39 vulnerabilities (10 low, 10 moderate, 16 high, 3 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
PS D:\UII\EHR\client>

```

73. Sekarang kita sudah berhasil menambahkan dependensi, selanjutnya adalah membuat **Alert Context**.

74. Pada folder context buatlah sebuah folder baru dengan nama AlertContext
75. Di dalam folder tersebut buatlah dua buah file yaitu **AlertContext.js** dan **useAlert.js**. masing-masing kode tiap file adalah berikut ini.

```
AlertContext.js
import { createContext, useState } from 'react';

const ALERT_TIME = 5000;

const initialState = {
  text: '',
  type: '',
};

const AlertContext = createContext({
  ...initialState,
  setAlert: () => {},
});

export const AlertProvider = (props) => {
  const { children } = props;
  const [text, setText] = useState('');
  const [type, setType] = useState('');

  const setAlert = (text, type) => {
    setText(text);
    setType(type);

    setTimeout(() => {
      setText('');
      setType('');
    }, ALERT_TIME);
  };

  return (
    <AlertContext.Provider
      value={{
        text,
        type,
        setAlert,
      }}
    >
      {children}
    </AlertContext.Provider>
  );
};

export default AlertContext;
```

```
useAlert.js
import { useContext } from 'react';
import AlertContext from './AlertContext';
```

```
const useAlert = () => useContext(AlertContext);

export default useAlert;
```

76. Selanjutnya ubah isi kode **App.js** menjadi berikut ini.

App.jsx

```
import { EthProvider } from './contexts/EthContext'
import routes from './routes'
import { useRoutes } from 'react-router-dom'
import { AlertProvider } from './contexts/AlertContext/AlertContext'

function App() {
  const content = useRoutes(routes)

  return (
    <EthProvider>
      <AlertProvider>{content}</AlertProvider>
    </EthProvider>
  )
}

export default App
```

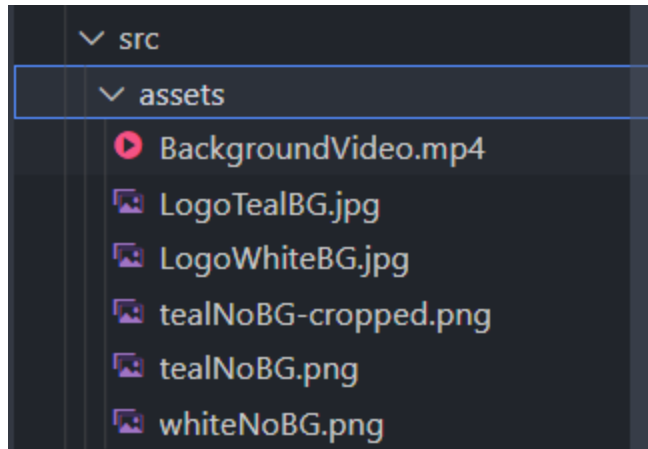
77. Ubah juga file **index.js** dengan kode berikut

Index.js

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import './index.css'
import App from './App'
import { BrowserRouter } from 'react-router-dom'

const root = ReactDOM.createRoot(document.getElementById('root'))
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
)
```

78. Pada folder src buatlah folder baru dengan nama **assets**. Isi folder tersebut adalah images yang sudah kami sediakan.



79. Masih di folder **src->components** buatlah file dengan nama **CustomButton.jsx** isikan kode berikut

```

CustomButton.jsx
import React from 'react'
import { Box, Button, Typography } from '@mui/material'
import { grey, teal } from '@mui/material/colors'

const CustomButton = ({ text, handleClick, disabled = false, children }) => {
  return (
    <Button
      startIcon={children}
      style={{
        backgroundColor: disabled ? grey[400] : teal['A700'],
        textTransform: 'none',
        padding: '10px 20px',
      }}
      onClick={handleClick}
      disabled={disabled}
    >
      <Typography variant='h5' color='white'>
        {text}
      </Typography>
    </Button>
  )
}

export default CustomButton

```

80. Buatlah file pada posisi yang sama dengan nama **Record.jsx**. isikan kode berikut

```

Record.jsx
import { Card, CardContent, IconButton, Typography, Grid, Box } from '@mui/material'
import React from 'react'
import DescriptionRoundedIcon from '@mui/icons-material/DescriptionRounded'
import { grey } from '@mui/material/colors'
import moment from 'moment'
import CloudDownloadRoundedIcon from '@mui/icons-material/CloudDownloadRounded'
import { useNavigate } from 'react-router-dom'

const Record = ({ record }) => {

```

```

const [cid, name, patientId, doctorId, timestamp] = record
const navigate = useNavigate()

return (
  <Card>
    <CardContent>
      <Grid container spacing={2}>
        <Grid item xs={1}>
          <DescriptionRoundedIcon style={{ fontSize: 40, color: grey[700] }} />
        </Grid>
        <Grid item xs={3}>
          <Box display='flex' flexDirection='column'>
            <Typography variant='h6' color={grey[600]}>
              Record name
            </Typography>
            <Typography variant='h6'>{name}</Typography>
          </Box>
        </Grid>
        <Grid item xs={5}>
          <Box display='flex' flexDirection='column'>
            <Typography variant='h6' color={grey[600]}>
              Doctor
            </Typography>
            <Typography variant='h6'>{doctorId}</Typography>
          </Box>
        </Grid>
        <Grid item xs={2}>
          <Box display='flex' flexDirection='column'>
            <Typography variant='h6' color={grey[600]}>
              Created time
            </Typography>
            <Typography variant='h6'>{moment.unix(timestamp).format('MM-DD-YYYY
HH:mm')}</Typography>
          </Box>
        </Grid>
        <Grid item xs={1}>
          <a href={`https://med-chain.infura-ipfs.io/ipfs/${cid}`} target='_blank'
rel='noopener noreferrer'>
            <IconButton>
              <CloudDownloadRoundedIcon fontSize='large' />
            </IconButton>
          </a>
        </Grid>
      </Grid>
    </CardContent>
  </Card>
)
}

export default Record

```

81. Masih dalam folder components, buatlah folder baru dengan nama layouts. Dalam folder layouts buatlah dua file yaitu **AlertPopup.jsx** dan **Layout.jsx**. Masing-masing kode adalah sebagai berikut.

AlertPopup.jsx

```

import { Box, Alert, Typography } from '@mui/material'
import useAlert from '../contexts/AlertContext/useAlert'

const AlertPopup = () => {
  const { text, type } = useAlert()

  if (text && type) {
    return (
      <Box
        sx={{

```

```

        display: 'flex',
        justifyContent: 'center',
        marginTop: '8px',
        width: '100%',
        height: 'auto',
      }}
    >
    <Alert
      severity={type}
      sx={{
        position: 'absolute',
        zIndex: 1000000,
        width: 'auto',
        paddingRight: '25px',
      }}
    >
    <Typography variant='h6'>{text}</Typography>
    </Alert>
  </Box>
)
} else {
  return <></>
}
}

export default AlertPopup

```

Layout.jsx

```

import { AppBar, Chip, Toolbar, Box, Typography } from '@mui/material'
import React from 'react'
import useEth from '../contexts/EthContext/useEth'
import PersonRoundedIcon from '@mui/icons-material/PersonRounded'
import logo from '../assets/tealNoBG-cropped.png'
import { grey, teal } from '@mui/material/colors'

const HeaderAppBar = () => {
  const {
    state: { accounts, role },
  } = useEth()

  return (
    <AppBar position='static' style={{ backgroundColor: 'white' }}>
      <Toolbar>
        <Box display='flex' justifyContent='space-between' alignItems='center' width='100%'>
          <a href='/'>
            <img src={logo} alt='med-chain-logo' style={{ height: 20, weight: 20 }} />
          </a>
          <Box flexGrow={1} />
          <Box display='flex' alignItems='center'>
            <Box mb={0.1}>
              <PersonRoundedIcon style={{ color: grey[700], fontSize: '22px' }} />
            </Box>
            <Box ml={0.5} mr={2}>
              <Typography variant='h6' color='black'>
                {accounts ? accounts[0] : 'Wallet not connected'}
              </Typography>
            </Box>
          </Box>
          <Chip
            label={role === 'unknown' ? 'not registered' : role}
            style={{ fontSize: '12px', backgroundColor: teal['A700'], color: 'white' }}
          />
        </Box>
      </Toolbar>
    </AppBar>
  )
}

```

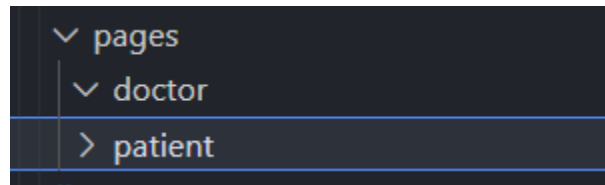


```

        </Box>
      </Box>
    </Toolbar>
  </AppBar>
)
}
export default HeaderAppBar

```

82. Pada folder **src** buatlah kembali folder **pages**. Di dalam folder pages kita akan membuat dua folder untuk user interface halaman dokter dan pasien. Folder tersebut adalah **doctor** dan **patient**.



83. Pada folder doctor buatlah dua buah file yaitu **index.js** dan **AddRecordModal.jsx**. Isi tiap file adalah sebagai berikut.

index.js

```

import { Box, Divider, FormControl, Modal, TextField, Typography, Backdrop, CircularProgress } from '@mui/material'
import React, { useCallback } from 'react'
import { useState } from 'react'
import CustomButton from '../../components/CustomButton'
import SearchRoundedIcon from '@mui/icons-material/SearchRounded'
import useEth from '../../contexts/EthContext/useEth'
import PersonAddAlt1RoundedIcon from '@mui/icons-material/PersonAddAlt1Rounded'
import useAlert from '../../contexts/AlertContext/useAlert'
import AddRecordModal from './AddRecordModal'
import CloudUploadRoundedIcon from '@mui/icons-material/CloudUploadRounded'
import ipfs from '../../ipfs'
import Record from '../../components/Record'

const Doctor = () => {
  const {
    state: { contract, accounts, role, loading },
  } = useEth()
  const { setAlert } = useAlert()

  const [patientExist, setPatientExist] = useState(false)
  const [searchPatientAddress, setSearchPatientAddress] = useState('')
  const [addPatientAddress, setAddPatientAddress] = useState('')
  const [records, setRecords] = useState([])
  const [addRecord, setAddRecord] = useState(false)

  //kode mencari pasien dengan public key
  const searchPatient = async () => {
    try {
      if (!/^(0x)?[0-9a-f]{40}$/i.test(searchPatientAddress)) {
        setAlert('Please enter a valid wallet address', 'error')
        return
      }
      const patientExists = await
      contract.methods.getPatientExists(searchPatientAddress).call({ from: accounts[0] })
      if (patientExists) {
        const records = await contract.methods.getRecords(searchPatientAddress).call({ from:
        accounts[0] })
        console.log('records :>> ', records)
        setRecords(records)
      }
    }
  }
}

```

```

        setPatientExist(true)
      } else {
        setAlert('Patient does not exist', 'error')
      }
    } catch (err) {
      console.error(err)
    }
  }
}

//kode mendaftarkan pasien baru dengan mengambil public key
const registerPatient = async () => {
  try {
    await contract.methods.addPatient(addPatientAddress).send({ from: accounts[0] })
  } catch (err) {
    console.error(err)
  }
}

const addRecordCallback = useCallback(
  async (buffer, fileName, patientAddress) => {
    if (!patientAddress) {
      setAlert('Please search for a patient first', 'error')
      return
    }
    try {
      const res = await ipfs.add(buffer)
      const ipfsHash = res[0].hash
      if (ipfsHash) {
        await contract.methods.addRecord(ipfsHash, fileName, patientAddress).send({ from:
accounts[0] })
        setAlert('New record uploaded', 'success')
        setAddRecord(false)

        // refresh records
        const records = await contract.methods.getRecords(patientAddress).call({ from:
accounts[0] })
        setRecords(records)
      }
    } catch (err) {
      setAlert('Record upload failed', 'error')
      console.error(err)
    }
  },
  [addPatientAddress, accounts, contract]
)

if (loading) {
  return (
    <Backdrop sx={{ color: '#fff', zIndex: theme => theme.zIndex.drawer + 1 }}
open={loading}>
    <CircularProgress color='inherit' />
    </Backdrop>
  )
} else {
  return (
    <Box display='flex' justifyContent='center' width='100vw'>
      <Box width='60%' my={5}>
        {!accounts ? (
          <Box display='flex' justifyContent='center'>
            <Typography variant='h6'>Open your MetaMask wallet to get connected, then
refresh this page</Typography>
          </Box>
        ) : (
          <>
            {role === 'unknown' && (
              <Box display='flex' justifyContent='center'>
                <Typography variant='h5'>You're not registered, please go to home
page</Typography>
              </Box>
            )}
          </>
        )}
      </Box>
    </Box>
  )
}

```

```

    })
    {role === 'patient' && (
      <Box display='flex' justifyContent='center'>
        <Typography variant='h5'>Only doctor can access this page</Typography>
      </Box>
    )}
    {role === 'doctor' && (
      <>
        <Modal open={addRecord} onClose={() => setAddRecord(false)}>
          <AddRecordModal
            handleClose={() => setAddRecord(false)}
            handleUpload={addRecordCallback}
            patientAddress={searchPatientAddress}
          />
        </Modal>

        <Typography variant='h4'>Patient Records</Typography>
        <Box display='flex' alignItems='center' my={1}>
          <FormControl fullWidth>
            <TextField
              variant='outlined'
              placeholder='Search patient by wallet address'
              value={searchPatientAddress}
              onChange={e => setSearchPatientAddress(e.target.value)}
              InputProps={{ style: { fontSize: '15px' } }}
              InputLabelProps={{ style: { fontSize: '15px' } }}
              size='small'
            />
          </FormControl>
          <Box mx={2}>
            <CustomButton text='Search' handleClick={() => searchPatient()}>
              <SearchRoundedIcon style={{ color: 'white' }} />
            </CustomButton>
          </Box>
          <CustomButton text='New Record' handleClick={() => setAddRecord(true)}
            disabled={!patientExist}>
            <CloudUploadRoundedIcon style={{ color: 'white' }} />
          </CustomButton>
        </Box>

        {patientExist && records.length === 0 && (
          <Box display='flex' alignItems='center' justifyContent='center' my={5}>
            <Typography variant='h5'>No records found</Typography>
          </Box>
        )}

        {patientExist && records.length > 0 && (
          <Box display='flex' flexDirection='column' mt={3} mb={-2}>
            {records.map((record, index) => (
              <Box mb={2}>
                <Record key={index} record={record} />
              </Box>
            ))}
          </Box>
        )}

        <Box mt={6} mb={4}>
          <Divider />
        </Box>

        <Typography variant='h4'>Register Patient</Typography>
        <Box display='flex' alignItems='center' my={1}>
          <FormControl fullWidth>
            <TextField
              variant='outlined'
              placeholder='Register patient by wallet address'
              value={addPatientAddress}
              onChange={e => setAddPatientAddress(e.target.value)}
              InputProps={{ style: { fontSize: '15px' } }}
            />
          </FormControl>
        </Box>
      </>
    )}
  </Box>
)

```

```

        InputLabelProps={{ style: { fontSize: '15px' } }}
        size='small'
      />
    </FormControl>
    <Box mx={2}>
      <CustomButton text={'Register'} handleClick={() => registerPatient()}>
        <PersonAddAlt1RoundedIcon style={{ color: 'white' }} />
      </CustomButton>
    </Box>
  </Box>
</>
)}
</>
)}
</Box>
</Box>
)
}
}

export default Doctor

```

AddRecordModal.jsx

```

import React, { useState } from 'react'
import CustomButton from '../components/CustomButton'
import { DropzoneAreaBase } from 'material-ui-dropzone'
import { Box, Chip, IconButton, Typography } from '@mui/material'
import CloseRoundedIcon from '@mui/icons-material/CloseRounded'
import useAlert from '../contexts/AlertContext/useAlert'

const AddRecordModal = ({ handleClose, handleUpload, patientAddress }) => {
  const { setAlert } = useAlert()
  const [file, setFile] = useState(null)
  const [buffer, setBuffer] = useState(null)

  const handleFileChange = fileObj => {
    const { file } = fileObj
    setBuffer(null)
    setFile(file)
    console.log('file.name :>> ', file.name)

    const reader = new FileReader()
    reader.readAsArrayBuffer(file)
    reader.onloadend = () => {
      const buffer = Buffer.from(reader.result)
      setBuffer(buffer)
    }
  }

  return (
    <Box
      sx={{
        display: 'flex',
        justifyContent: 'center',
        alignItems: 'center',
        height: '100vh',
        width: '100vw',
      }}
    >
      <Box
        width='50vw'
        style={{
          backgroundColor: 'white',
          boxShadow: 24,
          borderRadius: 10,

```

```

    }}
    p={2}
    pr={6}
    pb={0}
    position='relative'
  >
    <Box position='absolute' sx={{ top: 5, right: 5 }}>
      <IconButton onClick={() => handleClose()}>
        <CloseRoundedIcon />
      </IconButton>
    </Box>
    <Box display='flex' flexDirection='column' my={1}>
      <Typography variant='h4'>Add Record</Typography>
      <Box my={2}>
        <DropzoneAreaBase
          onAdd={fileObjs => handleFileChange(fileObjs[0])}
          onDelete={fileObj => {
            setFile(null)
            setBuffer(null)
          }}
          onAlert={(message, variant) => setAlert(message, variant)}
        />
      </Box>
      <Box display='flex' justifyContent='space-between' mb={2}>
        {file && <Chip label={file.name} onDelete={() => setFile(null)} style={{
fontSize: '12px' }} />}
      <Box flexGrow={1} />
      <CustomButton
        text='upload'
        handleClick={() => handleUpload(buffer, file.name, patientAddress)}
        disabled={!file || !buffer}
      />
    </Box>
  </Box>
</Box>
)
}

export default AddRecordModal

```

84. Pada folder **patient** buatlah file baru dengan nama **index.jsx** dengan kode berikut.

```

index.jsx
import React, { useState, useEffect } from 'react'
import { Box, Typography, Backdrop, CircularProgress } from '@mui/material'
import useEth from '../../contexts/EthContext/useEth'
import Record from '../../components/Record'

const Patient = () => {
  const {
    state: { contract, accounts, role, loading },
  } = useEth()

  const [records, setRecords] = useState([])
  const [loadingRecords, setLoadingRecords] = useState(true)

  useEffect(() => {
    const getRecords = async () => {
      try {
        const records = await contract.methods.getRecords(accounts[0]).call({ from: accounts[0]
      })
        setRecords(records)
        setLoadingRecords(false)
      } catch (err) {
        console.error(err)
      }
    }
  }, [contract, accounts])
}

```

```

        setLoadingRecords(false)
      }
    }
    getRecords()
  })

  if (loading || loadingRecords) {
    return (
      <Backdrop sx={{ color: '#fff', zIndex: theme => theme.zIndex.drawer + 1 }}
open={loading}>
      <CircularProgress color='inherit' />
    </Backdrop>
  )
} else {
  return (
    <Box display='flex' justifyContent='center' width='100vw'>
      <Box width='60%' my={5}>
        <Backdrop sx={{ color: '#fff', zIndex: theme => theme.zIndex.drawer + 1 }}
open={loading}>
          <CircularProgress color='inherit' />
        </Backdrop>
        {!accounts ? (
          <Box display='flex' justifyContent='center'>
            <Typography variant='h6'>Open your MetaMask wallet to get connected, then refresh
this page</Typography>
          </Box>
        ) : (
          <>
            {role === 'unknown' && (
              <Box display='flex' justifyContent='center'>
                <Typography variant='h5'>You're not registered, please go to home
page</Typography>
              </Box>
            )}
            {role === 'doctor' && (
              <Box display='flex' justifyContent='center'>
                <Typography variant='h5'>Only patient can access this page</Typography>
              </Box>
            )}
            {role === 'patient' && (
              <>
                <Typography variant='h4'>My Records</Typography>

                {records.length === 0 && (
                  <Box display='flex' alignItems='center' justifyContent='center' my={5}>
                    <Typography variant='h5'>No records found</Typography>
                  </Box>
                )}

                {records.length > 0 && (
                  <Box display='flex' flexDirection='column' mt={3} mb={-2}>
                    {records.map((record, index) => (
                      <Box mb={2}>
                        <Record key={index} record={record} />
                      </Box>
                    )
                    )}
                  </Box>
                )}
              </>
            )}
          </>
        )}
      </Box>
    </Box>
  )
}
}
}
export default Patient

```

85. Masih pada folder **src** buatlah sebuah file dengan nama **routes.js** dan **ipfs.js** dengan isi kode berikut

routes.js

```
// Guards
import Layout from './components/layouts/Layout'
import AlertPopup from './components/layouts/AlertPopup'

// Pages
import Home from './pages'
import Patient from './pages/patient'
import Doctor from './pages/doctor'
import HeaderAppBar from './components/layouts/Layout'

const routes = [
  {
    path: '/',
    children: [
      {
        path: '',
        element: (
          <>
            <AlertPopup />
            <Home />
          </>
        ),
      },
      {
        path: 'patient',
        element: (
          <>
            <HeaderAppBar />
            <AlertPopup />
            <Patient />
          </>
        ),
      },
      {
        path: 'doctor',
        element: (
          <>
            <HeaderAppBar />
            <AlertPopup />
            <Doctor />
          </>
        ),
      },
    ],
  },
]

export default routes
```

ipfs.js

```
const ipfsClient = require('ipfs-http-client')
```

```

const projectId = '2DKPh21wsRUiB27R526E0Co4eUV'
const projectSecret = '1bc14260235b4fb8844b4d86b0113876'
const auth = 'Basic ' + Buffer.from(projectId + ':' + projectSecret).toString('base64')

const ipfs = ipfsClient({
  host: 'ipfs.infura.io',
  port: 5001,
  protocol: 'https',
  headers: {
    authorization: auth,
  },
})

export default ipfs

```

86. Pada folder **pages** buatlah sebuah file dengan nama **index.js** dengan kode sbagai berikut.

```

index.jsx
import { Box, Typography, Backdrop, CircularProgress, Divider } from '@mui/material'
import React from 'react'
import AccountBalanceWalletRoundedIcon from '@mui/icons-material/AccountBalanceWalletRounded'
import VideoCover from 'react-video-cover'
import BackgroundVideo from '../assets/BackgroundVideo.mp4'
import logo from '../assets/tealNoBG-cropped.png'
import useEth from '../contexts/EthContext/useEth'
import PersonAddAlt1RoundedIcon from '@mui/icons-material/PersonAddAlt1Rounded'
import CustomButton from '../components/CustomButton'
import { useNavigate } from 'react-router-dom'
import LoginRoundedIcon from '@mui/icons-material/LoginRounded'
import { grey } from '@mui/material/colors'
import '../App.css'

const Home = () => {
  const {
    state: { contract, accounts, role, loading },
    dispatch,
  } = useEth()
  const navigate = useNavigate()

  const registerDoctor = async () => {
    try {
      await contract.methods.addDoctor().send({ from: accounts[0] })
      dispatch({
        type: 'ADD_DOCTOR',
      })
    } catch (err) {
      console.error(err)
    }
  }

  const ActionSection = () => {
    if (!accounts) {
      return (
        <Typography variant='h5' color='white'>
          Open your MetaMask wallet to get connected, then refresh this page
        </Typography>
      )
    }
  }

```



```

    } else {
      if (role === 'unknown') {
        return (
          <Box display='flex' flexDirection='column' alignItems='center'>
            <Box mb={2}>
              <CustomButton text='Doctor Register' handleClick={() =>
registerDoctor()}>
                <PersonAddAlt1RoundedIcon style={{ color: 'white' }} />
              </CustomButton>
            </Box>
            <Typography variant='h5' color='white'>
              If you are a patient, ask your doctor to register for you
            </Typography>
          </Box>
        )
      } else if (role === 'patient') {
        return (
          <CustomButton text='Patient Portal' handleClick={() =>
navigate('/patient')}>
            <LoginRoundedIcon style={{ color: 'white' }} />
          </CustomButton>
        )
      } else if (role === 'doctor') {
        return (
          <CustomButton text='Doctor Portal' handleClick={() =>
navigate('/doctor')}>
            <LoginRoundedIcon style={{ color: 'white' }} />
          </CustomButton>
        )
      }
    }
  }
}

if (loading) {
  return (
    <Backdrop sx={{ color: '#fff', zIndex: theme => theme.zIndex.drawer + 1 }}
open={loading}>
      <CircularProgress color='inherit' />
    </Backdrop>
  )
} else {
  return (
    <Box
      display='flex'
      flexDirection='column'
      justifyContent='center'
      alignItems='center'
      width='100vw'
      height='100vh'
      id='background'
    >
      <Box
        style={{
          position: 'absolute',
          width: '100%',
          height: '100%',
          overflow: 'hidden',
          top: 0,
          left: 0,
          zIndex: -1,
        }}
      >

```

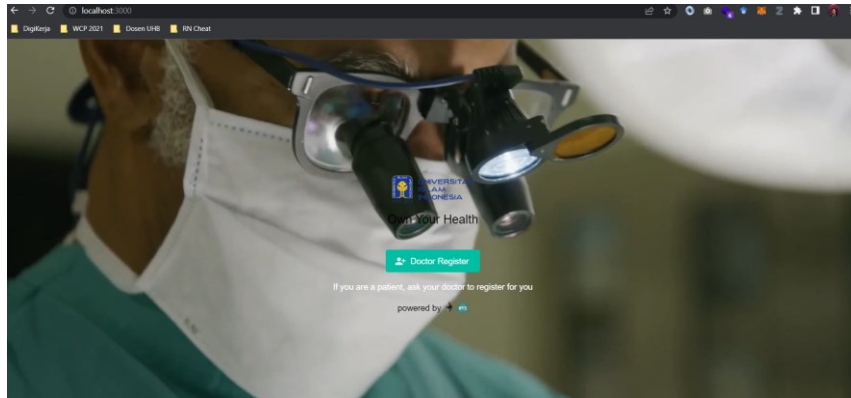
```

    >
    { /* <VideoCover
      videoOptions={{
        src: BackgroundVideo,
        autoPlay: true,
        loop: true,
        muted: true,
      }}
    } /> */
  </Box>
  <Box id='home-page-box' display='flex' flexDirection='column'
justifyContent='center' alignItems='center' p={5}>
    <img src={logo} alt='med-chain-logo' style={{ height: 50 }} />
    <Box mt={2} mb={5}>
      <Typography variant='h4' color='black'>
        Own Your Health
      </Typography>
    </Box>
    <ActionSection />
    <Box display='flex' alignItems='center' mt={2}>
      <Typography variant='h5' color='black'>
        powered by{' '}
      </Typography>
      <Box mx={1}>
        <img
          src='https://cdn.worldvectorlogo.com/logos/ethereum-1.svg'
          alt='Ethereum logo vector'
          style={{ height: 20 }}
        ></img>
      </Box>
      <img
        src='https://upload.wikimedia.org/wikipedia/commons/1/18/Ipfs-logo-
1024-ice-text.png'
        alt='Ethereum logo vector'
        style={{ height: 20 }}
      ></img>
    </Box>
  </Box>
)
}
}

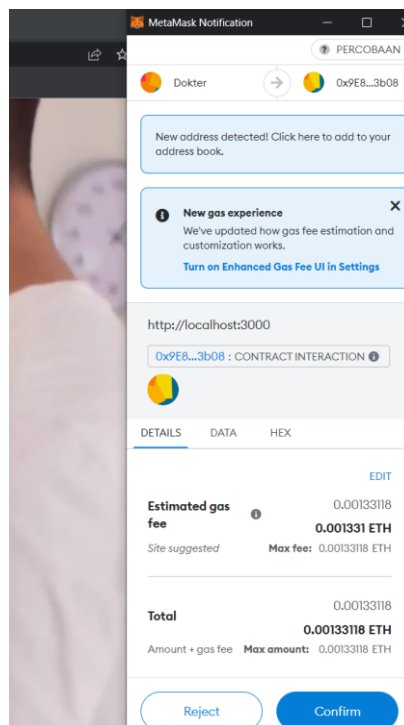
export default Home

```

87. Sekarang kita akan **jalankan** web appsnya dengan cara mengetikan perintah **npm start** pada terminal.
88. Kita kemudian diarahkan pada url **localhost:3000**
89. Secara otomatis wallet Metamask kita akan aktif dan kita disuruh memilih akun yang akan digunakan untuk menjalankan aplikasi.
90. Jika berhasil maka anda akan melihat tampilan sebagai berikut ini



91. Jika diklik Doctor Portal maka metamask akan aktif. Pastikan anda sudah mengimport 2 akun yaitu Dokter dan pasien dari Ganache.
92. Sekarang kita coba klik Doctor Register.
93. Secara otomatis metamask akan terbuka, saat ini di wallet kita aktifkan akun docter.
94. Aplikasi metamask selanjutnya akan melakukan konfirmasi di wallet.



95. Jika kita klik Confirm maka dokter akan terdaftar di Blockchain.
96. Selanjutnya kita akan reload aplikasi dan kemudian secara otomatis aplikasi ini menampilkan menu Dokter Portal.
97. Klik doctor portal dan aplikasi akan diarahkan ke tampilan portal.

0xACF289fc56a799A3b55FE7E93B9Bd7bF9b10c062 doctor

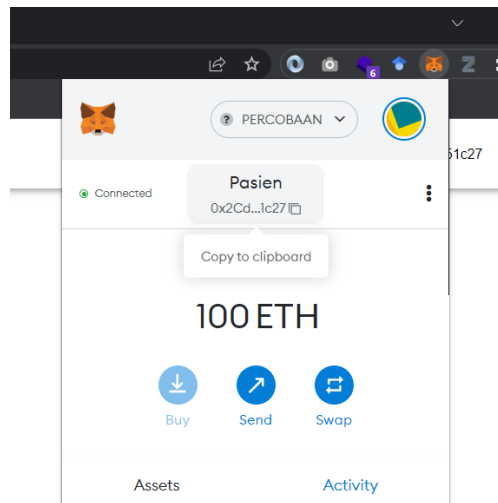
Patient Records

Search
New Record

Register Patient

Register

98. Sekarang kita akan menambahkan Pasien, caranya kita harus sudah memiliki akun pasien dari public keynya.
99. Kita dapat copy atau ambil dari metamask pasien.

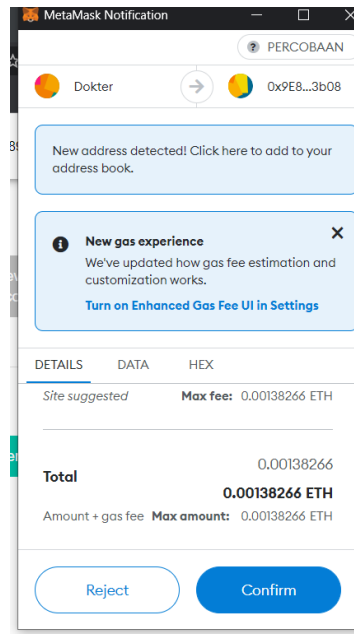


100. Pilih copy to clipboard dan sekarang anda sudah memiliki public key pasien.
101. Pindah kembali ke akun doctor dan tambahkan pasien

Register Patient

Register

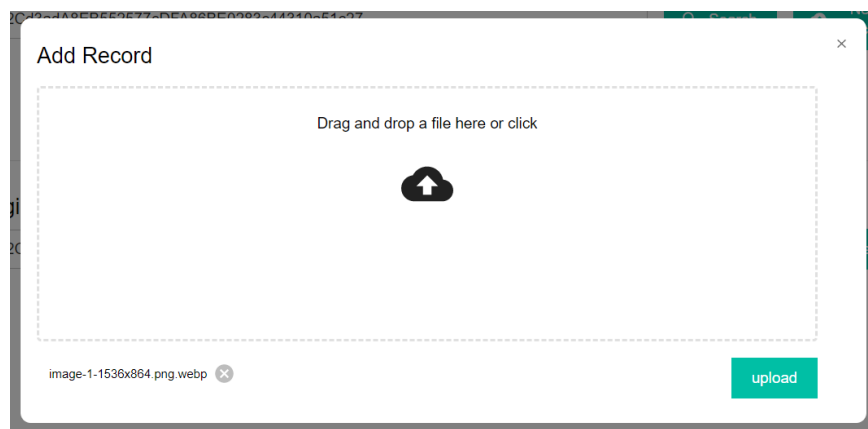
102. Klik register dan anda sekarang bisa menambahkan data pasien.
103. Metamask akan aktif untuk melakukan konfirmasi



104. Klik confirm untuk menambahkan pasien.
105. Masukan public key pasien pada kolom pencarian pasien.

Patient Records

106. Sekarang pilih New Record untuk menambahkan rekam medis.
107. Pilih salah satu file untuk diupload



108. Tunggu proses upload ke IPFS
109. Sekarang file rekam medis sudah bisa dilihat.

✔ New record uploaded

Patient Records

0x2Cd3adA8EB552577cDFA86BE0283c44310a51c27

Search

New Record

Record name

image-1-1536x864.png.webp

Doctor

0xACF289fc56a799A3b55FE7E93B9Bd7bF9b10c062

Created time

09-09-2022 23:03

110. Jika kita cek di Ganache maka blok akan terus bertambah.

ACCOUNTS	BLOCKS	TRANSACTIONS	CONTRACTS	EVENTS	LOGS	SEARCH FOR BLOCK NUMBERS OR TX HASHES	
CURRENT BLOCK 5	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK MUIRGLACIER	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:7545	MINING STATUS AUTOMINING	WORKSPACE PERCOBAAN
							SWITCH
BLOCK 5	MINED ON 2022-09-09 23:03:35				GAS USED 195911	1 TRANSACTION	
BLOCK 4	MINED ON 2022-09-09 23:00:44				GAS USED 24173	1 TRANSACTION	
BLOCK 3	MINED ON 2022-09-09 23:00:39				GAS USED 46089	1 TRANSACTION	
BLOCK 2	MINED ON 2022-09-09 22:55:37				GAS USED 44373	1 TRANSACTION	
BLOCK 1	MINED ON 2022-09-09 22:35:02				GAS USED 1505943	1 TRANSACTION	
BLOCK 0	MINED ON 2022-09-09 22:24:29				GAS USED 0	NO TRANSACTIONS	

111. Sekarang kita akan coba pindah ke sisi Pasien. Pada metamask pilih akun pasien pada url <http://localhost:3000/patient>

112. Sekarang anda bisa melihat data rekam medis anda.

My Records

Record name

image-1-1536x864.png.webp

Doctor

0xACF289fc56a799A3b55FE7E93B9Bd7bF9b10c062

Created time

09-09-2022 23:03