##### Building a Privacy-Preserving Blockchain-Based Bidding System: A Crypto Approach

***Submitting By***

|  |  |
| --- | --- |
| **MOHANA KRISHNAN R** | **(113119UG03058)** |
| **NIKHIL B** | **(113119UG03062)** |
| **SUNIL AHAMED S** | **(113119UG03105)** |
| **KANAGASABAI K** | **(113119UG03045)** |

***In partial fulfilment for the award of the degree of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**VEL TECH MULTI TECH Dr. RANGARAJAN Dr. SAKUNTHALA ENGINEERING COLLEGE, ALAMATHI ROAD, AVADI, CHENNAI-62 ANNA UNIVERSITY, CHENNAI 600 025.**

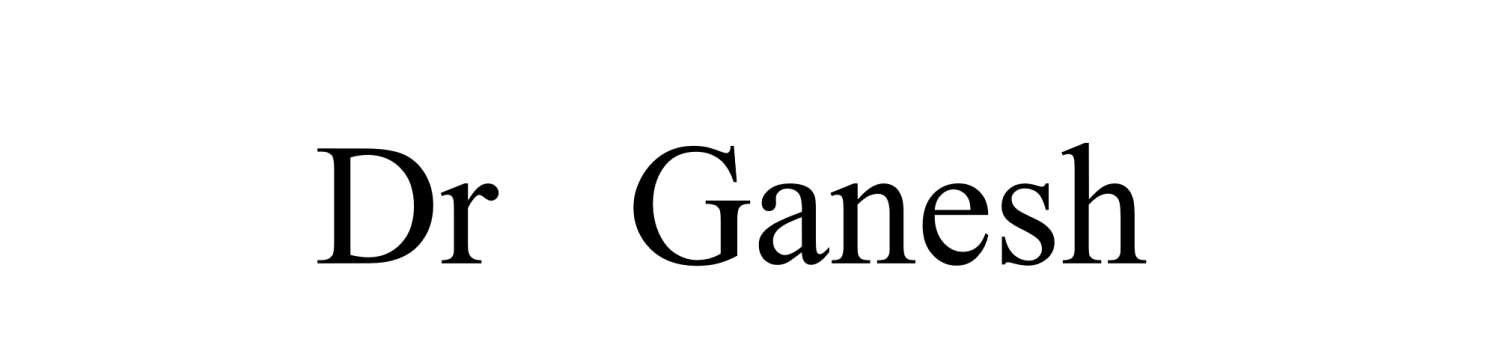
**DECEMBER 2022**

**ANNA UNIVERSITY, CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report of title “**BUILDING A PRIVACY-PRESERVING BLOCK CHAIN-BASED BIDIG SYSTEM: A CRYPTO APPROACH**” is the bonafide work of **MOHANA KRISHNAN R (113119UG03058), NIKHIL B (113119UG03062), SUNIL AHAMED S(113119UG03105),KANAGASABAI K(113119UG03045)** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**



**HEAD OF THE DEPARTMENT SUPERVISOR**

Mr.Victor Jose,M.E,Ph.D,

PROFESSOR

Department of Computer Science

and Engineering

Vel Tech Multi Tech Dr. Rangarajan

Dr. Sakunthala Engineering College,

Avadi, Chennai-600 062.

Mr.Saravanan, B.E,M.E(CSE),Ph.D,

PROFESSOR

Department of Computer Science and Engineering

Vel Tech Multi Tech Dr. Rangarajan

Dr. Sakunthala Engineering College,

Avadi, Chennai-600 062

**CERTIFICATE FOR EVALUATION**

This is to certify that the project entitled **“BUILDING A PRIIVACY-PRESERVING BLOCKCHAIN-BASED BIDIG SYSTEM: A CRYPTO APPROACH”** is the bonafide record of work done by following students to carry out the project work under our guidance during the year 2022-2023 in partial fulfilment for the award of Bachelor of Engineering degree in Computer Science and Engineering conducted by Anna University, Chennai.

|  |  |
| --- | --- |
| **MOHANA KRISHNAN R** | **(113119UG03058)** |
| **NIKHIL B** | **(113119UG03062)** |
| **SUNIL AHAMED S** | **(113119UG03105)** |
| **KANAGASABAI K** | **(113119UG03045)** |

This project report was submitted for viva voce held on \_\_\_\_\_\_\_\_\_\_

At VelTech Multi Tech Dr. Rangarajan and Dr. Sakunthala Engineering College.

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks to Almighty and the people who extended their help during the course of our work.

We are greatly and profoundly thankful to our honourable Chairman, **Col. Prof.Vel. Shri Dr.R.Rangarajan B.E.(ELEC), B.E.(MECH), M.S.(AUTO)., D.Sc.,** &Vice Chairman, **Dr.Mrs.Sakunthala Rangarajan M.B.B.S.**, for facilitating us with this opportunity.

We also record our sincere thanks to our honourable Principal, **Dr.V.Rajamani M.E.,Ph.D.,** for his kind support to take up this project and complete it successfully.

We would like to express our special thanks to our Head of the Department, **Dr.R.Saravanan,B.E,M.E(CSE).,Ph.D,** Department of Computer Science and Engineering and our project supervisor **Mr.Victor Jose,ME,Ph,D.,** for their moral support by taking keen interest on our project work and guided us all along, till the completion of our project work and also by providing with all the necessary information required for developing a good system with successful completion of the same.

Further, the acknowledgement would be incomplete if we would not mention a word of thanks to our most beloved Parents for their continuous support and encouragement all the way through the course that has led us to pursue the degree and confidently complete the project work.

**(MOHANA KRISHNAN R) (NIKHIL B) (SUNIL AHAMED S) (KANAGASABAI K)**

**ABSTRACT**

With the recent development in network technology over a few years digital works can be easily published online. One of the main issues in the field of digital technology is the infringement of digital works which can seriously damage the data owners rights and affects the enthusiasm of the owners to create original work. Blockchain is one of the core technologies of ICT (Information & Communication Technology) and is attracting attention as the next- generation network technology. In an attempt to counter illegal distribution content providers have turned to digital rights management (DRM). So far DRM is mainly used to restrict the actions of people who have legally purchased the digital content. In a traditional centralized digital content distribution model middlemen serve their purposes because a content creator often finds it hard to reach their end users. Blockchain technology makes possible the establishment of a decentralized infrastructure on which content owners and purchasers can trade digital content directly with each other without middlemen. Aiming at the mentioned problems this paper proposes a protection technique which can realize the automatic management of the complete digital content trading life cycle on the blockchain using smart contract technology. The proposed system is based on blockchain technology which leverages the distributed tamper-proof and traceable characteristics of blockchain. The system uses smart contracts to manage the full life cycle of digital content trading. The test results show that the proposed system provides effffective protection of the digital content trading system and can efficiently confirm the rights of digital content trading.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | **05** |
|  | **LIST OF ABBREVIATIONS** | **09** |
| **1.** | **INTRODUCTION** | **10** |
|  | 1.1. OBJECTIVE | 11 |
|  | 1.2. SCOPE OF THE PROJECT | 11 |
|  | 1.3. LITERATURE SURVEY | 11 |
| **2.** | **SYSTEM ANALYSIS** | **16** |
|  | 2.1. EXISTING SYSTEM | 17 |
|  | 2.1.1.EXISTING ALGORITHM | 17 |
|  | 2.2. PROPOSED SYSTEM | 17 |
|  | 2.2.1.ADVANTAGES OS PROPOSED SYSTEM | 18 |
|  | 2.3.PROPOSED ALGORITHM | 18 |
|  | 2.3.1.ADVANTAGE OF PROPOSED ALGORITHM | 18 |
| **3.** | **SYSTEM SPECIFICATIONS** | **19** |
|  | 3.1. HARDWARE SPECIFICATIONS | 20 |
|  | 3.2. SOFTWARE SPECIFICATIONS | 20 |
| **4.** | **SOFTWARE DESCRIPTION** | **21** |
|  | 4.1. FRONT END | 22 |
|  | 4.2. BACK END | 24 |

|  |  |  |
| --- | --- | --- |
|  | **MODULE DESCRIPTION** | **25** |
|  | 5.1. PROJECT DEFINITION | 26 |
|  | 5.2. OVERVIEW OF THE PROJECT | 26 |
|  | 5.4. ARCHITECTURE DIAGRAM | 27 |
|  | 5.5. MODULES | 28 |
|  | 5.5.1. User Authentication module | 28 |
|  | 5.5.2.Digital Owner Registration | 28 |
|  | 5.5.3.Digital Information Query | 29 |
|  | **SYSTEM TESTING** | **30** |
|  | 6.1. CODE REVIEW | 31 |
|  | 6.2. TESTING PROCESS | 31 |
|  | 6.2.1. Unit Testing | 32 |
|  | 6.2.2.System Testing | 33 |

|  |  |  |
| --- | --- | --- |
|  | **SYSTEM IMPLEMENTATION** | **34** |
|  | 7.1. IMPLEMENTATION PROCEDURE | 35 |
|  | **CONCLUSION AND FUTURE ENHANCEMENTS** | **36** |
|  | 8.1. CONCLUSION | 37 |
|  | 8.2. FUTURE ENHANCEMENTS | 37 |
|  | **APPENDICES** | **38** |
|  | **APPENDIX-1 SCREENSHOTS** | **39** |
|  | **APPENDIX-2 IMPLEMENTATION CODE** | **47** |
|  | **REFERENCES** | **69** |

**LIST OF ABBREVIATIONS**

**ABBREVATION FULL FORM**

SQL STRUCTURED QUERY LANGUAGE

HTML HYPER TEXT MARKUP LANGUAGE

CSS CASCADING STYLE SHEETS

**CHAPTER 1**

**INTRODUCTION**

**INTRODUCTION**

Recently, block chain has become highly popular, especially in commerce, and it has influenced the growth and usefulness of many commercial applications. With the decentralized and hash-chained nature of block chain, commercial applications can eliminate the problem of whether a third-party or communication channel is trustworthy, which is the most critical problem for a bidding system [1]–[3]. The difference between traditional and block chain-based bidding systems is shown in Figure 1. In traditional bidding systems, a buyer seals his or her bid to protect the information and sends it to a seller during the auction. The buyer must ensure that the sealed bid is securely delivered to the seller, without any distortion or information leakage. In a block chain-based bidding system, the buyer can confirm that his or her bid is safely delivered to the seller and cannot be changed by any adversaries. As aresult, the market share of block chain-based bidding systems is increased. In a bidding system, the most important property is the privacy of the identities of the transaction both sides and the bidding content. This is not the case in which the communication peers are required to be authenticated, such as the IoT (Internet of Things) scenario [4] or the normal cloud storage case [5]. Instead, for a fair competition for the price of the auction, no buyer should obtain any information about other buyers’ prices or identities; only the seller and the buyer who sealed the bid can decrypt the bid successfully, and the sealed bid should not leak any identifying information to others. If the buyer’s identity or their bid information leaks to other competitors, the auction becomes unfair and untrustworthy. Thus, an encryption system that can be operated without explicit identities is required. In 2019, A teniese *et al.* proposed an encryption scheme called matchmaking encryption (ME) [6], which can hide senders’ and receivers’ identities simultaneously. In an ME scheme, the sender and receiver identities are hashed and used in the encryption process. No outsider can obtain identity information, including the sender or the receiver, from a ciphertext. Only a party who knows the sender and the receiver of a ciphertext can correctly decrypt the ciphertext with the receiver’s secret key. Therefore, ME is a good candidate that completely fits into the blockchain-based bidding system.

* 1. **OBJECTIVE**
* To assess this empirical evidence in the context of a resilience framework based on the adaptive cycle.
* To define a syntax for specifying policy targets
* To define openness as the capacity of stakeholders to access stored data.
* To achieve credibility, truthfulness,
* To not to damage the reputation.

**1.2. SCOPE OF THE PROJECT**

The aim of this project is to introduce a new encryption scheme called deniable matchmaking encryption (DME). This new encryption scheme provides deniability not only for the message, but also for the identities. We use the chameleon hash function to make fake message and fake identities indistinguishable from the real message and the real identities. Therefore, the bidding system can use fake information to answer the coercer, and user privacy is kept by the blockchain-based bidding system.

**1.3. LITERATURE SURVEY**

Title 1 : Audio Watermarking for Security and Non-Security Applications

Authors :Maha Charfeddine , Eya Mezghani , Salma Masmoudi , Chokri Ben Amar and Hesham Alhumyani

Published Year : 2022

Description :The digitization of audiovisual data is significantly increasing. Thus, to guarantee the protection of the intellectual properties of this digital content, watermarking has appeared as a solution. Watermarking can be used in reality in several types of applications that target two different contexts: the first for security applications and the second for non-security ones. In this paper, we carry a big interest in studying these two types of applications. Moreover, we propose a first digital watermarking scheme for security copyright protection applications, where we have involved neural network architecture in the insertion and detection processes, and integrated some masking phenomena of the human psycho acoustic model with linear predictive coding spectral envelope estimation of the audio file.

Title 2 : HEVC Watermarking Techniques for Authentication and Copyright Applications: Challenges and Opportunities

Authors : Ali A. Elrowayati , Mohamed A. Alrshah , Mohammad Faiz Liew Abdullah and Rohaya LatipPublished Year : 2020

Description : Recently, High-Efficiency Video Coding (HEVC/H.265) has been chosen to replace previous video coding standards, such as H.263 and H.264. Despite the efficiency of HEVC, it still lacks reliable and practical functionalities to support authentication and copyright applications. In order to provide this support, several watermarking techniques have been proposed by many researchers during the last few years. However, those techniques are still suffering from many issues that need to be considered for future designs. In this paper, a Systematic Literature Review (SLR) is introduced to identify HEVC challenges and potential research directions for interested researchers and developers. The time scope of this SLR covers all research articles published during the last six years starting from January 2014 up to the end of April 2020. Forty-two articles have met the criteria of selection out of 343 articles published in this area during the mentioned time scope. Anew classification has been drawn followed by an identification of the challenges of implementing HEVC watermarking techniques based on the analysis and discussion of those chosen articles.Eventually, recommendations for HEVC watermarking techniques have been listed to help researchers to improve the existing techniques or to design new efficient ones.

Title 3 :A Blockchain Ethereum Technology-Enabled Digital Content: Development of Trading and Sharing Economy Data

Authors :Umair Khan , Zhang Yong An and Azhar Imran

Published Year : 2020

Description : The idea of a shared economy becomes one of the companies as an enterprise type. Especially with the advanced development of digital smart devices and the internet, several forms of the mutual economy have been advanced in accord with the need for sharing of separate income.Shareable commodity and digital content are also seeking to utilize. When digital content is used as a sharing economy, various possible threats may arise in the course of transactions, the potential for theft, alteration, and hacking of contents. This paper presents a comprehensive overview of the security and privacy of Blockchain. Blockchain promise transparent, tamper-proof and secure systems that can enable novel solutions, especially when combined with smartcontracts. In this research, we proposed a content protection and transaction method using Blockchain Ethereum Technology. The encryption algorithm is incorporated in proposed system to make transparent transactions and it is also implemented on content itself to prevent from smart forgery and hacking. The experimental results signify that the proposed method has strong potential to enhance transactions transparency by minimizing the security threats in digital content transactions.

Title 4 :Rate-Distortion-Preserving Forensic Watermarking Using Quantization Parameter Variation

Authors : Hannes Mareen , Martijn Courteaux , Johan De Praeter , Md. Asikuzzaman , Glenn Van Wallendael and Peter Lambert

Published Year : 2020

Description :Watermarking enables the identification of digital pirates that illegally redistribute copyright protected videos. One of the main challenges is for the watermark to be imperceptible, while not increasing the video bit rate. Additionally, the system should be robust to attacks that attempt to remove the watermark. Therefore, this paper proposes a robust watermarking technique that does not degrade the video quality nor negatively affect the video bit rate. In other words, it preserves the video encoders compression efficiency or rate-distortion performance. For watermark embedding, the quantization parameters are varied during video compression. As a result, different compression artifacts are introduced, although they do not distort the video more than those that occur during ordinary video compression. The collection of artifacts represents the watermark and is used for watermark detection. The experimental results prove that the proposed approach retains the rate-distortion performance better than state-of-the-art techniques. Furthermore, the watermarks are robust to recompression and noise attacks. In conclusion, the proposed method enables content providers to perform forensic watermarking without affecting the compression efficiency.

Title 5 :Obfuscation-Based Watermarking for Mobile Service Application Copyright Protection in the Cloud

Authors : Sun Guang , F. Xiaoping , J. Wangdong , L. Fenghua and J. Yuewei

Published Year : 20199

Description :The contributions of cloud computing in the prevention of software piracy are inadequate, and there are still rampant piratical mobile service applications in the cloud. This paper navigates mobile service application copyright protection in the cloud and sets a watermarking example to explain it. We use Mondens method to obfuscate the applications source code, remove a part of the semantics, and add it to a recovery module. Because these obfuscation rules come from watermarks, the watermarks are mapped into the rules. The recovery module is a recognizer to prove the watermarks when the original program is recovered. The experimental results indicate that the obfuscated code becomes difficult to reverse engineering and the watermarks are robust.

**CHAPTER 2**

**SYSTEM ANALYSIS**

**SYSTEM ANALYSIS**

**2.1.** **EXISTING SYSTEM**

* Due to the properties of blockchain, bidding records are unchangeable.
* With existing encryption techniques, these bidding records can only be shared by the bidder and the seller.
* Although this scenario sounds secure, it does not consider a coercion case.
* A powerful coercer may force the bidding system to open the records stored on the blockchain, and the system loses privacy.

**2.1.1.EXISTING ALGORITHM**

Deniable Matchmaking Encryption.

**2.2.** **PROPOSED SYSTEM**

The proposed system includes the digital rights owner the digital rights receiver and the distributed deployment of smart contracts which can achieve a variety of business functions. This proposed application automates the management of Data Sharing within associations communities and other organizations. The software helps them communicate with their constituents share resources promote information collect information and more. It is specifically designed to automate and streamline administrative tasks for data sharing application. This software usually offers an integrated system. It provides components or modules that work together and offer an online data sharing platform. Through this you can manage your web application citizen database and administrative functions public registration online verification and sharing data. You can access your data anytime anywhere. Even public who like to apply for an data sharing can do so through online self-service. This means that everyone is working on the same set of records. Hence when any changes are made to these data all records are simultaneously updated preventing duplication and it also eliminates the need to copy files or email anyone separately. When the whole process and data is integrated through proposed system you are able to track and report seamlessly. You can now instantly create any kind of records in real time.This places you in a position to feel view and control on what is happening when the process is actually going on or even after it is completed.

**2.2.1. ADVANTAGES OF PROPOSED SYSTEM**

* Lowering the Complexity Threshold
* Better operational efficiency
* Minimal time cost and memory usage.
* Improved traceability
* Simple to use and interpret

**2.3.PROPOSED ALGORITHM**

* Hashing Algorithm.

**2.3.1.ADVANTAGE OF PROPOSED ALGORITHM**

* Particularly efficient when the maximum number of entries
* Highly Secure
* Low Computation and Storage Cost

**CHAPTER 3**

**SYSTEM SPECIFICATIONS**

**SYSTEM SPECIFICATIONS**

**3.1. HARDWARE SPECIFICATIONS**

* Intel Quad Core Processor
* 500 GB Hard disk
* âœ“ 8 GB Ram

**3.2. SOFTWARE SPECIFICATIONS**

* HTML
* CSS,
* JavaScript
* Angular JS
* Python
* Python RESTful
* SQL Server









**CHAPTER 4**

**SOFTWARE DESCRIPTION**













**SOFTWARE DESCRIPTION**

**4.1.** **FRONT END**

**4.1.1.HTML**

**HTML** stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between web pages. A markup language is used to define the text document within the tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

HTML is a markup language used by the browser to manipulate text, images, and other content, in order to display it in the required format. HTML was created by Tim Berners-Lee in 1991. The first-ever version of HTML was HTML 1.0, but the first standard version was HTML 2.0, published in 1995.

**4.1.2.CSS**

**C**ascading **S**tyle **S**heets, fondly referred to as **CSS**, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page. It describes how a web page should look: it prescribes colors, fonts, spacing, and much more. In short, you can make your website look however you want. CSS lets developers and designers define how it behaves, including how elements are positioned in the browser.While HTML uses tags, CSS uses rule sets. CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML document.

**4.1.3.JAVASCRIPT**

**JavaScript** is a *lightweight,* *cross-platform*, *single-threaded,*and *interpreted compiled* programming language which is also known as the scripting language for web pages. It is well-known for the development of web pages, and many non-browser environments also use it. JavaScript is a[weakly typed language](https://www.geeksforgeeks.org/type-systemsdynamic-typing-static-typing-duck-typing/)**(dynamically typed)**. JavaScript can be used for [Client-side](https://www.geeksforgeeks.org/server-side-client-side-programming/) developments as well as [Server-side](https://www.geeksforgeeks.org/server-side-client-side-programming/) developments. JavaScript is both an imperative and declarative type of language. JavaScript contains a standard library of objects, like [Array](https://www.geeksforgeeks.org/arrays-in-javascript/), [Date](https://www.geeksforgeeks.org/javascript-date-objects/), and [Math](https://www.geeksforgeeks.org/javascript-math-object/), and a core set of language elements like [operators](https://www.geeksforgeeks.org/javascript-operators/), **control structures**, and [statements](https://www.geeksforgeeks.org/javascript-statements/).

**Client-side:** It supplies objects to control a browser and its [Document Object Model (DOM).](https://www.geeksforgeeks.org/dom-document-object-model/) Like if client-side extensions allow an application to place elements on an HTML form and respond to user events such as **mouse clicks**, **form input**, and **page navigation**. Useful libraries for the client side are [AngularJS](https://www.geeksforgeeks.org/introduction-to-angularjs/), [ReactJS](https://www.geeksforgeeks.org/react-js-introduction-working/), [VueJS,](https://www.geeksforgeeks.org/vue-js/) and so many others.

**Server-side:** It supplies objects relevant to running JavaScript on a server. For if the server-side extensions allow an application to communicate with a database, and provide continuity of information from one invocation to another of the application, or perform file manipulations on a server. The useful framework which is the most famous these days is [node.js](https://www.geeksforgeeks.org/introduction-to-nodejs/).

**Imperative language –**In this type of language we are mostly concerned about how it is to be done. It simply controls the flow of computation. The procedural programming approach, object, oriented approach comes under this as async await we are thinking about what is to be done further after the async call.

**Declarative programming –**In this type of language we are concerned about how it is to be done, basically here logical computation requires. Her main goal is to describe the desired result without direct dictation on how to get it as the arrow function does.

JavaScript can be added to your HTML file in [two ways](https://www.geeksforgeeks.org/where-to-put-javascript-in-an-html-document/):

**Internal JS:** We can add JavaScript directly to our HTML file by writing the code inside the <script> tag. The <script> tag can either be placed inside the <head> or the <body> tag according to the requirement.

**External JS:** We can write JavaScript code in another files having an extension.js and then link this file inside the <head> tag of the HTML file in which we want to add this code.

**4.1.4.ANGULAR JS**

It changes the static HTML to dynamic HTML. Its features like dynamic binding and dependency injection eliminate the need for code that we have to write otherwise. Angular JS is rapidly growing and because of this reason, we have different versions of Angular Js with the latest stable being 1.7.7. It is also important to note that Angular is different from Angular Js. It is an open-source project which can be freely used and changed by anyone. It extends HTML attributes with Directives, and data is bound with HTML.

* Angular is a JavaScript framework that is mainly used for Front end Development.
* It is used for making Single Page Applications(SPA).
* It is open source and is completely free for everyone.
* It uses the Model, View, Control(MVC) pattern for developing projects.

**CHAPTER 5**

**MODULE DESCRIPTION**

###### MODULE DESCRIPTION

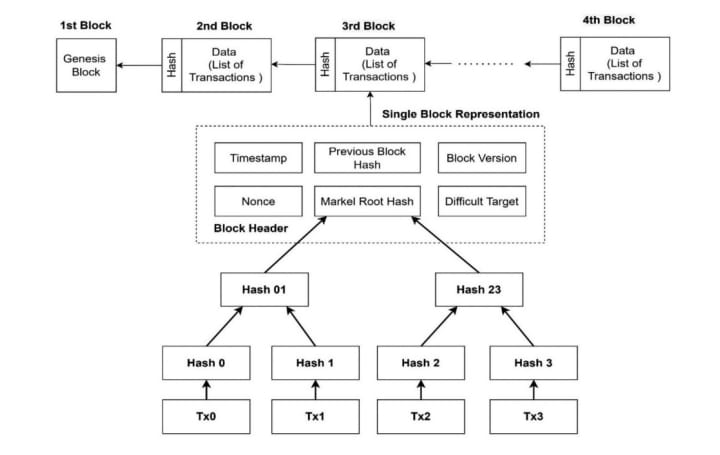
* 1. **PROJECT DEFINITION**

Blockchain-based bidding systems are becoming increasingly popular nowadays. Due to the properties of blockchain, bidding records are unchangeable. With existing encryption techniques, these bidding records can only be shared by the bidder and the seller. Although this scenario sounds secure, it does not consider a coercion case. A powerful coercer may force the bidding system to open the records stored on the blockchain, and the system loses privacy. To solve this problem, in this paper, we introduce a new encryption scheme called deniable matchmaking encryption (DME). This new encryption scheme provides deniability not only for the message, but also for the identities. We use the chameleon hash function to make fake message and fake identities indistinguishable from the real message and the real identities. Therefore, the bidding system can use fake information to answer the coercer, and user privacy is kept by the blockchain-based bidding system.

* 1. **OVERVIEW OF THE PROJECT**

Blockchain-based bidding systems are becoming increasingly popular nowadays. Due to the properties of blockchain, bidding records are unchangeable. With existing encryption techniques, these bidding records can only be shared by the bidder and the seller. Although this scenario sounds secure, it does not consider a coercion case. A powerful coercer may force the bidding system to open the records stored on the blockchain, and the system loses privacy. To solve this problem, in this paper, we introduce a new encryption scheme called deniable matchmaking encryption (DME). This new encryption scheme provides deniability not only for the message, but also for the identities. We use the chameleon hash function to make fake message and fake identities indistinguishable from the real message and the real identities. Therefore, the bidding system can use fake information to answer the coercer, and user privacy is kept by the blockchain-based bidding system.

* 1. **ARCHITECTURE DIAGRAM**



**Fig. 5.1: Architecture Diagram**

* 1. **MODULES**
     1. **USER AUTHENTICATION**

All users must register before using the system; The authenticator is responsible for verifying the legitimacy of requests sent by requestors to the data owners system. On a shared server, a security management model designed for password-authenticated users allows access to shared files. Although the scheme simplifies security management efforts, all backups and sharing files are still reside on the sharing server. Also, the file sharing policy relies on the authentication mechanism on the sharing server and is based on the users password to extract the key.

* + 1. **DIGITAL OWNER REGISTRATION**

Data sharing is one of important applications in computing, especially for enterprise. Usually, an enterprise may authorize some entities to share its remote data under the its defined policy. Data owner is an entity whose massive data will be uploaded to the servers for storage and processing. The data owner registers the copyright of the digital works on the system. During the registration process, the data information of the digital works needs to be extracted and stored; The data owner uploads these data blocks along with their corresponding signatures to the server. The data stored in the server is often shared across multiple users in many storage applications. In such a scheme, the sensitive information can be protected and the other information can be published. It makes the file stored in the server able to be shared and used by others on the condition that the sensitive information is protected, while the remote data integrity is still able to be efficiently executed.

* + 1. **DIGITAL INFORMATION QUERY**

Anyone needs to verify the Digital Information Ownership information, the proposed system is used to trace the entire process of circulation to investigate and retrieve the information. Hence, the system provides the query functions. The data query layer consists of sets of querying structures that access, process, forward or respond to queries posed on the system. Queries on the systems may be requests to access data from the existing database infrastructure. The data query layer directly interfaces with the data structuring and provenance layer and has

mechanisms, implemented to interpret and translate actions between the data structuring and provenance layer and the outside environment.

# CHAPTER 6

**SYSTEM TESTING**

* 1. **CODE REVIEW**

After successful completion of coding, Code review was done with the objective of identifying and correcting deviations from standards, Identifying and fixing logical bugs and fall through and recording code walk through findings. The programs were checked and the code structure was made readable. The variable names were meaningful. It follows certain naming conventions, which makes the program readable.

* + - Variable names are prefixed with their scope and data type.
    - Checking out for the correct scope for various function.
    - All possible explanation for the code were given as comment.
    - Sufficient labels and comments were included in the code as the description of it for the benefit of the developer and other programmers who might examine it later.
    - Checking out the connectivity of the database.
    - Code optimization was carried out.
  1. **TESTING PROCESS**
     + Testing is the process of executing the program with the intent of finding an error
     + A good test has a high probability of finding an as yet undiscovered error.
     + A successful test is one that uncovers an as yet undiscovered error.
     1. **Unit Testing**

Number of input parameters should be equal to the number of arguments Parameter and argument attributes must match .Parameter passed should be in correct order. Global variable definition consistent across module. If module does input, File attribute should be correct. Open /close statement should be correct, Buffer size should match record size, Input error should handle. File should be open before used, Any textual error in output information should be checked.

* + 1. **Integration Testing**

Modules integrated by moving down the program design hierarchy. Top-level structure tested most. Depth first implementation allows the complete function to be implemented, tested and demonstrated. Integration Testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

* + 1. **Validation Testing**

Validation testing is aims to demonstrate that the software function in a manner that can be easily reasonably expected by the customer. A set of black box test is to demonstrate conformance with requirements to check that all functional requirements are satisfied, all performance requirements achieved and other requirements are met. When validation testing fails it may be too late to correct the error prior to scheduled delivery.

* + 1. **System Testing**

Software is only one component of a system. Software will be incorporated with other system components and system integration and validation test performance. Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test meets the requirements that guided its design and development, responds correctly to all kinds of inputs, performs its functions within an acceptable time, it is sufficiently usable, can be installed and run in its intended [environments](https://en.wikipedia.org/wiki/Operating_environment), and Achieves the general result its stakeholder’s desire.

* + 1. **Performance Testing**

For real-time and embedded systems, functional requirements may be satisfied but performance problems make the system unacceptable Performance testing checks the runtime performance in the context of integrated system.

# CHAPTER 7

### SYSTEM IMPLEMENTATION

###### SYSTEM IMPLEMENTATION

**7.1. IMPLEMENTATION PROCEDURE**

Project implementation is the process of putting a project plan into action to produce the deliverable, otherwise known as the products or services, for clients or stakeholders. It takes place after the planning phase, during which a team determines the key objectives for the project, as well as the timeline and budget. Implementation involves coordinating resources and measuring performance to ensure the project remains within its expected scope and budget. It also involves handling any unforeseen issues in a way that keeps a project running smoothly. Project implementation that relies on strategic planning outlined earlier in the process can help a team achieve the project objectives while staying within budget and meeting relevant deadlines. Implementation is the part of the project cycle that bridges the planning process and the project outcomes. This step of the process, and how well it's executed, can ultimately determine the success of a project.

# CHAPTER 8

### CONCLUSION AND FUTUREENHANCEMENT

###### CONCLUSION AND FUTURE ENHANCEMENTS

* 1. **CONCLUSION**

The proposed study can be used to establish a database for copyright protected works and to transfer rights through smart contracts without relying on trusted third parties or authorities. Although a block chain-based database is slower and more expensive than a traditional database it provides more security stability transparency

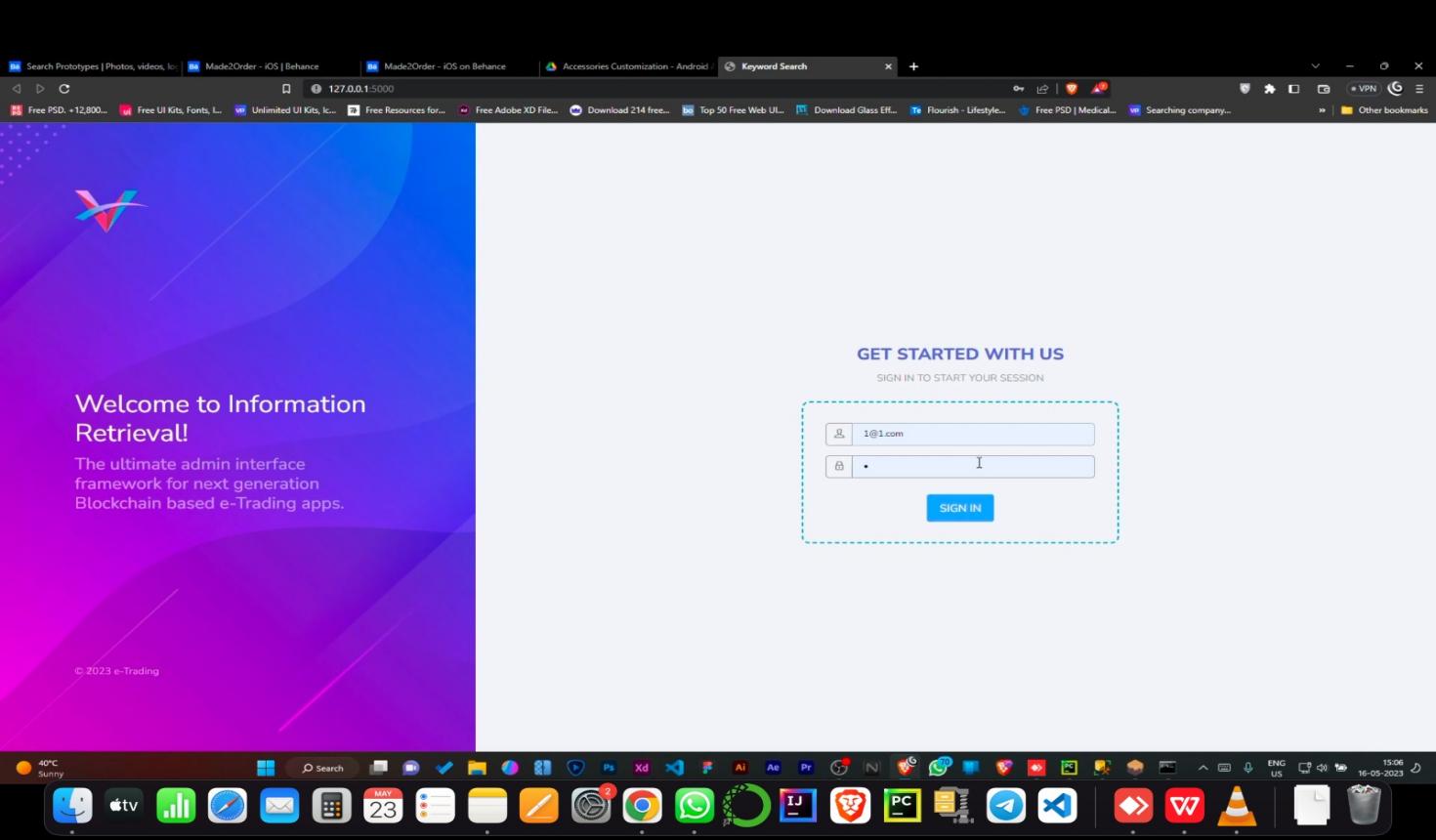
and tamper protection. In this paper we proposed a data sharing scheme which can achieve the anonymity and data confidentiality in server. We formalize the definition and the security model. Then we designed a concrete data sharing scheme and gave the security proof. In our scheme the file stored in the server can be shared and used by others on the condition that the sensitive information of the file is protected. Besides the remote data integrity auditing is still able to be efficiently executed.

* 1. **FUTURE ENHANCEMENTS**
* In future we like to implement Identity-Based Auditing for Shared Cloud Data

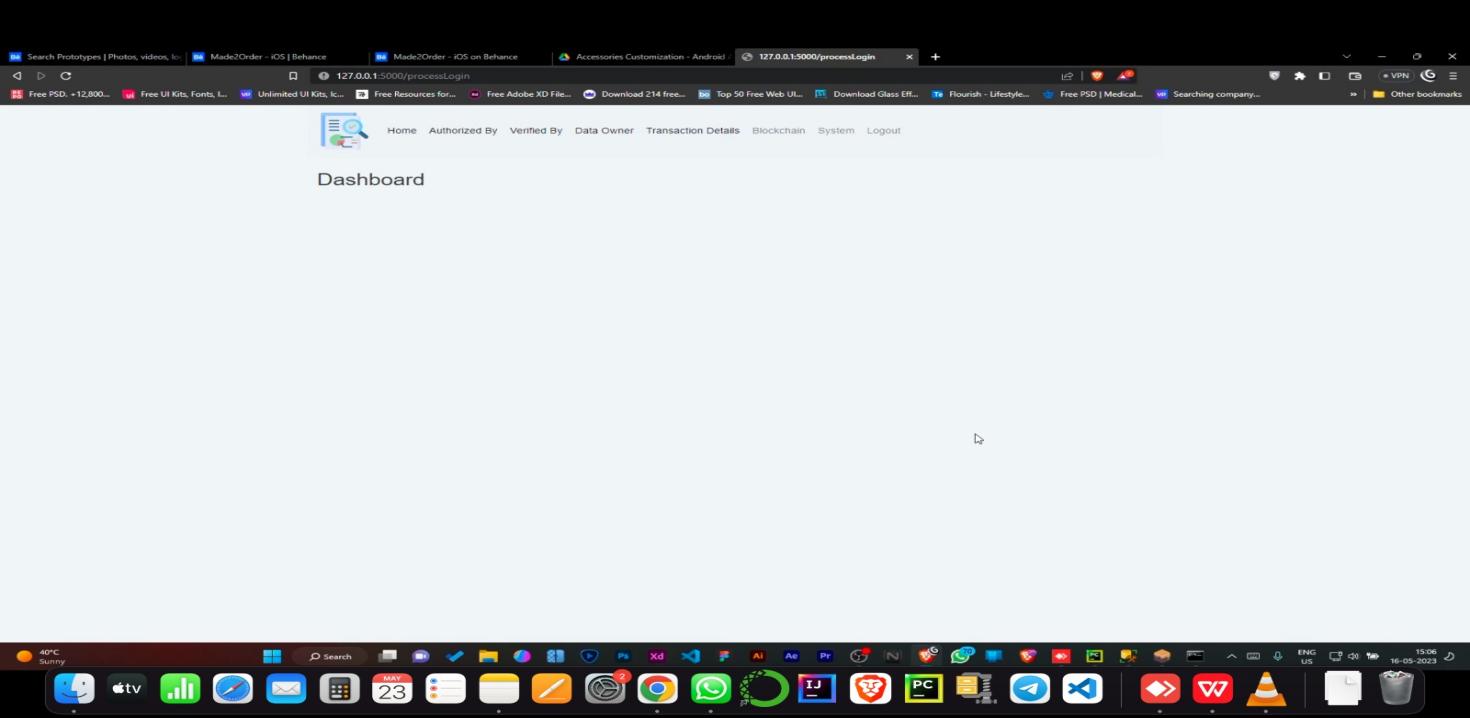
**APPENDICES**

**SCREENSHOTS**

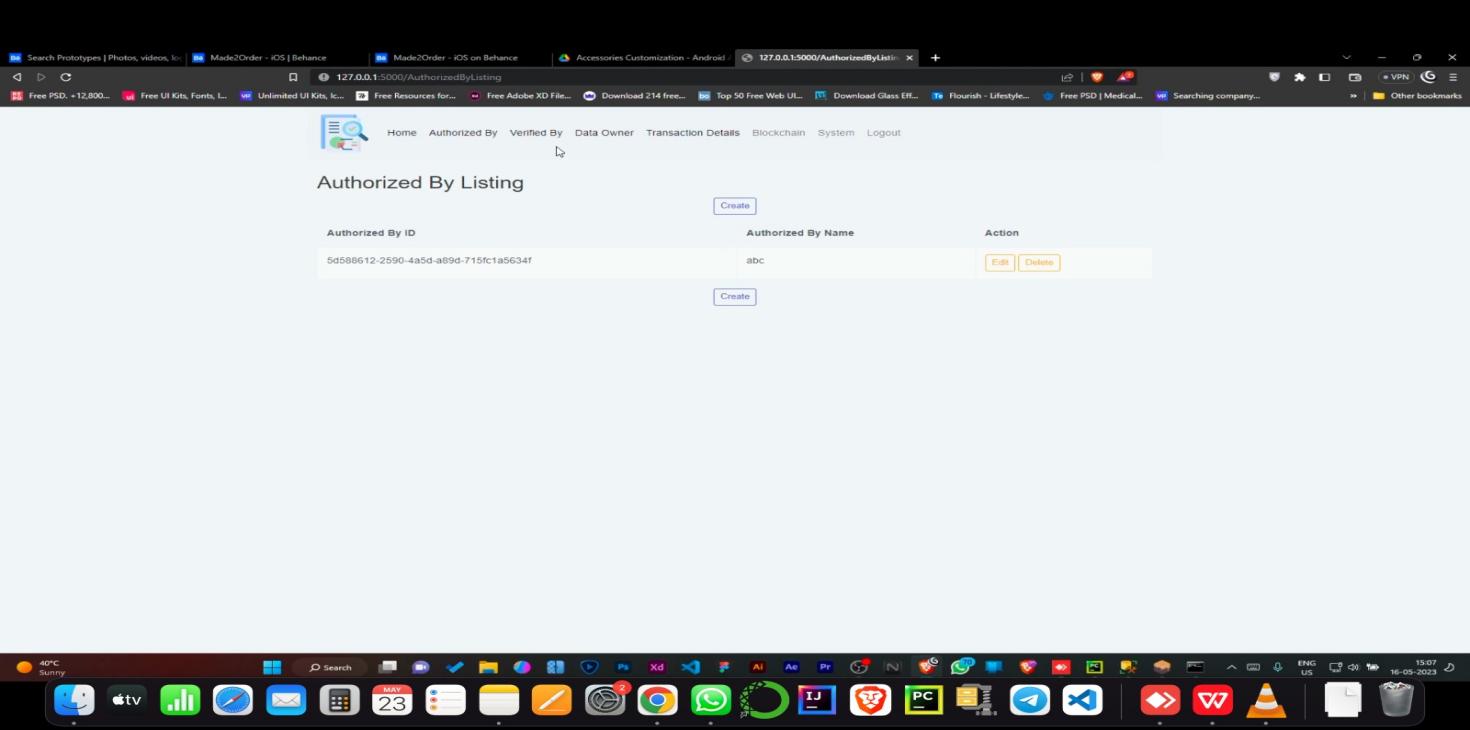
**Register Page :**

****

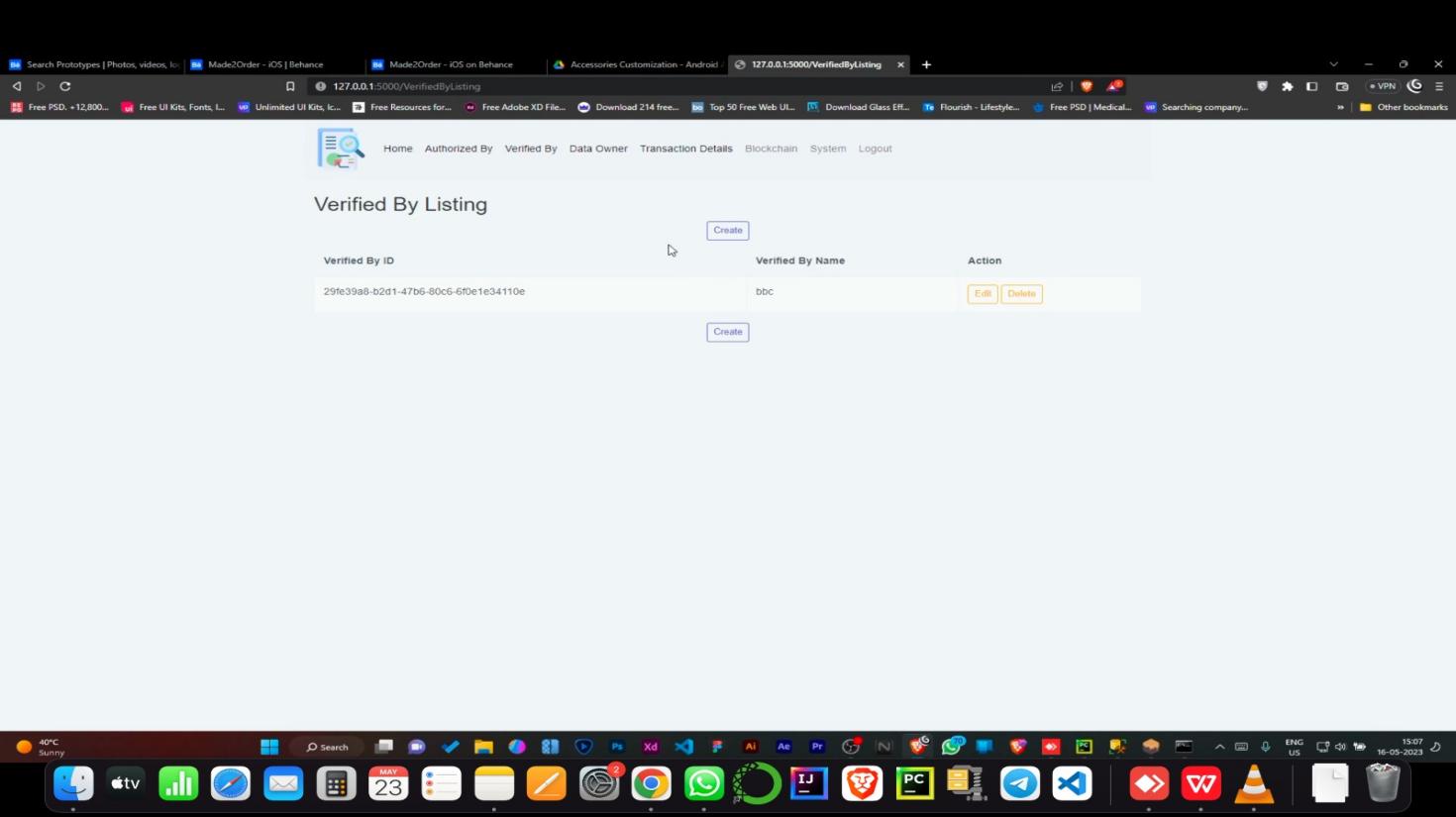
**Home Dashboard :**



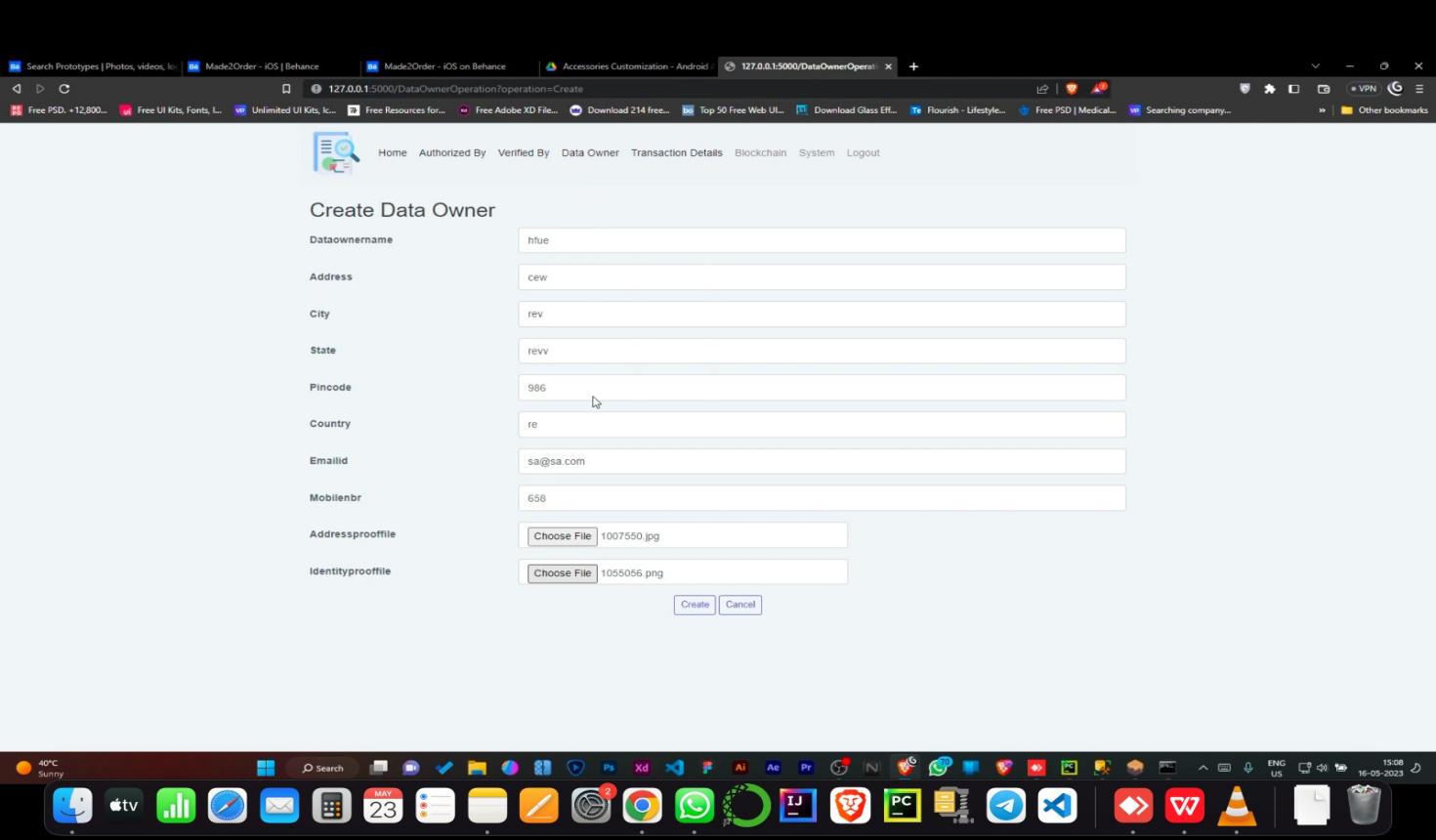
**Authorized By Listing:**

****

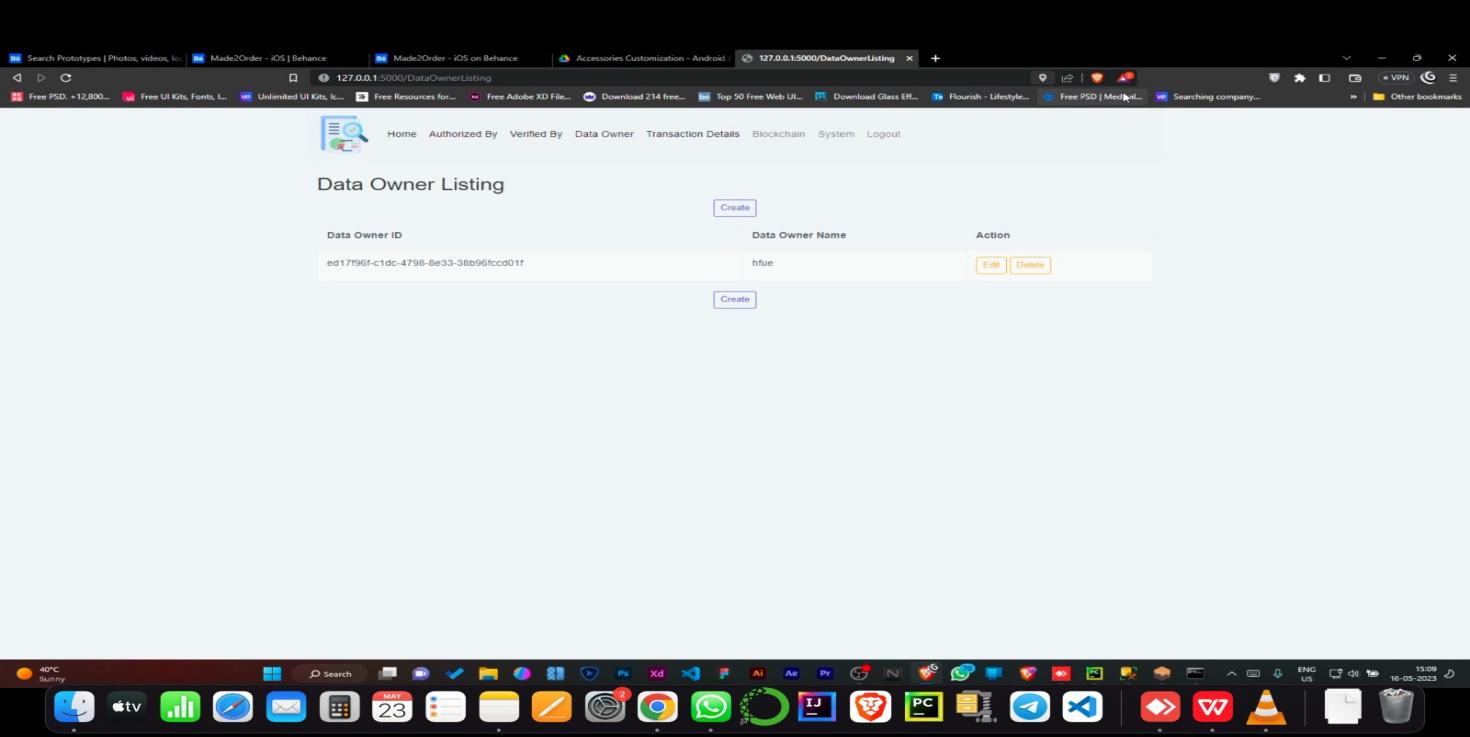
**Verified By Listing:**

****

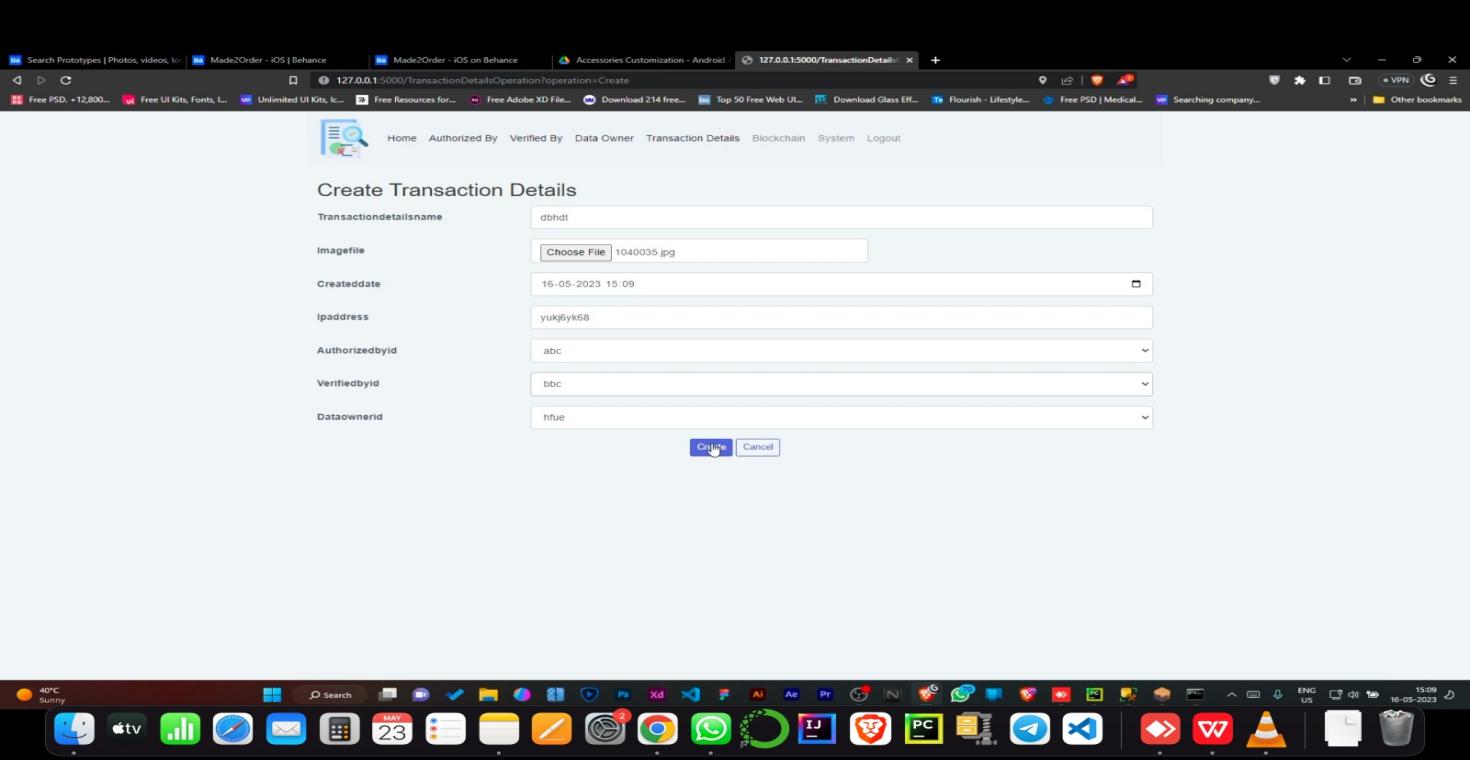
**Create Data Owner**:



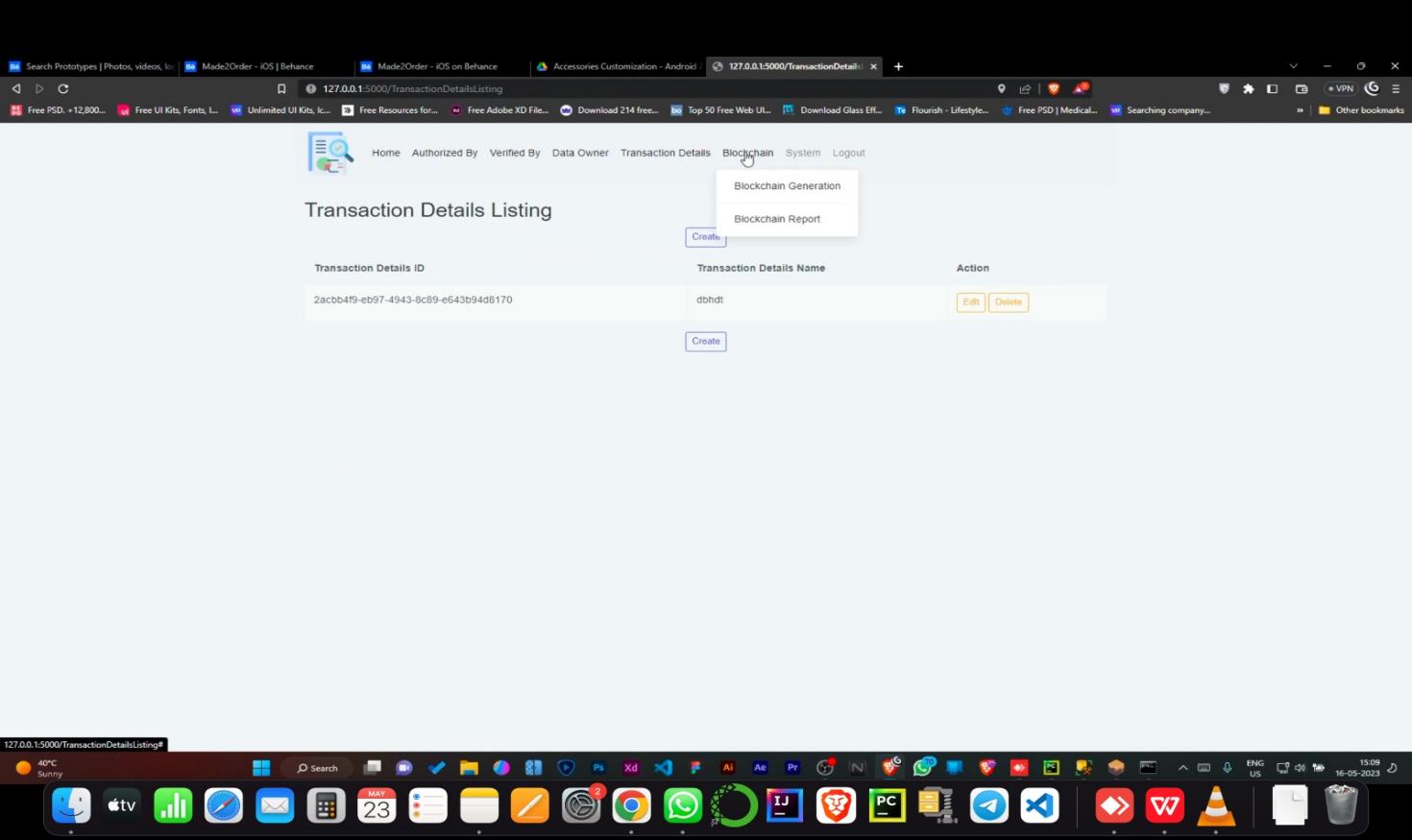
**Data Owner Listing:**



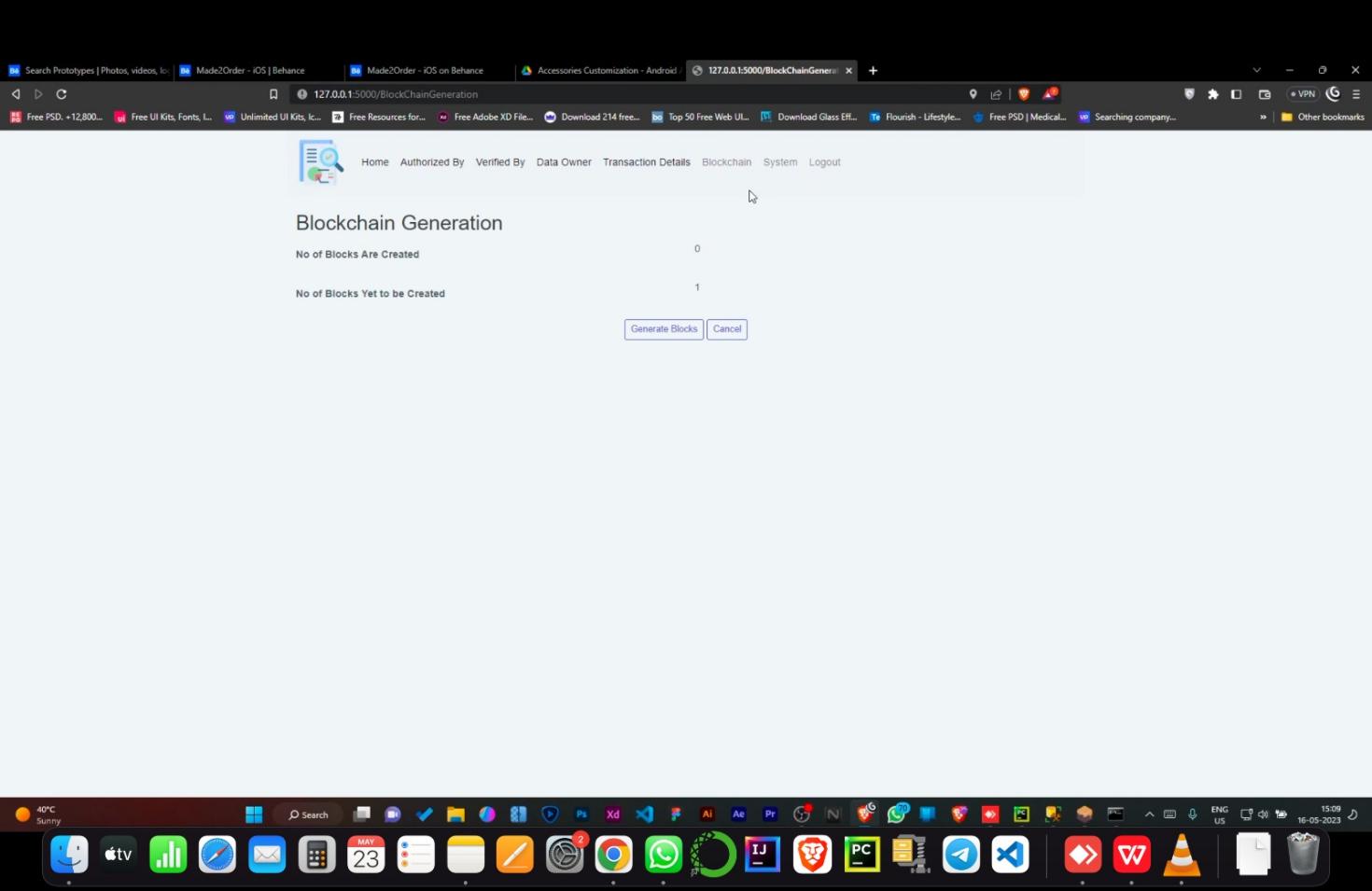
**Create transaction Details:**



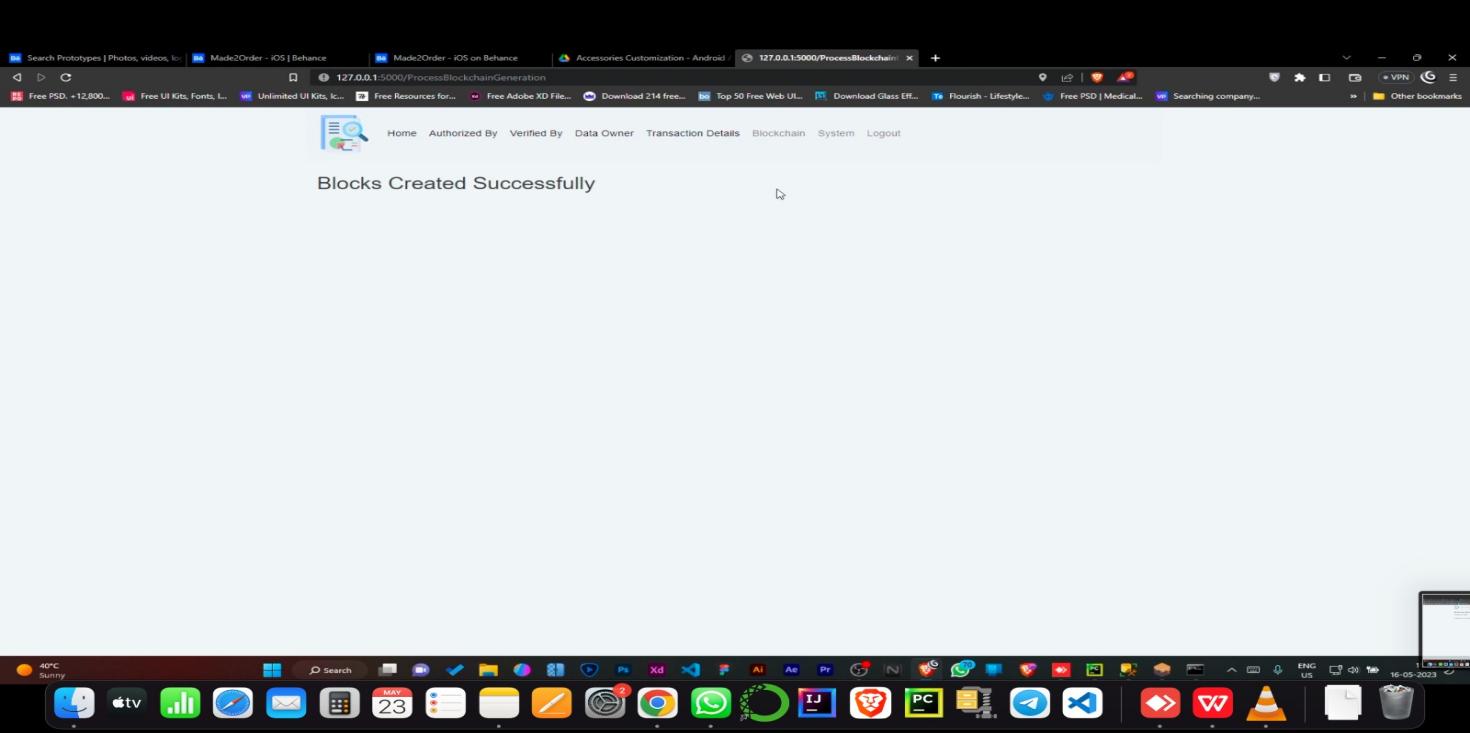
**Transaction Details Listing:**



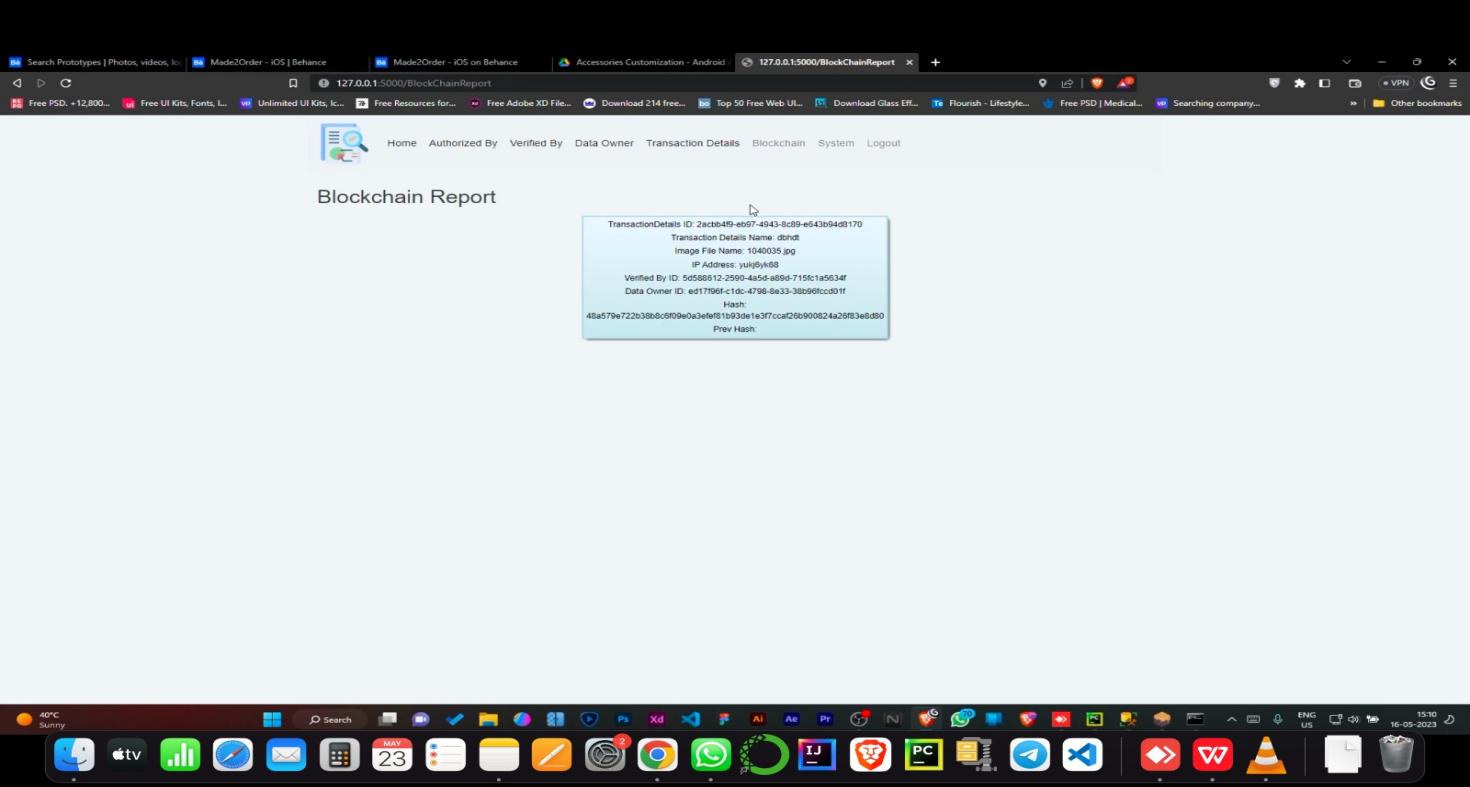
**Blockchain Generation:**

****

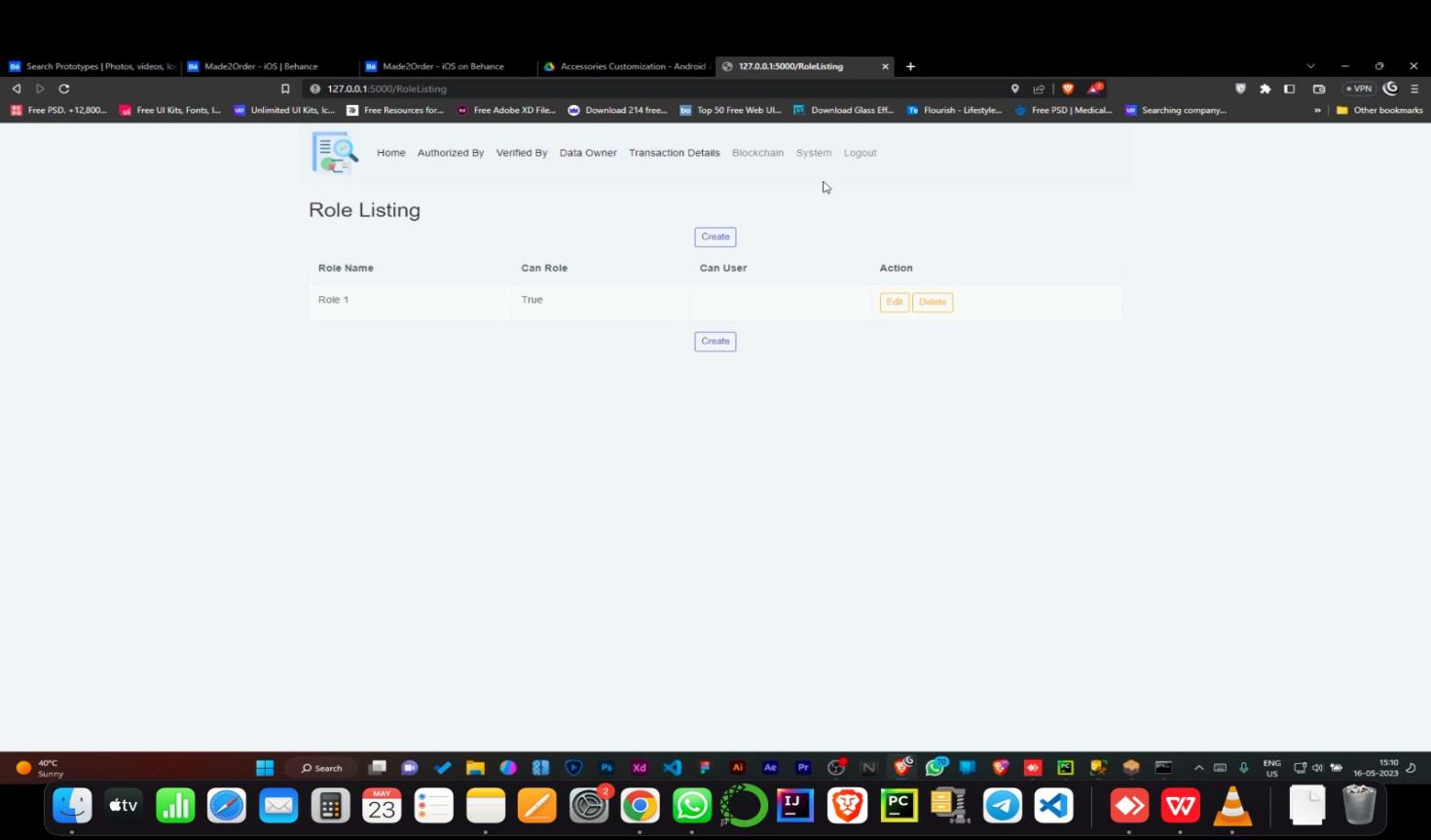
**Block Creation:**

****

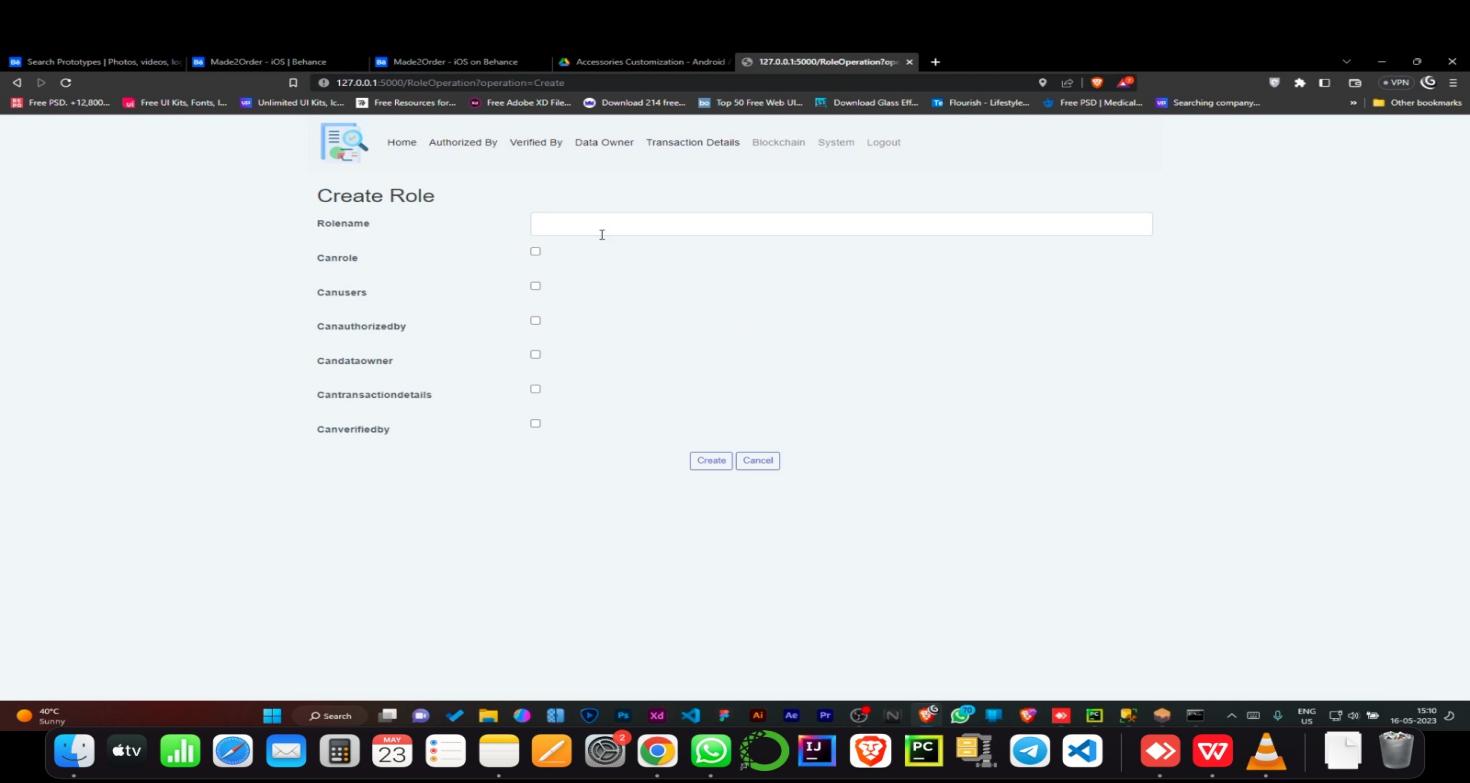
**Blochchain Report:**



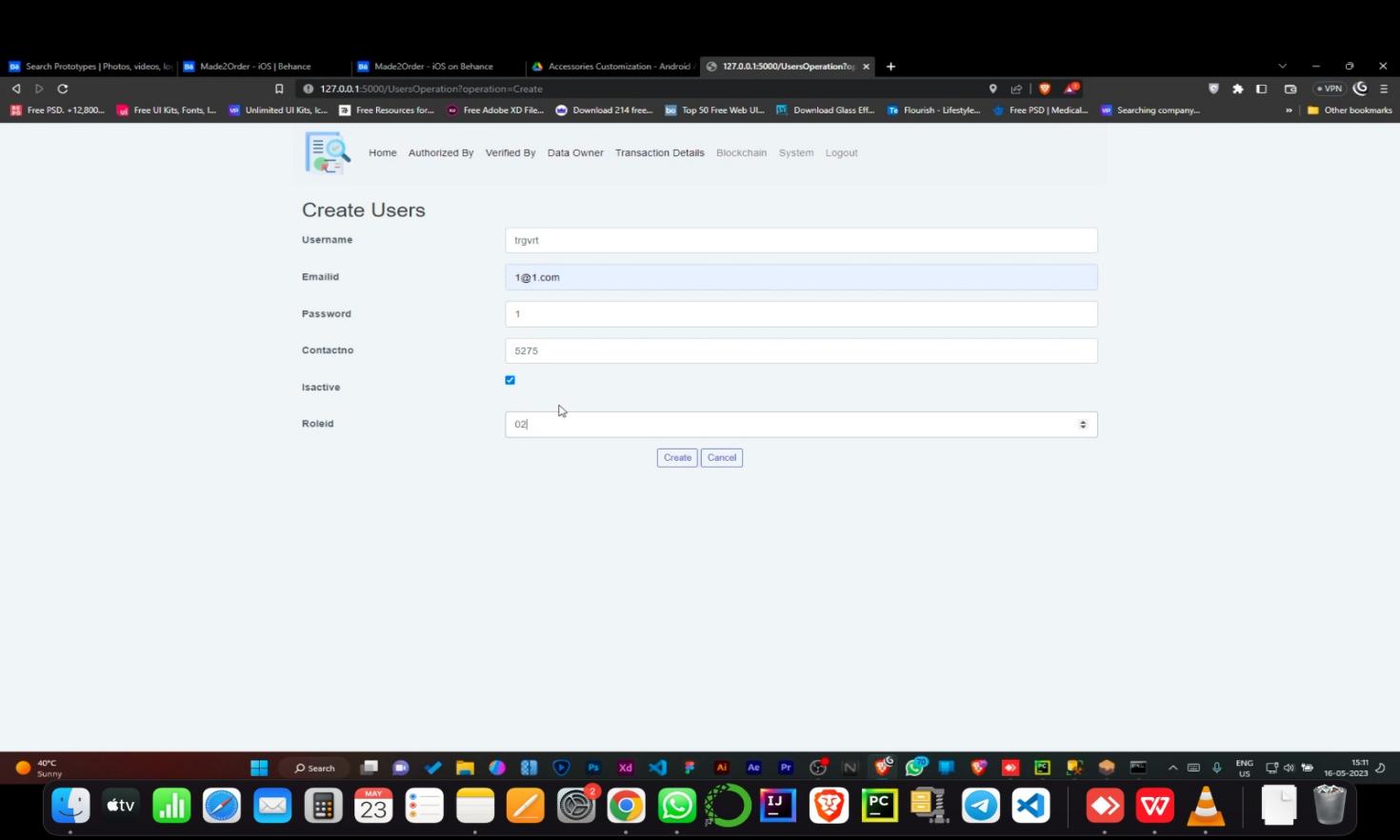
**Role Listing:**



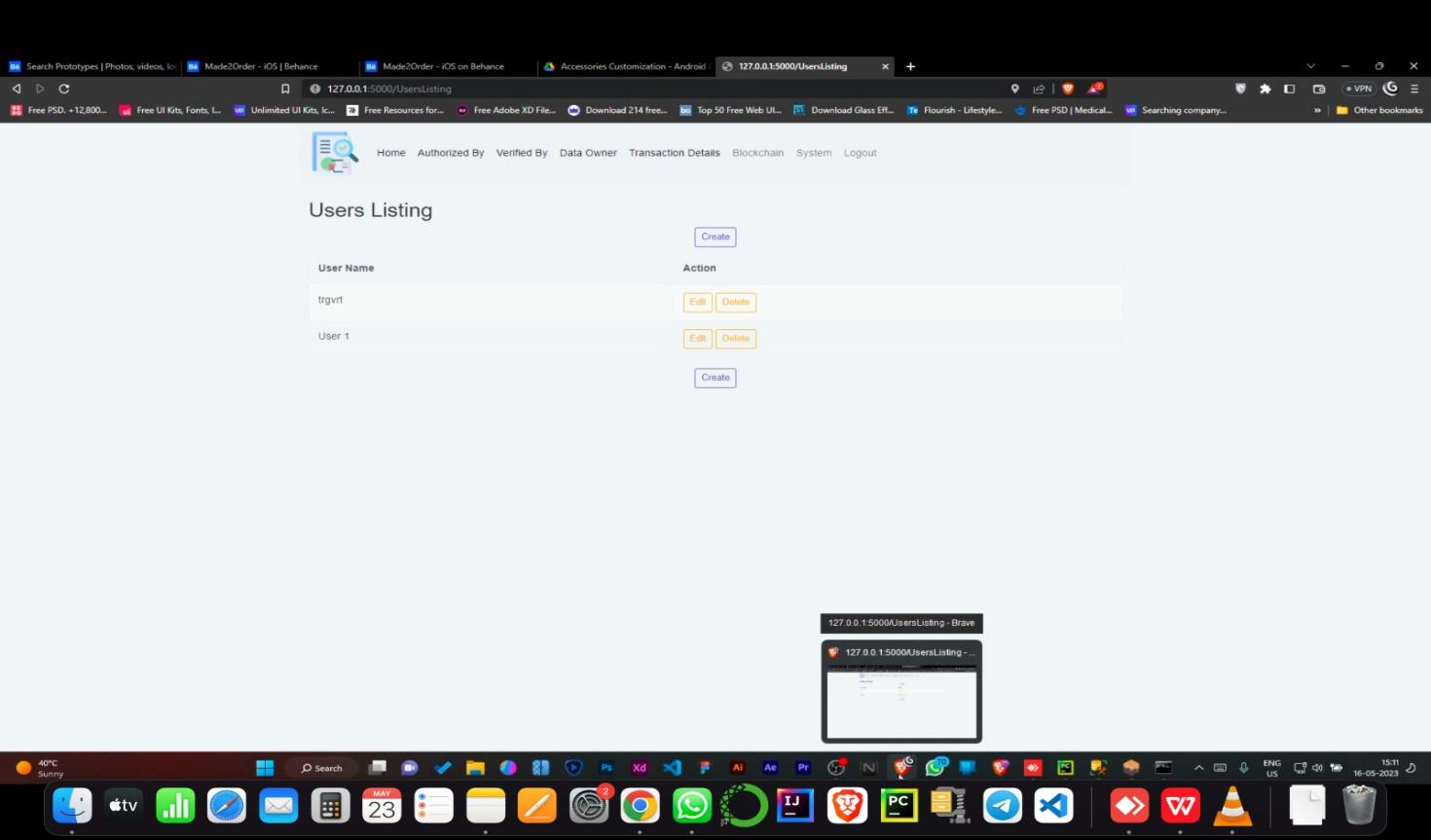
**Create Role:**



**Create Users:**



**Users Listing:**

****

###### APPENDIX – 2 IMPLEMENTATION CODE

**Index.py**

from flask import Flask, request, render\_template, redirect, url\_for import os

import pyodbc import uuid import time

from datetime import datetime from Constants import connString

from AuthorizedByModel import AuthorizedByModel from DataOwnerModel import DataOwnerModel

from RoleModel import RoleModel

from TransactionDetailsModel import TransactionDetailsModel from UsersModel import UsersModel

from VerifiedByModel import VerifiedByModel

app = Flask( name ) app.secret\_key = "MySecret" ctx = app.app\_context() ctx.push()

with ctx: pass

user\_id = "" emailid = "" role\_object = None message = "" msgType = ""

uploaded\_file\_name = ""

def initialize():

global message, msgType message = ""

msgType = ""

def process\_role(option\_id):

if option\_id == 0:

if role\_object.canAuthorizedBy == False: return False

if option\_id == 1:

if role\_object.canDataOwner == False: return False

if option\_id == 2:

if role\_object.canRole == False: return False

if option\_id == 3:

if role\_object.canTransactionDetails == False: return False

if option\_id == 4:

if role\_object.canUsers == False: return False

if option\_id == 5:

if role\_object.canVerifiedBy == False: return False

return True

@app.route("/") def index():

global user\_id, emailid

return render\_template("Login.html")

@app.route("/processLogin", methods=["POST"]) def processLogin():

global user\_id, emailid, role\_object emailid = request.form["emailid"] password = request.form["password"]

conn1 = pyodbc.connect(connString, autocommit=True) cur1 = conn1.cursor()

sqlcmd1 = "SELECT \* FROM Users WHERE emailid = '" + emailid + "' AND password = '" + password + "' AND isActive = 1"; cur1.execute(sqlcmd1)

row = cur1.fetchone()

cur1.commit() if not row:

return render\_template("Login.html", processResult="Invalid Credentials") user\_id = row[0]

cur2 = conn1.cursor()

sqlcmd2 = "SELECT \* FROM Role WHERE RoleID = '" + str(row[6]) + "'" cur2.execute(sqlcmd2)

row2 = cur2.fetchone()

if not row2:

return render\_template("Login.html", processResult="Invalid Role")

role\_object = RoleModel(row2[0], row2[1], row2[2], row2[3], row2[4], row2[5], row2[6], row2[7]) return render\_template("Dashboard.html")

@app.route("/ChangePassword") def changePassword():

global user\_id, emailid

return render\_template("ChangePassword.html")

@app.route("/ProcessChangePassword", methods=["POST"]) def processChangePassword():

global user\_id, emailid

oldPassword = request.form["oldPassword"] newPassword = request.form["newPassword"] confirmPassword = request.form["confirmPassword"] conn1 = pyodbc.connect(connString, autocommit=True) cur1 = conn1.cursor()

sqlcmd1 = "SELECT \* FROM Users WHERE emailid = '" + emailid + "' AND password = '" + oldPassword + "'"; cur1.execute(sqlcmd1)

row = cur1.fetchone()

cur1.commit() if not row:

return render\_template("ChangePassword.html", msg="Invalid Old Password")

if newPassword.strip() != confirmPassword.strip():

return render\_template("ChangePassword.html", msg="New Password and Confirm Password are NOT same")

conn2 = pyodbc.connect(connString, autocommit=True) cur2 = conn2.cursor()

sqlcmd2 = "UPDATE Users SET password = '" + newPassword + "' WHERE emailid = '" + emailid + "'"; cur1.execute(sqlcmd2)

cur2.commit()

return render\_template("ChangePassword.html", msg="Password Changed Successfully")

@app.route("/Dashboard") def Dashboard():

global user\_id, emailid

return render\_template("Dashboard.html")

@app.route("/Information") def Information():

global message, msgType

return render\_template("Information.html", msgType=msgType, message=message)

@app.route("/AuthorizedByListing") def AuthorizedBy\_listing():

global user\_id, email id

global message, msgType, role\_object if role\_object == None:

message = "Application Error Occurred. Logout" msgType = "Error"

return redirect(url\_for("Information")) canAuthorizedBy = process\_role(0)

if canAuthorizedBy == False:

message = "You Don't Have Permission to Access AuthorizedBy" msgType = "Error"

return redirect(url\_for("Information")) records = AuthorizedByModel.get\_all()

return render\_template("AuthorizedByListing.html", records=records)

@app.route("/AuthorizedByOperation") def AuthorizedBy\_operation():

global user\_id, user\_name, message, msgType, role\_object if role\_object is None:

message = "Application Error Occurred. Logout" msgType = "Error"

return redirect(url\_for("Information")) canAuthorizedBy = process\_role(0)

if not canAuthorizedBy:

message = "You Don't Have Permission to Access AuthorizedBy" msgType = "Error"

return redirect(url\_for("Information")) unique\_id = ""

operation = request.args.get("operation") row = AuthorizedByModel("", "")

AuthorizedBy = AuthorizedByModel.get\_all() if operation != "Create":

unique\_id = request.args.get("unique\_id").strip() row = AuthorizedByModel.get\_by\_id(unique\_id)

return render\_template("AuthorizedByOperation.html", row=row, operation=operation, AuthorizedBy=AuthorizedBy, ) @app.route("/ProcessAuthorizedByOperation", methods=["POST"])

def process\_AuthorizedBy\_operation():

global user\_id, user\_name, message, msgType, role\_object

if role\_object is None:

message = "Application Error Occurred. Logout" msgType = "Error"

return redirect(url\_for("/"))

canAuthorizedBy = process\_role(0) if not canAuthorizedBy:

message = "You Don't Have Permission to Access AuthorizedBy" msgType = "Error"

return redirect(url\_for("Information"))

operation = request.form["operation"] obj = AuthorizedByModel("", "")

if operation != "Delete":

obj.authorizedByID = request.form['authorizedByID'] obj.authorizedByName = request.form['authorizedByName']

if operation == "Create": obj.insert(obj)

if operation == "Edit":

obj.authorizedByID = request.form["authorizedByID"] obj.update(obj)

if operation == "Delete":

authorizedByID = request.form["authorizedByID"] obj.delete(authorizedByID)

return redirect(url\_for("AuthorizedBy\_listing")) @app.route("/DataOwnerListing")

def DataOwner\_listing(): global user\_id, emailid

global message, msgType, role\_object if role\_object == None:

message = "Application Error Occurred. Logout" msgType = "Error"

return redirect(url\_for("Information")) canDataOwner = process\_role(1)

if canDataOwner == False:

message = "You Don't Have Permission to Access DataOwner" msgType = "Error"

return redirect(url\_for("Information")) records = DataOwnerModel.get\_all()

return render\_template("DataOwnerListing.html", records=records)

@app.route("/DataOwnerOperation") def DataOwner\_operation():

global user\_id, user\_name, message, msgType, role\_object if role\_object is None:

message = "Application Error Occurred. Logout" msgType = "Error"

return redirect(url\_for("Information")) canDataOwner = process\_role(1)

if not canDataOwner:

message = "You Don't Have Permission to Access DataOwner" msgType = "Error"

return redirect(url\_for("Information")) unique\_id = ""

operation = request.args.get("operation") row = DataOwnerModel("", "")

DataOwner = DataOwnerModel.get\_all() if operation != "Create":

unique\_id = request.args.get("unique\_id").strip() row = DataOwnerModel.get\_by\_id(unique\_id)

return render\_template("DataOwnerOperation.html", row=row, operation=operation, DataOwner=DataOwner, ) @app.route("/ProcessDataOwnerOperation", methods=["POST"])

def process\_DataOwner\_operation():

global user\_id, user\_name, message, msgType, role\_object

if role\_object is None:

message = "Application Error Occurred. Logout" msgType = "Error"

return redirect(url\_for("/"))

canDataOwner = process\_role(1) if not canDataOwner:

message = "You Don't Have Permission to Access DataOwner" msgType = "Error"

return redirect(url\_for("Information"))

operation = request.form["operation"] obj = DataOwnerModel("", "")

if operation != "Delete":

obj.dataOwnerID = request.form['dataOwnerID'] obj.dataOwnerName = request.form['dataOwnerName'] obj.address = request.form['address']

obj.city = request.form['city'] obj.state = request.form['state'] obj.pincode = request.form['pincode'] obj.country = request.form['country'] obj.emailID = request.form['emailID']

obj.mobileNbr = request.form['mobileNbr'] if len(request.files) != 0 :

file = request.files['addressProofFile'] if file.filename != '':

addressProofFile = file.filename obj.addressProofFile = addressProofFile

f = os.path.join('static/UPLOADED\_FILES', addressProofFile) file.save(f)

else:

obj.addressProofFile = request.form['haddressProofFile']

file = request.files['identityProofFile'] if file.filename != '':

identityProofFile = file.filename obj.identityProofFile = identityProofFile

f = os.path.join('static/UPLOADED\_FILES', identityProofFile) file.save(f)

else:

obj.identityProofFile = request.form['hidentityProofFile']

if operation == "Create": obj.insert(obj)

if operation == "Edit":

obj.dataOwnerID = request.form["dataOwnerID"] obj.update(obj)

if operation == "Delete":

dataOwnerID = request.form["dataOwnerID"] obj.delete(dataOwnerID)

return redirect(url\_for("DataOwner\_listing")) @app.route("/RoleListing")

def Role\_listing():

global user\_id, emailid

global message, msgType, role\_object if role\_object == None:

message = "Application Error Occurred. Logout" msgType = "Error"

return redirect(url\_for("Information")) canRole = process\_role(2)

if canRole == False:

message = "You Don't Have Permission to Access Role" msgType = "Error"

return redirect(url\_for("Information")) records = RoleModel.get\_all()

return render\_template("RoleListing.html", records=records)

@app.route("/RoleOperation") def Role\_operation():

global user\_id, user\_name, message, msgType, role\_object

if role\_object is None:

message = "Application Error Occurred. Logout" msgType = "Error"

return redirect(url\_for("Information")) canRole = process\_role(2)

if not canRole:

message = "You Don't Have Permission to Access Role" msgType = "Error"

return redirect(url\_for("Information")) unique\_id = ""

operation = request.args.get("operation") row = RoleModel("", "")

Role = RoleModel.get\_all() if operation != "Create":

if canVerifiedBy == False:

message = "You Don't Have Permission to Access VerifiedBy" msgType = "Error"

return redirect(url\_for("Information")) records = VerifiedByModel.get\_all()

return render\_template("VerifiedByListing.html", records=records)

@app.route("/VerifiedByOperation") def VerifiedBy\_operation():

global user\_id, user\_name, message, msgType, role\_object if role\_object is None:

message = "Application Error Occurred. Logout" msgType = "Error"

return redirect(url\_for("Information")) canVerifiedBy = process\_role(5)

if not canVerifiedBy:

message = "You Don't Have Permission to Access VerifiedBy" msgType = "Error"

return redirect(url\_for("Information")) unique\_id = ""

operation = request.args.get("operation") row = VerifiedByModel("", "")

VerifiedBy = VerifiedByModel.get\_all() if operation != "Create":

unique\_id = request.args.get("unique\_id").strip() row = VerifiedByModel.get\_by\_id(unique\_id)

return render\_template("VerifiedByOperation.html", row=row, operation=operation, VerifiedBy=VerifiedBy, ) @app.route("/ProcessVerifiedByOperation", methods=["POST"])

def process\_VerifiedBy\_operation():

global user\_id, user\_name, message, msgType, role\_object

if role\_object is None:

message = "Application Error Occurred. Logout" msgType = "Error"

return redirect(url\_for("/"))

canVerifiedBy = process\_role(5) if not canVerifiedBy:

message = "You Don't Have Permission to Access VerifiedBy" msgType = "Error"

@app.route("/BlockChainGeneration") def BlockChainGeneration():

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd = "SELECT COUNT(\*) FROM TransactionDetails WHERE isBlockChainGenerated = 1" cursor.execute(sqlcmd)

while True:

dbrow = cursor.fetchone() if not dbrow:

break

blocksCreated = dbrow[0]

sqlcmd = "SELECT COUNT(\*) FROM TransactionDetails WHERE isBlockChainGenerated = 0 or isBlockChainGenerated is null" cursor.execute(sqlcmd)

sqlcmd = "SELECT \* FROM TransactionDetails WHERE isBlockChainGenerated = 0 or isBlockChainGenerated is null ORDER BY sequenceNumber" cursor.execute(sqlcmd)

while True:

sqlcmd1 = ""

dbrow = cursor.fetchone() if not dbrow:

break

unqid = str(dbrow[11])

bdata = str(dbrow[1]) + str(dbrow[2]) + str(dbrow[3]) + str(dbrow[4]) block\_serialized = json.dumps(bdata, sort\_keys=True).encode('utf-8') block\_hash = hashlib.sha256(block\_serialized).hexdigest()

conn1 = pyodbc.connect(connString, autocommit=True) cursor1 = conn1.cursor()

sqlcmd1 = "UPDATE TransactionDetails SET isBlockChainGenerated = 1, hash = '" + block\_hash + "', prevHash = '" + prevHash + "' WHERE sequenceNumber = '" + unqid + "'" cursor1.execute(sqlcmd1)

cursor1.close() conn1.close() prevHash = block\_hash

return render\_template('BlockchainGenerationResult.html')

@app.route("/BlockChainReport") def BlockChainReport():

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT \* FROM TransactionDetails WHERE isBlockChainGenerated = 1" cursor.execute(sqlcmd1)

conn2 = pyodbc.connect(connString, autocommit=True) cursor = conn2.cursor()

sqlcmd1 = "SELECT \* FROM TransactionDetails ORDER BY sequenceNumber DESC" cursor.execute(sqlcmd1)

records = []

while True:

dbrow = cursor.fetchone() if not dbrow:

break

row = TransactionDetailsModel(dbrow[0],dbrow[1],dbrow[2],dbrow[3],dbrow[4],dbrow[5],dbrow[6],dbrow[7],dbrow[8],dbrow[9],dbrow[10],dbrow[11]) records.append(row)

return render\_template('BlockChainReport.html', records=records)

if name == " main ": app.run()

**Data Owner Model.py**

from Constants import connString import pyodbc

import datetime import uuid import time

class DataOwnerModel:

def init (self, dataOwnerID = '',dataOwnerName = '',address = '',city = '',state = '',pincode = '',country = '',emailID = '',mobileNbr = '',addressProofFile = '',identityProof self.dataOwnerID = dataOwnerID

self.dataOwnerName = dataOwnerName self.address = address

self.city = city self.state = state self.pincode = pincode self.country = country self.emailID = emailID self.mobileNbr = mobileNbr

self.addressProofFile = addressProofFile self.identityProofFile = identityProofFile self.emailModel = emailModel

@staticmethod def get\_all():

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT \* FROM DataOwner ORDER BY dataOwnerName" cursor.execute(sqlcmd1)

records = []

for dbrow in cursor.fetchall():

row = DataOwnerModel(dbrow[0],dbrow[1],dbrow[2],dbrow[3],dbrow[4],dbrow[5],dbrow[6],dbrow[7],dbrow[8],dbrow[9],dbrow[10]) records.append(row)

cursor.close() conn.close() return records

@staticmethod

def get\_name\_id():

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT dataOwnerID, dataOwnerName FROM DataOwner ORDER BY dataOwnerName" cursor.execute(sqlcmd1)

records = []

for dbrow in cursor.fetchall():

row = DataOwnerModel(dbrow[0],dbrow[1]) records.append(row)

cursor.close() conn.close() return records

@staticmethod

def get\_by\_id(unique\_id):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT \* FROM DataOwner WHERE dataOwnerID = ?" cursor.execute(sqlcmd1, unique\_id)

record = None

for dbrow in cursor.fetchall():

record = DataOwnerModel(dbrow[0],dbrow[1],dbrow[2],dbrow[3],dbrow[4],dbrow[5],dbrow[6],dbrow[7],dbrow[8],dbrow[9],dbrow[10]) cursor.close()

conn.close() return record

@staticmethod def insert(obj):

obj.dataOwnerID = str(uuid.uuid4())

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "INSERT INTO DataOwner (dataOwnerID,dataOwnerName,address,city,state,pincode,country,emailID,mobileNbr,addressProofFile,identityProofFile) VALUES(?,?,?,?,?,?,?,?,?,?, cursor.execute(sqlcmd1, (obj.dataOwnerID,obj.dataOwnerName,obj.address,obj.city,obj.state,obj.pincode,obj.country,obj.emailID,obj.mobileNbr,obj.addressProofFile,obj.identityPr cursor.close()

conn.close()

@staticmethod def update(obj):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "UPDATE DataOwner SET dataOwnerName = ?,address = ?,city = ?,state = ?,pincode = ?,country = ?,emailID = ?,mobileNbr = ?,addressProofFile = ?,identityProofFile = ? WH cursor.execute(sqlcmd1, (obj.dataOwnerName,obj.address,obj.city,obj.state,obj.pincode,obj.country,obj.emailID,obj.mobileNbr,obj.addressProofFile,obj.identityProofFile,obj.dat cursor.close()

conn.close()

@staticmethod

def delete(unique\_id):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "DELETE FROM DataOwner WHERE dataOwnerID = ?" cursor.execute(sqlcmd1, (unique\_id))

cursor.close() conn.close()

**Role Model.py**

from Constants import connString import pyodbc

import datetime import uuid import time

class RoleModel:

def init (self, roleID = 0,roleName = '',canRole = False,canUsers = False,canAuthorizedBy = False,canDataOwner = False,canTransactionDetails = False,canVerifiedBy = False): self.roleID = roleID

self.roleName = roleName self.canRole = canRole self.canUsers = canUsers

self.canAuthorizedBy = canAuthorizedBy self.canDataOwner = canDataOwner self.canTransactionDetails = canTransactionDetails self.canVerifiedBy = canVerifiedBy

@staticmethod def get\_all():

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT \* FROM Role ORDER BY roleName" cursor.execute(sqlcmd1)

records = []

for dbrow in cursor.fetchall():

row = RoleModel(dbrow[0],dbrow[1],dbrow[2],dbrow[3],dbrow[4],dbrow[5],dbrow[6],dbrow[7]) records.append(row)

cursor.close() conn.close() return records

@staticmethod

def get\_name\_id():

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT roleID, roleName FROM Role ORDER BY roleName" cursor.execute(sqlcmd1)

records = []

for dbrow in cursor.fetchall():

row = RoleModel(dbrow[0],dbrow[1]) records.append(row)

cursor.close() conn.close() return records

@staticmethod

def get\_by\_id(unique\_id):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT \* FROM Role WHERE roleID = ?" cursor.execute(sqlcmd1, unique\_id)

record = None

for dbrow in cursor.fetchall():

record = RoleModel(dbrow[0],dbrow[1],dbrow[2],dbrow[3],dbrow[4],dbrow[5],dbrow[6],dbrow[7]) cursor.close()

conn.close() return record

@staticmethod def insert(obj):

obj.roleID = str(uuid.uuid4())

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "INSERT INTO Role (roleName,canRole,canUsers,canAuthorizedBy,canDataOwner,canTransactionDetails,canVerifiedBy) VALUES(?,?,?,?,?,?,?)" cursor.execute(sqlcmd1, (obj.roleName,obj.canRole,obj.canUsers,obj.canAuthorizedBy,obj.canDataOwner,obj.canTransactionDetails,obj.canVerifiedBy)) cursor.close()

conn.close()

@staticmethod def update(obj):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "UPDATE Role SET roleName = ?,canRole = ?,canUsers = ?,canAuthorizedBy = ?,canDataOwner = ?,canTransactionDetails = ?,canVerifiedBy = ? WHERE roleID = ?" cursor.execute(sqlcmd1, (obj.roleName,obj.canRole,obj.canUsers,obj.canAuthorizedBy,obj.canDataOwner,obj.canTransactionDetails,obj.canVerifiedBy,obj.roleID)) cursor.close()

conn.close()

@staticmethod

def delete(unique\_id):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "DELETE FROM Role WHERE roleID = ?" cursor.execute(sqlcmd1, (unique\_id)) cursor.close()

conn.close()

**Verified Model.py**

[6:38 PM, 5/19/2023] Hulk: from Constants import connString import pyodbc

import datetime import uuid import time

class VerifiedByModel:

def init (self, verifiedByID = '',verifiedByName = ''): self.verifiedByID = verifiedByID

self.verifiedByName = verifiedByName

@staticmethod def get\_all():

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT \* FROM VerifiedBy ORDER BY verifiedByName" cursor.execute(sqlcmd1)

records = []

for dbrow in cursor.fetchall():

row = VerifiedByModel(dbrow[0],dbrow[1]) records.append(row)

cursor.close() conn.close() return records

@staticmethod

def get\_name\_id():

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT verifiedByID, verifiedByName FROM VerifiedBy ORDER BY verifiedByName" cursor.execute(sqlcmd1)

records = []

for dbrow in cursor.fetchall():

row = VerifiedByModel(dbrow[0],dbrow[1]) records.append(row)

cursor.close() conn.close() return records

@staticmethod

def get\_by\_id(unique\_id):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT \* FROM VerifiedBy WHERE verifiedByID = ?" cursor.execute(sqlcmd1, unique\_id)

record = None

for dbrow in cursor.fetchall():

record = VerifiedByModel(dbrow[0],dbrow[1]) cursor.close()

conn.close() return record

@staticmethod def insert(obj):

obj.verifiedByID = str(uuid.uuid4())

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "INSERT INTO VerifiedBy (verifiedByID,verifiedByName) VALUES(?,?)" cursor.execute(sqlcmd1, (obj.verifiedByID,obj.verifiedByName)) cursor.close()

conn.close()

@staticmethod def update(obj):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "UPDATE VerifiedBy SET verifiedByName = ? WHERE verifiedByID = ?" cursor.execute(sqlcmd1, (obj.verifiedByName,obj.verifiedByID)) cursor.close()

conn.close() @staticmethod

def delete(unique\_id):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "DELETE FROM VerifiedBy WHERE verifiedByID = ?" cursor.execute(sqlcmd1, (unique\_id))

cursor.close() conn.close(

**User Model.py**

from Constants import connString import pyodbc

import datetime import uuid import time

class UsersModel:

def init (self, userID = 0,userName = '',emailid = '',password = '',contactNo = '',isActive = False,roleID = 0,roleModel = None): self.userID = userID

self.userName = userName self.emailid = emailid self.password = password self.contactNo = contactNo self.isActive = isActive self.roleID = roleID self.roleModel = roleModel

@staticmethod def get\_all():

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT \* FROM Users ORDER BY userName" cursor.execute(sqlcmd1)

records = []

for dbrow in cursor.fetchall():

row =

UsersModel(dbrow[0],dbrow[1],dbrow[2],dbrow[3],dbrow[4],dbrow[5],dbrow[6]) records.append(row)

cursor.close() conn.close() return records

@staticmethod

def get\_name\_id():

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT userID, userName FROM Users ORDER BY userName" cursor.execute(sqlcmd1)

records = []

for dbrow in cursor.fetchall():

row = UsersModel(dbrow[0],dbrow[1]) records.append(row)

cursor.close() conn.close() return records

@staticmethod

def get\_by\_id(unique\_id):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT \* FROM Users WHERE userID = ?" cursor.execute(sqlcmd1, unique\_id)

record = None

for dbrow in cursor.fetchall():

record = UsersModel(dbrow[0],dbrow[1],dbrow[2],dbrow[3],dbrow[4],dbrow[5],dbrow[6]) cursor.close()

conn.close() return record

@staticmethod def insert(obj):

obj.userID = str(uuid.uuid4())

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "INSERT INTO Users (userName,emailid,password,contactNo,isActive,roleID) VALUES(?,?,?,?,?,?)" cursor.execute(sqlcmd1, (obj.userName,obj.emailid,obj.password,obj.contactNo,obj.isActive,obj.roleID)) cursor.close()

conn.close()

@staticmethod def update(obj):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "UPDATE Users SET userName = ?,emailid = ?,password = ?,contactNo = ?,isActive = ?,roleID = ? WHERE userID = ?" cursor.execute(sqlcmd1, (obj.userName,obj.emailid,obj.password,obj.contactNo,obj.isActive,obj.roleID,obj.userID)) cursor.close()

conn.close()

@staticmethod

def delete(unique\_id):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "DELETE FROM Users WHERE userID = ?" cursor.execute(sqlcmd1, (unique\_id)) cursor.close()

conn.close()

**Transaction Details Model.py**

from Constants import connString import pyodbc

import datetime import uuid import time

from Constants import contract\_address from web3 import Web3, HTTPProvider import json

import pprint

class TransactionDetailsModel:

def init (self, transactionDetailsID = '',transactionDetailsName = '',imageFile = '',createdDate = None,ipAddress = '',authorizedByID = '',verifiedByID = '',dataOwnerID = '',i self.transactionDetailsID = transactionDetailsID

self.transactionDetailsName = transactionDetailsName self.imageFile = imageFile

self.createdDate = createdDate self.ipAddress = ipAddress self.authorizedByID = authorizedByID self.verifiedByID = verifiedByID self.dataOwnerID = dataOwnerID

self.isBlockChainGenerated = isBlockChainGenerated self.hash = hash

self.prevHash = prevHash self.sequenceNumber = sequenceNumber self.authorizedByModel = authorizedByModel self.verifiedByModel = verifiedByModel self.dataOwnerModel = dataOwnerModel

@staticmethod def get\_all():

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT \* FROM TransactionDetails ORDER BY transactionDetailsName" cursor.execute(sqlcmd1)

records = []

for dbrow in cursor.fetchall():

row = TransactionDetailsModel(dbrow[0],dbrow[1],dbrow[2],dbrow[3],dbrow[4],dbrow[5],dbrow[6],dbrow[7],dbrow[8],dbrow[9],dbrow[10],dbrow[11]) records.append(row)

cursor.close() conn.close() return records

@staticmethod

def get\_name\_id():

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT transactionDetailsID, transactionDetailsName FROM TransactionDetails ORDER BY transactionDetailsName" cursor.execute(sqlcmd1)

records = []

for dbrow in cursor.fetchall():

row = TransactionDetailsModel(dbrow[0],dbrow[1]) records.append(row)

cursor.close() conn.close() return records

@staticmethod

def get\_by\_id(unique\_id):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "SELECT \* FROM TransactionDetails WHERE transactionDetailsID = ?" cursor.execute(sqlcmd1, unique\_id)

record = None

for dbrow in cursor.fetchall():

record = TransactionDetailsModel(dbrow[0],dbrow[1],dbrow[2],dbrow[3],dbrow[4],dbrow[5],dbrow[6],dbrow[7],dbrow[8],dbrow[9],dbrow[10],dbrow[11]) cursor.close()

conn.close() return record

@staticmethod def insert(obj):

obj.transactionDetailsID = str(uuid.uuid4())

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "INSERT INTO TransactionDetails (transactionDetailsID,transactionDetailsName,imageFile,createdDate,ipAddress,authorizedByID,verifiedByID,dataOwnerID,isBlockChainGener cursor.execute(sqlcmd1, (obj.transactionDetailsID,obj.transactionDetailsName,obj.imageFile,datetime.datetime.strptime(obj.createdDate.replace('T', ' '), '%Y-%m-%d %H:%M'),obj. cursor.close()

conn.close()

w3 = Web3(HTTPProvider('http://localhost:7545'))

compiled\_contract\_path = '../../../eTrading-Truffle/build/contracts/TransactionDetailsContract.json' deployed\_contract\_address = contract\_address

with open(compiled\_contract\_path) as file: contract\_json = json.load(file) contract\_abi = contract\_json["abi"]

contract = w3.eth.contract(address=deployed\_contract\_address, abi=contract\_abi) accounts = w3.eth.accounts

tx\_hash = contract.functions.perform\_transactions(obj.transactionDetailsID, obj.transactionDetailsName, obj.imageFile, obj.ipAddress, obj.authorizedByID, obj.dataOwnerID).tran receipt = w3.eth.wait\_for\_transaction\_receipt(tx\_hash)

@staticmethod def update(obj):

conn = pyodbc.connect(connString, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "UPDATE TransactionDetails SET transactionDetailsName = ?,imageFile = ?,createdDate = ?,ipAddress = ?,authorizedByID = ?,verifiedByID = ?,dataOwnerID = ?,isBlockChain cursor.execute(sqlcmd1, (obj.transactionDetailsName,obj.imageFile,datetime.datetime.strptime(obj.createdDate.replace('T', ' '), '%Y-%m-%d %H:%M'),obj.ipAddress,obj.authorized ring, autocommit=True) cursor = conn.cursor()

sqlcmd1 = "DELETE FROM TransactionDetails WHERE transactionDetailsID = ?" cursor.execute(sqlcmd1, (unique\_id))

cursor.close() conn.close()

**REFERENCES**

1. Dannen, Introducing Ethereum and Solidity: Foundations of Cryptocur- rency and Blockchain Programming for Beginners, 1st ed. Berkeley, CA, USA: Apress, 2017, doi: 10.1007/978-1-4842-2535-6.
2. Wood. (Aug. 6, 2017). The Yellow Paper: Ethereums Formal [Online]. Available: Specication github.io/yellowpaper/paper.pdf [Internet].
3. -S. Sung, A study on the protection plan of digital contents, J. Secur.
4. Acquier, T. Daudigeos, and J. Pinkse, Promises and para- the sharing economy: An organizing framework, Tech- doxes of 10.1016/j.techfore.2017.07.006.
5. Kogure, K. Kamakura, T. Shima, and T. Kubo, Blockchain /fstj/archives/vol53-5/paper09.pdf [20] F. Hawlitschek, B. Notheisen, and T. Teubner, The limits of trust-free systems: A literature review on blockchain technology and trust in the May 2018, doi: 10.1016/j.elerap.2018.03.005.
6. Lansiti and K. R. Lakhani, The truth about blockchain, in Harvard Business Review. Cambridge, MA, USA: Harvard Univ., Jan. 2017.
7. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, The eigentrust algorithm for reputation management in P2P networks, in Proc. 12th Int.

M. Haferkorn and J. M. Q. Diaz, Seasonality and interconnectivity within cryptocurrenciesAn analysis on the basis of Bitcoin, Litecoin and Name- coin, in Enterprise Applications and Services in the Finance Industry.

Introduction of Blockchain for Activation of Shared Economy, Survey, Seoul Digit. Found., Seoul, South Korea, 2016.

G. Wood. Ethereum: A Secure Decentralised Generalised Transac- tion Ledger (EIP-150). Accessed: Feb. 3,2018. [Online]. Available: [40] J. Filiba. (Dec. 22, 2017). Ethereum Breaks One Million Transactions in a U. Khan et al.: Blockchain Ethereum Technology-Enabled Digital Content: Development of Trading and Sharing Economy Data ZHANG YONG AN received the Ph.D. degree in management from Xian Jiaotong University, China. Since 2002, he has been a Professor with the School of Economics and Management Science, Beijing University of Technology. He has participated in and presided over the completion of the National Natural Science Foundation of China, four provincial and ministerial soft science projects, and won the second prize of outstanding achievements in provincial philosophy and social sciences. He is currently the President of the Key Project of the Ministry of Education, the Humanities and Social Sciences Project of the Beijing Municipal Education Commission, and a total of six cooperation projects of the enterprise. He has completed two books and two textbooks and published dozens of articles. Some articles were included in EI and others, and some articles were reprinted by the National Peoples Congress.

**13th INTERNATIONAL CONFERENCE ON SCIENCE AND INNOVATIVE ENGINEERING 2023 CERTIFICATES**







