|  |
| --- |
| CIS 600 – Blockchain and Cryptocurrencies |
| Blockchain Technology for Electronic Health Records |
| Final Report |

|  |
| --- |
| -Krupa Mavani (351960088)  -Vinu Kundnani (470316255)  -Sonali Ratnam (600148631) |

Table of Contents

1. Objective 2

2. Design2

2.1 Model3

2.2 Flow Chart3

3. Use Cases4

4. Implementation6

4.1 Features6

4.2 Technical Details7

4.3 Proof of Concept7

5. Advanced Features11

6. Novelty11

7. Output Explanation12

8. Output Screenshots12

9. Future Scope19

10. References19

1. **Objective**

One of the biggest problems in healthcare today is that healthcare organizations must maintain multiple and fragmented health records for the patients. Our project aims to solve this problem by putting all the medical records as transactions on the blockchain to create a smart healthcare ecosystem. Our goal is to create a user focused electronic health record system whilst maintaining a single true version of the user’s data.

With this idea, we intend to explore and learn the capabilities of blockchain. So, we focused on implementing many features like smart contracts, hashing technique, time limited access, logging data on blockchain, making payments through smart contracts.

One important point to note here is that, smart contracts have mostly been used as a decentralized method of payment in many applications. But here we are exploiting capabilities of smart contracts by using them not only as a payment method but also as an electronic storage system for patient’s data.

1. **Design**

Our system will have 4 basic entities in health care system. There will be one smart contract between all the entities. All the communication between entities will happen as blockchain transactions. So, any transaction sent on blockchain serves as a proof of communication between entities. This ensures a trust factor among entities in the system.

System has following entities:

1. Patient:
   1. Upload his health data as a transaction on the blockchain which can be used by the doctors to review and provide the feedback or the prescription.
   2. Patient pays the doctor for his service.
   3. Patient can give claim insurance money using his payment transaction as a proof of payment.
   4. Patient can give access of his data to research lab for research purposes.

2. Doctor: Upload patient’s health records on the blockchain.

3. Insurance Company: Release insurance money upon verifying the health records.

4. Research Lab: Pay the patient for accessing his health records.

* 1. **Model**

Fig 2.1 Model of Healthcare System

* 1. **Flowchart**

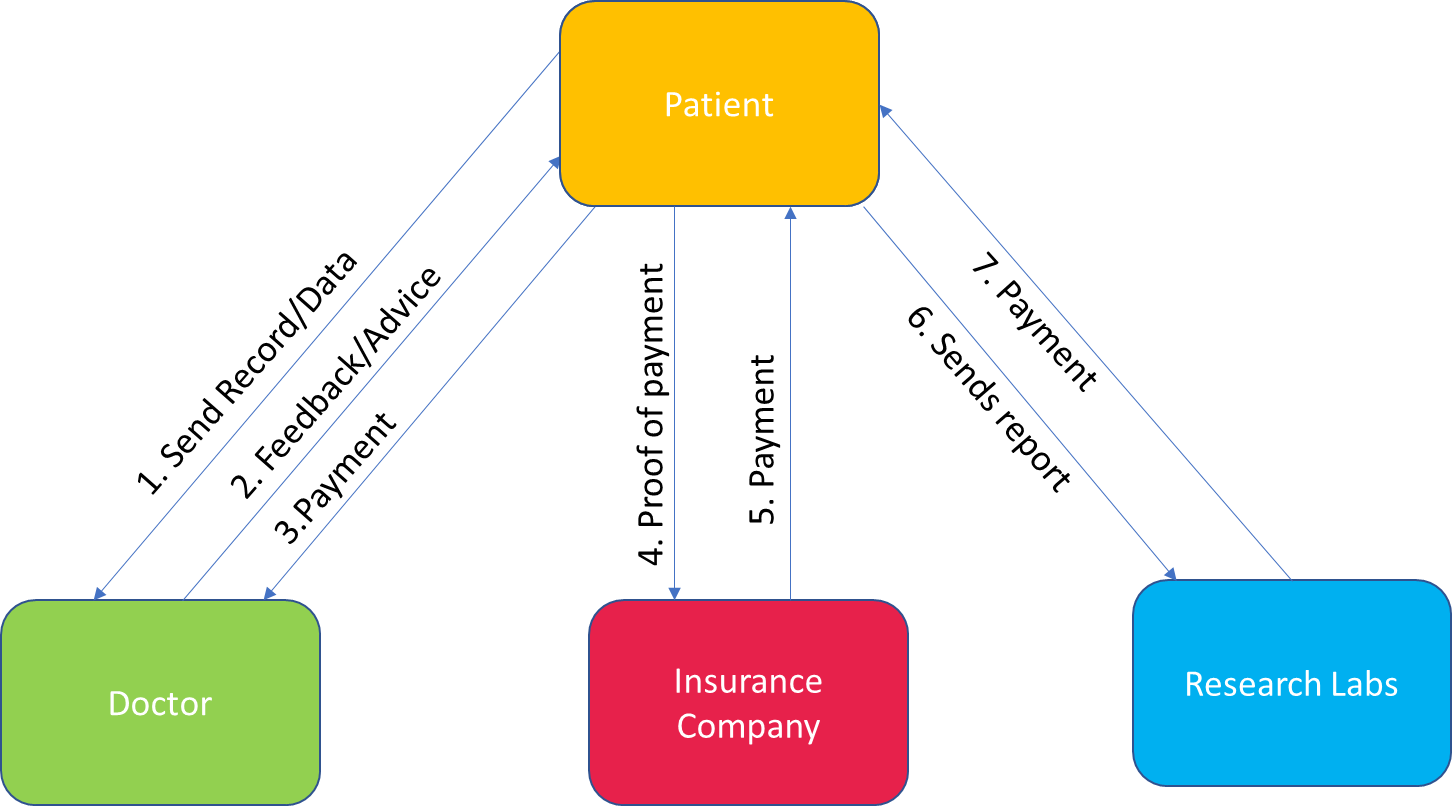
****

Fig 2.2 Flowchart of Healthcare System

1. **Use Cases**

In this section we are showing the real-life scenarios which are handled by our application.

**Use case 1** - Communication between Patient and Doctor.

Real-life scenario: Patient shares his data to doctor and ask for advice. In return of advice, he pays the doctor fees.

Some examples of patient data:

1. Patient can share his BMI report and doctor can suggest some diets or exercises if needed.
2. Patient can share his sugar readings/blood pressure readings/pulse rate readings/etc. and doctors can suggest some tests or prescribe some medicines if needed.
3. Patient can share any symptoms and doctor can suggest him to have a physical check-up done or take tests or ask him to personally visit the doctor for detailed checkup or prescribe medicines.

Our implementation: Patient will send his data to Doctor as a blockchain transaction. Doctor will provide his advice on the received data as a blockchain transaction. In return of doctor’s advice, patient will pay doctor’s fees as a blockchain transaction.

Advantages: As the transactions are part of a smart contract, doctor is sure to receive payment for his feedback, and patient is sure to receive feedback for the payment he does. As hashing is used, patient’s data will be confidential on blockchain.

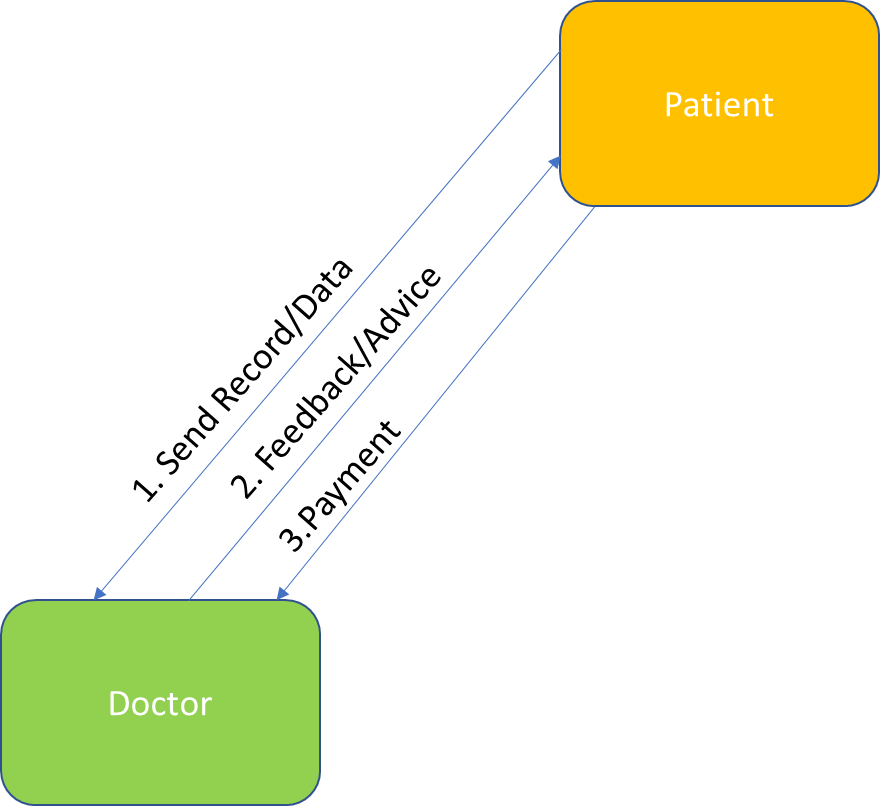


Fig. 3.1 Communication between Patient and Doctor

**Use Case 2** - Communication between Patient and Insurance Company.

Real-life scenario: Patient claims the fees that he has paid the doctor from the insurance company. Patient uses the fees receipt as a proof of payment. Insurance company reimburses the amount that patient is eligible to receive for that claim by using the receipt as a proof or it can reject the claim.

Our implementation: Patient can claim the fees that he has paid to doctor from the insurance company by sending a blockchain transaction. Insurance company can verify the proof and can either accept or reject a claim. If accepted, insurer will reimburse the patient by sending a blockchain transaction. If rejected, insurer will send a transaction to patient to let them know that claim is rejected with a valid reason.

Advantages: Patient can use his fess payment transaction as a proof of payment with the insurance company instead of submitting a separate fees receipt. As hashing is used, patient’s payment details will be confidential on blockchain.

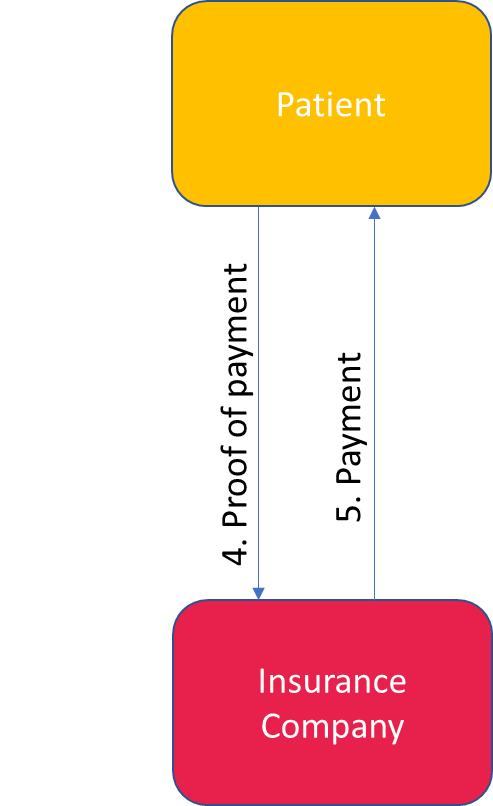


Fig. 3.2 Communication between Patient and Insurance Company

**Use Case 3 –** Communication between Patient and Research Lab.

Real-life scenario: Usually, in today’s scenario, majority of the patients are not aware of research labs and how they operate or how to connect with them or how they can be benefitted by such labs. Research labs on the other hand depend on hospitals and doctors to get patients’ data for conducting their research. This happens without patient’s consent in most cases as patients are not aware that their data is being shared to someone else. This in some kind is a violation of the patients’ right to privacy. Also, the patient is not getting any benefit out of it. Instead, hospitals and doctors are making profit by sharing patients’ data and getting paid by research labs.

Our implementation: This part of our system is unique as it is not addressed by many other available systems. This part of our system is focusing on solving the above real-life problem by making use of blockchain. Patient can provide access to his records to the research labs as per his wish by sending a transaction. Research labs will pay the patient for providing his records by sending a transaction.

Advantages: As the transactions are part of a smart contract, patient is sure to receive payment for his record sharing, and research lab is sure to receive patient records for the payment they make. As hashing is used, patient’s data will be confidential on blockchain. The important point here is that, Patient himself will be benefitted for sharing his records instead of some third-party profiting from it.

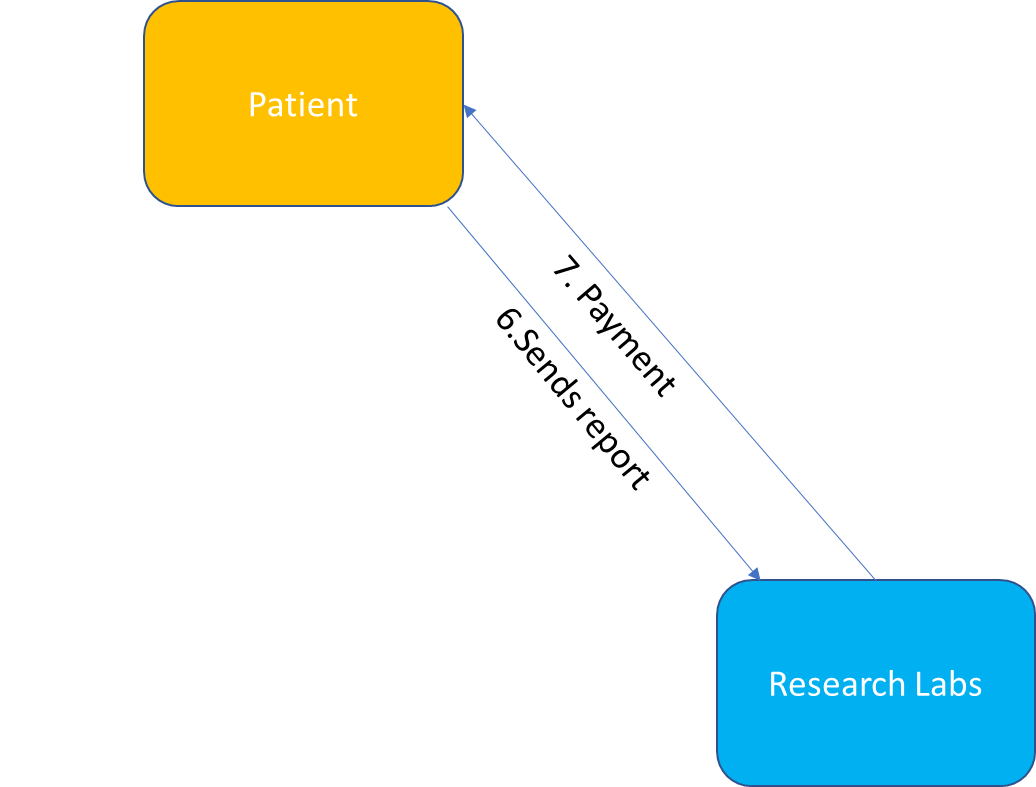


Fig. 3.3 Communication between Patient and Research Labs

1. **Implementation**

**4.1 Features**

1. We are using blockchain as an **electronic medium** to store heath records. It is like a replacement of traditional database systems.
2. We are using smart contracts feature of blockchain to **decentralize** the health system.
3. We are using smart contracts as **payment** **method** between entities.
4. We are using hashing technique of blockchain to provide for **security**.
5. We are adding **time limited access** feature to avoid misuse of data.
6. We are using **logging** concept of blockchain to pass data between entities.

**4.2 Technical details**

1. We will use Ethereum blockchain.
2. We will use Smart Contracts feature of blockchain to implement the system.
3. Ethers will be the currency of payment.
4. We will use Solidity as the Programming language.
5. We will use Remix IDE for development.
6. We will use ‘struct’ data structure of contracts to store data that is to be sent in transaction.

*(We have implemented this feature in the current application of Rock, Paper, Scissor that we did in Lab 2 as a Proof of Concept. We will extend this feature on our system during development).*

Note: Other alternatives for sending data that are available:

* 1. Transaction data - We can store the data that is to be sent in the data field of the transaction.
  2. Map/Arrays - We can use Map/Arrays data structure of contracts to store and send data in transaction.

1. We will use SHA-3 hashing technique for data encryption and decryption as it cheaper in terms of gas usage compared to other available techniques but less secure than them. SHA-3 costs 30 gas + 6 gas for each word (rounded up) for input data to a SHA3 Keccak-256 operation.

Note: Other hashing techniques available:

* 1. SHA-256- It costs 60 gas + 12 gas for each word (rounded up) for input data to a SHA2-256 operation.
  2. Ripemd- It costs 600 gas + 120 gas for each word (rounded up) for input data to a RIPEMD-160 operation.

**4.3 Proof of Concept:**

We have extended the Lab 2 code to show it is possible to use struct, arrays and security. Since, we are familiar with the Lab 2 implemented in an easy way, we are demonstrating how advanced features can be incorporated in the same code. These concepts will be extended to be used on our system during development.

1. Using ‘struct’ as data structure

Storing choice of player:

In lab2, we stored the choice in the string. However, for proof of concept, we have made a **struct** that would store the hashKey and choice of each player. Furthermore, we have also used mapping of address and the struct for each player.

1. Security

Using sha3:

We are using sha3 to encrypt and store the choice of each player through mapping. Since sha3 is one-way hashing, hence we have stored the key as well as choice as string in struct. This is just to show that we are aware how to integrate security in our system, where we will store the encrypted data and its key for each transaction.

|  |
| --- |
| *pragma solidity ^ 0.4.13;*  *contract RockPaperScissorsGame*  *{*  *//PROOF\_OF\_CONCEPT\_FOR\_FINAL\_PROJECT*  *struct storekey{*  *bytes32 key;*  *string choice;*  *}*  *mapping (string => mapping(string => int)) payoffMatrix;*    *//PROOF\_OF\_CONCEPT\_FOR\_FINAL\_PROJECT*  *mapping (address => storekey[2]) public stores;*    *address firstPlayer;*  *address secondPlayer;*  *bytes32 enctrypedChoiceFirst;*  *bytes32 enctrypedChoiceSecond;*  *string public firstPlayerChoice;*  *string public secondPlayerChoice;*  *function RockPaperScissorsGame()*  *{ // constructor decides which one wins*  *payoffMatrix["rock"]["rock"] = 0;*  *payoffMatrix["rock"]["paper"] = 2;*  *payoffMatrix["rock"]["scissors"] = 1;*  *payoffMatrix["paper"]["rock"] = 1;*  *payoffMatrix["paper"]["paper"] = 0;*  *payoffMatrix["paper"]["scissors"] = 2;*  *payoffMatrix["scissors"]["rock"] = 2;*  *payoffMatrix["scissors"]["paper"] = 1;*  *payoffMatrix["scissors"]["scissors"] = 0;*  *}*    *// requirement 1 contract address where both the players deposit the initial amount*  *function getContractBalance () constant returns (uint amount)*  *{*  *return this.balance;*  *}*    *// requirement 2 each player deposits an initial amount of 5 Ethers into the owner account.*  *modifier verifyMinimumAmount(uint amount)*  *{*  *if (msg.value < amount)*  *throw;*  *else*  *\_;*  *}*    *modifier checkRegistered()*  *{*  *if (msg.sender == firstPlayer || msg.sender == secondPlayer)*  *throw;*  *else*  *\_;*  *}*    *//requirement 3 once both the players deposit the money, allow them to play. While depositing the money, make sure you keep track of who is depositing and make him/her the player1 or player2 accordingly.*  *function registerPlayer() payable verifyMinimumAmount(5) checkRegistered()*  *{*  *if (firstPlayer == 0)*  *firstPlayer = msg.sender;*  *else if (secondPlayer == 0)*  *secondPlayer = msg.sender;*  *}*    *//requirement 4 Write a function play which takes the string parameter (Choice of the player - Rock, paper, scissors) and consider their choice only if they have deposited successfully.*    *//PROOF\_OF\_CONCEPT\_FOR\_FINAL\_PROJECT*  *function play(string choice) returns (int w)*  *{*  *if (msg.sender == firstPlayer){*  *storekey memory s = storekey(sha3(choice), choice);*  *//stores[msg.sender].key = sha3(choice);*  *//stores[msg.sender].choice = choice;*  *firstPlayerChoice = choice;*  *}*  *else if (msg.sender == secondPlayer){*    *secondPlayerChoice = choice;*  *}*    *if (checkBothNotNull())*  *{*    *int winner = payoffMatrix[firstPlayerChoice][secondPlayerChoice];*  *if (winner == 1)*  *firstPlayer.send(this.balance);*  *else if (winner == 2)*  *secondPlayer.send(this.balance);*  *else*  *{*  *firstPlayer.send(this.balance/2);*  *secondPlayer.send(this.balance);*  *}*    *// unregister players and choices*  *firstPlayerChoice = "";*  *secondPlayerChoice = "";*  *firstPlayer = 0;*  *secondPlayer = 0;*  *return winner;*  *}*  *else*  *return -1;*  *}*    *//validate choices*  *function checkBothNotNull() constant returns (bool x)*  *{*  *if(bytes(firstPlayerChoice).length != 0 && bytes(secondPlayerChoice).length != 0)*  *{*  *return true;*  *}*  *else*  *return false;*  *}*    *// get winner*  *function decideWinner() constant returns (int x)*  *{*  *return payoffMatrix[firstPlayerChoice][secondPlayerChoice];*  *}*  *function getMyBalance () constant returns (uint amount)*  *{*  *return msg.sender.balance;*  *}*    *//PROOF\_OF\_CONCEPT\_FOR\_FINAL\_PROJECT*  *function getMyChoice () constant returns (string choice){*  *if(msg.sender == firstPlayer)*  *{*  *if(stores[msg.sender].key == sha3(stores[msg.sender].choice))*  *{*  *return (stores[msg.sender].choice);*  *}*  *}*  *else if (msg.sender == secondPlayer)*  *{*  *if(stores[msg.sender].key == sha3(stores[msg.sender].choice))*  *{*  *return (stores[msg.sender].choice);*  *}*  *}*  *}*  *}* |

1. **Advanced Features**
2. Security - All the transactions on blockchain are visible to everyone. Therefore, we want to implement an additional feature of using a hashing technique to encrypt data and then send in a transaction so that it is secure and visible only to target audience. This feature will help in preventing the misuse of any data. *(We have implemented this feature in the current application of Rock, Paper, Scissor that we did in Lab 2 as a Proof of Concept. We will extend this feature on our system during development).*
3. Time limit - All the transactions on the blockchain are available to everyone for the whole time. Therefore, we want to implement a new feature of time limited access. This feature will give access of any data to target audience only for a pre-defined amount of time decided by the owner. This feature will help in preventing the misuse of any data. *(We did some research on this topic and we think it is feasible to implement it. So, we are planning to implement it on our system during development).*
4. Front end - Our system will be implemented in Remix and it can be run from Remix IDE. But it is possible to integrate the back-end solidity code with front end.

There are 2 possible ways:

1. Running Ethereum on local node and setting the environment value in the Remix IDE to web3 provider which then connects to the localhost where Ethereum node is running and incorporate it with NodeJS application that uses web3js library.
2. We can simply install the Metamask which will run the Ethereum node in the browser itself and incorporate it with NodeJS application that uses web3js library.

*(We did some research on this topic and we think it is feasible to implement it. But we will give priority to other features of the system so that the system is fully functional even without the front-end. After all features are implemented, implementing front-end will be an additional feature to facilitate ease of use and testing).*

**COMPLETED!**

1. **Novelty**
2. Electronic Storage – Instead of using smart contracts only as a medium of payment on blockchain, we are using it as a single **electronic storage** medium for patient’s data on blockchain. Using blockchain for this purpose makes data management easy for patients. It allows data to be persisted for a longer duration and in a secure way on a reliable medium. Data can be accessed by the patient whenever required and patient doesn’t have to worry about the data getting lost or misplacing it.
3. Research labs - Our inclusion of the **entity** of **Research Labs** is a unique idea in the sense that this side of health system domain is not popularly known to a common man. Not much work is done to improve its access to common man either. Our system addresses this issue and handles it using blockchain.
4. Time limited access – Our **feature** of **Time Limited Access** to transactions on blockchain is something that is not very commonly thought of or implemented. Things that are on blockchain stay there forever and are accessible forever. This might invite the risk of misuse of the data. So, instead of this infinitely availability of data, we restrict its availability to target audience for some time limit. Patient can provide access to its records to other entities for pre-defined time limit so that he can avoid misuse of his data.
5. **Output Explanation**

Flow:

1. Each entity will have to register itself.
2. Patient starts with adding a new health record on blockchain.
3. Patient deposits initial money on the blockchain which is held by smart contract.
4. Doctor gives his feedback to the patient based on his record and symptoms and this feedback is recorded on blockchain as a transaction.
5. After giving feedback, if all the constraints are satisfied then the contract will release patient’s deposit to doctor or else it will be given back to patient.

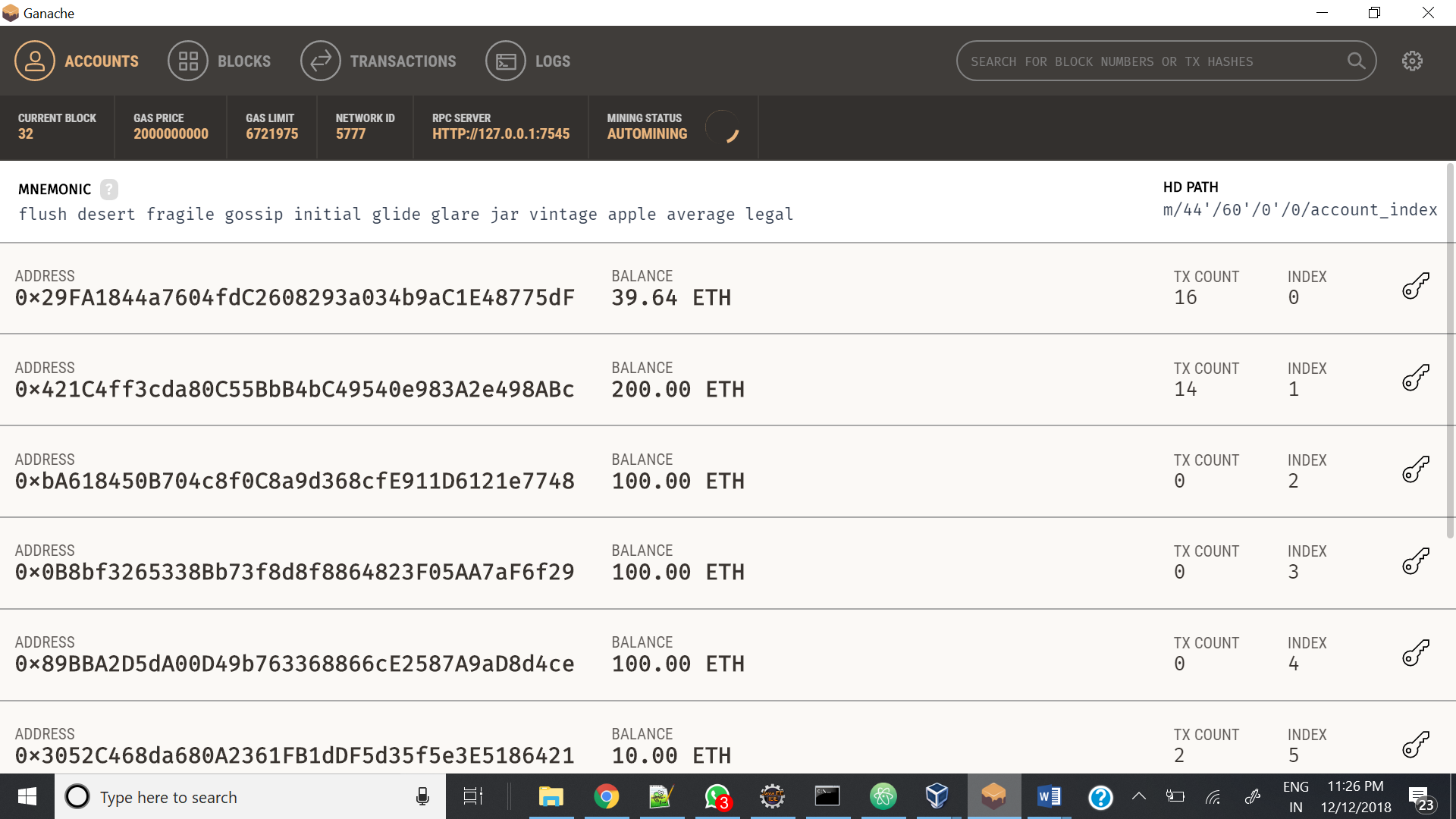
Technology Details:

1. We have used Truffle Framework to connect front end and back end.
2. We are using Ganache as client to connect to test network.
3. We are using Metamask for handling transactions.
4. Our GUI is developed in HTML, CSS, Bootstrap, jQuery, JavaScript.
5. Our business logic is in a Smart Contract written in Solidity-Remix.
6. We are using Node as the server to deploy our application.
7. **Output Screenshots**

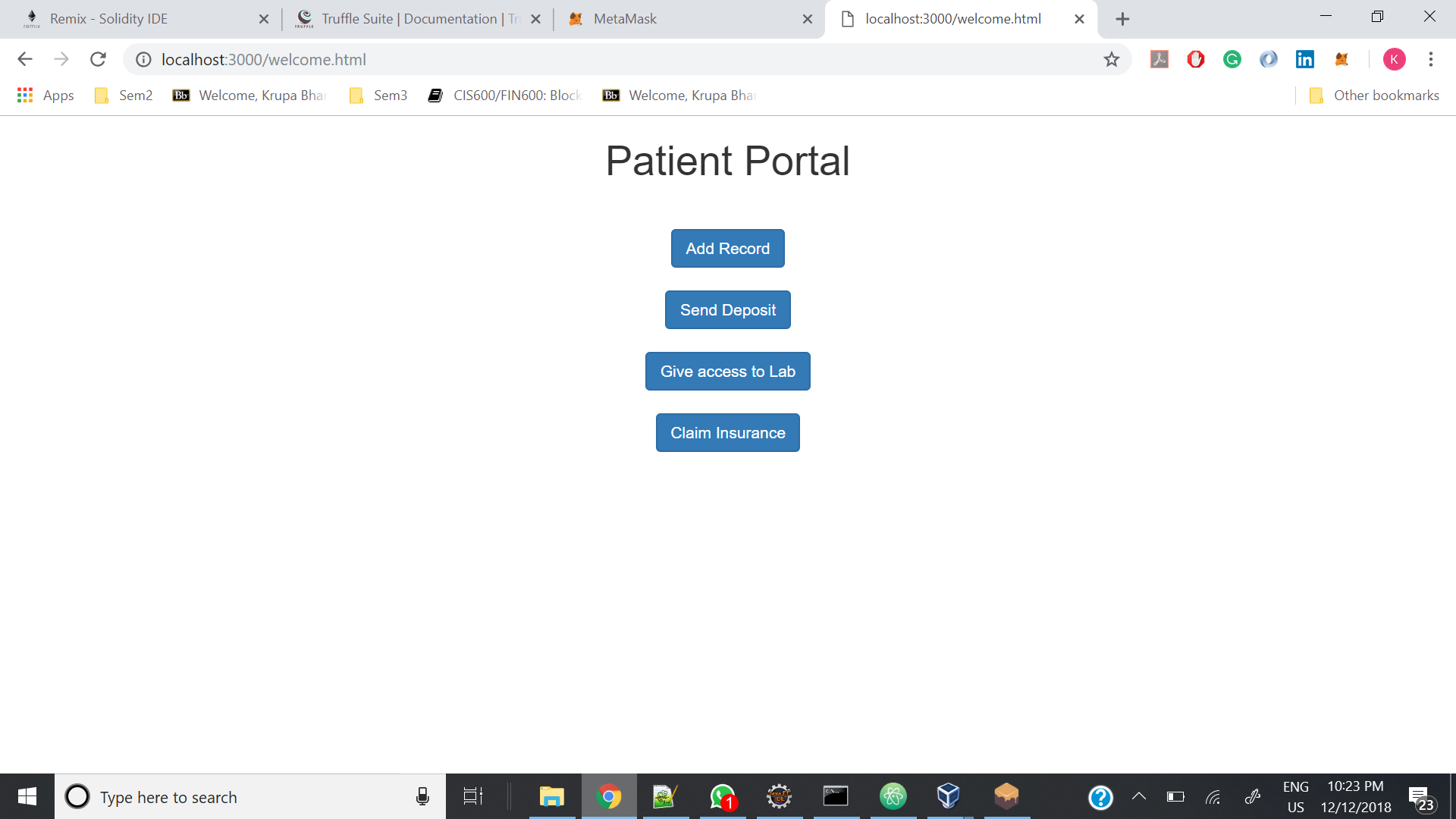
We are using Ganache client through which we are accessing addresses for patient and doctor.

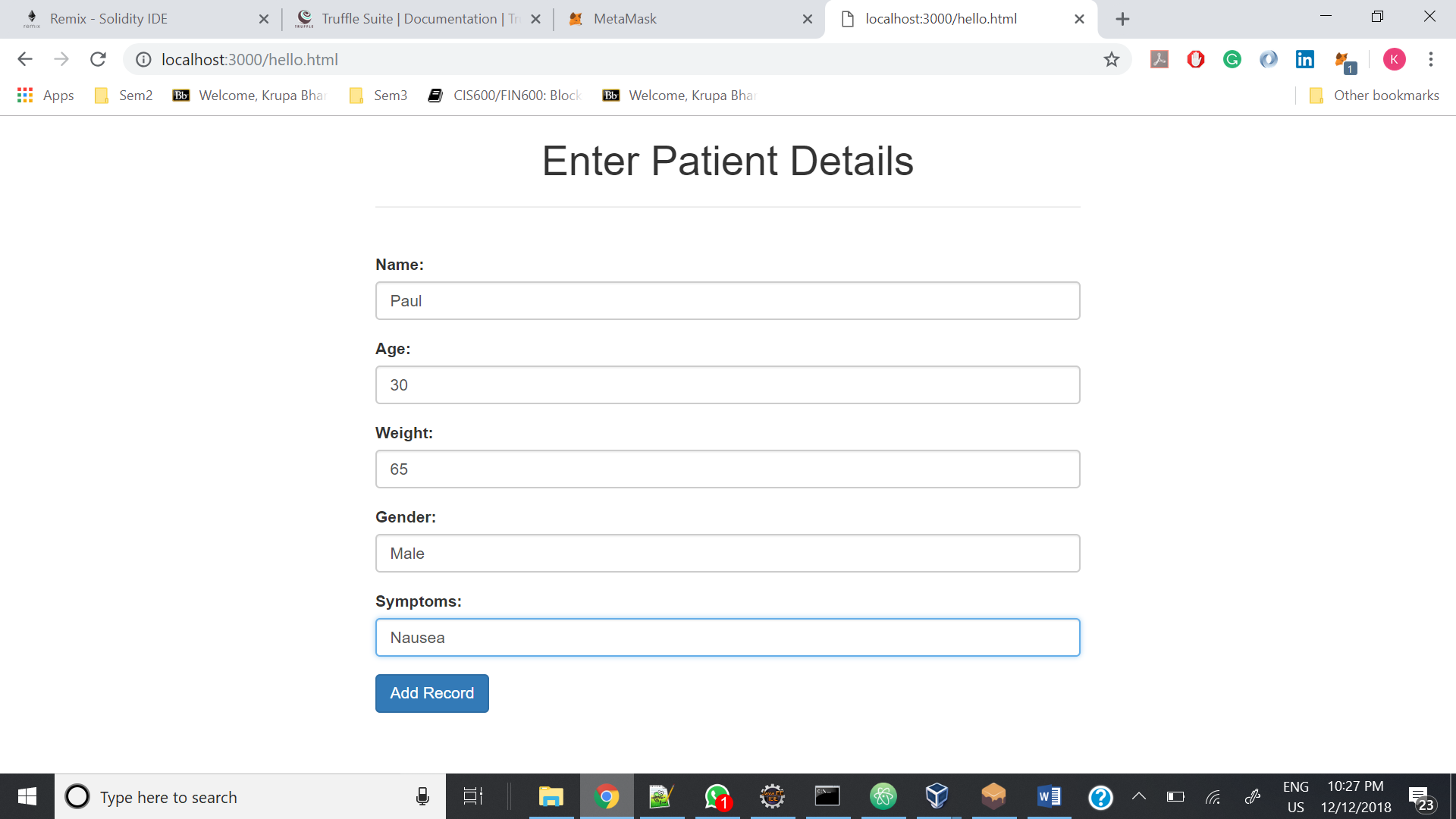
Address 1 is our Patient.

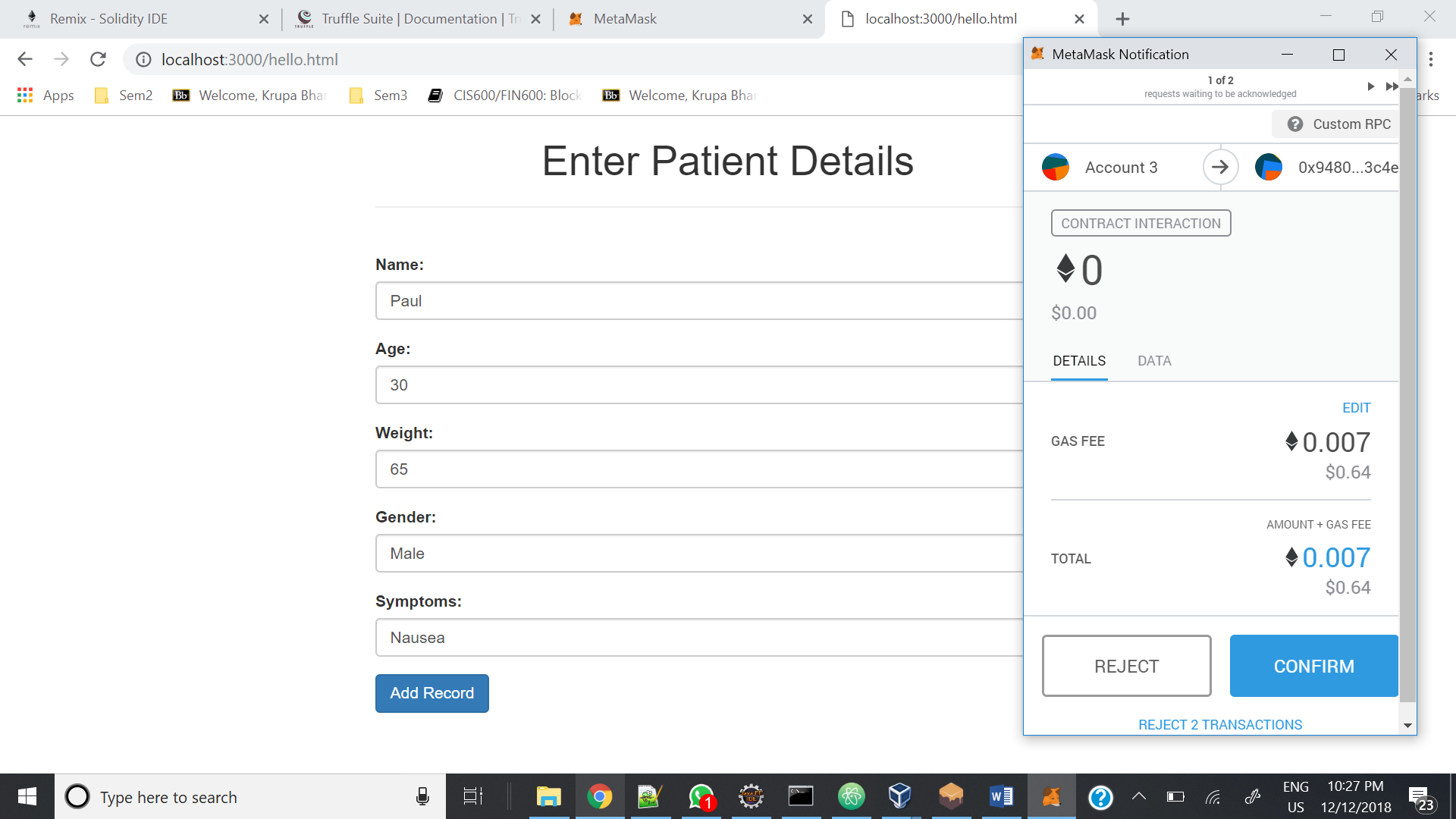
Address 2 is our Doctor.

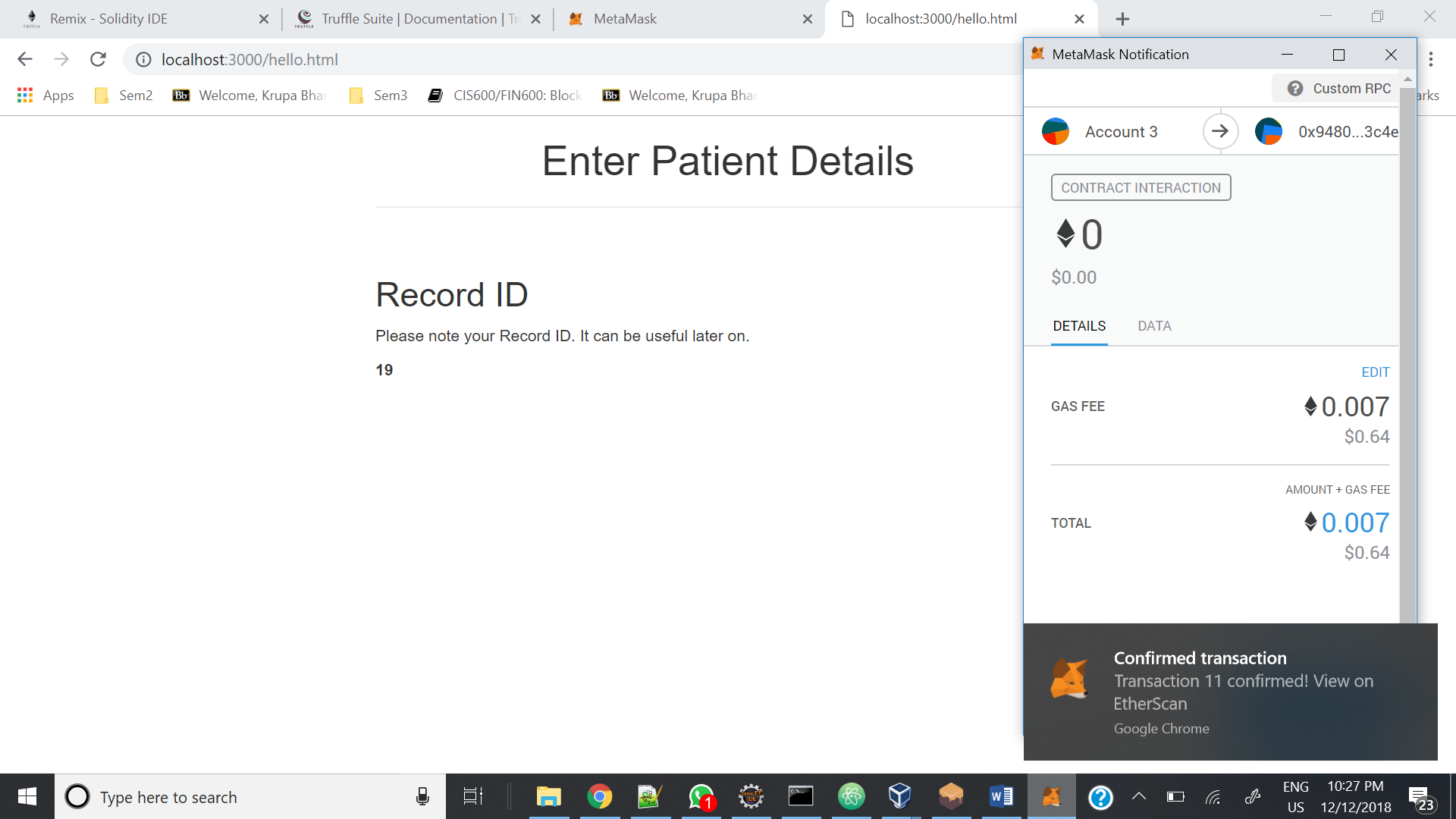


Step 1: Patient enters a new record from Patient Portal.

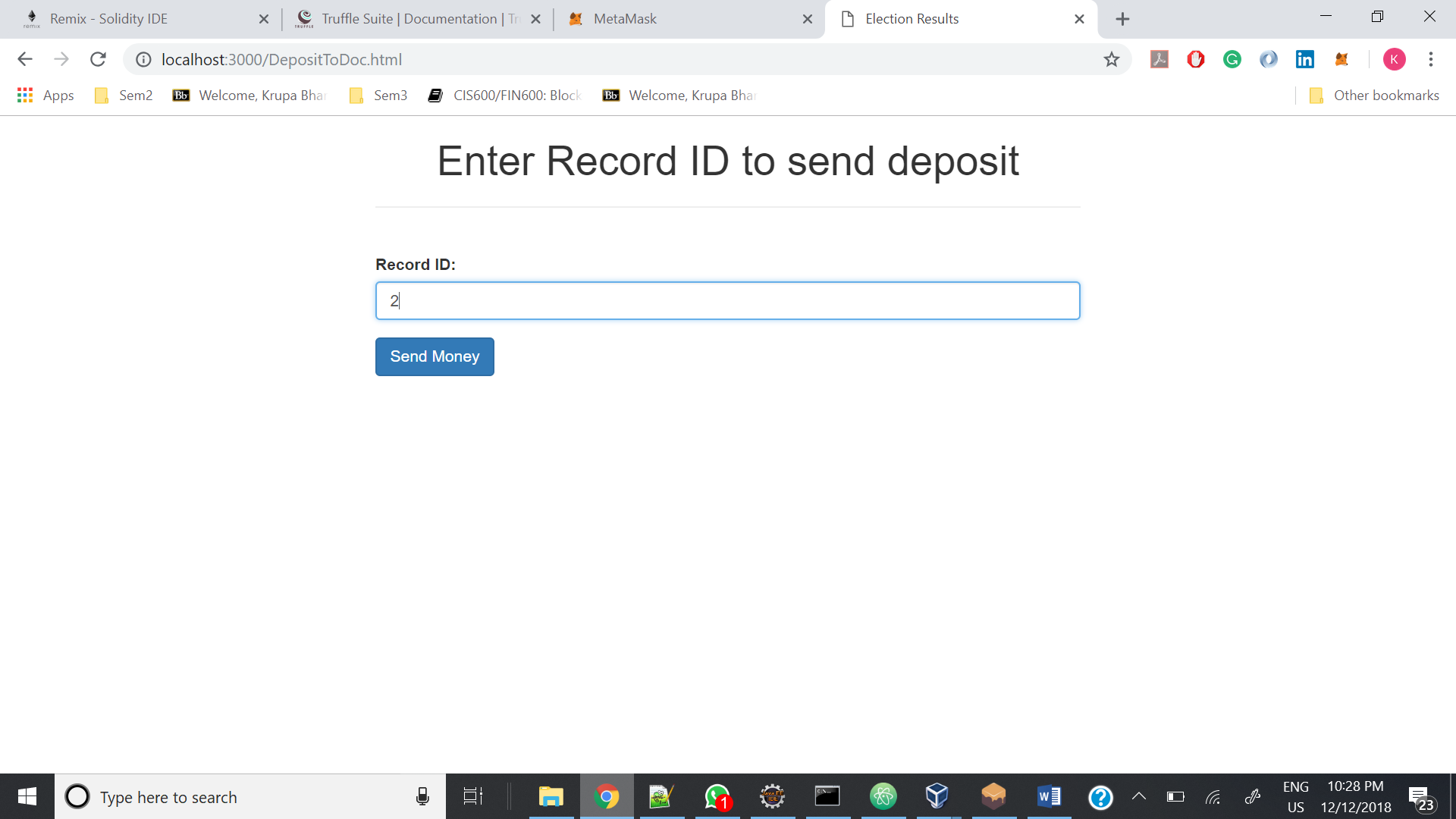


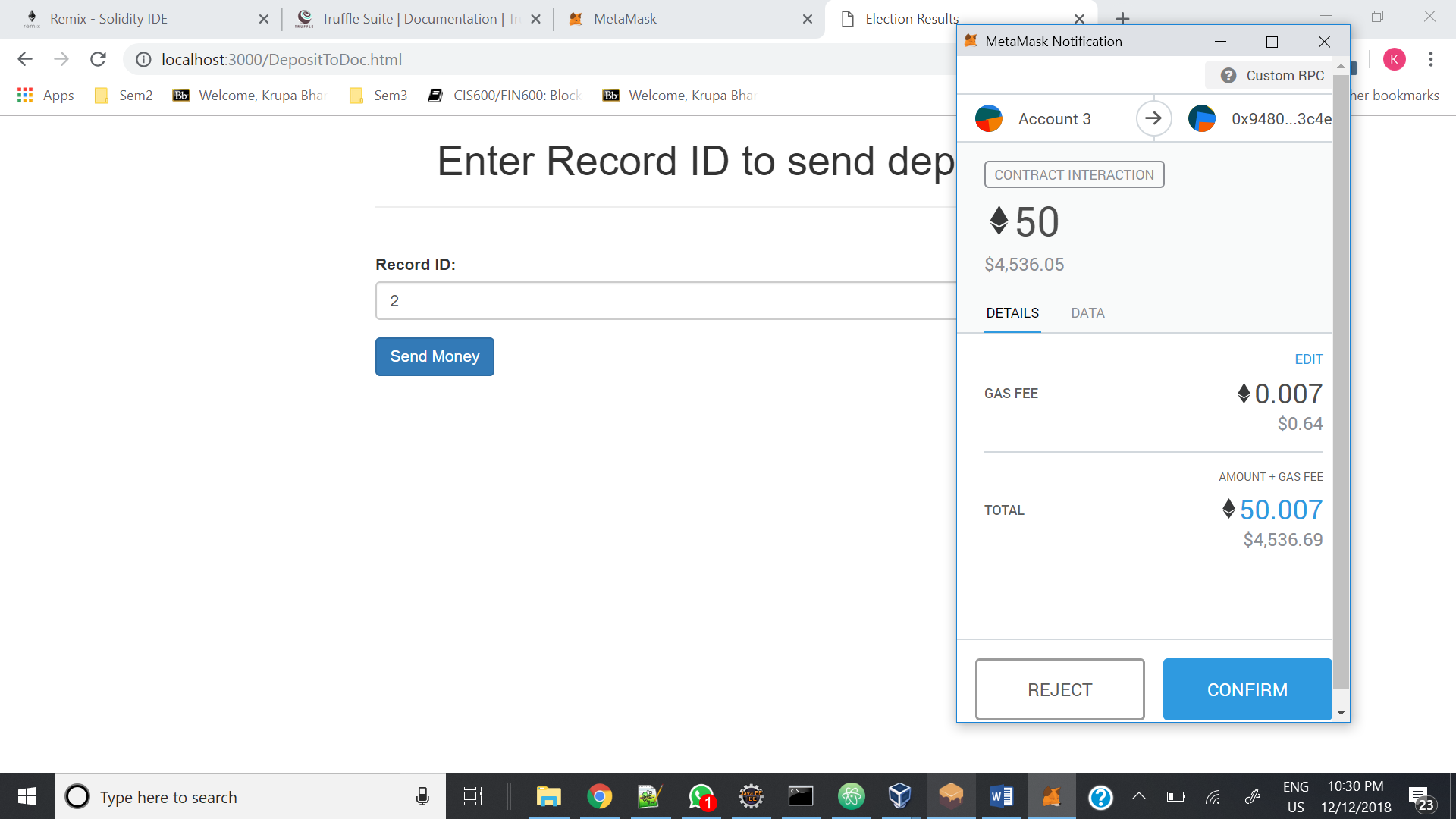


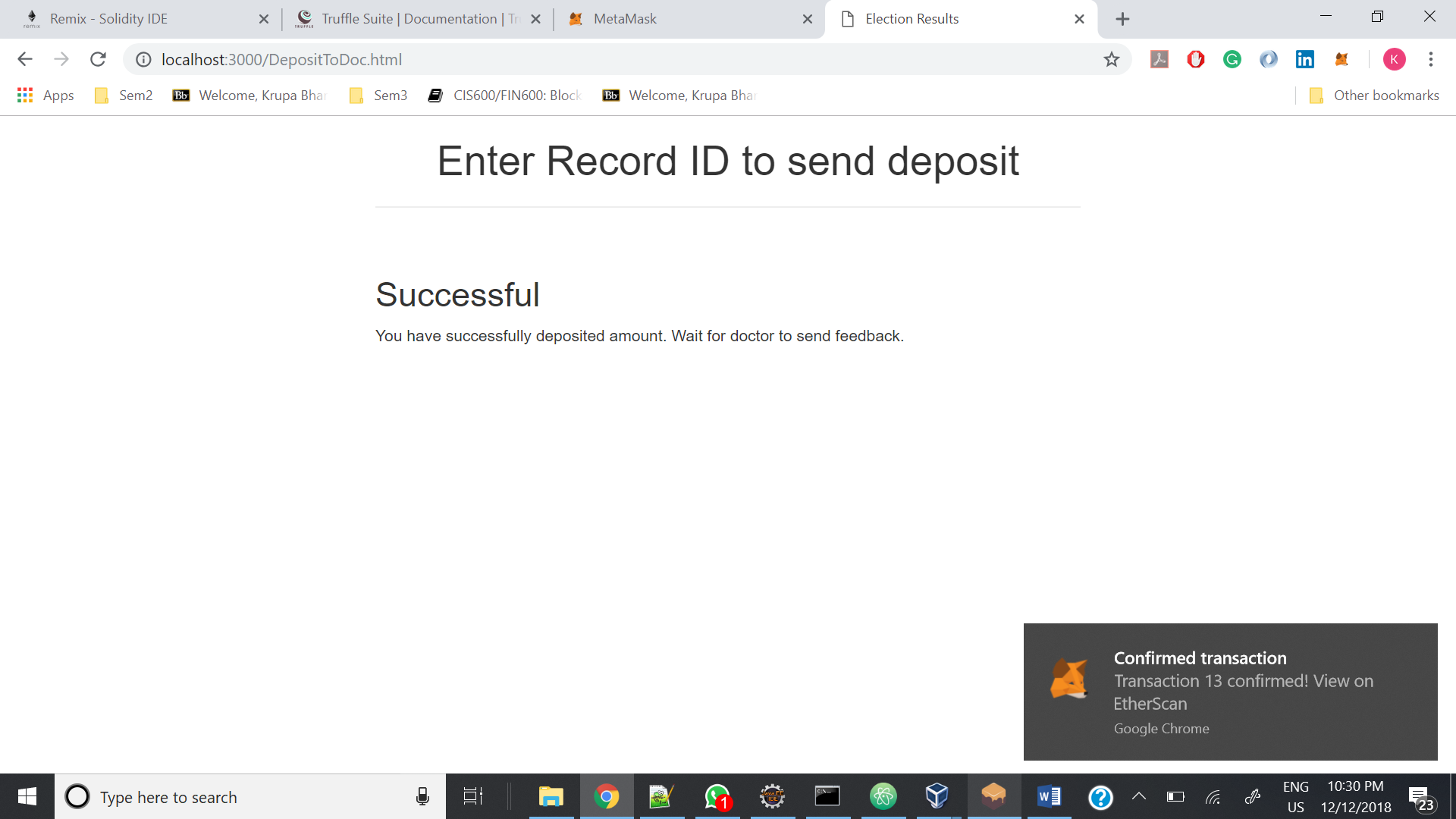


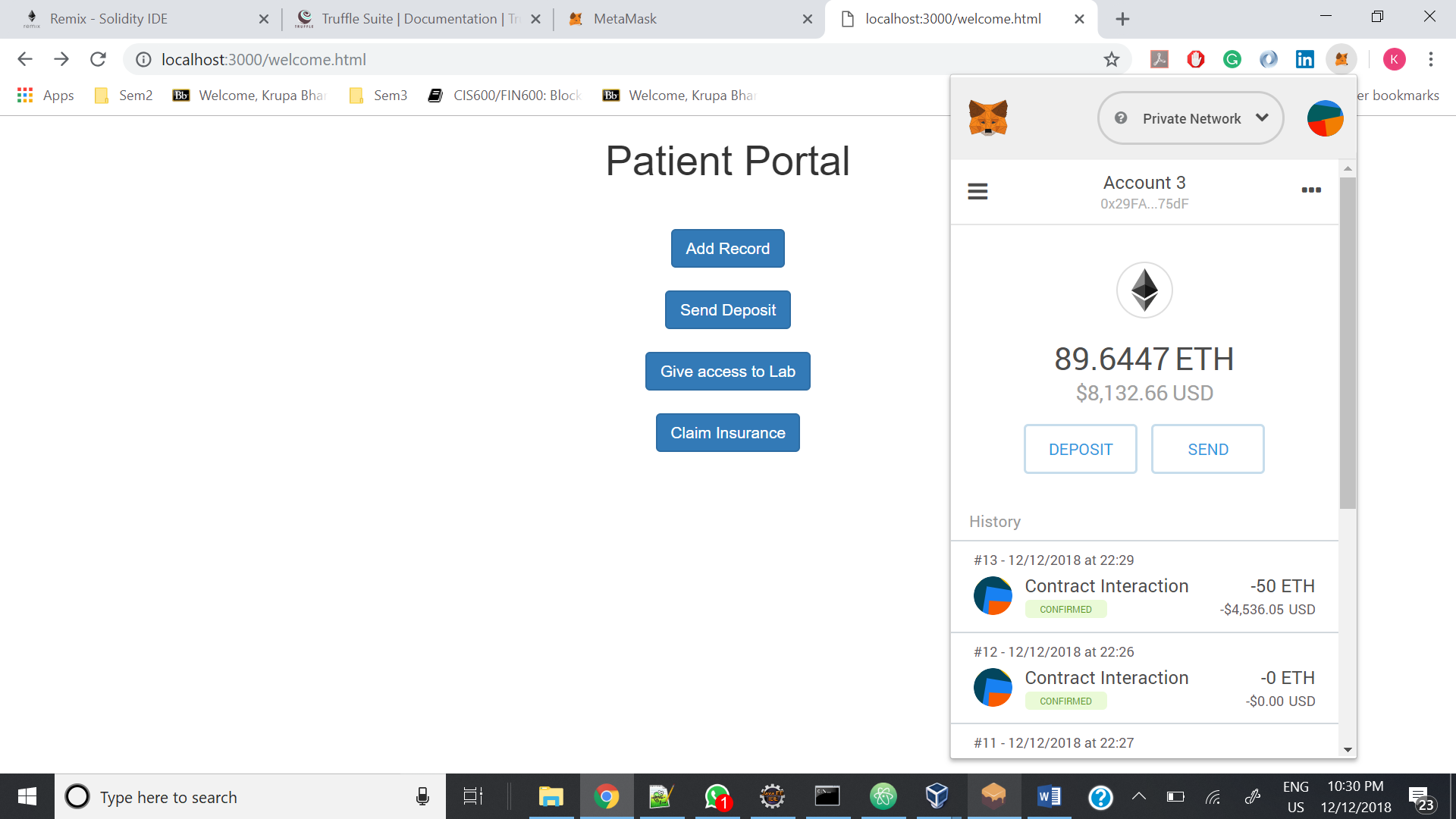


Step 2: Patient deposits money from Patient Portal for a particular record so that he can get feedback from doctor for that record.

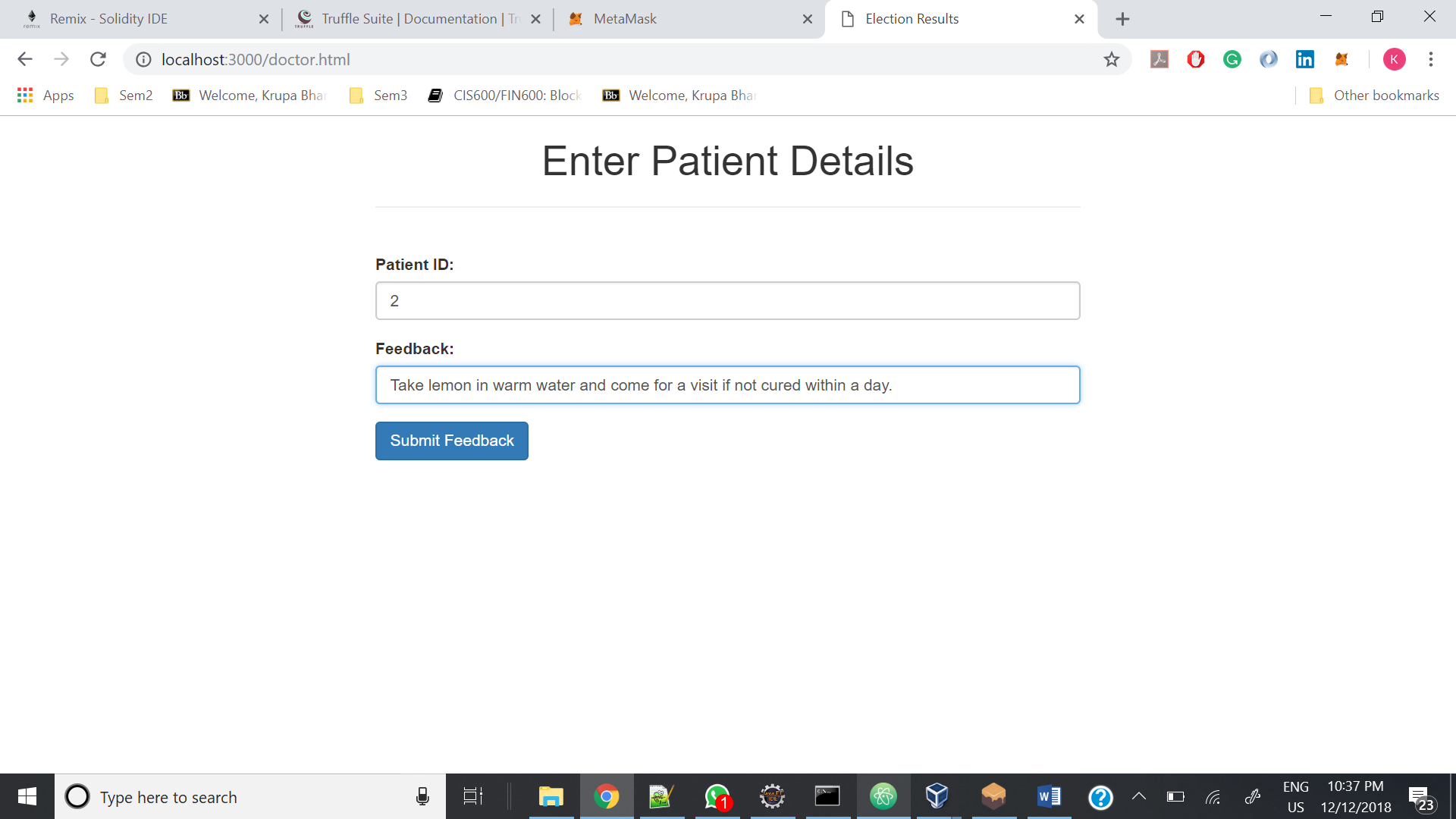


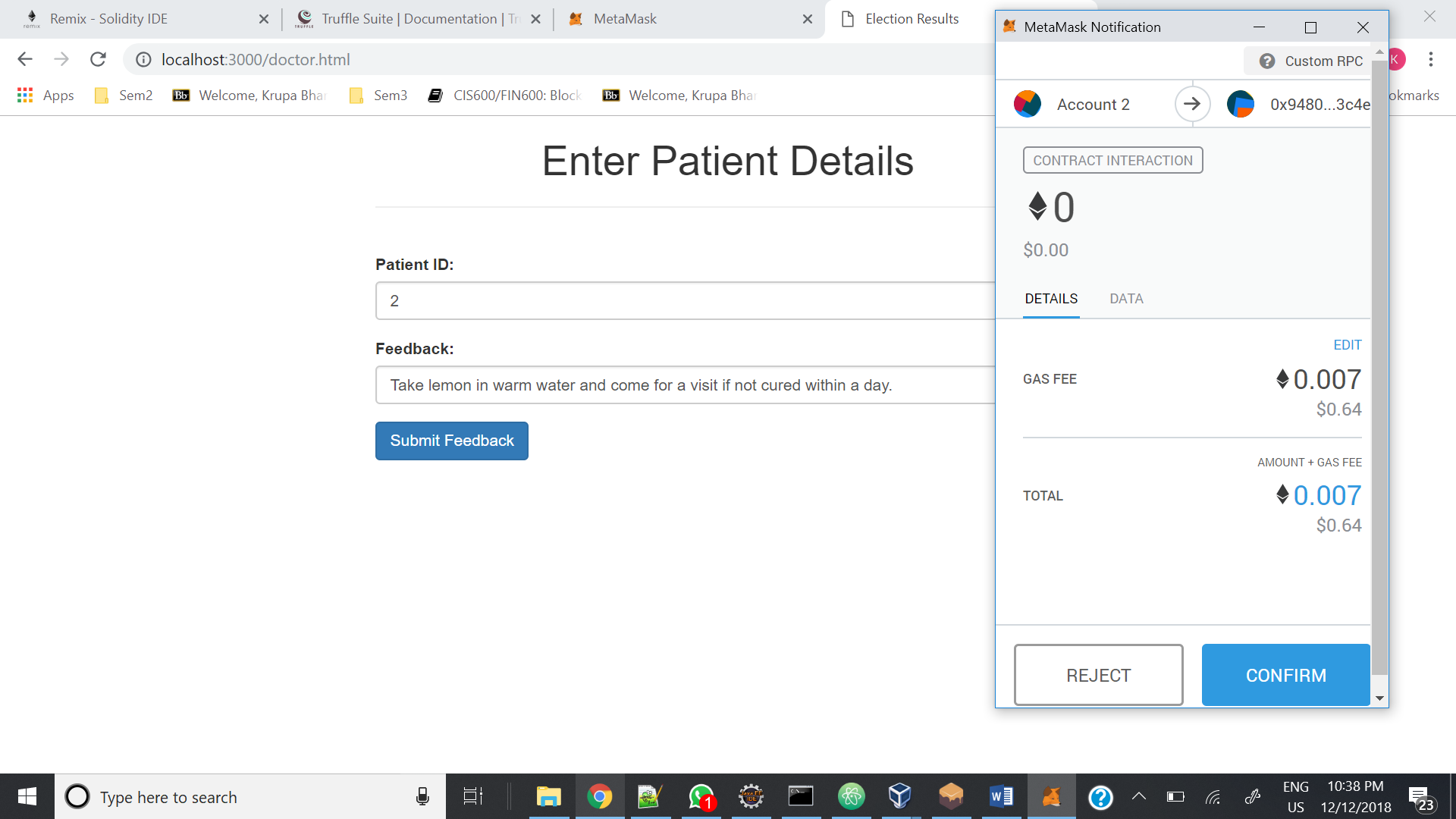


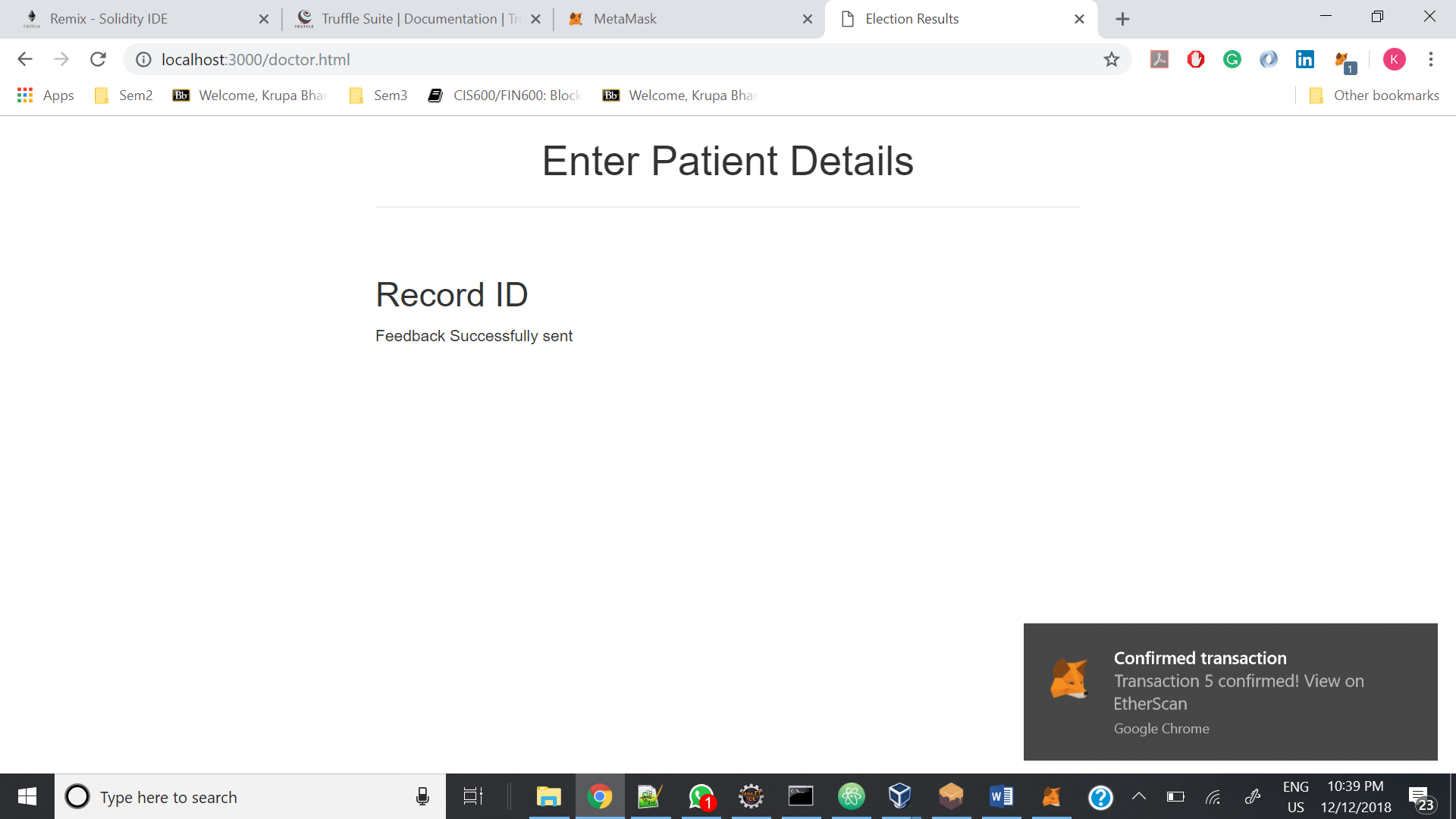


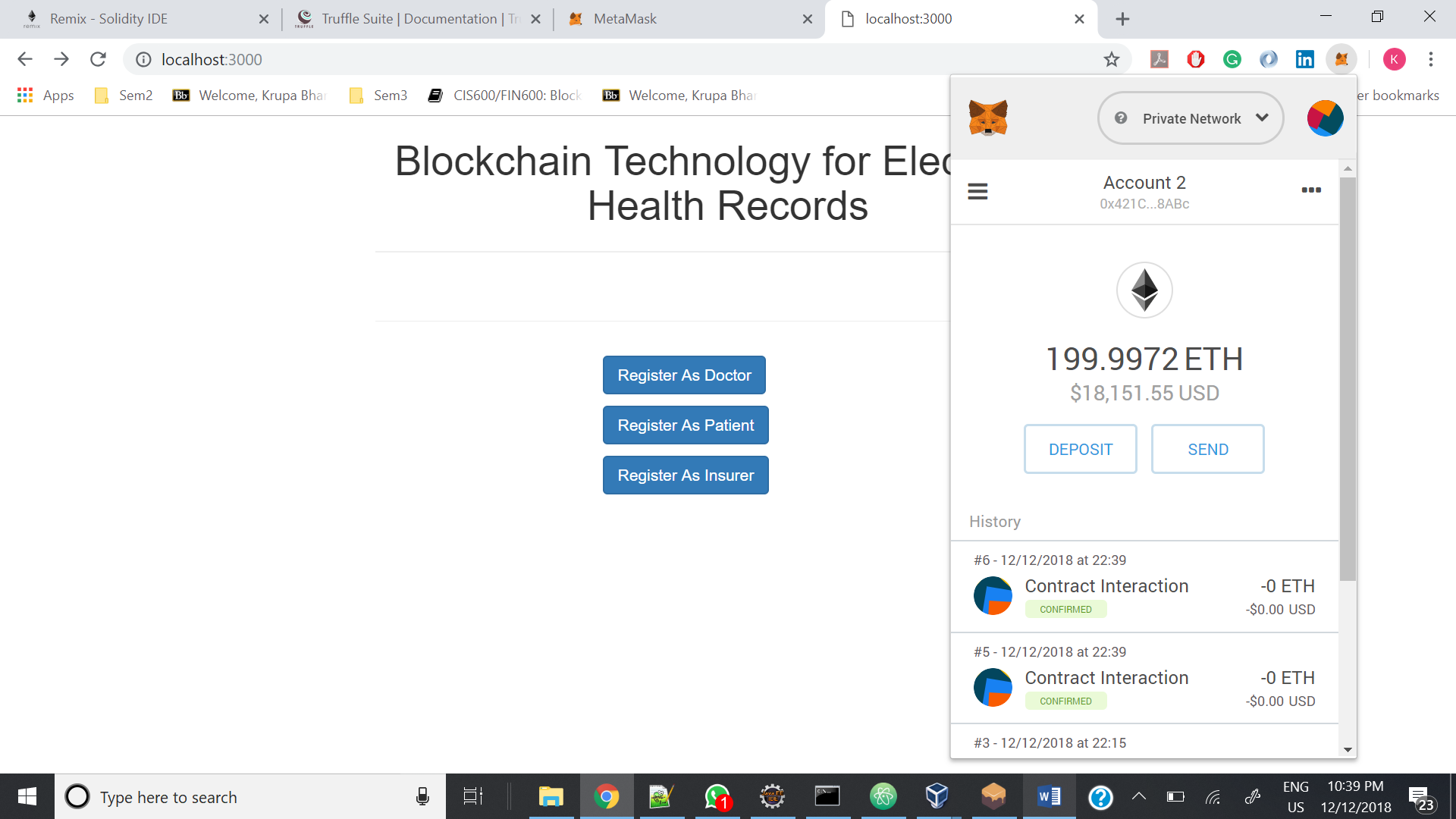


Step 3: Doctor gives feedback for a particular patient record.









1. **Future Scope**
2. Currently we wanted to focus on learning and implementing more and more features of blockchain with minimum required entities instead of adding more entities and implementing same feature between them.

We can add more entities like hospitals, test labs, pharmacy, blood banks, organ donors, etc. to expand the system.

1. Currently patient manually creates/enters data to send. Our system can be integrated with other smart devices that records patient data e.g. smart watches. This way patient’s data will be automatically sent to respective entities directly from the smart devices as blockchain transactions.
2. We have implemented a simple application with one entity each. This can be easily expanded for multiple entities. Also, each entity will have access to all the records that they are allowed to access, on their own client web site.
3. **References**
4. <https://www.prolifics.com/blog/healthcare-blockchain-how-smart-contracts-could-revolutionize-care-delivery>
5. Blockchain Title: How Blockchain Can Transform The Pharmaceutical And Healthcare Industries,PhUSE-WP005-<https://www.phuse.eu/documents//working-groups/deliverables/phuse-blockchain-white-paper-final-version-1-18843.pdf>