**Decentralized Patient Record Storage**

The "**Decentralized Patient Record Storage**" project aims to leverage block chain technology to create a secure, transparent, and tamper-resistant system for storing and managing patient health records. The patient record is one of the important assets that is currently centralized, maintained, and managed by hospitals.

The smart contract defines a “**PatientRecord**” contract to store patient, doctors and medicines details.

Explanation of PatientRecord Smart Contract

1. Struct
2. User Roles and Modifier
3. Events
4. Mapping
5. State Variable
6. Constructor
7. Function
8. **Struct Details -**

These are the struct we have used in the PatientRecord smart contract.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # |  | Struct Name | | Remarks |
| 1. | Patient | | | It is used to store the patient related data along with diseases, authorized doctors and prescribed medicines. |
| **Data Type** | **Variable Name** | **Description** |
| string | firstName | Store First Name of the Patient. |
| string | lastName | Store Last Name of the Patient. |
| uint8 | age | Store the age of the Patient. |
| string[] | diseases | Store the list of disease. |
| address[] | authorizedDoctorAddresses | Store the list of authorized doctor address. |
| mapping(address => bool) | authorizedDoctors | Store the list of authorized doctors. |
| mapping(address => Medicine[]) | patientMedicines | Store the list of prescribed medicine. |
|  | bool | isRegistered | Check patient is registered or not. |  |
| 2. | **Medicine** | | | It is used to store the medicine details. |
| **Data Type** | **Variable Name** | **Description** |
| uint | id | Store Id of the medicine. |
| string | name | Store name of the medicine. |
| string | expiryDate | Store expiry date of a medicine. |
| string | dose | Store dosage of the medicine |
| uint | price | Store price of the medicine. |
| 3. | **Doctor** |  |  | It is used to store the doctor details and patient alignment to a doctor. |
| **Data Type** | **Variable Name** | **Description** |
| bool | isRegistered | Check doctor is registered or not. |
| uint | doctorId | Store the id of the doctor. |
| string | firstName | Store the first name of the doctor. |
| string | lastName | Store the last name of the doctor. |
| string | contactDetails | Store contact details of the doctor. |
| string | specialization | Store the doctor’s specialization. |
| string | workPlace | Store the working location of a doctor. |
| address[] | patients | Store the address of the patient assigned to a doctor. |

1. **User Roles and modifier**

We have used the below role and modifiers while developing this smart contract

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Role Name | Access | | |
| 1. | onlyPortalAdmin | It is accessible by portal Admin | | |
| **#** | **Function** | |
| 1. | addMedicine | |
| 2. | onlyPatient | It is accessible by a registered patient. | | |
| **#** | | **Function** |
| 1. | | addDisease |
| 2. | | authorizeDoctor |
| 3. | | getPrescribedMedicine |
| 4. | | updatePatientData |
| 3. | onlyDoctor | It is accessible by a registered doctor | | |
| # | | Function |
| 1. | | prescribeMedicine |
| 2. | | getPatientByDoctor |

1. **Events -**  We have used the underneath events in the PatientRecord Smart Contract to log and notify external systems.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Events Name | | Remarks |
| 1. | **DoctorRegistered** | | This event is emitted when a doctor is successfully registered in the system. It provides information about the registered doctor, such as their unique ID, first name, and last name. |
| **#** | **Parameters** |
| **1.** | **“uint indexed doctorId”:** The unique identifier of the registered doctor. The indexed keyword allows efficient filtering based on this parameter. |
| **2.** | **“string firstName” :** The first name of the registered doctor. |
| **3.** | **“string lastName” :** The last name of the registered doctor. |
| 2. | **PatientRegistered** | | This event is emitted when a patient is successfully registered in the system. It provides information about the registered patient, such as their Ethereum address, first name, and last name. |
| **#** | **Parameters** |
| **1.** | **“address indexed patientAddress”:** The Ethereum address of the registered patient. The indexed keyword allows efficient filtering based on this parameter. |
| **2.** | **“string firstName” :** The first name of the registered patient. |
| **3.** | **“string lastName” :** The last name of the registered patient. |
| 3. | **DiseaseAdded** | | This event is emitted when a disease is added to a patient's record. It informs external systems that a new disease has been added to a patient's medical history. |
| **#** | **Parameters** |
| **1.** | **“address indexed patientAddress”** : The Ethereum address of the patient for whom the disease was added. |
| **2.** | **“string disease”:** The name of the disease that was added. |
| 4. | **PatientAuthorized** | | This event is emitted when a patient authorizes a doctor to access their medical records. It informs external systems that a patient has authorized a doctor. |
| **#** | **Parameters** |
| **1.** | **“address indexed patientAddress” :** The Ethereum address of the patient who granted the **authorization.** |
| **2.** | **“address indexed doctorAddress”** : The Ethereum address of the doctor who was granted access. |
| 5. | **MedicineAdded** | | This event is emitted when a new medicine is added to the system. It provides information about the added medicine, such as its unique ID, name, expiry date, dose, and price. |
| **#** | **Parameters** |
| **1.** | **“uint indexed medicineId**”: The unique identifier of the newly added medicine. |
| **2.** | **“string name**” : The name of the newly added medicine. |
| **3.** | **“string name**” : The name of the newly added medicine. |
| **4.** | **“string dose”** : The dose prescribed for the medicine. |
| **5.** | **“uint price”** : The price of the newly added medicine. |
| 6. | **MedicinePrescribed** | | This event is emitted when a doctor prescribes a medicine for a patient. It provides information about the doctor, patient, and the prescribed medicine. |
| **#** | **Parameters** |
| **1.** | **“address indexed doctorAddress**” : The Ethereum address of the doctor who prescribed the medicine. |
| **2.** | **“address indexed patientAddress**” : The Ethereum address of the patient for whom the medicine was prescribed. |
| **3.** | **“string medicineName**” : The name of the prescribed medicine. |

1. **Mapping** – we have used the below mapping to store the patient, doctors and medicine details.

|  |  |  |
| --- | --- | --- |
| # | Mapping Name | Description |
| 1. | patients | This mapping stores patient details based on their Ethereum addresses. Each patient's address is used as the key to retrieve their corresponding **Patient** struct, which contains information about the patient, including registration status, name, diseases, authorized doctors, and medicines. This mapping allows efficient access to patient information without the need for iteration. |
| 2. | doctors | Similar to the **patients** mapping, this mapping stores doctor details based on their Ethereum addresses. Each doctor's address is used as the key to retrieve their corresponding **Doctor** struct, which likely contains information about the doctor's credentials, specialization, and more. Just like with patients, this mapping enables quick access to doctor information. |
| 3. | medicines | This mapping stores medicine details based on unique identifiers. Each medicine's ID is used as the key to retrieve its corresponding **Medicine** struct, which holds information about the medicine, such as its name, expiry date, dose, and price. This mapping is useful for efficiently retrieving information about specific medicines without the need for iteration. |

1. **State Variable**

|  |  |  |
| --- | --- | --- |
| # | Variable Name | Description |
| 1. | portalAdmin | This state variable represents the Ethereum address of the portal's admin. It is declared as **public**, which means it will automatically generate a getter function to retrieve the value of this variable. The portal admin is likely responsible for managing the overall functionality and administration of the decentralized patient record system. |
| 2. | nextDoctorId | This state variable holds the identifier that will be assigned to the next registered doctor. By using this variable, you can ensure that each newly registered doctor receives a unique ID. It is declared as **public**, allowing anyone to read the value. |

1. **Constructor:** The constructor we have provided initializes the state variables of our smart contract when it is deployed. When the smart contract is deployed to the Ethereum block chain, this constructor is automatically executed, setting the initial values for the **portalAdmin** and **nextDoctorId** state variables. This ensures that the smart contract is properly configured and ready to use.
2. **Function**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Parameters | Description |
| 1. | registerDoctor | string memory \_firstName, string memory \_lastName, string memory \_contactDetails,  string memory \_specialization,  string memory \_workPlace,  address \_doctorAddress | This function allows registration of a doctor in the system. It verifies that the doctor is not already registered and assigns them a unique ID. It then emits the **DoctorRegistered** event to notify the network about the registration. |
| 2. | getDoctor | address \_doctorAddress | This function retrieves and returns information about a registered doctor using their Ethereum address. |
| 3. | registerPatient | string memory \_firstName,  string memory \_lastName,  uint8 \_age,  address \_patientAddress | This function allows registration of a patient in the system. It verifies that the patient is not already registered and initializes their basic details. It emits the **PatientRegistered** event to inform the network about the registration. |
| 4. | updatePatientData | string memory \_newFirstName,  string memory \_newLastName,  uint8 \_newAge | This function allows patients to update their personal information such as first name, last name, and age. |
| 5. | addDisease | string memory disease | Patients can use this function to add a disease to their medical history. The function emits the **DiseaseAdded** event to signal the addition of a new disease. |
| 6. | getPatientData | address \_patientAddress | This function retrieves patient information, including personal details, diseases, and authorized doctor addresses. |
| 7. | authorizeDoctor | address \_doctorAddress | Patients can authorize doctors to access their medical records. It also updates the patient's authorized doctor list and the doctor's patient list. |
| 8. | addMedicine | uint \_medId,  string memory \_name, string memory \_expiryDate,  string memory \_dose, uint \_price | This function allows the admin to add a new medicine to the system. It emits the **MedicineAdded** event to indicate the addition of a new medicine. |
| 9. | getMedicine | uint \_medId | This function retrieves information about a specific medicine based on its ID. |
| 10. | getPatientByDoctor | address \_doctorAddress | Doctors can use this function to retrieve a list of patients they are authorized to access. |
| 11. | prescribeMedicine | address \_patientAddress,  uint \_medId,  string memory \_medicineName,  string memory \_expiryDate,  string memory \_dose, uint \_price | Doctors can prescribe medicine to patients. The function verifies authorization, adds the prescribed medicine to the patient's list, and emits the **MedicinePrescribed** event. |
| 12. | getPrescribedMedicine | address \_patientAddress | Patients can use this function to retrieve the list of medicines prescribed to them. |