**BUILDING A PRIVACY-PRESERVING BLOCKCHAIN-BASED BIDDING SYSTEM: A CRYPTO APPROACH**

*A Project Report Submitted in partial fulfillment of the requirements*

*for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

In

**COMPUTER SCIENCE AND ENGINEERING**

***Submitted By***

**N LAKSHMI POOJITHA 19221A0574**

**M S G KARTHIK GUPTHA 19221A0571**

**M DAVID PAUL 19221A0573**

**G GOPI RAJU 19221A0529**

**Under the Esteemed Guidance of**

**M JEEVANA JYOTHI**

**M.Tech**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BONAM VENKATA CHALAMAYYA ENGINEERING COLLEGE**

**(AUTONOMOUS)**

(Approved by A.I.C.T.E, New Delhi & Affiliated to J.N.T.U.K, Kakinada)

(Accredited by NBA & NAAC with “A‟ Grade)

**ODALAREVU-533210**

**2019 – 2023**

**BONAM VENKATA CHALAMAYYA ENGINEERING COLLEGE**

**(AUTONOMOUS)**

**ODALAREVU-533210**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



CERTIFICATE

This is to certify that the project work entitled **Building a Privacy-Preserving Blockchain-based Bidding System: A crypto Approach** is being submitted for the partial fulfilment of the requirements for the award of the degree of Bachelor Of Technology in **Computer Science and Engineering** at BVC Engineering College(A), Odalarevu is a bonafide work done by **N LAKSHMI POOJITHA (19221A0574), M S G KARTHIK GUPTHA(19221A0571), M DAVID PAUL (19221A0573), G GOPI RAJU (19221a0529)** under my guidance during the academic year **2019- 2023** and it has been found suitable for acceptance according to the requirement of the University. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree.

Project Guide Head of the Department

M JEEVANA JYOTHI Dr. GUNAMANI JENAM.E, Ph.D  Professor

**EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We profoundly grateful to express our deep sense of gratitude and respect towards our guide M JEEVANA JYOTHI, B V C Engineering College(A), Odalarevu, for his excellent guidance right from selection of the project and his valuable suggestions throughout the project work. His constant encouragement and support has been the cause of our success, in completing this thesis in the college.

We also grateful to the project Coordinator **Dr. B S N MURTHY,** **Professor** of CSE Department, B V C Engineering College(A), Odalarevu, for her excellent guidance right from selection of the project and her valuable suggestions throughout the project work.

We really thankful to **Dr. GUNAMANI JENA,** **Professor** and **Head of the Department of Computer Science and Engineering** for providing the laboratory facilities to the fullest extent as and when required and also for giving us the opportunity to carry out the project work in the college.

We also thankful to **Dr. Chandramouli V S A**, **Principal,** B V C Engineering College(A), Odalarevu, for his support during and till the completion of the project.

We thankful to all the **Teaching and Non-Teaching Staff** of Computer Science and Engineering Department, **Management**, B V C Engineering College(A), Odalarevu, and our **friends** for their direct and indirect help provided to us in completing the project.

**Project Associates………**

**N LAKSHMI POOJITHA(19221A0574)**

**M S G KARTHIK GUPTHa(19221A0571)**

**M DAVID PAUL(19221A0573)**

**G GOPI RAJU(19221A0529)**

**ABSTRACT**

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.

.

**OBJECTIVE:**

In a bidding system, the most important property is the privacy of the identities of the transaction both sides and the bidding content.

**MOTIVATION:**

Encryption scheme called deniable matchmaking encryption (DME). This new encryption scheme provides deniability not only for the message, but also for the identities.

**PROBLEM STATEMENT:**

Blockchain technique to create a secure and reliable data exchange platform across multiple data providers, where IoT data is encrypted and recorded in a distributed ledger.

Blockchain technique to create a secure and reliable data exchange platform across multiple data providers, where IoT data is encrypted and recorded in a distributed ledger

Block chain-based bidding systems, there are still some ‘‘out-of-rules’’ problems that can break the security and privacy of applications. try to trace an opposing party’s messages with the excuse of circumventing fake news. The government may force some service providers to hand over specified users’ secrets

**SCOPE OF WORK:**

Using DME on block chain-based bidding systems, we can preserve the privacy of the buyer and seller identities during the auction.

.

**Table of Contents**

[**1. INTRODUCTION 1**](#_Toc131091204)

[**2. LITERATURE SURVEY 10**](#_Toc131091205)

[**3. SYSTEM ANALYSIS 14**](#_Toc131091206)

[3.1 EXISTING SYSTEM 14](#_Toc131091207)

[3.1.1 Disadvantages of Existing system 14](#_Toc131091208)

[3.2 PROPOSED SYSTEM 14](#_Toc131091209)

[3.2.1 Advantages of Proposed System 14](#_Toc131091210)

[3.3 FUNCTIONAL REQUIREMENTS 15](#_Toc131091211)

[3.4 NON-FUNCTIONAL REQUIREMENTS 16](#_Toc131091212)

[3.5 FEASIBILITY STUDY 18](#_Toc131091216)

[3.6 SOCIAL FEASIBILITY 19](#_Toc131091217)

[3.6 REQUIREMENT SPECIFICATION 19](#_Toc131091218)

[3.6.1 Hardware Requirements 19](#_Toc131091219)

[3.6.2 Software Requirements 19](#_Toc131091220)

[**4. SYSTEM DESIGN 20**](#_Toc131091221)

[4.1 MODULE DESCRIPTION 20](#_Toc131091222)

[4.2 SYSTEM ARCHITECTURE: 20](#_Toc131091223)

[4.3 UML DIAGRAMS 20](#_Toc131091224)

[4.3.1 USE CASE DIAGRAM 22](#_Toc131091225)

[4.3.2 CLASS DIAGRAM 24](#_Toc131091226)

[4.3.3 SEQUENCE DIAGRAM 25](#_Toc131091227)

[4.4.4 COLLABORATION DIAGRAM 26](#_Toc131091228)

[**5. TECHNOLOGY DESCRIPTIONS 27**](#_Toc131091229)

[5.1 Python Introduction 27](#_Toc131091230)

[**6. SAMPLE CODE 42**](#_Toc131091239)

[**7. TESTING 47**](#_Toc131091240)

[7.1 INTRODUCTION 47](#_Toc131091241)

[**8. RESULTS 51**](#_Toc131091242)

[**9. CONCLUSION 62**](#_Toc131091243)

[**10.BIBLIOGRAPHY 63**](#_Toc131091244)

# 1. INTRODUCTION

Recently, blockchain 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 blockchain, 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 blockchain-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 blockchain-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 a result, the market share of blockchain-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, Ateniese 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.

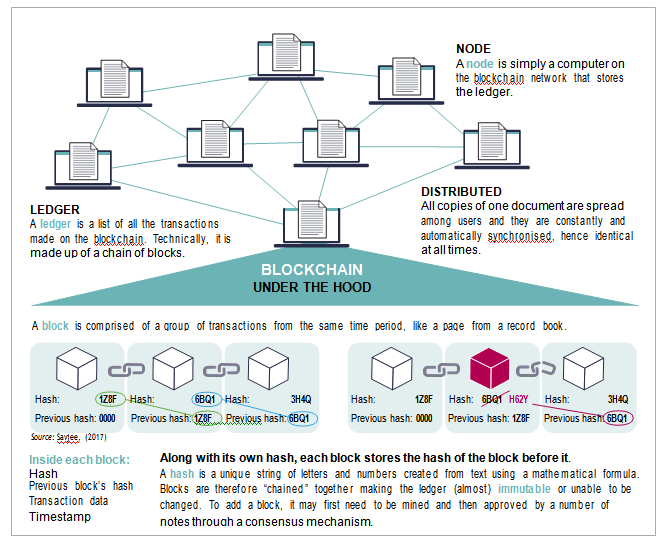
Although ME fits blockchain-based bidding systems, there are still some ‘‘out-of-rules’’ problems that can break the security and privacy of applications. For example, in some countries, the government may try to trace an opposing party’s messages with the excuse of circumventing fake news. The government may force some service providers to hand over specified users’ secrets. The bidding system is also a case that faces problems. If there is a coercer who forces the buyer, seller or key publisher to reveal their private secrets, the ME system crashes, and the bidding system becomes unsafe. Therefore, we need other tools to enhance the ME system.

To address the problem of coercing users into revealing their information, we introduce the idea of deniable encryption. Deniable encryption is now widely used when a user wants to prevent a message from being coerced. To deal with a malicious coercer, the deniable encryption technique can make the user convince the coercer with a fake message and keep the true message secret. This is a better way to protect user privacy than simply denying coercion requests.

Regarding ME, not only the message, but also the identities should be deniable. That is, given an ME ciphertext, we want to make the coercer believe the fake message, the fake sender and the fake receiver, so that the real sender, the real receiver and the real message are protected. For this reason, in this paper, we propose an encryption scheme called deniable matchmaking encryption (DME). The scheme is enhanced from identity-based matchmaking encryption (IB-ME), which was proposed by Giuseppe Ateniese et al. in 2019 [6]. The concept is shown in Figure 2. We use the chameleon hash function [7] as a verification tool in the decryption phase. The chameleon hash is a cryptographic hash function which can easily find a collision with a secret trapdoor. The trapdoor collision property of chameleon hash makes it possible to forge a fake message that has the same hash value as the real message; therefore, both the true message and the fake message can pass the verification check in the decryption phase. When encryption is performed, the true message is protected with the true sender and the true receiver by IB-ME, while the fake message is protected by the fake entities. When being coerced, the fake message and the fake identities are provided to the coercer.

**WHAT IS BLOCK CHAIN?**

A block chain is a shared ledger of transactions between parties in a network, not controlled by a single central authority. You can think of a ledger like a record book: it records and stores all transactions between users in chronological order. Instead of one authority controlling this ledger (like a bank), an identical copy of the ledger is held by all users on the network, called nodes.

****

**DIFFERENT TYPES OF BLOCK CHAIN:**

**Before going further, it is important to note that not every blockchain is made the same.** While there are a number of variable features, two of the most important are the **“openness” of the platform** (public or private) and the **level of permissions required to add information** to the blockchain (permissioned or permissionless). **Public** blockchains (like Bitcoin) are open for anyone to read and view, while **private** blockchains can only be viewed by a chosen group of people. Similarly, **permissioned** blockchains permit just a select group of users to write (i.e. generate transactions for the ledger to record) and commit (i.e. verify new blocks for addition to the chain). In contrast, **permissionless** blockchains allow anyone to contribute and add data to the ledger.

**THE HARDWARE LAYER**

The first layer of the blockchain consists of hardware, like network connections, the computers within the network and data servers. The data stored inside a blockchain is hosted by data servers, and computers on the blockchain network can share this data with each other. This leads to the creation of a P2P network where information is validated by individual nodes (or computers) on the network.

**THE DATA LAYER**

The second layer of this house is the data layer, where information stored on the network is managed. This layer is made up of blocks of information with each block connected to the previous one. The only block that is not linked back to another is the genesis block (the first block in the network).

Each transaction written on these blocks is protected through a private key and a public key. A private key is a digital signature known only by the owner for authorizing a transaction; a public key is used to verify who has signed for the transaction. To put it simply, if someone sends you some crypto, they will need to know your public key; for you to receive the crypto, you have to use your private key to verify the transaction and prove your ownership to your block chain wallet.

**THE NETWORK LAYER**

This layer facilitates communication between the different nodes within the block chain network. It is also in this layer that blocks are created and added to the block chain. As a result, this layer is also referred to as the propagation layer.

**THE CONSENSUS LAYER**

This layer ensures that the rules of the network are effectively enforced to preserve uniformity within the network. One node cannot simply add a transaction to the block chain; to do so, all nodes within the network need to agree on it. This level of verification lowers the risk of fraudulent transactions being added to the block chain.

**THE APPLICATION LAYER**

This layer facilitates the use of the block chain for a wide variety of purposes. It is made up of smart contracts and decentralized applications (DApps). This layer acts as the front end of the block chain and is essentially what a user would typically encounter when operating within a block chain network.

**BLOCK CHAIN PROTOCOL:**

**Layer 0**

Layer zero is where the [network hardware](https://medium.com/@nick.5montana/blockchain-layers-l0-l1-l2-l3-in-a-diagram-569162398db) (the internet and connected devices) coexist. It is the [foundation](https://insights.tokenmetrics.com/layer-0-blockchain-protocols/#:~:text=Layer%200%20protocols%20are%20the,built%20on%20top%20of%20them.) on which the rest of the layers are built.

**Layer 1**

The first layer of the protocol consists of the different blockchains (like Bitcoin, Ethereum and Binance Smart Chain) that can process transactions. This layer of the protocol [ensures](https://www.linkedin.com/pulse/crypto-101-4-layers-blockchain-protocol-david-mcneal/?trk=pulse-article_more-articles_related-content-card) the security of the blockchain with different [consensus mechanisms](https://www.jumpstartmag.com/the-proof-systems-of-cryptos/), like proof of work and proof of stake being a part of this layer.

**Layer 2**

This layer is also known as the execution layer. As a blockchain grows, the number of transactions being performed on it increases. To [support](https://www.cnbctv18.com/cryptocurrency/what-is-a-layer-2-blockchain-13267932.htm) the increased number of transactions, we need scalability (ability to [handle](https://www.wipro.com/blogs/hitarshi-buch/improving-performance-and-scalability-of-blockchain-networks/#:~:text=Scalability%20of%20blockchain%20networks%20is,of%20nodes%20in%20the%20network.)the increased load) Layer 2 solutions. Often, off-chain (or third party) solutions are [implemented](https://coinmarketcap.com/alexandria/article/what-are-application-layer-protocols) to address any issues within the first layer of the protocol. These solutions don’t hamper the features of the first layer but rather add to them.

**Layer 3**

This is the application layer of the blockchain protocol. It is made up of the different blockchain-based applications (Dapps and [decentralized autonomous organizations](https://www.jumpstartmag.com/are-daos-the-future-of-work-and-startups/) [DAOs]) that we see on the market today, such as Decentraland and [Crypto Kitties](https://www.jumpstartmag.com/animoca-brands-signs-exclusive-licence-and-distribution-agreement-with-axiom-zen-to-publish-cryptokitties-in-china/).

**BLOCK CHAIN KEY CHARACTERISTICS:**

**DISTRIBUTED:**

One of the core aspects of a blockchain is that it is a distributed ledger, meaning that the database is maintained and held by all nodes in the network. No central authority holds or updates the ledger, rather each node independently constructs its own record by processing every block (group of transactions), deciding if it is valid, then voting via the consensus mechanism on their conclusions. Once a change in the record is agreed, each node updates its own ledger. In contrast, traditional databases are stored and maintained centrally, which can make them high-value targets for hackers and criminals.

**IMMUTABLE:**

In general, once a transaction is added to a block chain ledger, it cannot be undone. This immutability is one of the principal aspects that contribute to the trustworthiness of block chain transactions. A block chain’s immutability is secured through its use of cryptography (see below for an explanation of hashing). In a traditional, centralized database, an authorized user can connect to the server to add or modify the data without the approval or detection of other users. Because all the data is held in one place, if the security of the server or the authority that runs the server is compromised, data can be modified or permanently deleted. This may sometimes be irreversible and occur without anyone else realizing it.

**AGREED BY CONSENSUS**:

No block can be added to the ledger without approval from specified nodes in the network. Rules regarding how this consent is collected are called consensus mechanisms. Consensus protocols are crucial in ensuring that every block is valid and that all participants agree and maintain the same version of the ledger. They heavily affect the incentives for nodes to act honestly and are therefore the most important variables when designing a block chain.

**Block chain is an emerging technology with many advantages in an increasingly digital world:**

**Highly Secure**

It uses a digital signature feature to conduct fraud-free transactions making it impossible to corrupt or change the data of an individual by the other users without a specific digital signature.

**Decentralized System**

Conventionally, you need the approval of regulatory authorities like a government or bank for transactions; however, with Block chain, transactions are done with the mutual consensus of users resulting in smoother, safer, and faster transactions.

**Automation Capability**

It is programmable and can generate systematic actions, events, and payments automatically when the criteria of the trigger are met.

**How Does Block chain Works?**

In recent years, you may have noticed many businesses around the world integrating Block chain technology. But how exactly does Block chain technology work? Is this a significant change or a simple addition? The advancements of Block chain are still young and have the potential to be revolutionary in the future; so, let’s begin demystifying this technology.

**Block chain is a combination of three leading technologies**:

Cryptographic keys

A peer-to-peer network containing a shared ledger

A means of computing, to store the transactions and records of the network

[Cryptography](https://www.simplilearn.com/understanding-cryptography-article) keys consist of two keys – Private key and Public key. These keys help in performing successful transactions between two parties. Each individual has these two keys, which they use to produce a secure digital identity reference. This secured identity is the most important aspect of Block chain technology. In the world of crypto currency, this identity is referred to as ‘digital signature’ and is used for authorizing and controlling transactions.

The digital signature is merged with the peer-to-peer network; a large number of individuals who act as authorities use the digital signature in order to reach a consensus on transactions, among other issues. When they authorize a deal, it is certified by a mathematical verification, which results in a successful secured transaction between the two network-connected parties. So to sum it up, Block chain users employ cryptography keys to perform different types of digital interactions over the peer-to-peer network.

**Advantages and Disadvantages of Block chain:**

**Advantages**

One major advantage of block chains is the level of security it can provide, and this also means that block chains can protect and secure sensitive data from online transactions. For anyone looking for speedy and convenient transactions, block chain technology offers this as well. In fact, it only takes a few minutes, whereas other transaction methods can take several days to complete. There is also no third-party interference from financial institutions or government organizations, which many users look at as an advantage.

**Disadvantages**

Block chain and cryptography involves the use of public and private keys, and reportedly, there have been problems with private keys. If a user loses their private key, they face numerous challenges, making this one disadvantage of block chains. Another disadvantage is the scalability restrictions, as the number of transactions per node is limited. Because of this, it can take several hours to finish multiple transactions and other tasks. It can also be difficult to change or add information after it is recorded, which is another significant disadvantage of block chain

# 2. LITERATURE SURVEY

**2.1] Identity-based cryptosystems and signature schemes**

**(Shamir)**

In this paper we introduce a novel type of cryptographic scheme, which enables any pair of users to communicate securely and to verify each other’s signatures without exchanging private or public keys, without keeping key directories, and without using the services of a third party. The scheme assumes the existence of trusted key generation centers, whose sole purpose is to give each user a personalized smart card when he first joins the network. The information embedded in this card enables the user to sign and encrypt the messages he sends and to decrypt and verify the messages he receives in a totally independent way, regardless of the identity of the other party. Previously issued cards do not have to be updated when new users join the network, and the various centers do not have to coordinate their activities or even to keep a user list. The centers can be closed after all the cards are issued, and the network can continue to function in a completely decentralized way for an indefinite period.

**2.2] A realization scheme for the identity-based cryptosystem**

**(H. Tanaka)**

At the Crypto’84, Shamir has presented a new concept of the identity-based cryptosystem, but no idea is presented on the realization scheme. In this paper a new realization scheme of the modified identity-based cryptosystem has been proposed. The basic idea of the scheme is based on the discrete logarithm problem and the difficulty of factoring a large integer composed of two large primes. The scheme seems to be very secure if all members of the system keep their secret keys safe, but if a constant number of users conspire, the center secret will be disclosed, Then it has a close relation to the well-known “threshold scheme”. To cope with the conspiracy, the basic system is extended to get a new scheme of which “threshold” becomes higher. Detail considerations on the scheme are also given.

**2.3] Bi-deniable public-key encryption**

**O’Neill, C. Peikert, and B. Waters)**

In 1997, Canetti et al. (CRYPTO 1997) put forward the intriguing notion of *deniable encryption*, which (informally) allows a sender and/or receiver, having already performed some encrypted communication, to produce ‘fake’ (but legitimate-looking) random coins that open the ciphertext to another message. Deniability is a powerful notion for both practice and theory: apart from its inherent utility for resisting coercion, a deniable scheme is also noncommitting (a useful property in constructing adaptively secure protocols) and secure under selective-opening attacks on whichever parties can equivocate. To date, however, known constructions have achieved only limited forms of deniability, requiring at least one party to withhold its randomness, and in some cases using an interactive protocol or external parties.

In this work we construct *bi-deniable* public-key cryptosystems, in which both the sender and receiver can simultaneously equivocate; we stress that the schemes are noninteractive and involve no third parties. One of our systems is based generically on “simulatable encryption” as defined by Damgård and Nielsen (CRYPTO 2000), while the other is lattice-based and builds upon the results of Gentry, Peikert and Vaikuntanathan (STOC 2008) with techniques that may be of independent interest. Both schemes work in the so-called “multi-distributional” model, in which the parties run alternative key-generation and encryption algorithms for equivocable communication, but claim under coercion to have run the prescribed algorithms. Although multi-distributional deniability has not attracted much attention, we argue that it is meaningful and useful because it provides credible coercion resistance in certain settings, and suffices for all of the related properties mentioned above.

**2.4] Blockchain based smart contract for bidding system**

**(Y.-H. Chen, S.-H. Chen, and I.-C. Lin)**

Because of the popularity of the Internet, the integration services have gradually changed people daily life, such as e-commerce activities on transactions, transportation and so on. The E-auction, one of the popular e-commerce activities, allows bidders to directly bid the products over the Internet. As for sealed bid, the extra transaction cost is required for the intermediaries because the third-party is the important role between the buyers and the sellers help to trade both during the auction. In addition, it never guarantees whether the third-party is trust. To resolve the problems, the blockchain technology with low transaction cost is used to develop the smart contract of public bid and sealed bid. The smart contract, proposed in 1990 and implements via Ethereum platform, can ensure the bill secure, private, non-reputability and inalterability owing to all the transactions are recorded in the same but decentralized ledgers. The smart contract is composed of the address of Auctioneer, the start auction time, deadline, the address of current winner, the current highest price. In the experiments, the accounts are created through Ethereum wallet. In miner stage, the Miner Gate is used in miner stage for obtaining money to pay the transaction fee. At recorder stage, the nodes of blockchain are synchronized to generate smart contract.

**2.5] Blockchain-based smart contract for E-bidding system**

**(P. Manimaran and R. Dhanalakshmi)**

Electronic bidding systems have become widespread since the advent of the internet and mobile phones. In the Electronic bidding systems, the seller will sell an item and many buyers will bid for that item and the highest bidder will get the item. One of the main issue with this E-Bidding system is the introduction of third-party mainly a company or set of companies which will develop and host either the website or smartphone application. The Buyers and the Sellers have to trust this company because all the bidding process will be handled by the company. The company can manipulate the bidding process if it wants. So, to avoid the trust issues blockchain based electronic bidding system is introduced in this paper. In this model, there is no need for third party. Smart Contract will handle all the bidding transactions. Since blockchains are known for its integrity this system makes sure that the integrity of the bidding process is preserved.

# 3. SYSTEM ANALYSIS

## 3.1 EXISTING SYSTEM

Block chain has inbuilt support for data unchangeable and encryption which is making Block chain more secure and these advantages forcing application developers to migrate towards Block chain. Online Bidding is one of the applications where bidders will bid for particular tenders and some malicious internal employees will change bidding or tender details to make their favorable person to win bidding and this manipulation will cost huge loss to the bidding system.

### 3.1.1 Disadvantages of Existing system

* Privacy issues.
* DME is not implemented

## 3.2 PROPOSED SYSTEM

Block chain-based bidding systems are becoming increasingly popular nowadays. Due to the properties of Block chain, 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 Block chain, and the system loses privacy. To solve this problem, in this paper, in propose paper 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 Block chain-based bidding system.

### 3.2.1 Advantages of Proposed System

* Higher accuracy
* Chameleon hash function will be implemented

## 3.3 FUNCTIONAL REQUIREMENTS

In [software engineering](https://en.wikipedia.org/wiki/Software_engineering) and [systems engineering](https://en.wikipedia.org/wiki/Systems_engineering), a functional requirement defines a function of a [system](https://en.wikipedia.org/wiki/System) or its component, where a function is described as a specification of behavior between outputs and inputs.[[1]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-FultonAirborne17-1)

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish.[[2]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-2) Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in [use cases](https://en.wikipedia.org/wiki/Use_case). Functional requirements are supported by [non-functional requirements](https://en.wikipedia.org/wiki/Non-functional_requirement) (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>," while non-functional requirements take the form "system shall be <requirement>." The plan for implementing functional requirements is detailed in the system design, whereas *non-functional* requirements are detailed in the system architecture.[[4]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-AdamsNon15-4)[[5]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-J%C3%B6nssonImpact06-5)

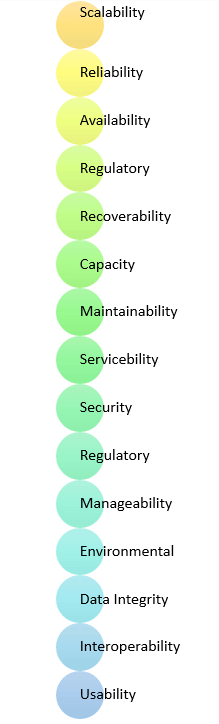
As defined in [requirements engineering](https://en.wikipedia.org/wiki/Requirements_analysis), functional requirements specify particular results of a system. This should be contrasted with non-functional requirements, which specify overall characteristics such as cost and [reliability](https://en.wikipedia.org/wiki/Reliability_engineering). Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.[[4]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-AdamsNon15-4)

In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements collection and change, broadly speaking, is: user/[stakeholder](https://en.wikipedia.org/wiki/Project_stakeholder) request → analyze → use case → incorporate. Stakeholders make a request; systems engineers attempt to discuss, observe, and understand the aspects of the requirement; use cases, entity relationship diagrams, and other models are built to validate the requirement; and, if documented and approved, the requirement is implemented/incorporated.[[6]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-MITRESys14-6) Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

## ****3.4 NON-FUNCTIONAL REQUIREMENTS****

(NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are> 10000. Description of non-functional requirements is just as critical as a functional requirement.



**Advantages of Non-Functional Requirement**

Benefits/pros of Non-functional testing are:

* The nonfunctional requirements ensure the software system follow legal and compliance rules.
* They ensure the reliability, availability, and performance of the software system
* They ensure good user experience and ease of operating the software.
* They help in formulating security policy of the software system.

**Disadvantages of Non-functional requirement**

Cons/drawbacks of non-function requirement are:

* None functional requirement may affect the various high-level software subsystem
* They require special consideration during the software architecture/high-level design phase which increases costs.
* Their implementation does not usually map to the specific software sub-system,
* It is tough to modify non-functional once you pass the architecture phase.

**\*KEY LEARNING**

* A non-functional requirement defines the performance attribute of a software system.
* Types of Non-functional requirement are Scalability Capacity, Availability, Reliability, Recoverability, Data Integrity, etc.
* Example of Non-Functional Requirement is Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.
* Functional Requirement is a verb while Non-Functional Requirement is an attribute
* The advantage of Non-functional requirement is that it helps you to ensure good user experience and ease of operating the software
* The biggest disadvantage of Non-functional requirement is that it may affect the various high-level software subsystems.

## 3.5 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

## 3.6 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## 3.6 REQUIREMENT SPECIFICATION

### 3.6.1 Hardware Requirements

* System : i3/i5.
* Hard Disk : 5000 GB.
* Ram : 4 GB.

### 3.6.2 Software Requirements

* Operating system : Windows 10, Windows 11.
* Coding Language : Python

# 4. SYSTEM DESIGN

## 4.1 MODULE DESCRIPTION

1. **Bidding Officer Login:**

bidding officer will login to application using username and password as ‘admin’ and ‘admin’ and then create tender with initial bidding amount. This user will evaluate bidding to select bidder with LOWEST amount and then will run ‘Winner Selection’ module to select winner.

1. **Bidder Login:**

Bidder can sign up with the application and then login and then view all tenders and bidding and can bid for desired tender. Tender status can be view by clicking on ‘View Status’ link which will display all tender, bidding and winner details.

## **4.2 SYSTEM ARCHITECTURE**:

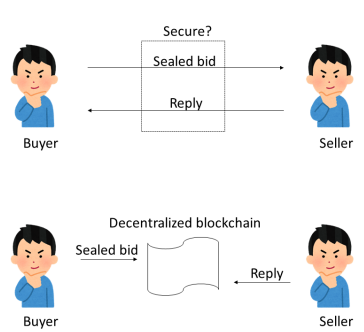


FIG1: Architecture

## 4.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

## 4.3.1 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted





FIGURE 4.2.1 USECASE DIAGRAM

## 4.3.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



FIGURE 4.2.2 CLASS DIAGRAM

## 4.3.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

****

FIGURE 4.2.3 SEQUENCE DIAGRAM

## 4.4.4 COLLABORATION DIAGRAM

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.



4.4.4 COLLABORATION DIAGRAM

# 5. TECHNOLOGY DESCRIPTIONS

## 5.1 Python Introduction

Python is a general purpose, dynamic, high level and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is easy to learn yet powerful and versatile scripting language which makes it attractive for Application Development.

Python's syntax and dynamic typing with its interpreted nature, makes it an ideal language for scripting and rapid application development.

Python supports multiple programming pattern, including object oriented, imperative and functional or procedural programming styles.

Python is not intended to work on special area such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD etc.

We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable.

Python makes the development and debugging fast because there is no compilation step included in python development and edit-test-debug cycle is very fast.

# Python History

* Python laid its foundation in the late 1980s.
* The implementation of Python was started in the December 1989 by Guido VanRossum at CWI in Netherland.
* In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
* In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
* \*Python 2.0 added new features like: list comprehensions, garbage collection system.
* On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
* ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
* Python is influenced by following programming languages:
  + ABC language.
  + Modula-3

# Python Features

Python provides lots of features that are listed below.

#### 1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high-level programming language.

#### 2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

#### 3) Interpreted Language

Python is an interpreted language i.e., interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

#### 4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

#### 5) Free and Open Source

Python language is freely available at address. The source-code is also available. Therefore, it is open source.

#### 6) Object-Oriented Language

Python supports object-oriented language and concepts of classes and objects come into existence.

#### 7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

#### 8) Large Standard Library

Python has a large and broad library and provides rich set of module and functions for rapid application development.

#### 9) GUI Programming Support

Graphical user interfaces can be developed using Python.

#### 10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

# Python Applications

Python is known for its general-purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development.

Here, we are specifying applications areas where python can be applied.

#### 1) Web Applications

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautiful Soup, Feed parser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develop web-based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

#### 2) Desktop GUI Applications

Python provides Tk GUI library to develop user interface in python-based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

#### 3) Software Development

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

#### 4) Scientific and Numeric

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

#### 5) Business Applications

Python is used to build Business applications like ERP and e-commerce systems. Tryton is a high-level application platform.

#### 6) Console Based Application

We can use Python to develop console-based applications. For example: IPython.

#### 7) Audio or Video based Applications

Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.

#### 8) 3D CAD Applications

To create CAD application Fandango is a real application which provides full features of CAD.

#### 9) Enterprise Applications

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

#### 10) Applications for Images

Using Python several applications can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

There are several such applications which can be developed using Python

# How to Install Python (Environment Set-up)

In this section of the tutorial, we will discuss the installation of python on various operating systems.

### Why Python

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

### Good to know

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, PyCharm, NetBeans or Eclipse which are particularly useful when managing larger collections of Python files.

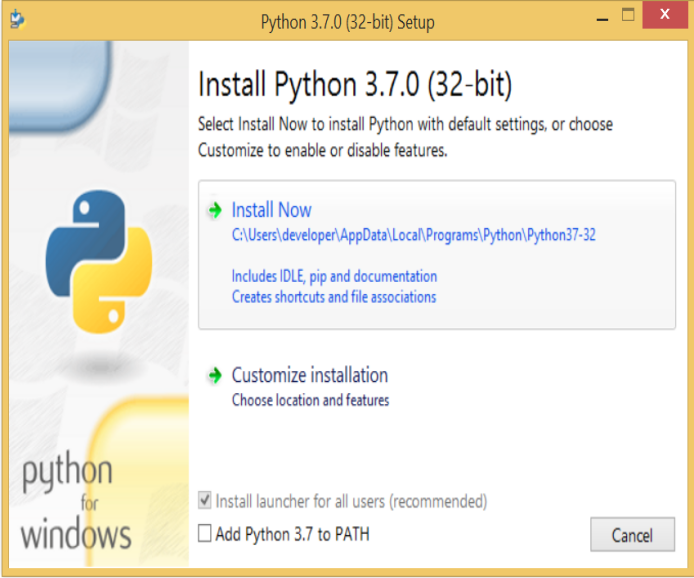
### Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

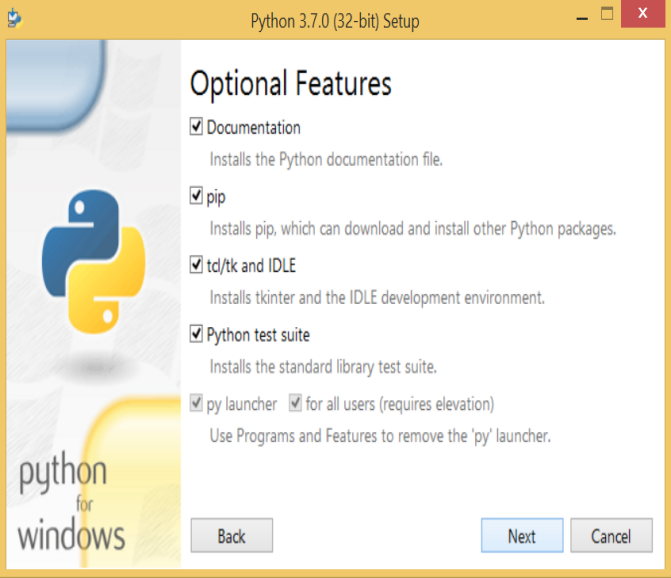
### Installation on Windows

Visit the link <https://www.python.org/downloads/> to download the latest release of Python. In this process, we will install Python 3.6.7 on our Windows operating system.

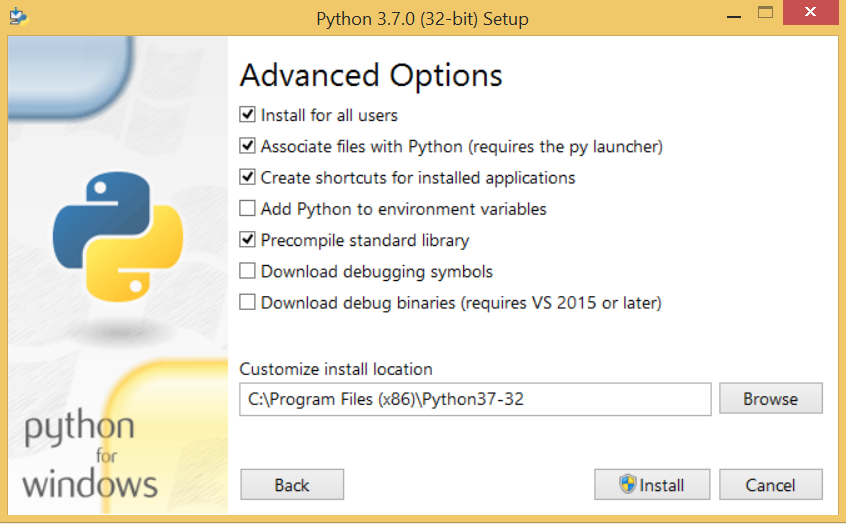
Double-click the executable file which is downloaded; the following window will open. Select Customize installation and proceed.



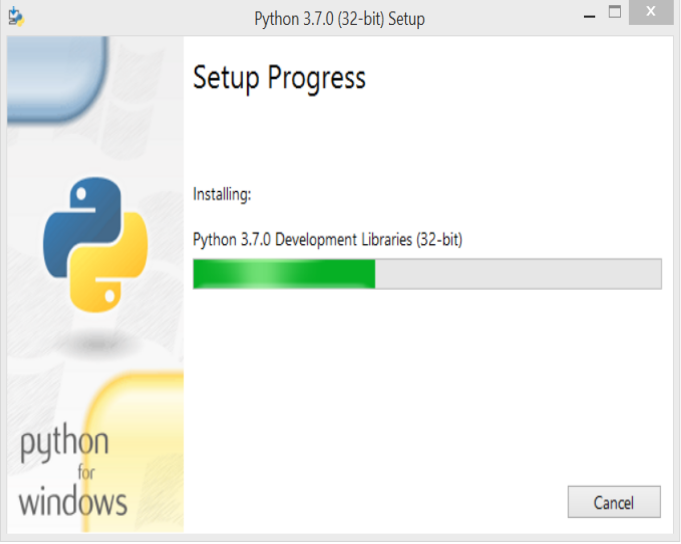
The following window shows all the optional features. All the features need to be installed and are checked by default; we need to click next to continue.



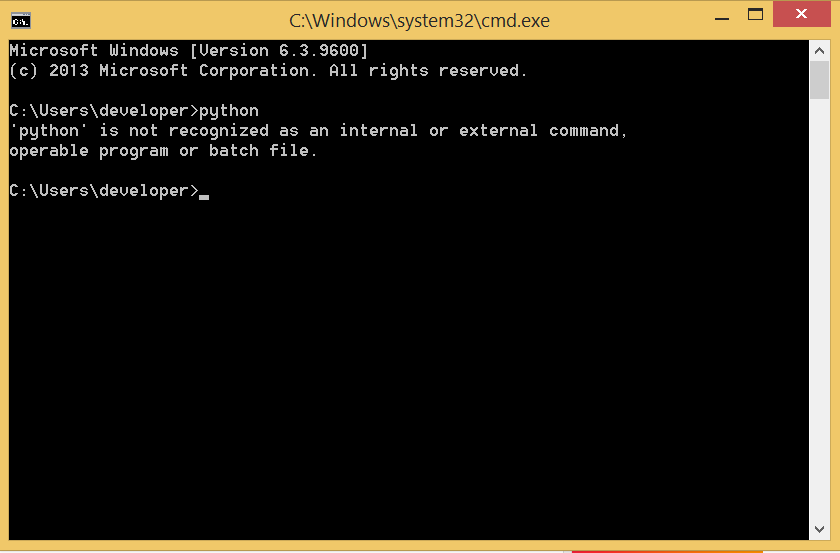
The following window shows a list of advanced options. Check all the options which you want to install and click next. Here, we must notice that the first check-box (install for all users) must be checked.



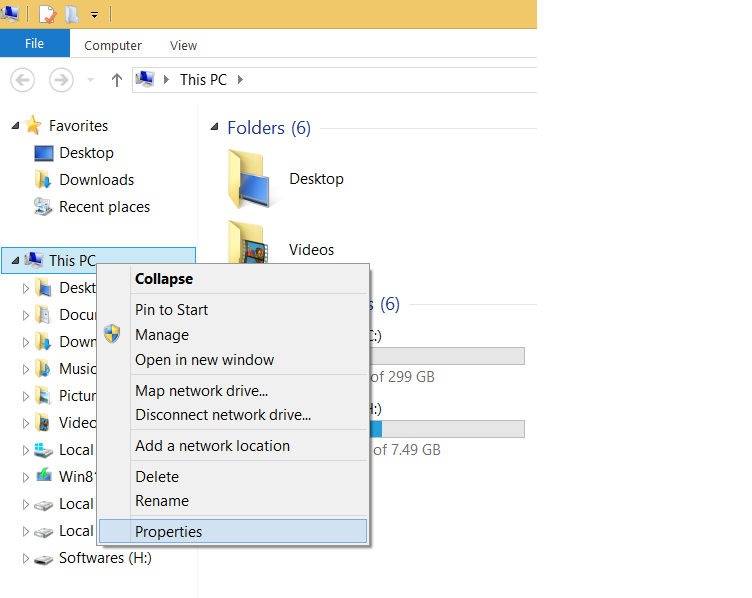
Now, we are ready to install python-3.6.6. Lets install it.

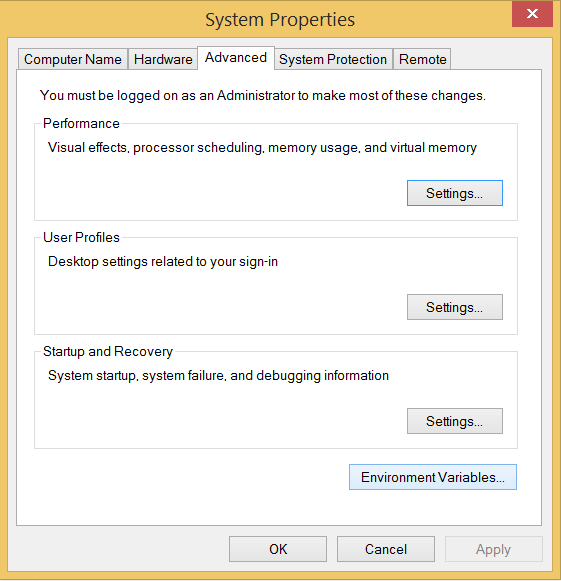


Now, try to run python on the command prompt. Type the command python in case of python2 or python3 in case of python3. It will show an error as given in the below image. It is because we haven't set the path.

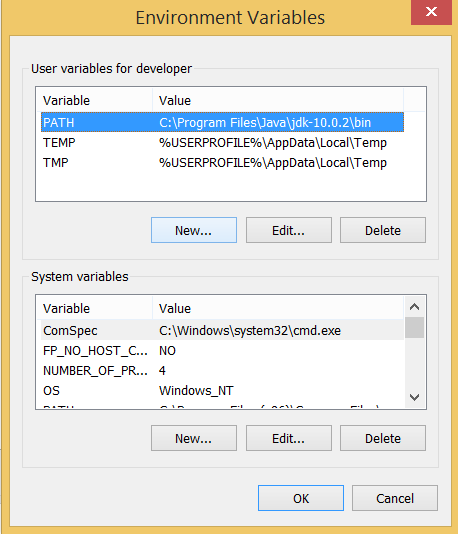


To set the path of python, we need to the right click on "my computer" and go to Properties → Advanced → Environment Variables.

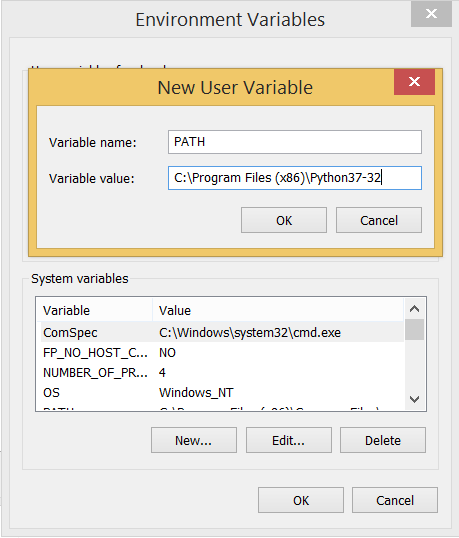




Add the new path variable in the user variable section.



Type PATH as the variable name and set the path to the installation directory of the python shown in the below image.



Now, the path is set, we are ready to run python on our local system. Restart CMD, and type python again. It will open the python interpreter shell where we can execute the python statements.

**Virtual Environments and Packages**

**Introduction**

Python applications will often use packages and modules that don’t come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library’s interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A’s environment.

**Creating Virtual Environments**

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the venv module as a script with the directory path:

python3 -m venv tutorial-env

This will create the tutorial-env directory if it doesn’t exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you’ve created a virtual environment, you may activate it.

On Windows, run:

tutorial-env\Scripts\activate.bat

On Unix or MacOS, run:

source tutorial-env/bin/activate

(This script is written for the bash shell. If you use the csh or fish shells, there are alternate activate.csh and activate. fish scripts you should use instead.)

Activating the virtual environment will change your shell’s prompt to show what virtual environment you’re using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

$ source ~/envs/tutorial-env/bin/activate

(Tutorial-env) $ python

Python 3.5.1 (default, May 6 2016, 10:59:36)

...

>>> import sys

>>>sys.path

['', '/usr/local/lib/python35.zip', ...,

'~/envs/tutorial-env/lib/python3.5/site-packages']

>>>

**Managing Packages with pip**

You can install, upgrade, and remove packages using a program called pip. By default pip will install packages from the Python Package Index, <https://pypi.org>. You can browse the Python Package Index by going to it in your web browser, or you can use pip’s limited search feature:

(tutorial-env) $ pip search astronomy

sky field - Elegant astronomy for Python

Gary - Galactic astronomy and gravitational dynamics.

novas - The United States Naval Observatory NOVAS astronomy library

astroobs - Provides astronomy ephemeris to plan telescope observations

PyAstronomy - A collection of astronomy related tools for Python.

...

pip has a number of subcommands: “search”, “install”, “uninstall”, “freeze”, etc. (Consult the Installing Python Modules guide for complete documentation for pip.)

You can install the latest version of a package by specifying a package’s name:

(tutorial-env) $ pip install novas

Collecting novas

Downloading novas-3.1.1.3.tar.gz (136kB)

Installing collected packages: novas

Running setup.py install for novas

Successfully installed novas-3.1.1.3

You can also install a specific version of a package by giving the package name followed by == and the version number:

(tutorial-env) $ pip install requests==2.6.0

Collecting requests==2.6.0

Using cached requests-2.6.0-py2.py3-none-any.whl

Installing collected packages: requests

Successfully installed requests-2.6.0

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install --upgrade to upgrade the package to the latest version:

(tutorial-env) $ pip install --upgrade requests

Collecting requests

Installing collected packages: requests

Found existing installation: requests 2.6.0

Uninstalling requests-2.6.0:

Successfully uninstalled requests-2.6.0

Successfully installed requests-2.7.0

pip uninstall followed by one or more package names will remove the packages from the virtual environment.

pip show will display information about a particular package:

(tutorial-env) $ pip show requests

---

Metadata-Version: 2.0

Name: requests

Version: 2.7.0

Summary: Python HTTP for Humans.

Home-page: http://python-requests.org

Author: Kenneth Reitz

Author-email: me@kennethreitz.com

License: Apache 2.0

Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages

Requires:

pip list will display all of the packages installed in the virtual environment:

(tutorial-env) $ pip list

novas (3.1.1.3)

numpy (1.9.2)

pip (7.0.3)

requests (2.7.0)

setuptools (16.0)

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

(tutorial-env) $ pip freeze > requirements.txt

(tutorial-env) $ cat requirements.txt

novas==3.1.1.3

numpy==1.9.2

requests==2.7.0

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

(tutorial-env) $ pip install -r requirements.txt

Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))

...

Collecting numpy==1.9.2 (from -r requirements.txt (line 2))

...

Collecting requests==2.7.0 (from -r requirements.txt (line 3))

...

Installing collected packages: novas, numpy, requests

Running setup.py install for novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you’ve written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.

# 6. SAMPLE CODE

from hashlib import sha256

import json

import time

import pickle

from datetime import datetime

import random

from ecies.utils import generate\_eth\_key, generate\_key

from ecies import encrypt, decrypt

import base64

import os

class Block:

def \_\_init\_\_(self, index, transactions, timestamp, previous\_hash):

self.index = index

self.transactions = transactions

self.timestamp = timestamp

self.previous\_hash = previous\_hash

self.nonce = 0

def compute\_hash(self):

block\_string = json.dumps(self.\_\_dict\_\_, sort\_keys=True)

return sha256(block\_string.encode()).hexdigest()

class Blockchain:

# difficulty of our PoW algorithm

difficulty = 2 #using difficulty 2 computation

def \_\_init\_\_(self):

self.unconfirmed\_transactions = []

self.chain = []

self.create\_genesis\_block()

self.peer = []

self.translist = []

def create\_genesis\_block(self): #create genesis block

genesis\_block = Block(0, [], time.time(), "0")

genesis\_block.hash = genesis\_block.compute\_hash()

self.chain.append(genesis\_block)

@property

def last\_block(self):

return self.chain[-1]

def add\_block(self, block, proof): #adding data to block by computing new and previous hashes

previous\_hash = self.last\_block.hash

if previous\_hash != block.previous\_hash:

return False

if not self.is\_valid\_proof(block, proof):

return False

block.hash = proof

#print("main "+str(block.hash))

self.chain.append(block)

return True

def is\_valid\_proof(self, block, block\_hash): #proof of work

return (block\_hash.startswith('0' \* Blockchain.difficulty) and block\_hash == block.compute\_hash())

def proof\_of\_work(self, block): #proof of work

block.nonce = 0

computed\_hash = block.compute\_hash()

while not computed\_hash.startswith('0' \* Blockchain.difficulty):

block.nonce += 1

computed\_hash = block.compute\_hash()

return computed\_hash

def add\_new\_transaction(self, transaction):

self.unconfirmed\_transactions.append(transaction)

def addPeer(self, peer\_details):

self.peer.append(peer\_details)

def addTransaction(self,trans\_details): #add transaction

self.translist.append(trans\_details)

def mine(self):#mine transaction

if not self.unconfirmed\_transactions:

return False

last\_block = self.last\_block

new\_block = Block(index=last\_block.index + 1,

transactions=self.unconfirmed\_transactions,

timestamp=time.time(),

previous\_hash=last\_block.hash)

proof = self.proof\_of\_work(new\_block)

self.add\_block(new\_block, proof)

self.unconfirmed\_transactions = []

return new\_block.index

def save\_object(self,obj, filename):

with open(filename, 'wb') as output:

pickle.dump(obj, output, pickle.HIGHEST\_PROTOCOL)

#generate deniable encryption keys

def generateDMEKeys(self):

if os.path.exists("keys/keys.txt"):

f = open('keys/keys.txt', 'rb')

keys = pickle.load(f)

f.close()

private\_key = keys[0]

public\_key = keys[1]

else:

secret\_key = generate\_eth\_key()

private\_key = secret\_key.to\_hex() # hex string

public\_key = secret\_key.public\_key.to\_hex()

keys = [private\_key, public\_key]

f = open('keys/keys.txt', 'wb')

pickle.dump(keys, f)

f.close()

return private\_key, public\_key

#Deniable encryption will encrypt data using plain text adn public key

def deniableEncrypt(self, plainText, public\_key):

dme\_encrypt = encrypt(public\_key, plainText.encode())

dme\_encrypt = str(base64.b64encode(dme\_encrypt),'utf-8')

return dme\_encrypt

#denyable encryption will decrypt data using private key and encrypted text

def deniableDecrypt(self, encrypt, private\_key):

encrypt = base64.b64decode(encrypt)

dme\_decrypt = decrypt(private\_key, encrypt)

return dme\_decrypt

# 7. TESTING

## 7.1 INTRODUCTION

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

**System Testing and Implementation**

The purpose is to exercise the different parts of the module code to detect coding errors. After this the modules are gradually integrated into subsystems, which are then integrated themselves too eventually forming the entire system. During integration of module integration testing is performed. The goal of this is to detect designing errors, while focusing the interconnection between modules. After the system was put together, system testing is performed. Here the system is tested against the system requirements to see if all requirements were met and the system performs as specified by the requirements. Finally accepting testing is performed to demonstrate to the client for the operation of the system.

For the testing to be successful, proper selection of the test case is essential. There are two different approaches for selecting test case. The software or the module to be tested is treated as a black box, and the test cases are decided based on the specifications of the system or module. For this reason, this form of testing is also called “black box testing”.

The focus here is on testing the external behavior of the system. In structural testing the test cases are decided based on the logic of the module to be tested. A common approach here is to achieve some type of coverage of the statements in the code. The two forms of testing are complementary: one tests the external behavior, the other tests the internal structure.

Testing is an extremely critical and time-consuming activity. It requires proper planning of the overall testing process. Frequently the testing process starts with the test plan. This plan identifies all testing related activities that must be performed and specifies the schedule, allocates the resources, and specifies guidelines for testing. The test plan specifies conditions that should be tested; different units to be tested, and the manner in which the module will be integrated together. Then for different test unit, a test case specification document is produced, which lists all the different test cases, together with the expected outputs, that will be used for testing. During the testing of the unit the specified test cases are executed and the actual results are compared with the expected outputs. The final output of the testing phase is the testing report and the error report, or a set of such reports. Each test report contains a set of test cases and the result of executing the code with the test cases. The error report describes the errors encountered and the action taken to remove the error.

**Testing Techniques**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected. In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

**Black Box Testing**

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structure or external database access
* Performance errors
* Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

**White Box Testing**

In this testing, the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

1. Guarantee that all independent paths have been executed.
2. Execute all logical decisions on their true and false sides
3. Execute all loops at their boundaries and within their operational
4. Execute internal data structures to ensure their validity.

**Testing Strategies**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

This System consists of 3 modules. Those are Reputation module, route discovery module, audit module. Each module is taken as unit and tested. Identified errors are corrected and executable unit are obtained.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**Functional Testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

**Functional testing is centered on the following items**

Valid Input: identified classes of valid input must be accepted

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised

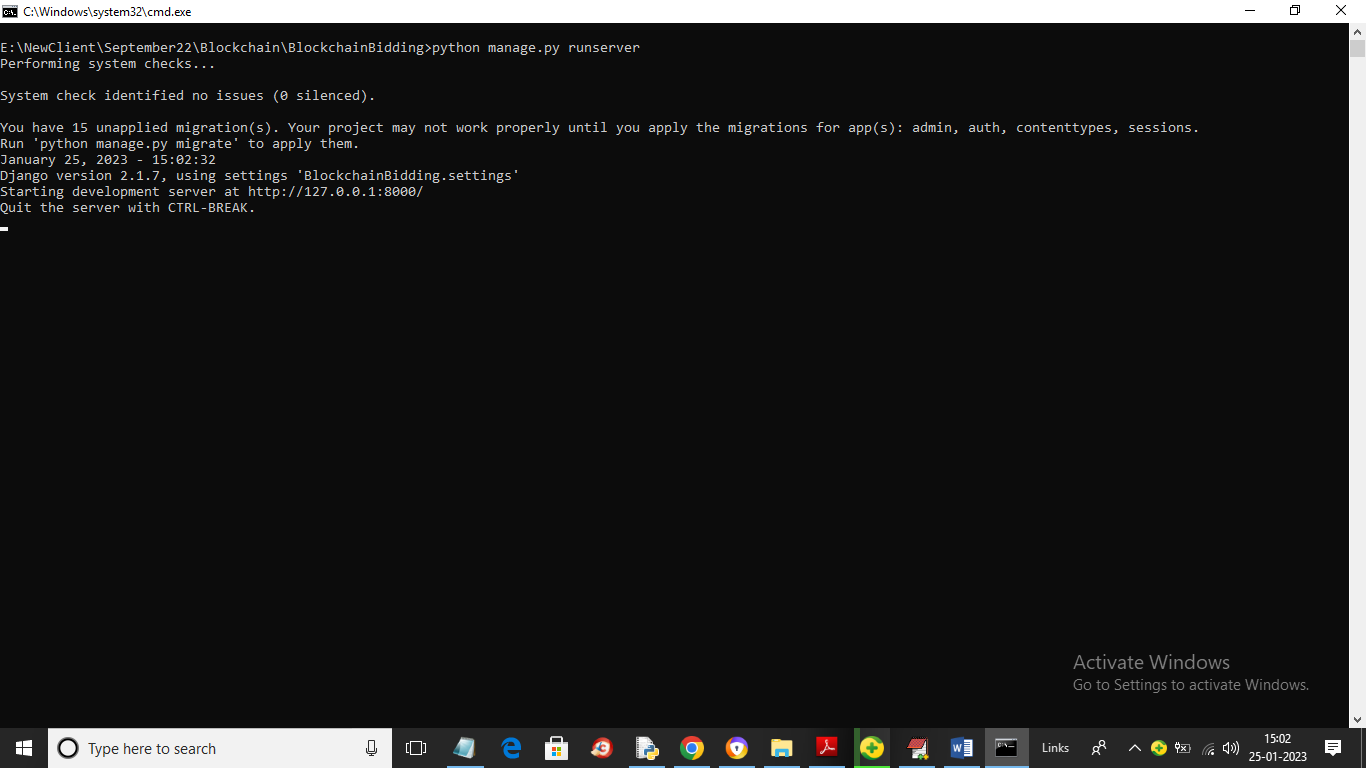
Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked

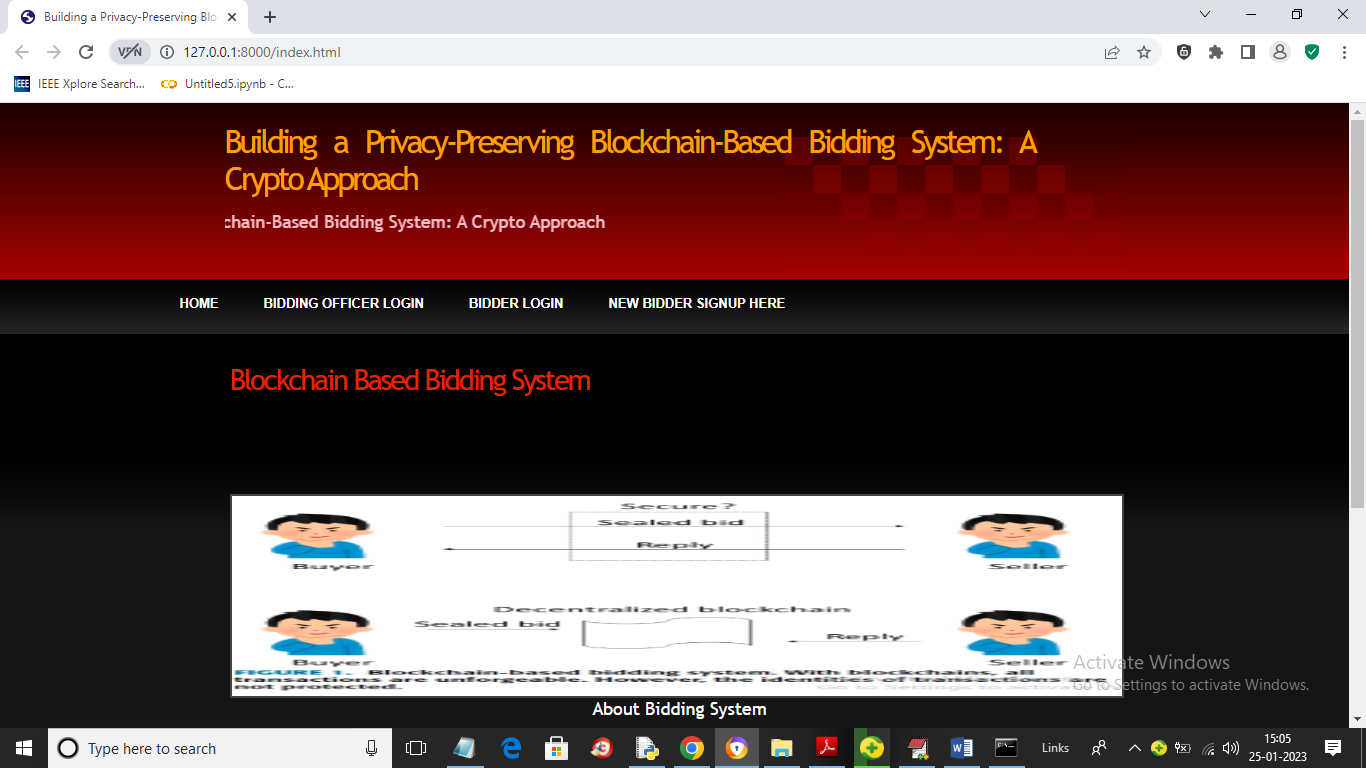
Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes.

# 8. RESULTS

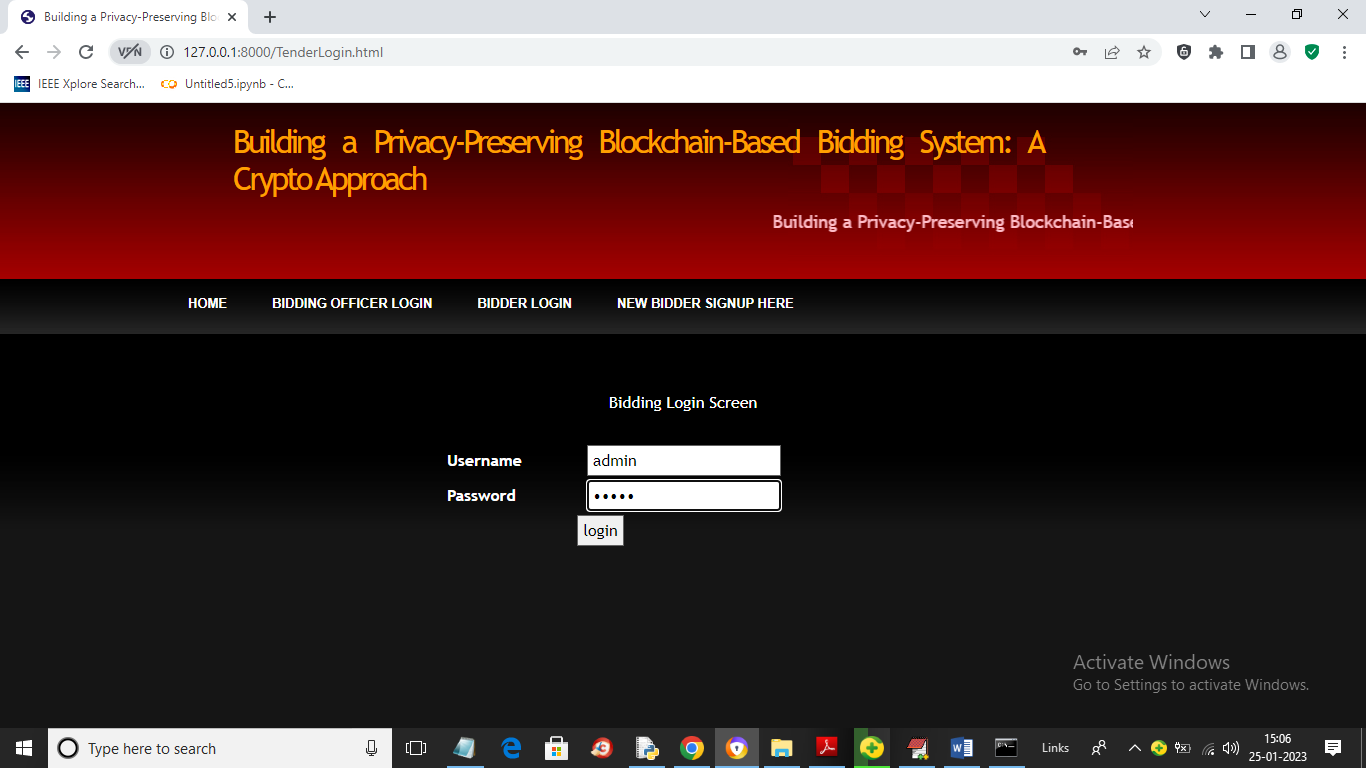
To run project double click on ‘run.bat’ file to start python server and get below screen



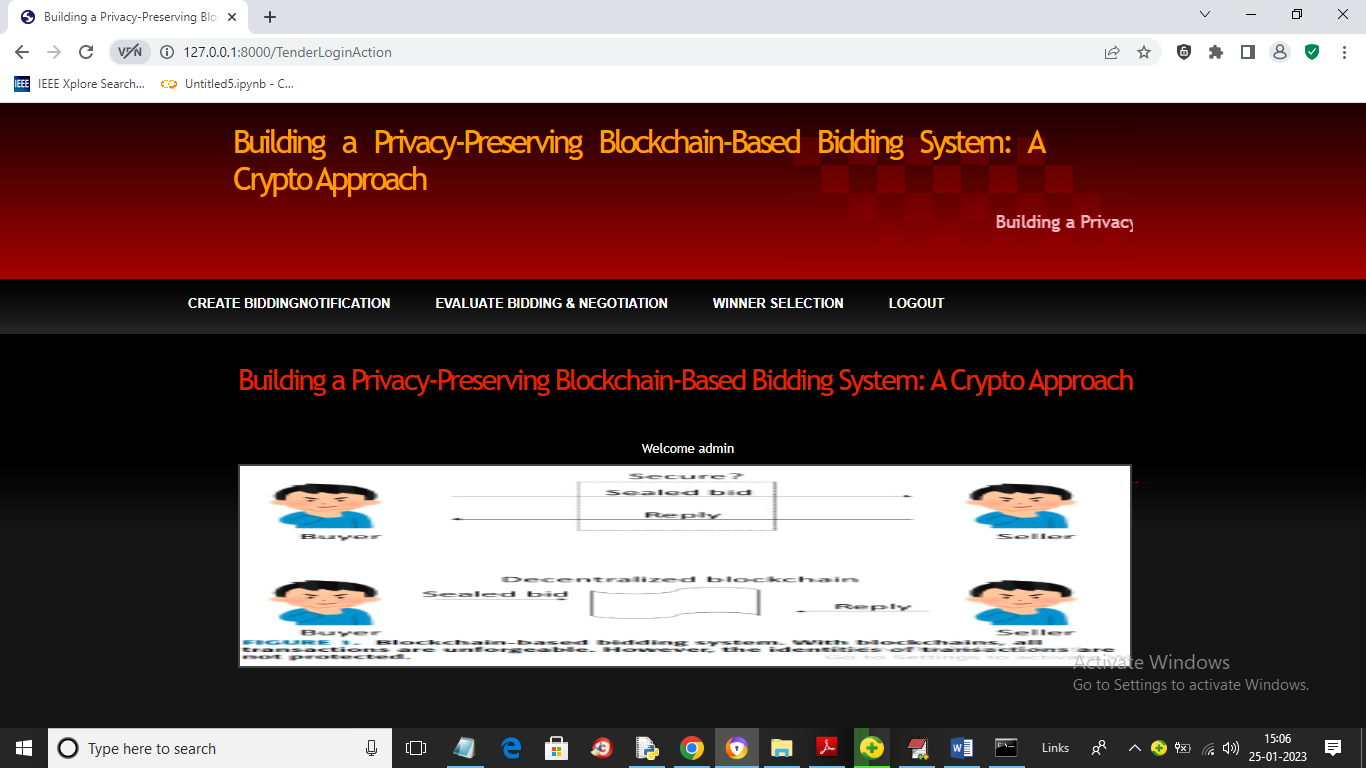
In above screen python web server started and now open browser and enter URL ashttp://127.0.0.1:8000/index.html and press enter key to get below page



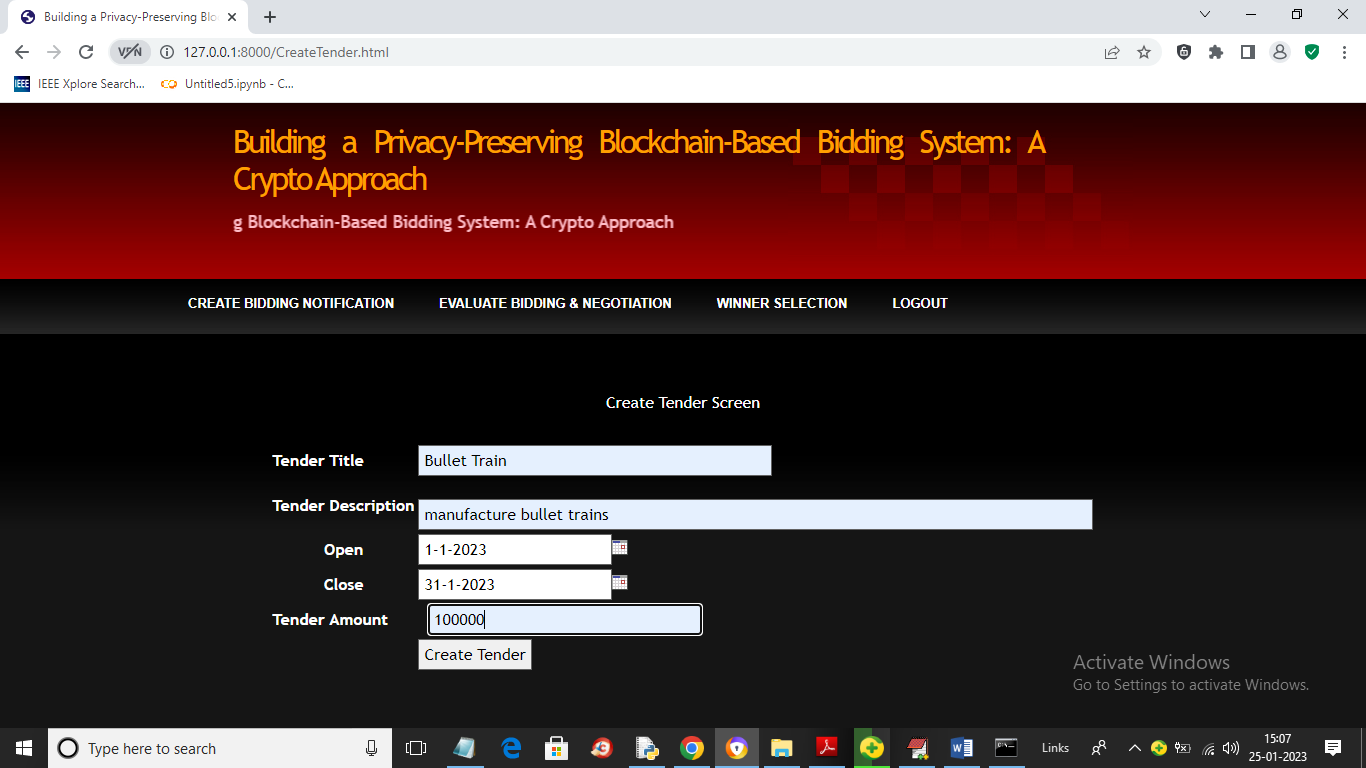
In above screen click on ‘Bidding Officer Login’ link to get below page



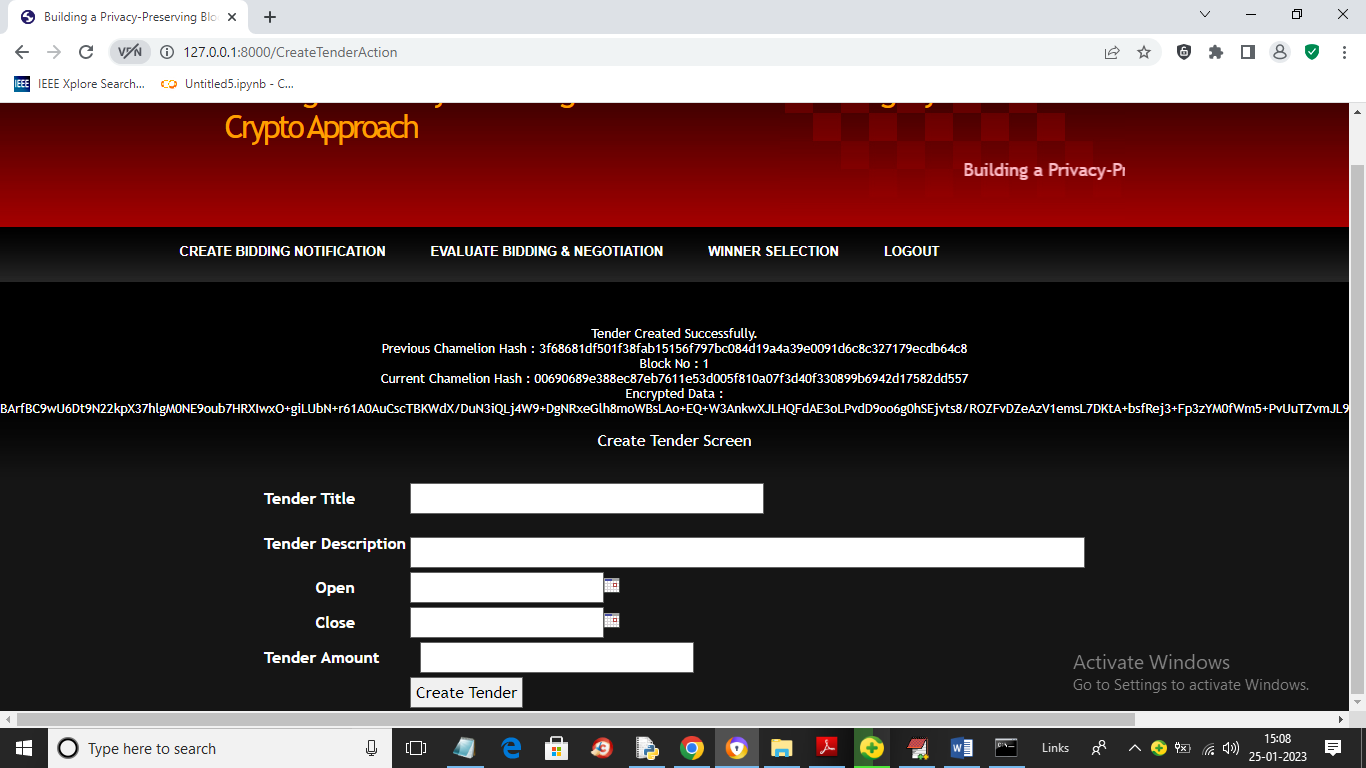
In above screen bidding officer is login and after login will get below page



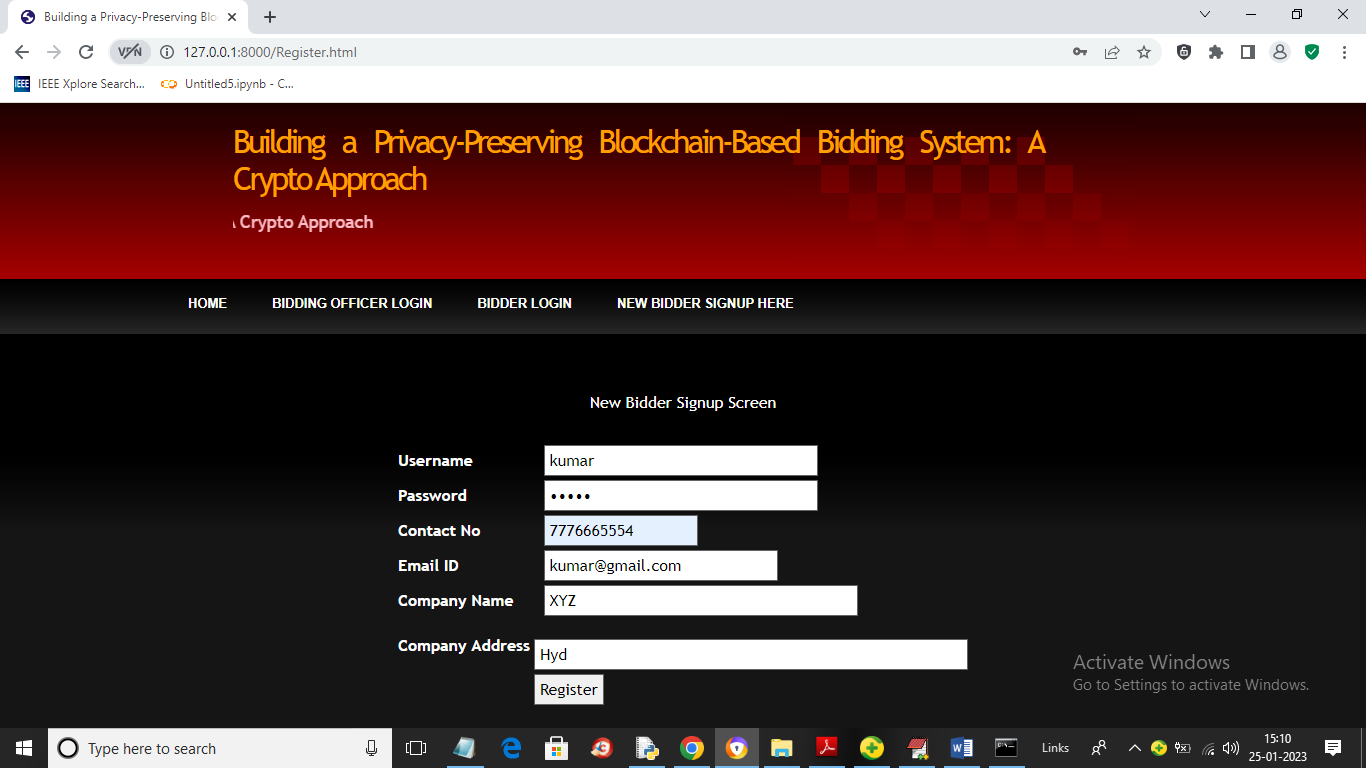
In above screen bidding officer can click on ‘Create Bidding Notification’ link to create tender like below screen



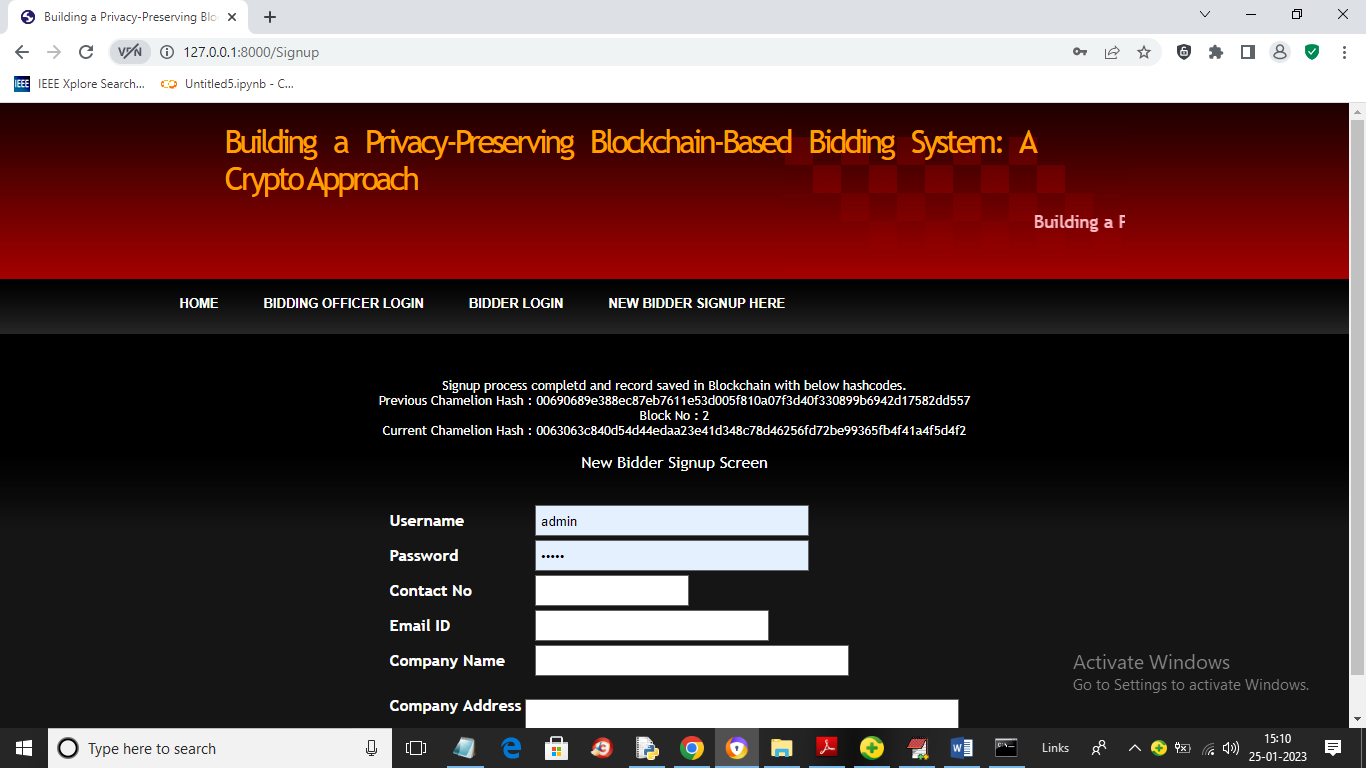
In above screen we are creating one tender with start and end date with initial tender amount and now press button to save data in Blockchain and will get below output



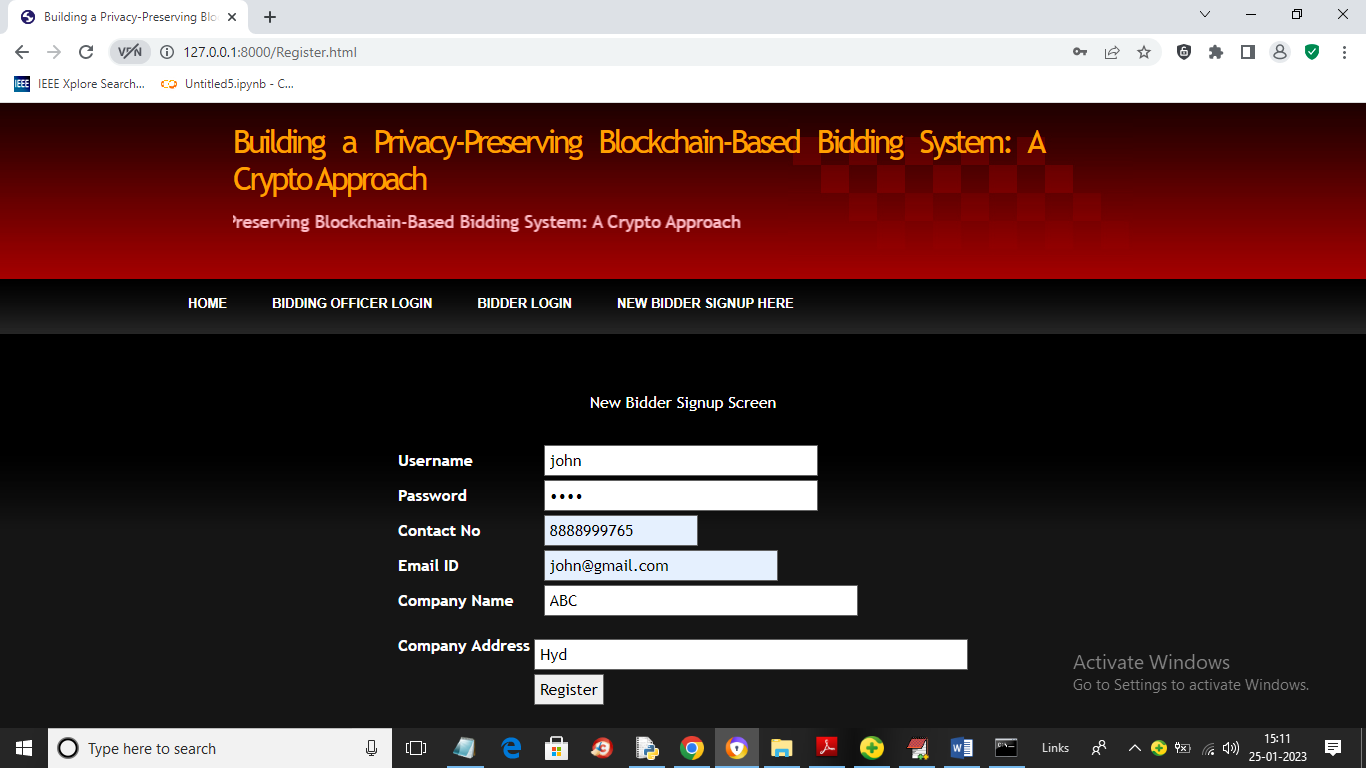
In above screen in first line we can see ‘Tender Created’ and then we can see generated previous and current block Chameleon hash code and then displaying encrypted data and now logout and signup and login bidders to bid for above tender



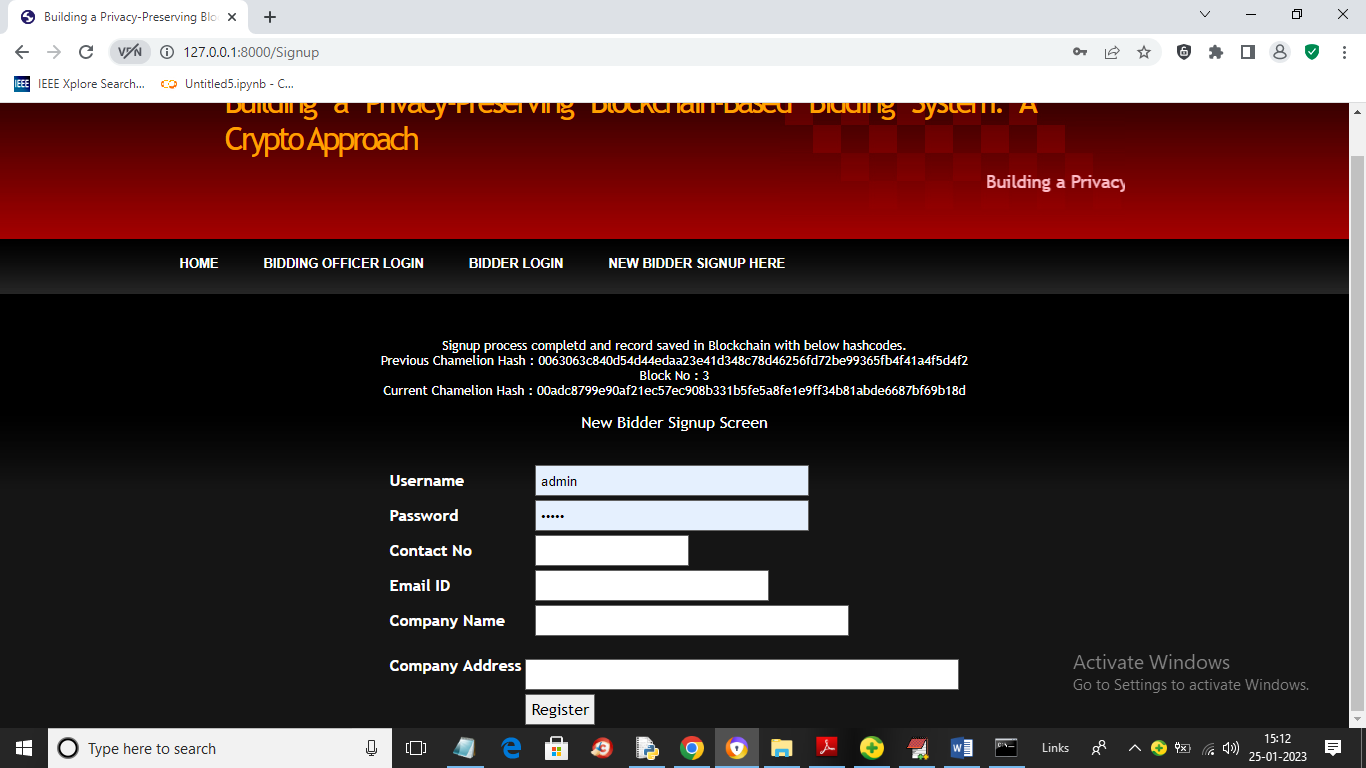
In above screen one user is signing up and then press button to get below page



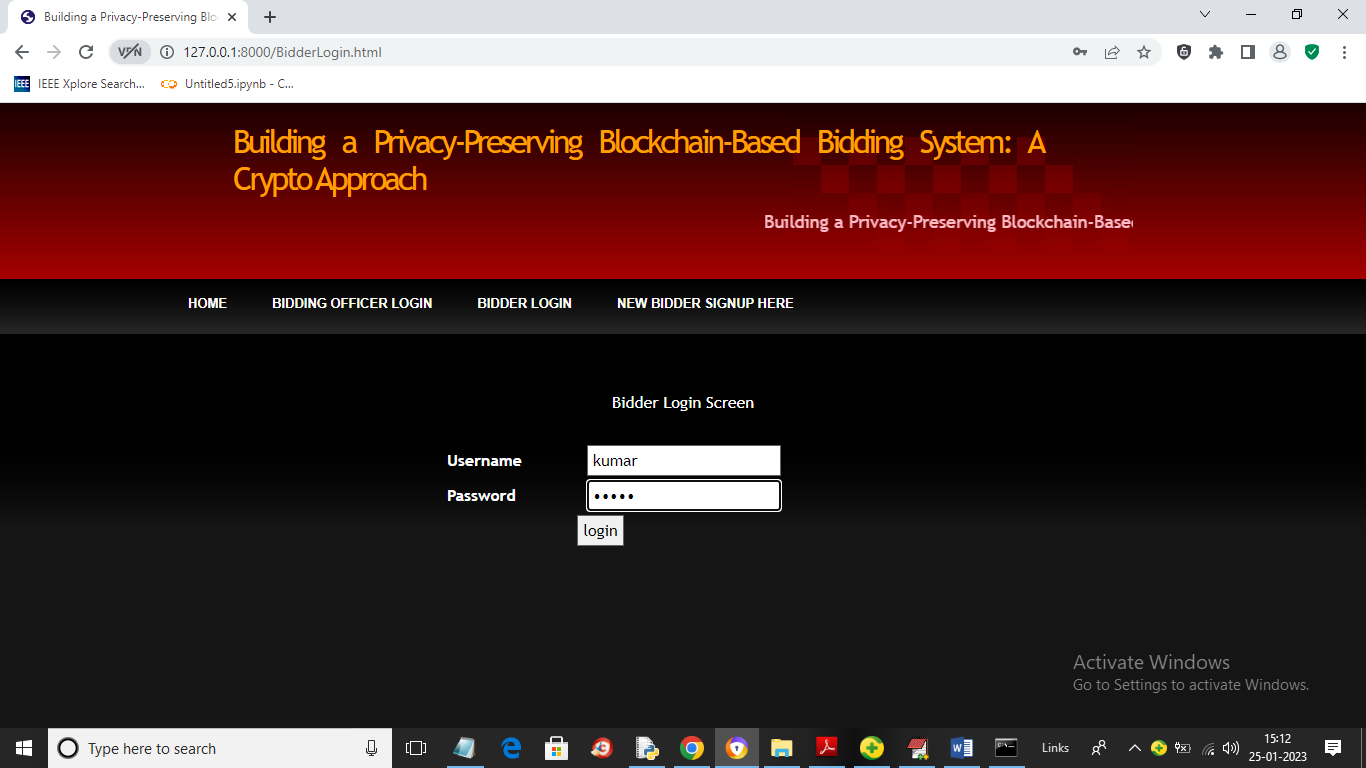
In above screen we can see signup task completed and we can see hash code returned by Blockchain and similarly design any number of bidders



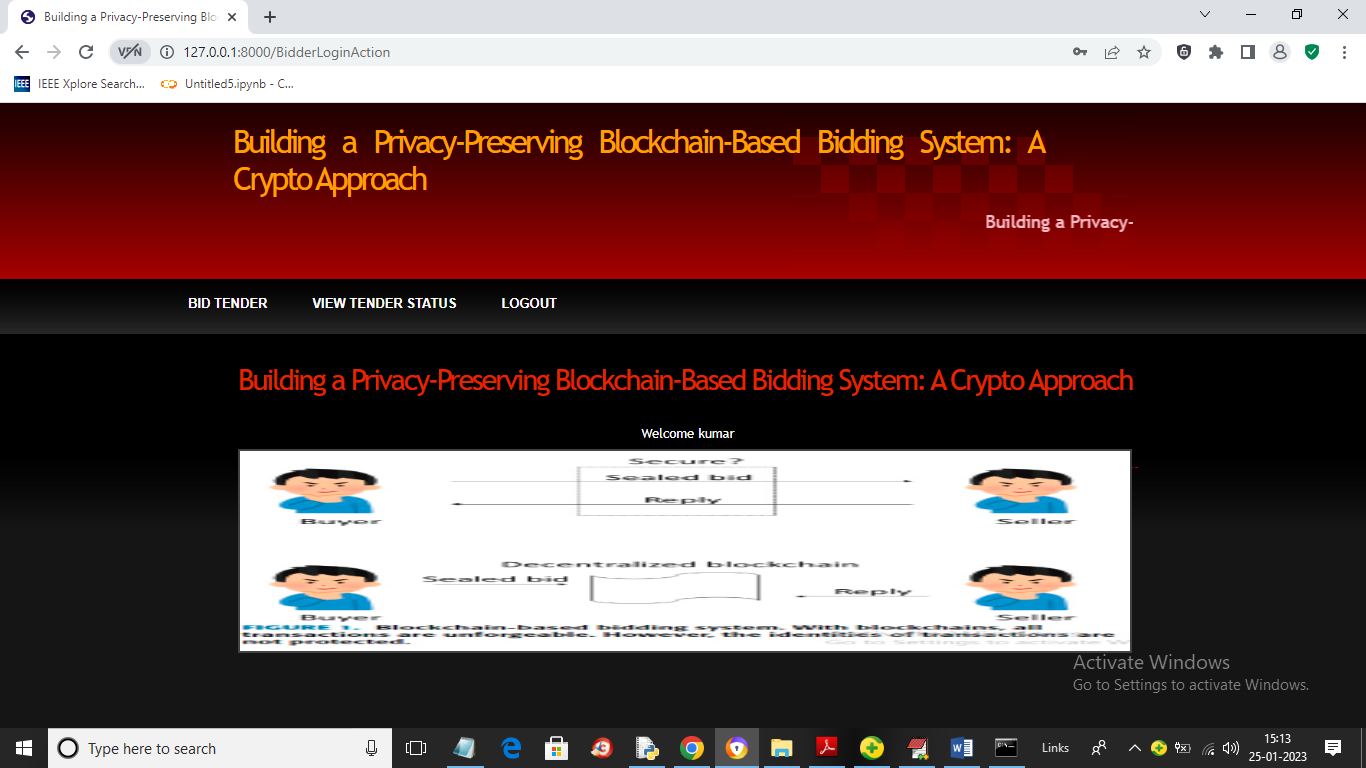
In above screen another user ‘John’ is signing up and press button to get below page



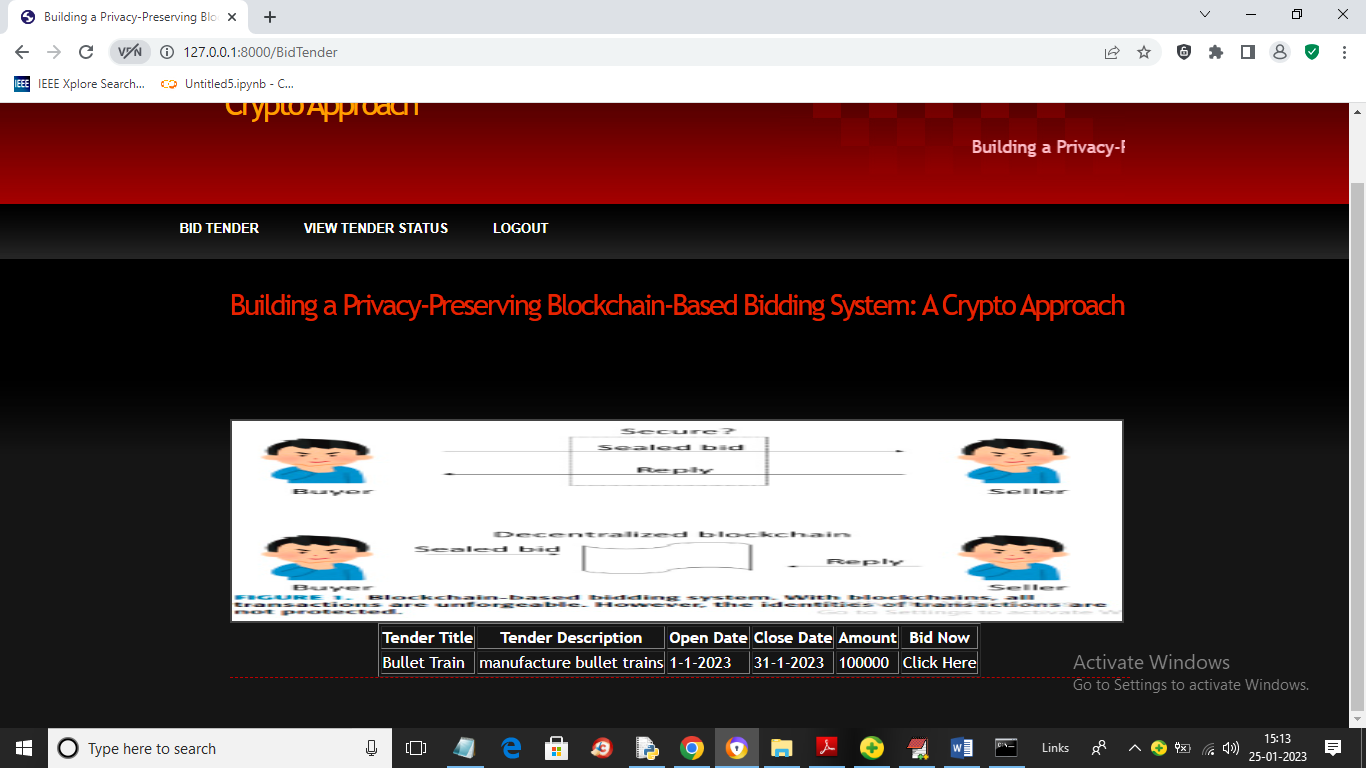
In above screen signup completed and now login as one user



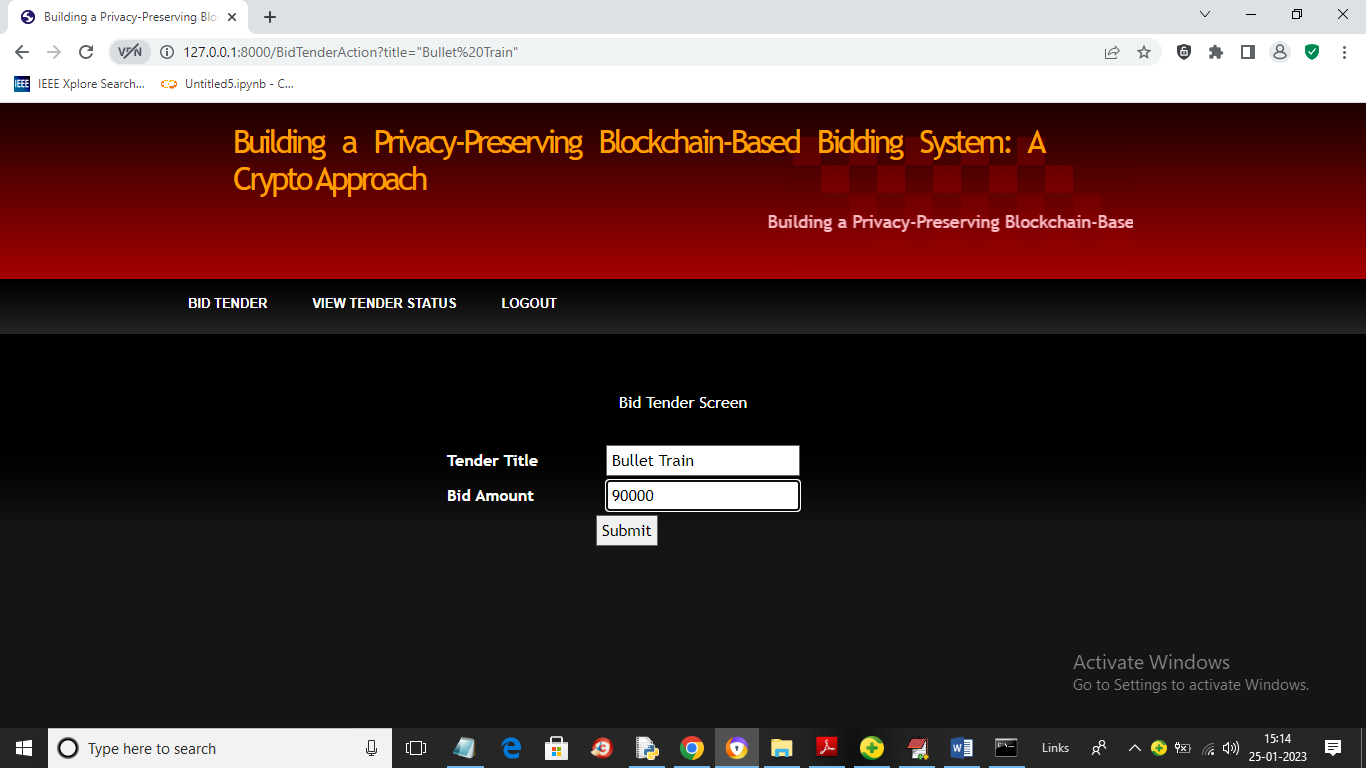
In above screen ‘kumar’ user is login and after login will get below page



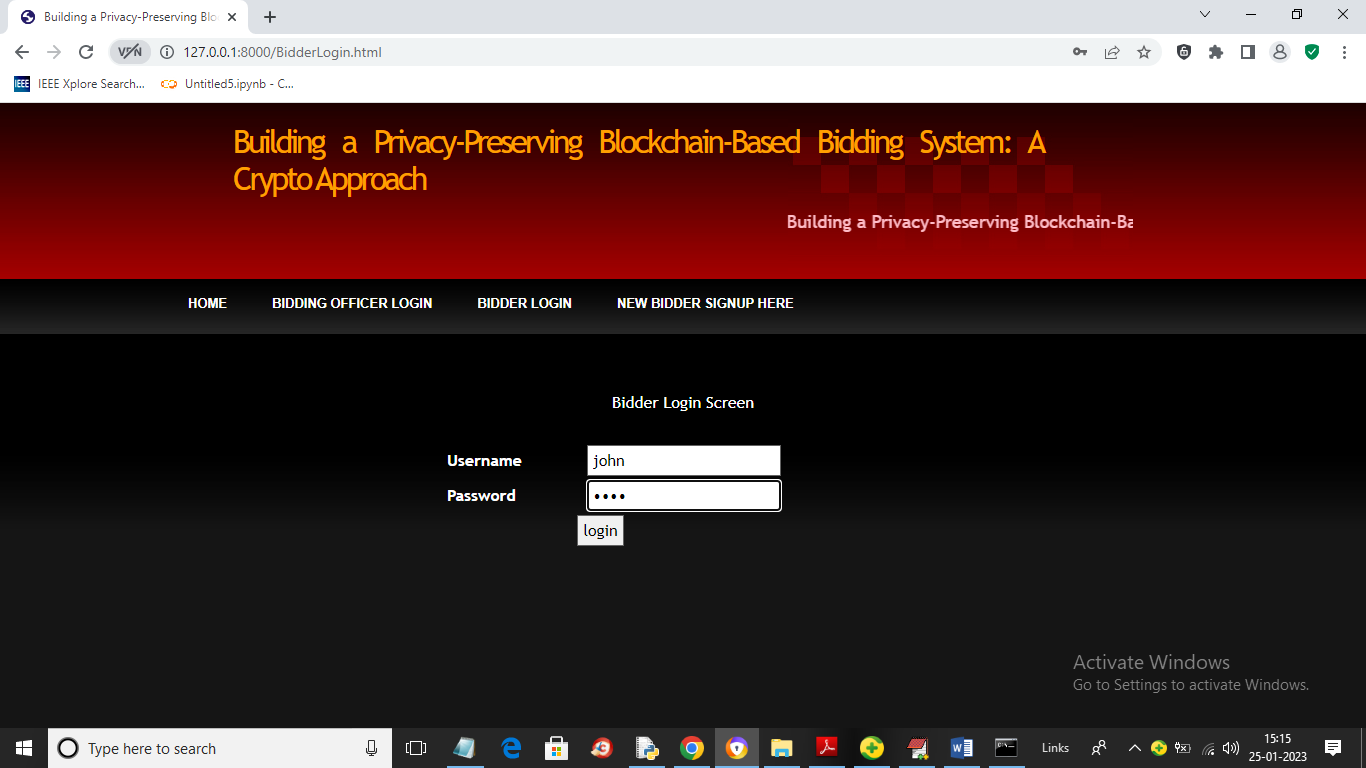
In above screen user can click on ‘Bid Tender’ link to give bidding on available tenders



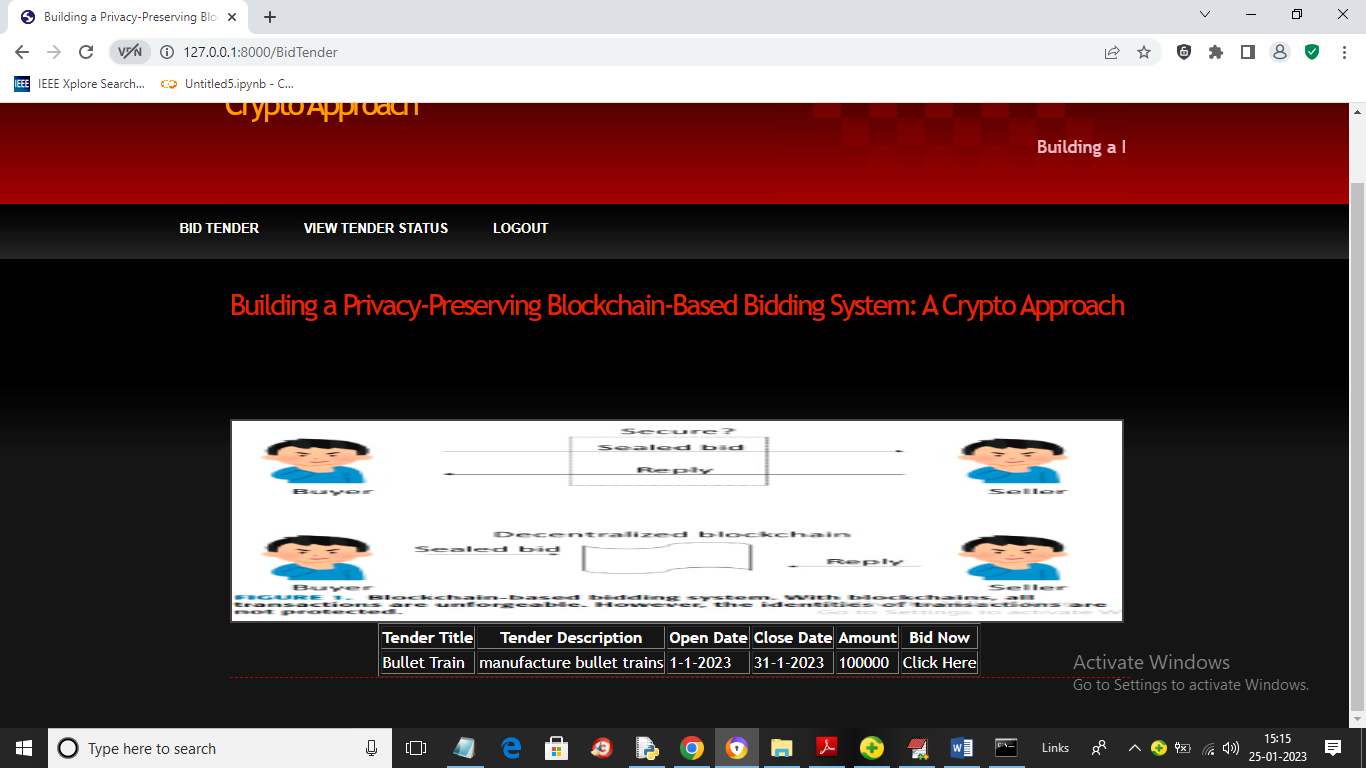
In above screen user can view list of tenders and can click on ‘Click Here’ link to give bids like below screen



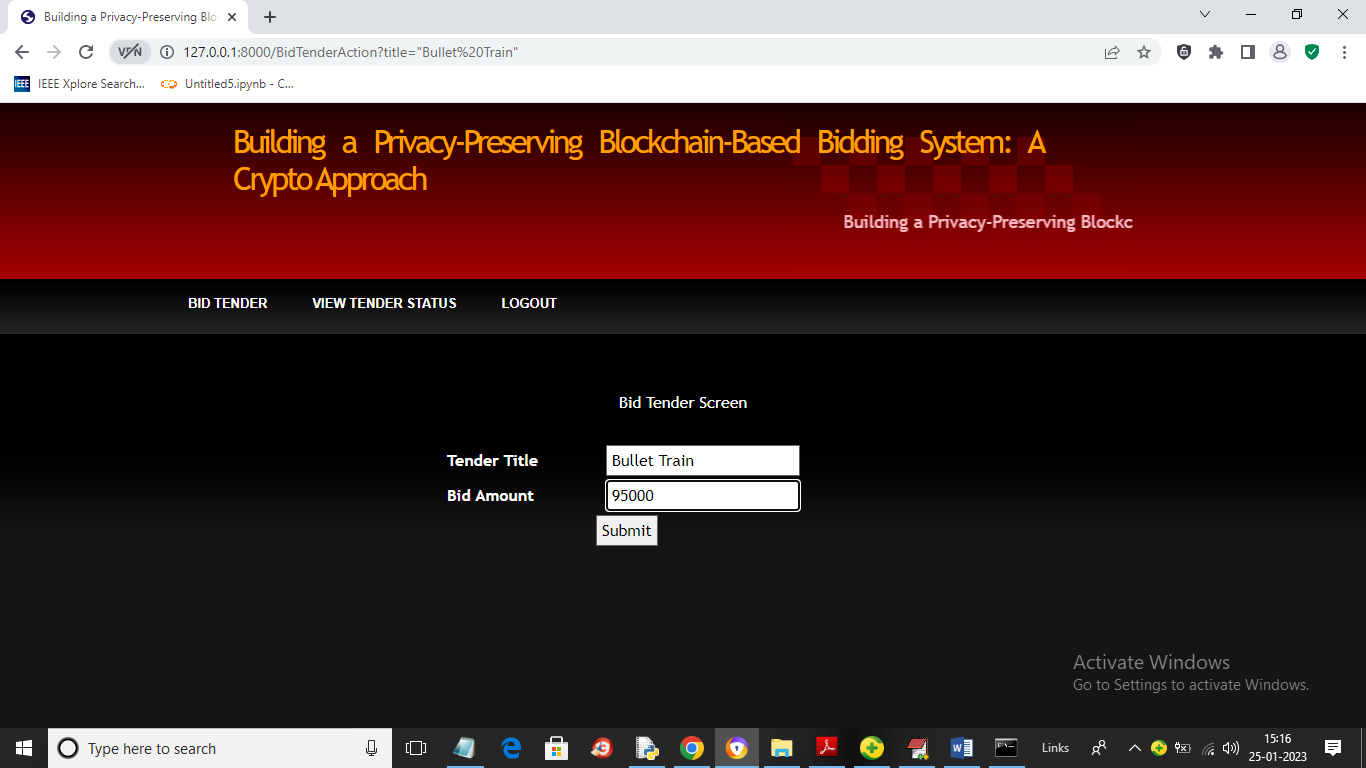
In above screen kumar user giving bidding as 90000 and now login as another user and give bidding



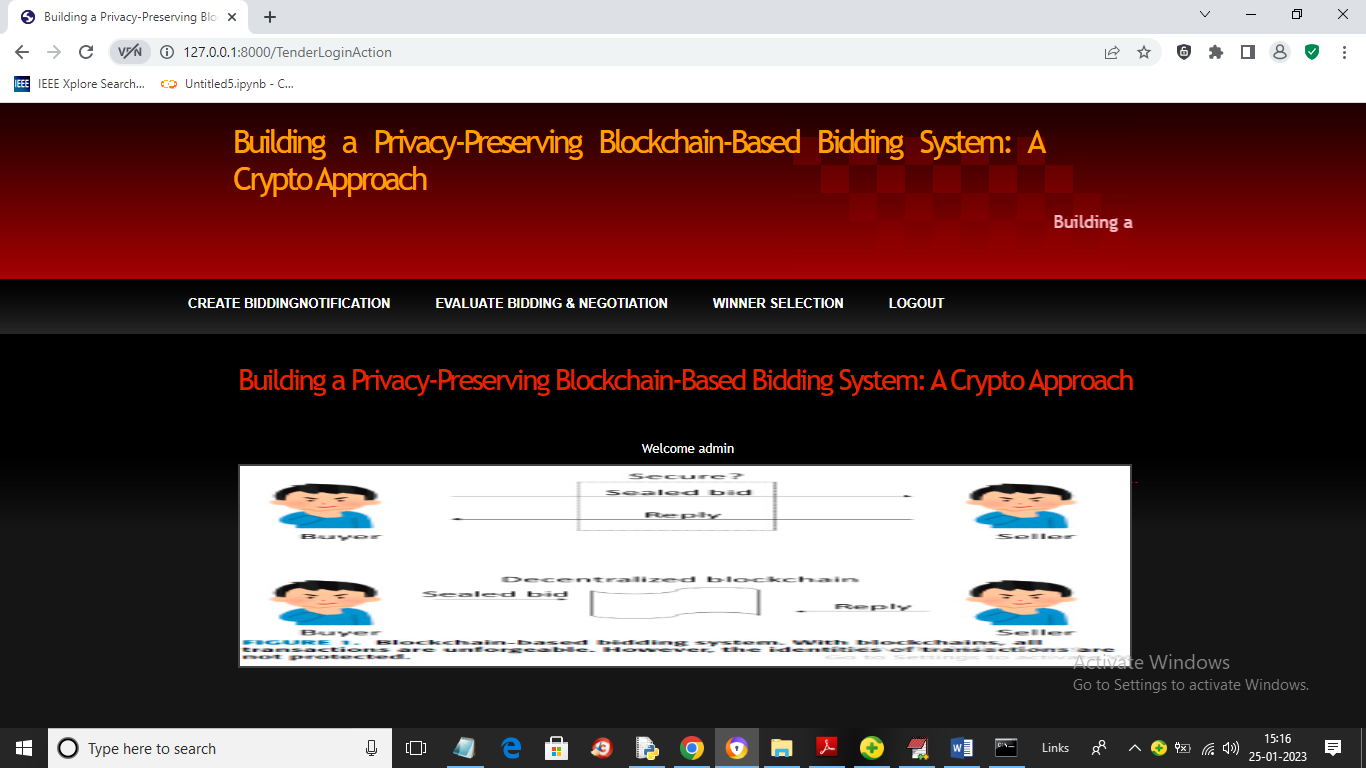
In above screen user John is login and after login will get below page



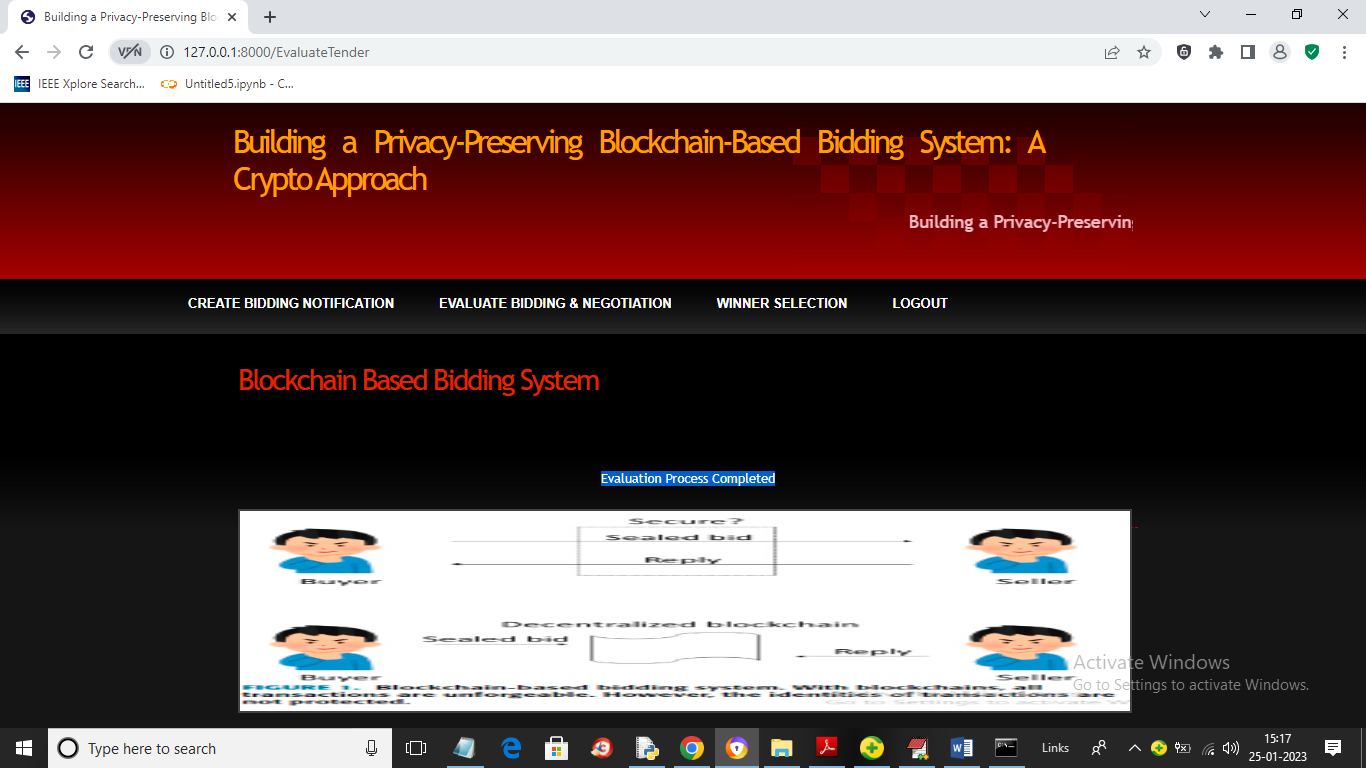
In above screen John user can click on ‘Bid Tender’ link to give bidding



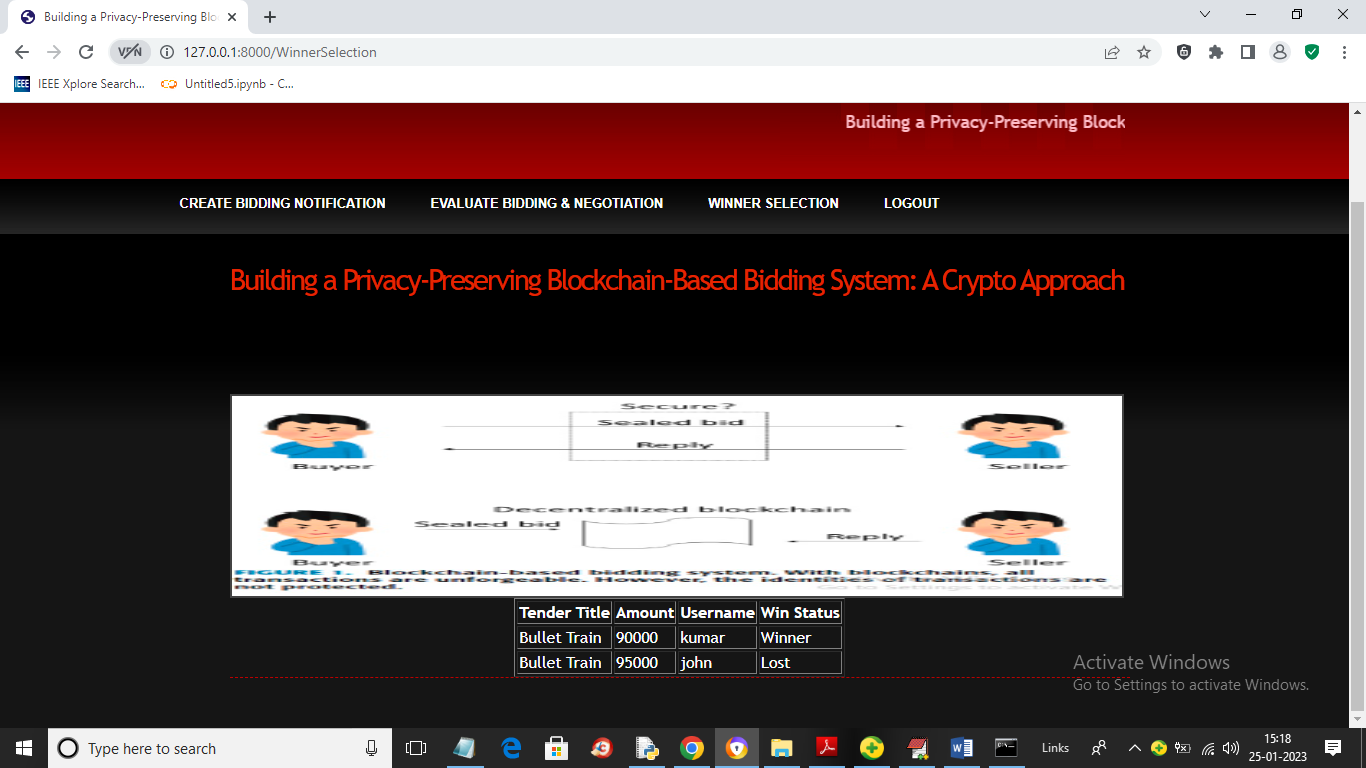
In above screen John is giving 95000 bidding and now logout and login as ‘Bidding officer’ to select winner with low bidding



In above screen bidding officer can click on ‘Evaluate Bidding & Negotiation’ link to select winner like below screen



In above screen evaluation process completed and now click on ‘Winner Selection’ link to view winner like below screen



In above screen we can see kumar and john bid for same tender but kumar gave lowest bid as 90000 so he got selected as winner and similarly bidder can login and view status of winner.

In above screens we can see all users who bid for same tender can decrypt and view bidding and winner details.

# 9. CONCLUSION

We proposed a new encryption called DME to solve the coercion problem on blockchain-based bidding systems. Using DME on blockchain-based bidding systems, we can preserve the privacy of the buyer and seller identities during the auction. We use a fake buyer and seller pair to cover the real transaction, including the identities and the content. We show that our scheme costs approximately twice the base scheme in both space and computational time for the deniability feature. We believe it is worth the cost to protect the privacy of transaction identities.

# 10.BIBLIOGRAPHY

[1] Y.-H. Chen, S.-H. Chen, and I.-C. Lin, ‘‘Blockchain based smart contract for bidding system,’’ in Proc. IEEE Int. Conf. Appl. Syst. Invention (ICASI), Apr. 2018, pp. 208–211.

[2] P. Manimaran and R. Dhanalakshmi, ‘‘Blockchain-based smart contract for E-bidding system,’’ in Proc. 2nd Int. Conf. Intell. Commun. Comput. Techn. (ICCT), Sep. 2019, pp. 55–59. [3] N. Diallo, W. Shi, L. Xu, Z. Gao, L. Chen, Y. Lu, N. Shah, L. Carranco, T.-C. Le, A. B. Surez, and G. Turner, ‘‘EGov-DAO: A better government using blockchain based decentralized autonomous organization,’’ in Proc. Int. Conf. eDemocracyeGovernment (ICEDEG), Apr. 2018, pp. 166–171.

[4] L. Malina, G. Srivastava, P. Dzurenda, J. Hajny, and R. Fujdiak, ‘‘A secure publish/subscribe protocol for Internet of Things,’’ in Proc. 14th Int. Conf. Availability, Rel. Secur., New York, NY, USA, 2019, doi: 10.1145/3339252.3340503.

[5] B. Seth, S. Dalal, V. Jaglan, D. Le, S. Mohan, and G. Srivastava, ‘‘Integrating encryption techniques for secure data storage in the cloud,’’ Trans. Emerg. Telecommun. Technol., Sep. 2020, Art. no. e4108, doi: 10.1002/ett.4108.

[6] G. Ateniese, D. Francati, D. Nu nez, and D. Venturi, ‘‘Match me if you can: Matchmaking encryption and its applications,’’ in Advances in Cryptology, A. Boldyreva and D. Micciancio, Eds. Cham, Switzerland: Springer, 2019, pp. 701–731. [7] H. Krawczyk and T. Rabin. (1998). Chameleon Hashing and Signatures. Accessed: Mar. 17, 1998. [Online]. Available: http://eprint.iacr. org/1998/010

[8] A. Shamir, ‘‘Identity-based cryptosystems and signature schemes,’’ in Advances in Cryptology, G. R. Blakley and D. Chaum, Eds. Berlin, Germany: Springer, 1985, pp. 47–53.

[9] Y. Desmedt and J.-J. Quisquater, ‘‘Public-key systems based on the difficulty of tampering (Is there a difference between DES and RSA?),’’ in Advances in Cryptology, A. M. Odlyzko, Ed. Berlin, Germany: Springer, 1987, pp. 111–117.

[10] H. Tanaka, ‘‘A realization scheme for the identity-based cryptosystem,’’ in Advances in Cryptology, C. Pomerance, Ed. Berlin, Germany: Springer, 1988, pp. 340–349.

[11] S. Tsujii and T. Itoh, ‘‘An ID-based cryptosystem based on the discrete logarithm problem,’’ IEEE J. Sel. Areas Commun., vol. 7, no. 4, pp. 467–473, May 1989.

[12] U. M. Maurer and Y. Yacobi, ‘‘Non-interactive public-key cryptography,’’ in Advances in Cryptology, D. W. Davies, Ed. Berlin, Germany: Springer, 1991, pp. 498–507.

[13] D. Boneh and M. Franklin, ‘‘Identity-based encryption from the Weil pairing,’’ in Advances in Cryptology, J. Kilian, Ed. Berlin, Germany: Springer, 2001, pp. 213–229.

[14] A. Sahai and B. Waters, ‘‘Fuzzy identity-based encryption,’’ in Advances in Cryptology, R. Cramer, Ed. Berlin, Germany: Springer, 2005, pp. 457–473.

[15] B. Waters, ‘‘Efficient identity-based encryption without random oracles,’’ in Advances in Cryptology, R. Cramer, Ed. Berlin, Germany: Springer, 2005, pp. 114–127.

[16] C.-I. Fan, L.-Y. Huang, and P.-H. Ho, ‘‘Anonymous multireceiveridentitybased encryption,’’ IEEE Trans. Comput., vol. 59, no. 9, pp. 1239–1249, Jan. 2010.

[17] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, ‘‘Identity-based encryption with outsourced revocation in cloud computing,’’ IEEE Trans. Comput., vol. 64, no. 2, pp. 425–437, Feb. 2015. [18] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, ‘‘Deniable encryption,’’ in Advances in Cryptology, B. S. Kaliski, Ed. Berlin, Germany: Springer, 1997, pp. 90–104.

[19] M. Klonowski, P. Kubiak, and M. Kutyłowski, ‘‘Practical deniable encryption,’’ in Theory and Practice of Computer Science, V. Geffert, J. Karhumäki, A. Bertoni, B. Preneel, P. Návrat, and M. Bieliková, Eds. Berlin, Germany: Springer, 2008, pp. 599–609.

[20] A. O’Neill, C. Peikert, and B. Waters, ‘‘Bi-deniable public-key encryption,’’ in Advances in Cryptology, P. Rogaway, Ed. Berlin, Germany: Springer, 2011, pp. 525–542.