

## OPTIMIZATION OF E-COMMERCE PLATFORM STRUCTURE BASED ON ARTIFICIAL INTELLIGENCE AND BLOCKCHAIN TECHNOLOGY

### A PROJECT REPORT

***Submitted by***

## ARUNA. M [REGISTER NO:211417104024] HARINI.T.M [REGISTER NO:211417104080] KRITHIKAA.K [REGISTER NO:211417104124]

***in partial fulfillment for the award of the degree of***

## BACHELOR OF ENGINEERING

**IN**

### COMPUTER SCIENCE AND ENGINEERING

**PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.**

**ANNA UNIVERSITY: CHENNAI 600 025**

**AUGUST 2021**

**BONAFIDE CERTIFICATE**

Certified that this project report **“Structure Optimization of E-Commerce Platform Based On Artificial Intelligence and Blockchain Technology**” is the bonafide work of “**Aruna.M (211417104024) ,Harini.T.M (211417104080) Krithikaa.K(211417104124)”** who carried out the project work under my supervision.

### SIGNATURE SIGNATURE

**Dr.S.MURUGAVALLI,M.E.,Ph.D., Mrs.C.Vijayalakshmi,M.Tech.,Ph.D,**

**HEAD OF THE DEPARTMENT ASSISTANT PROFESSOR**

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI, NAZARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on **05.08.2021**

### INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like express our heartfelt and sincere thanks to our Directors

### Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHIKUMAR, M.E., Ph.D., and

**Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide Mrs.C.VIJAYALAKSHMI ,**

**M.Tech., 𝐏h.D.,** and all the faculty members of the Department of CSE for their advice and suggestions for the successfulcompletion of the project.

### NAME OF THE STUDENTS

**Aruna.M** **Harini.T.M Krithikaa.K**

**ABSTRACT**

E-Commerce has become more and more popular because of rich products, fast transactions, and free from time, locations, stores, and so on. However, the disclosure of personal data such as their IDs, addresses, and phone numbers has become a major concern for online activities. The current e-commerce model is at the crossroads of ownership and privacy. To address this, this article creates an enterprise protocol that uses smart personal contracts to protect privacy during the negotiation phase. This protocol allows contracting parties to conduct business without disclosing personal information such as identity, address, and phone number. Furthermore, we employ the zero-knowledge proof to ensure ownership.

## TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** |  |
|  | **LIST OF TABLES** |  |
|  | **LIST OF FIGURES** |  |
|  | **LIST OF SYMBOLS, ABBREVIATIONS** |  |
| **1.** | **INTRODUCTION** |  |
|  | 1.1 Overview | 1 |
|  | 1.2 Problem Definition | 2 |
| **2.** | **LITERATURE SURVEY** | 3 |
| **3.** | **SYSTEM ANALYSIS** |  |
|  | 2.1 Existing System | 7 |
|  | 2.2 Proposed system | 8 |
|  | 2.3 Requirement Analysis and Specification |  |
|  | 2.3.1 Input Requirements | 8 |
|  | 2.3.2 Output Requirements | 9 |
|  | 2.3.3 Functional Requirements | 9 |
|  | 2.4 Technology Stack | 10 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| **4.** | **SYSTEM DESIGN** |  |
|  | 4.1 Data Dictionary | 15 |
|  | 4.2 UML Diagrams | 16 |
| **5.** | **SYSTEM ARCHITECTURE** |  |
|  | 5.1 Architecture Overview | 25 |
|  | 5.2 Module Design | 25 |
| **6.** | **SYSTEM IMPLEMENTATION** |  |
|  | 6.1 Client-side coding | 27 |
|  | 6.2 Server-side coding | 41 |
| **7.** | **SYSTEM TESTING** |  |
|  | 7.1 Unit Testing | 67 |
|  | 7.2 Integration Testing | 67 |
|  | 7.3 Test Cases & Reports / Performance Analysis | 68 |
| **8.** | **CONCLUSION** |  |
|  | 8.1 Conclusion and Future Enhancements | 70 |
|  | **APPENDICES** |  |
|  | A.1 Sample Screens | 71 |
|  | A.2 Publications | 75 |
|  | **REFERENCES** | 76 |

**LIST OF TABLES**

4.1.1 Product Table

4.1.2 Register Table

4.1.3 Smart Contract Table

4.1.4 Trans Table

4.1.5 USD Token Table

7.3.1 Seller and Buyer - Registration validation page Testing

7.3.2 Seller and Buyer - Hash Code Generation Testing

7.3.3 Seller and Buyer - View Transaction Testing

**LIST OF FIGURES**

4.2.1 Use case diagram

4.2.2 Collaboration Diagram

4.2.3 Sequence Diagram

4.2.4 Activity Diagram

4.2.5 State Chart Diagram

5.1 Architecture Overview

**LIST OF ABBREVIATIONS**

**CA** - Central Authority

**PKI** - Public Key Infrastructure

**ZK** - Zero-Knowledge

**P2P** - Peer to Peer

# CHAPTER 1

# INTRODUCTION

### Overview

E-commerce has become more and more popular because of rich products, fast transactions, and free from time, locations, stores and so on. However, the disclosure of users’ personal information such as 1) identities, 2) addresses, and 3) phone numbers has become a big concern of online activity . Factually, it has formed a huge “gray industry” that seriously endangers users’ safety and privacy. It is not uncommon for sellers to threaten and enforce customers to make, modify or delete product reviews which are against their wills. Meanwhile, online shopping websites are also suffering from malicious bad reviews or spurious praise reviews , which reduce the Quality of Experience (QoE). There are already some researches and practices addressing these issues. Janice Tsai et al. show that consumers are more willing to pay a premium for privacy protection. Berendt et al. propose PET which provides more timely privacy protection and trust tools for Web applications. Li et al. present a coherent adaptive trust model based on a transaction feedback system to quantify and compare the credibility of users in peer-to-peer (P2P) e-commerce communities. Vandervort proposes three different models by which a reputation system could be implemented based on the blockchain technology. However, none of them consider the issue of privacy disclosure of buyers during the delivery of products. In order to address privacy issues in product delivery, “Privacy Waybill” is invented to hide consumer’s information so that private information will not appear in the waybill. Only couriers can obtain recipients’ information including identities, addresses, and phone numbers through authorized devices. Although it prevents the disclosure of personal information in some extent, this technology cannot hide the addresses, phone numbers and other users’ information from the sellers. As far as we know, there are no existing models protecting private information from beginning to end. In this paper, we address this problem by focusing on the trade and certiﬁcation of collections such as artworks, luxuries and limited edition products. These goods are much easier to encounter privacy issues in E-commerce since buyers are usually more desired to protect their private information such as identities or addresses in these transactions. We would like to build a uniﬁed platform for collectible

collections and items to exchange, evaluate and certiﬁcate their ownership. Just like the CryptoKitties.co does that on the Fancy Cat which is maintained on blockchain, our platform aims to give, certiﬁcate and protect the value of these “collections”. We protect users’ private information by employing the blockchain technologies including private smart contracts and zero-knowledge proof. Speciﬁcally, each trade is represented by a private smart contract, which deﬁnes the business logics, types of trade, counterparties, underlying assets, price, and any other relevant information.

**1.2 Problem Definition**

E-Commerce has become more and more popular because of rich products, fast transactions, and free from time, locations, stores, and so on. However, the disclosure of personal data such as their IDs, addresses, and phone numbers has become a major concern for online activities. The current e-commerce model is at the crossroads of ownership and privacy. To address this, this article creates an enterprise protocol that uses smart personal contracts to protect privacy during the negotiation phase. This protocol allows contracting parties to conduct business without disclosing personal information such as identity, address, and phone number. Furthermore, we employ the zero-knowledge proof to ensure ownership

**CHAPTER 2**

**LITERATURE SURVEY**

* 1. **A Reputation-Based Trust Model for Peer-to-Peer E-Commerce Communities**

Peer-to-Peer eCommerce communities are commonly perceived as an environment offering both opportunities and threats. One way to minimize threats in such an open community is to use community-based reputations to help evaluating the trustworthiness and predicting the future behavior of peers. This paper presents PeerTrust a coherent adaptive trust model for quantifying and comparing the trustworthiness of peers based on a transaction-based feed-back system. There are two main features of our model. First, we introduce three basic trust parameters in computing trustworthiness of peers. In addition to feedback a peer receives through its transactions with other peers, we in- corporate the total number of transactions a peer performs, and the credibility of the feedback sources into the model for evaluating the trustworthiness of peers. We argue that the trust models based solely on feedback from other peers in the community is inaccurate and ineffective. Second, we introduce two adaptive trust factors, the transaction context factor and the community context factor, to allow the basic trust metric to incorporate different contexts (situations) and to address common problems encountered in a variety of online eCommerce communities. We present a concrete method to validate the proposed trust model and report the set of simulation-based experiments, showing the feasibility and benefit of the PeerTrust model.

### Drawbacks:

* + - There is an extensive amount of research focused on building trust for electronic markets through trusted third parties or intermediaries. However, it is not applicable to P2P eCommerce communities where peers are equal in their roles and are independent entities, thus no peers can serve as trusted third parties or intermediaries.

**Author**: Li Xiong, Ling Liu.

**Published In**: 2003

* 1. **Challenges and Opportunities Associated with a Bitcoin-Based Transaction Rating System**

It has been shown that seller ratings given by previous buyers give new customers useful information when making purchasing decisions. Bitcoin, however, is designed to obfuscate the link between buyer and seller with a layer of limited anonymity, thus preventing buyers from finding or validating this information. While this level of anonymity is valued by the Bitcoin community, as Bitcoin moves toward greater adoption there will be pressure from buyers who wish to know more about who they are doing business with, and sellers who consider their reputation a strong selling point, to allow greater transparency. We consider three different models by which a reputation/rating system could be implemented in conjunction with Bitcoin transactions and consider pros and cons of each. We find that each presents challenges on both the technological and social fronts.

### Drawbacks:

* + - The Bitcoin community points to this anonymity, often referred to as pseudonymity because it is not absolute, as an asset, a way of circumventing surveillance and cumbersome regulatory regimes. It is also considered a defense against the user pro- filing/data mining practiced by large merchants such as K-Mart.

**Author**: David Vandervort

**Published In**: 2014

## Zerocash: Decentralized Anonymous Payments from Bitcoin

Bitcoin is the first digital currency to see widespread adoption. While payments are conducted between pseudonyms, Bitcoin cannot offer strong privacy guarantees: payment transactions are recorded in a public decentralized ledger, from which much information can be deduced. Zerocoin (Miers et al., IEEE S&P 2013) tackles some of these privacy issues by unlinking transactions from the payment’s origin. Yet, it still reveals payments’ destinations and amounts, and is limited in functionality. In this paper, we construct a full-fledged ledger-based digital currency with strong privacy guarantees. Our results leverage recentadvances in zero-knowledge Succinct Non-interactive Arguments of Knowledge (zk- SNARKs) First, we formulate and construct decentralized anonymous payment schemes (DAP schemes). A DAP scheme enables users to directly pay each other privately: the corresponding transaction hides the payment’s origin, destination, and transferred amount. We provide formal definitions and proofs of the construction’s security. Second, we build Zerocash, a practical instantiation of our DAP scheme construction. In Zerocash, transactions are less than 1 kB and take under 6 ms to verify — orders of magnitude more efficient than the less-anonymous Zerocoin and competitive with plain Bitcoin.

### Drawback:

* + - Zerocoin does not rely on digital signatures to validate coins, nor does it require a central bank to prevent double spending. Instead, Zerocoin authenticates coins by proving, in zero- knowledge, that they belong to a public list of valid coins (which can be maintained on the block chain).

**Author:** Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, Madars Virza.

**Published In:** 2014

## The Effect of Online Privacy Information on Purchasing Behavior: An Experimental Study

Although online retailers detail their privacy practices in online privacy policies, this information often remains invisible to consumers, who seldom make the effort to read and understand those policies. This paper reports on research undertaken to determine whether a more prominent display of privacy information will cause consumers to incorporate privacy considerations into their online purchasing decisions. We designed an experiment in which a shopping search engine interface clearly and compactly displays privacy policy information. When such information is made available, consumers tend to purchase from online retailers who better protect their privacy. In fact, our study indicates that when privacy information is made more salient and accessible, some consumers are willing to pay a premium to purchase from privacy protective websites. This result suggests that businesses may be able to leverage privacy protection

### Drawbacks:

* + - After the purchase, the consumer may not know how the merchant will use the personal information she revealed as part of the transaction (Acquisti and Grossklags 2005b). This lack of information arguably affects individual behavior in different ways.

**Author:** Janice Y. Tsai, Serge Egelman, Lorrie Cranor and Alessandro Acquisti

**Publlished In**: 2016

**CHAPTER 3**

**SYSTEM ANALYSIS**

### EXISTING SYSTEM:

Green et al. show how to construct anonymous payment channels between two mutually distrustful parties. Their protocols are instantiated using efficient cryptographic primitives with no trusted third parties. Payments of arbitrary value are conducted directly between parties, or via an intermediate connection who learns neither the participants identities nor the amount involved. Coupled with a decentralized anonymous payment scheme for funding the channels, they provide for private instantaneous anonymous payments without a trusted bank. They use the idea of lightning networks and address the anonymity problem of payment channels based on the zero- knowledge technology. Gallay et al. design a novel platform for decentralized logistics, the aim of which is to magnify and accelerate the impact offered by the integration of the most recent advances in Information and Communication Technologies (ICTs) to multi-modal freight operations. Besides, the platform allows for an implementation that is not affected by scalability issues and is not limited by geographical borders. They also employ the IDS and blockchain technology to construct a decentralized logistics platform.

### DISADVANTAGES OF EXISTING SYSTEM:

* + - However, they did not consider the payment problem in the event of a dispute.
    - The platform allows for an implementation that is not affected by scalability issues and is not limited by geographical borders.

### PROPOSED SYSTEM:

We make the following contributions:

We design a privacy-preserving model for E-commerce based on the private smart contract technology. Our model can provide the proof of ownership while protect users’ private information from other participants. Besides, we build implementations of model using two existing blockchain application platforms. We also evaluate the performance and validate its effectiveness and efficiency of the model with experiments. Performance analysis of the blockchain platforms provides further considerations for deploying a usable implementation. **ADVANTAGES OF PROPOSED SYSTEM:**

* + - We employ a escrow protocol to address the problem of dispute. In addition, our model is based on a permissioned blockchain which in nature has superior performance efficiency to public chains.
    - We further tap the potential of blockchain and expand it to E-commerce systems based on permissioned blockchain. Additionally, we enhance the anonymity of the logistics model to better protect users’ privacy.

### Requirements Analysis and Specification

* + 1. **Input Requirements:**

**1.ANACONDA:**

Anaconda is a distribution of the Python and R programming languages for

scientific computing (data science, machine learning applications, large-scale data

processing, predictive analytics, etc.), that aims to simplify package management

and deployment. The distribution includes data-science packages suitable for

Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc.,

which was founded by Peter Wang and Travis Oliphant in 2012.[8] As an

Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda

Individual Edition, while other products from the company are Anaconda Team

Edition and Anaconda Enterprise Edition, both of which are not free.

**2.** **SQLyog**

SQLyog is a GUI tool for the RDBMS MySQL. **It** is the most powerful manager,

admin and GUI tool for MySQL, combining the features of MySQL Query

Browser, Administrator, phpMyAdmin and other MySQL Front Ends and MySQL

GUI tools in a single intuitive interface.

* + 1. **Output Requirements:**

System with64 bit distribution capable of running 32 bit application and 1200\*800

minimum screen resolution with stable internet connection

* + 1. **Functional Requirements Hardware Requirements**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following sub- sections discuss the various aspects of hardware requirements

### Hardware requirements for present project:

* + - * System : Intel Pentium.
      * Hard Disk : 120 GB.
      * Monitor : 15’’ LED
      * Input Devices : Keyboard, Mouse
      * Ram : 2 GB

### Software Requirements:

Software Requirements deal with defining software resource requirements and pre- requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

### Software requirements for present project:

* Operating system : Windows 7 or more
* Coding Language : Python
* Backend : Anaconda

### Technology Stack

**PYTHON:**

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language), [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [general-purpose](https://en.wikipedia.org/wiki/General-purpose_programming_language) [programming language](https://en.wikipedia.org/wiki/Programming_language). Python is [dynamically typed](https://en.wikipedia.org/wiki/Dynamic_programming_language) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming](https://en.wikipedia.org/wiki/Programming_paradigms) [paradigms](https://en.wikipedia.org/wiki/Programming_paradigms), including [procedural](https://en.wikipedia.org/wiki/Procedural_programming), object-oriented, and [functional programming](https://en.wikipedia.org/wiki/Functional_programming). Python is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library). Python [interpreters](https://en.wikipedia.org/wiki/Interpreter_(computing)) are available for many [operating](https://en.wikipedia.org/wiki/Operating_system) [systems](https://en.wikipedia.org/wiki/Operating_system). A global community of programmers develops and maintains [CPython](https://en.wikipedia.org/wiki/CPython), an [open source](https://en.wikipedia.org/wiki/Open-source_software) [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation). A [non-profit organization](https://en.wikipedia.org/wiki/Nonprofit_organization), the [Python](https://en.wikipedia.org/wiki/Python_Software_Foundation) [Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation), manages and directs resources for Python and CPython development.

* 1. **Software Description**

**PYTHON**

Python is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language). [Object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming)

[programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support [functional programming](https://en.wikipedia.org/wiki/Functional_programming) and [aspect-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming). Many other paradigms are supported via extensions, including [design by contract](https://en.wikipedia.org/wiki/Design_by_contract) and [logic](https://en.wikipedia.org/wiki/Logic_programming) [programming.](https://en.wikipedia.org/wiki/Logic_programming)

Python uses [dynamic typing](https://en.wikipedia.org/wiki/Dynamic_typing), and a combination of [reference counting](https://en.wikipedia.org/wiki/Reference_counting) and a cycle-detecting garbage collector for [memory management](https://en.wikipedia.org/wiki/Memory_management). It also features dynamic [name resolution](https://en.wikipedia.org/wiki/Name_resolution_(programming_languages)) ([late binding](https://en.wikipedia.org/wiki/Late_binding)), which binds method and variable names during program execution.

Python's design offers some support for [functional programming](https://en.wikipedia.org/wiki/Functional_programming) in

the [Lisp](https://en.wikipedia.org/wiki/Lisp_(programming_language)) tradition. It has filter map and functions; [list](https://en.wikipedia.org/wiki/List_comprehension)

reduce

[comprehensions](https://en.wikipedia.org/wiki/List_comprehension), [dictionaries](https://en.wikipedia.org/wiki/Associative_array), sets and [generator](https://en.wikipedia.org/wiki/Generator_(computer_programming)) expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)) and [Standard ML](https://en.wikipedia.org/wiki/Standard_ML).

The language's core philosophy is summarized in the document The [Zen of](https://en.wikipedia.org/wiki/Zen_of_Python) [Python](https://en.wikipedia.org/wiki/Zen_of_Python) (PEP 20), which includes [aphorisms](https://en.wikipedia.org/wiki/Aphorism) such as:

* + - Beautiful is better than ugly.
    - Explicit is better than implicit.
    - Simple is better than complex.
    - Complex is better than complicated.
    - Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly [extensible](https://en.wikipedia.org/wiki/Extensibility). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with [ABC](https://en.wikipedia.org/wiki/ABC_(programming_language)), which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to [Perl'](https://en.wikipedia.org/wiki/Perl)s "[there is more](https://en.wikipedia.org/wiki/There_is_more_than_one_way_to_do_it)

[than one way to do it](https://en.wikipedia.org/wiki/There_is_more_than_one_way_to_do_it)" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" design philosophy. [Alex Martelli,](https://en.wikipedia.org/wiki/Alex_Martelli) a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid [premature optimization](https://en.wikipedia.org/wiki/Premature_optimization), and reject patches to non-critical parts of the [CPython](https://en.wikipedia.org/wiki/CPython) reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use [PyPy,](https://en.wikipedia.org/wiki/PyPy) a [just-in-time compiler](https://en.wikipedia.org/wiki/Just-in-time_compilation). [Cython](https://en.wikipedia.org/wiki/Cython) is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonistas.

### Libraries

Python's large [standard library](https://en.wikipedia.org/wiki/Standard_library), commonly cited as one of its greatest strengths, provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as [MIME](https://en.wikipedia.org/wiki/MIME) and [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) are supported. It includes modules for creating [graphical user interfaces](https://en.wikipedia.org/wiki/Graphical_user_interface), connecting to [relational databases](https://en.wikipedia.org/wiki/Relational_database), [generating](https://en.wikipedia.org/wiki/Pseudorandom_number_generator)

[pseudorandom numbers](https://en.wikipedia.org/wiki/Pseudorandom_number_generator), arithmetic with arbitrary precision decimals, manipulating [regular expressions](https://en.wikipedia.org/wiki/Regular_expression), and [unit testing.](https://en.wikipedia.org/wiki/Unit_testing)

Some parts of the standard library are covered by specifications, but most modules are not. They are specified by their code, internal documentation, and test suites (if supplied). However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.

As of March 2018, the [Python Package Index](https://en.wikipedia.org/wiki/Python_Package_Index) (PyPI), the official repository for third-party Python software, contains over 130,000 packages with a wide range of functionality, including:

* + - Graphical user interfaces
    - Web frameworks
    - Multimedia
    - Databases
    - Networking
    - Test frameworks
    - Automation
    - Web scraping
    - Documentation
    - System administration
    - Scientific computing
    - Text processing
    - Image processing

### Development

Python's development is conducted largely through the Python Enhancement Proposal (PEP) process, the primary mechanism for proposing major new features, collecting community input on issues and documenting Python design decisions. Python coding style is covered in PEP 8. Outstanding PEPs are reviewed

and commented on by the Python community and the steering council.

Enhancement of the language corresponds with development of the CPython reference implementation. The mailing list python-dev is the primary forum for the language's development. Specific issues are discussed in the [Roundup](https://en.wikipedia.org/wiki/Roundup_(issue_tracker)) [bug](https://en.wikipedia.org/wiki/Bug_tracker) [tracker](https://en.wikipedia.org/wiki/Bug_tracker) maintained at python.org. Development originally took place on a [self-](https://en.wikipedia.org/wiki/Self-hosting_(web_services)) [hosted](https://en.wikipedia.org/wiki/Self-hosting_(web_services)) source-code repository running [Mercurial](https://en.wikipedia.org/wiki/Mercurial), until Python moved to [GitHub](https://en.wikipedia.org/wiki/GitHub) in January 2017.

CPython's public releases come in three types, distinguished by which part of the version number is incremented:

* + - Backward-incompatible versions, where code is expected to break and need to be manually [ported](https://en.wikipedia.org/wiki/Ported). The first part of the version number is incremented. These releases happen infrequently for example, version

3.0 was released 8 years after 2.0.

* + - Major or "feature" releases, about every 18 months, are largelycompatible but introduce new features. The second part of the version number is incremented. Each major version is supported by bugfixes for several years after its release.
    - Bugfix releases, which introduce no new features, occur about every 3 months and are made when a sufficient number of bugs have been fixed upstream since the last release. Security vulnerabilities are also patched in these releases. The third and final part of the version number is incremented.

Many [alpha, beta, and release-candidates](https://en.wikipedia.org/wiki/Beta_release) are also released as previews and for testing before final releases. Although there is a rough schedule for each release, they are often delayed if the code is not ready. Python's development team monitors the state of the code by running the large [unit test](https://en.wikipedia.org/wiki/Unit_test) suite during development, and using the [BuildBot](https://en.wikipedia.org/wiki/BuildBot) [continuous integration](https://en.wikipedia.org/wiki/Continuous_integration) system.

The community of Python developers has also contributed over 86,000 software modules (as of 20 August 2016) to the [Python Package Index](https://en.wikipedia.org/wiki/Python_Package_Index) (PyPI),

he official repository of third-party Python libraries.

The major [academic conference](https://en.wikipedia.org/wiki/Academic_conference) on Python is [PyCon](https://en.wikipedia.org/wiki/PyCon). There are also special Python mentoring programmes, such as [Pyladies](https://en.wikipedia.org/wiki/Pyladies).

### CHAPTER 4

### SYSTEM DESIGN

### 4.1Data dictionary

**4.1.1 Product Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **Description** | **Constraints** |
| id | int | 99 | Id for each product | Primary Key |
| sid | Varchar | 9999 | Sender address | Null |
| pkey | Varchar | 9999 | public key for the product | Null |
| product | Varchar | 9999 | Name of the product | Null |
| quantity | Varchar | 88 | Quantity of the product | Null |
| amount | Varchar | 88 | Amount of the product | Null |
| token | Varchar |  | Unique token for every product | Null |

**4.1.2 Register Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **Description** | **Constraints** |
| id | int | 66 | Id for each buyer who register to buy the product | Primary Key |
| name | Varchar | 777 | Name of the buyer | Null |
| mno | Varchar | 777 | Mobile number for the buyer | Null |
| address | Varchar | 777 | Address of the buyer for delivery purpose | Null |
| master | Varchar | 9999 | public key | Null |
| private | Varchar | 9999 | private key | Null |

**4.1.3 Smart Contract Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **Description** | **Constraints** |
| id | int | 88 | Id for each contract | Primary Key |
| sid | Varchar | 7777 | Id for each Sender address | Null |
| key1 | Varchar | 7777 | Sender private key | Null |
| rid | Varchar | 7777 | Recipient address | Null |
| pid | Varchar | 6666 | Product ID | Null |
| status | Varchar | 6666 | Delivery status of the product | Null |

**4.1.4. Trans Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **Description** | **Constraints** |
| id | int | 90 | Id for each transaction | Primary Key |
| sid | Varchar | 9090 | Sender address | Null |
| rid | Varchar | 9090 | Recipient address | Null |
| pid | Varchar | 9090 | ID for each product | Null |

**4.1.5. USD Token Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **Description** | **Constraints** |
| id | int | 66 | Id for each token | Primary Key |
| sid | Varchar | 7777 | Id for all the Sender address | Null |
| pkey | Varchar | 7777 | Public key for all the product | Null |
| amount | Varchar | 7777 | Amount of the product | Null |
| token | Varchar | 7777 | Token for all the product | Null |

**4.2.UML Diagrams**

UML stands for **U**nified **M**odeling **L**anguage. 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.

### 4.2.1.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.

**Negotiation phase**



### Delivery Phase



**4.2.2.COLLABORATION DIAGRAM**

### Negotiation Phase

4: instruct to pay

Alice

Negotation phase

1: Create Contract

5: instruct to send cards

Bob

2: Shield USD token

3: establish private contract-private

### Delivery Phase

3: Trade is finished and send key

2: Change the status of the box

Bob

Delivery Phase

5: fetch the ownership of the card token

4: fetch the ownership of the USD tokens 1: Send box count

Alice

### 4.2.3.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.

### Negotiation Phase

Alice

Logistics center

Trading chain

payment chain

Private Contract

logistic chain

Bob

* + - 1. Create a contract

6. send proof to USD n...

* + - 1. Shield USD token

8. verify payment

7.provide private contract evidence

3.establish private contract-private

* + - * 1. Instruct place token into escrow

5. instruct to pay

4.accept terms

* + - * 1. transfer card token
        2. provide evidence to transfer
        3. verify the transfer
        4. instruct to send cards

### Delivery Phase

Alice

Logistic center

private contract

Trading chain

payment chain

logistic chain

Bob

* + - * 1. send the box cent

er

* + - * 1. send the number of the box

16.send the num

ber to bob

nquires the box status

18. i

e logistic info

17.record th

shed

dicate the trade fini

21.in

1. fetch cards
2. change the status of the box
   1. fetch the ownership of the USD tokens

y

22.send the ke

* 1. fetch the ownership of the card token

### 4.2.4.ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by- step workflows of components in a system. An activity diagram shows the overall flow of control.

### Negotiation Phase



New Contract Creation

Shield USD token

Instruct the to pay

Send Proof to USD contract with evidence

Payment Verification

Transfer the card token with evidence

Instruct to send the card

**Delivery Phase**



Send the box Count

Inquire the Box status

Trade Finished

Send the Key

fetch the ownership of the USD tokens

fetch the ownership of the card token

**4.2.5STATE CHART DIAGRAM**



Create a new contract

Shield the USD Token

Establish Private contract

Use escrow protocol for payment

Instruct to send Card

Send box count and its status

Inquire the box and fetch the card

Indicate trade finished

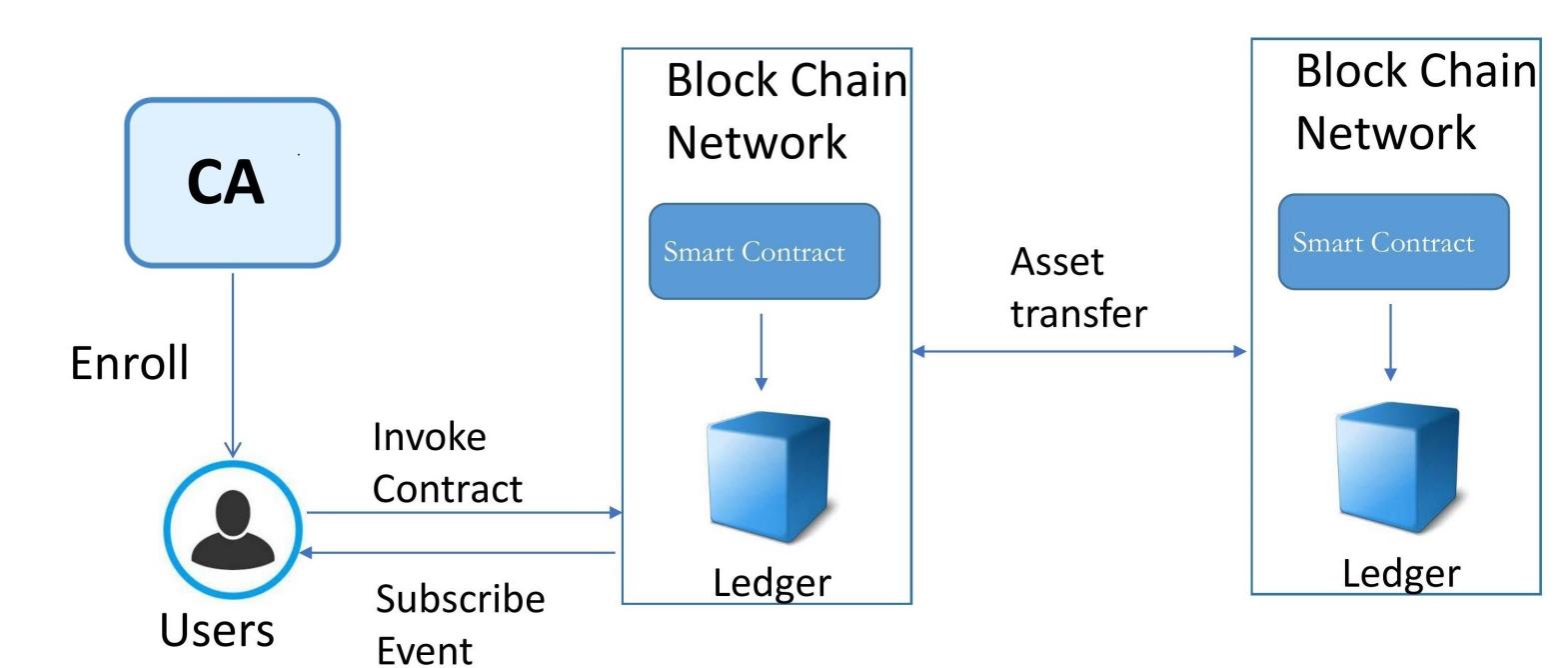
Send the key

Fetch the ownership of USD and Card Token

**CHAPTER 5**

**SYSTEM ARCHITECTURE**

* 1. **Architecture Overview:**



**Figure 5.1.1 Architecture Overview**

**Modules in the project**

The entire project mainly consists of 4 modules, which are

* + - Central Authority
    - Token Creation
    - Negotiation phase
    - Delivery phase

## Central Authority:

The CA component is in charge of issuing PKI-based certificates to organization members and their users. A root certificate (rootCert) is issued to each member and one enrollment certificate (ECert) to each authorized user. The Client interacts with the blockchain network and smart contracts. It has to obtain a valid identity certificate from CA before joining the application channel/chains in the network. Both users and intelligent logistics centers act as clients.

## Token Creation:

Thus, Alice has to create a note-contract for the set of collector cards and issue a shielded token for herself . Now Alice owns a CARD token. At a same time, Bob shields some USD tokens by the USD note- contract.

## Negotiation phase:

Alice establishes a private contract with Bob in a private channel . The private contract specifies the trade of the cards at a specific price in USD between Alice and Bob. The private contract also refers to the cards and USD note- contracts. Besides, the private contract also receives the rel- evant public keys and payment addresses of the two parties (including the Hash of physical addresses). When Alice initializes the contract, Bob can send to the private contract a transaction indicating acceptance of the terms We assume that the USD must be paid first. After the private contract receives the confirmation transaction, the private contract issues an instruction to Bob to pay the relevant amount of USD to Alice (Bob places the USD tokens into the escrow.USD tokens to a mediator’s payment address by generating the necessary zk-SNARK proof and sends it to the USD note-contract(Miner). **Delivery phase:**

Alice places the cards into a delivery box which has a unique number.Then Alice sends the delivery box to the intelligent logistics center (suppose the intelligent center is reliable). Transport companies are responsible for inspecting and monitoring the legitimacy of the items. Alice’s client sends the Hash of Bob’s address to this box and sends the number of the box to the private contract. The private contract then sends the number of the delivery box to Bob’s client.

**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

* 1. **Client-side coding**

**Blockchain.py**

import OrderedDict

import binascii

import Crypto

import Crypto.Random

from Crypto.Hash import SHA

from Crypto.PublicKey import RSA

from Crypto.Signature import PKCS1\_v1\_5

import hashlib

import json

from time import time

from urllib.parse import urlparse

from uuid import uuid4

import requests

from flask import Flask, jsonify, request, render\_template

from flask\_cors import CORS

MINING\_SENDER = "THE BLOCKCHAIN"

MINING\_REWARD = 1

MINING\_DIFFICULTY = 2

class Blockchain:

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

self.transactions = []

self.chain = []

self.nodes = set()

#Generate random number to be used as node\_id

self.node\_id = str(uuid4()).replace('-', '')

#Create genesis block

self.create\_block(0, '00')

def register\_node(self, node\_url):

"""

Add a new node to the list of nodes

"""

#Checking node\_url has valid format

parsed\_url = urlparse(node\_url)

if parsed\_url.netloc:

self.nodes.add(parsed\_url.netloc)

elif parsed\_url.path:

# Accepts an URL without scheme like '192.168.0.5:5000'.

self.nodes.add(parsed\_url.path)

else:

raise ValueError('Invalid URL')

def verify\_transaction\_signature(self, sender\_address, signature, transaction):

"""

Check that the provided signature corresponds to transaction

signed by the public key (sender\_address)

"""

public\_key = RSA.importKey(binascii.unhexlify(sender\_address))

verifier = PKCS1\_v1\_5.new(public\_key)

h = SHA.new(str(transaction).encode('utf8'))

return verifier.verify(h, binascii.unhexlify(signature))

def submit\_transaction(self, sender\_address, recipient\_address, value, signature):

"""

Add a transaction to transactions array if the signature verified

"""

transaction = OrderedDict({'sender\_address': sender\_address,

'recipient\_address': recipient\_address,

'value': value})

#Reward for mining a block

if sender\_address == MINING\_SENDER:

self.transactions.append(transaction)

return len(self.chain) + 1

#Manages transactions from wallet to another wallet

else:

transaction\_verification = self.verify\_transaction\_signature(sender\_address, signature, transaction)

if transaction\_verification:

self.transactions.append(transaction)

return len(self.chain) + 1

else:

return False

def create\_block(self, nonce, previous\_hash):

"""

Add a block of transactions to the blockchain

"""

block = {'block\_number': len(self.chain) + 1,

'timestamp': time(),

'transactions': self.transactions,

'nonce': nonce,

'previous\_hash': previous\_hash}

# Reset the current list of transactions

self.transactions = []

self.chain.append(block)

return block

def hash(self, block):

"""

Create a SHA-256 hash of a block

"""

# We must make sure that the Dictionary is Ordered, or we'll have inconsistent hashes

block\_string = json.dumps(block, sort\_keys=True).encode()

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

def proof\_of\_work(self):

"""

Proof of work algorithm

"""

last\_block = self.chain[-1]

last\_hash = self.hash(last\_block)

nonce = 0

while self.valid\_proof(self.transactions, last\_hash, nonce) is False:

nonce += 1

return nonce

def valid\_proof(self, transactions, last\_hash, nonce, difficulty=MINING\_DIFFICULTY):

"""

Check if a hash value satisfies the mining conditions. This function is used within the proof\_of\_work function.

"""

guess = (str(transactions)+str(last\_hash)+str(nonce)).encode()

guess\_hash = hashlib.sha256(guess).hexdigest()

return guess\_hash[:difficulty] == '0'\*difficulty

def valid\_chain(self, chain):

"""

check if a bockchain is valid

"""

last\_block = chain[0]

current\_index = 1

while current\_index < len(chain):

block = chain[current\_index]

#print(last\_block)

#print(block)

#print("\n-----------\n")

# Check that the hash of the block is correct

if block['previous\_hash'] != self.hash(last\_block):

return False

# Check that the Proof of Work is correct

#Delete the reward transaction

transactions = block['transactions'][:-1]

# Need to make sure that the dictionary is ordered. Otherwise we'll get a different hash

transaction\_elements = ['sender\_address', 'recipient\_address', 'value']

transactions = [OrderedDict((k, transaction[k]) for k in transaction\_elements) for transaction in transactions]

if not self.valid\_proof(transactions, block['previous\_hash'], block['nonce'], MINING\_DIFFICULTY):

return False

last\_block = block

current\_index += 1

return True

def resolve\_conflicts(self):

"""

Resolve conflicts between blockchain's nodes

by replacing our chain with the longest one in the network.

"""

neighbours = self.nodes

new\_chain = None

# We're only looking for chains longer than ours

max\_length = len(self.chain)

# Grab and verify the chains from all the nodes in our network

for node in neighbours:

print('http://' + node + '/chain')

response = requests.get('http://' + node + '/chain')

if response.status\_code == 200:

length = response.json()['length']

chain = response.json()['chain']

# Check if the length is longer and the chain is valid

if length > max\_length and self.valid\_chain(chain):

max\_length = length

new\_chain = chain

# Replace our chain if we discovered a new, valid chain longer than ours

if new\_chain:

self.chain = new\_chain

return True

return False

# Instantiate the Node

app = Flask(\_\_name\_\_)

CORS(app)

# Instantiate the Blockchain

blockchain = Blockchain()

@app.route('/')

def index():

return render\_template('./index.html')

@app.route('/configure')

def configure():

return render\_template('./configure.html')

@app.route('/transactions/new', methods=['POST'])

def new\_transaction():

values = request.form

# Check that the required fields are in the POST'ed data

required = ['sender\_address', 'recipient\_address', 'amount', 'signature']

if not all(k in values for k in required):

return 'Missing values', 400

# Create a new Transaction

transaction\_result = blockchain.submit\_transaction(values['sender\_address'], values['recipient\_address'], values['amount'], values['signature'])

if transaction\_result == False:

response = {'message': 'Invalid Transaction!'}

return jsonify(response), 406

else:

response = {'message': 'Transaction will be added to Block '+ str(transaction\_result)}

return jsonify(response), 201

@app.route('/transactions/get', methods=['GET'])

def get\_transactions():

#Get transactions from transactions pool

transactions = blockchain.transactions

response = {'transactions': transactions}

return jsonify(response), 200

@app.route('/chain', methods=['GET'])

def full\_chain():

response = {

'chain': blockchain.chain,

'length': len(blockchain.chain),

}

return jsonify(response), 200

@app.route('/mine', methods=['GET'])

def mine():

# We run the proof of work algorithm to get the next proof...

last\_block = blockchain.chain[-1]

nonce = blockchain.proof\_of\_work()

# We must receive a reward for finding the proof.

blockchain.submit\_transaction(sender\_address=MINING\_SENDER, recipient\_address=blockchain.node\_id, value=MINING\_REWARD, signature="")

# Forge the new Block by adding it to the chain

previous\_hash = blockchain.hash(last\_block)

block = blockchain.create\_block(nonce, previous\_hash)

response = {

'message': "New Block Forged",

'block\_number': block['block\_number'],

'transactions': block['transactions'],

'nonce': block['nonce'],

'previous\_hash': block['previous\_hash'],

}

return jsonify(response), 200

@app.route('/nodes/register', methods=['POST'])

def register\_nodes():

values = request.form

nodes = values.get('nodes').replace(" ", "").split(',')

if nodes is None:

return "Error: Please supply a valid list of nodes", 400

for node in nodes:

blockchain.register\_node(node)

response = {

'message': 'New nodes have been added',

'total\_nodes': [node for node in blockchain.nodes],

}

return jsonify(response), 201

@app.route('/nodes/resolve', methods=['GET'])

def consensus():

replaced = blockchain.resolve\_conflicts()

if replaced:

response = {

'message': 'Our chain was replaced',

'new\_chain': blockchain.chain

}

else:

response = {

'message': 'Our chain is authoritative',

'chain': blockchain.chain

}

return jsonify(response), 200

@app.route('/nodes/get', methods=['GET'])

def get\_nodes():

nodes = list(blockchain.nodes)

response = {'nodes': nodes}

return jsonify(response), 200

if \_\_name\_\_ == '\_\_main\_\_':

from argparse import ArgumentParser

parser = ArgumentParser()

parser.add\_argument('-p', '--port', default=5000, type=int, help='port to listen on')

args = parser.parse\_args()

port = args.port

app.run(host='127.0.0.1', port=port)

**Blockchain\_client.py**

from collections import OrderedDict

import binascii

import functools

import Crypto

import Crypto.Random

from Crypto.Hash import SHA

from Crypto.PublicKey import RSA

from Crypto.Signature import PKCS1\_v1\_5

from dbconnect import connection

import requests

from flask import Flask, jsonify, request, render\_template

class Transaction:

def \_\_init\_\_(self, sender\_address, sender\_private\_key, recipient\_address, value):

self.sender\_address = sender\_address

self.sender\_private\_key = sender\_private\_key

self.recipient\_address = recipient\_address

self.value = value

def \_\_getattr\_\_(self, attr):

return self.data[attr]

def to\_dict(self):

return OrderedDict({'sender\_address': self.sender\_address,

'recipient\_address': self.recipient\_address,

'value': self.value})

def sign\_transaction(self):

"""

Sign transaction with private key

"""

private\_key = RSA.importKey(binascii.unhexlify(self.sender\_private\_key))

signer = PKCS1\_v1\_5.new(private\_key)

h = SHA.new(str(self.to\_dict()).encode('utf8'))

return binascii.hexlify(signer.sign(h)).decode('ascii')

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return render\_template('./index.html')

@app.route('/make/transaction')

def make\_transaction():

return render\_template('./make\_transaction.html')

@app.route('/make/transaction1')

def make\_transaction1():

return render\_template('./make\_transaction1.html')

@app.route('/make/transaction2')

def make\_transaction2():

return render\_template('./make\_transaction2.html')

@app.route('/contract/private')

def make\_transaction3():

return render\_template('./privatecontract.html')

@app.route('/view/transactions')

def view\_transaction():

return render\_template('./view\_transactions.html')

@app.route('/wallet/new', methods=['GET'])

def new\_wallet():

random\_gen = Crypto.Random.new().read

private\_key = RSA.generate(1024, random\_gen)

public\_key = private\_key.publickey()

response = {

'private\_key': binascii.hexlify(private\_key.exportKey(format='DER')).decode('ascii'),

'public\_key': binascii.hexlify(public\_key.exportKey(format='DER')).decode('ascii')

}

return jsonify(response), 200

@app.route('/generate/transaction', methods=['POST'])

def generate\_transaction():

sender\_address = request.form['sender\_address']

sender\_private\_key = request.form['sender\_private\_key']

recipient\_address = request.form['recipient\_address']

value = request.form['amount']

pid = request.form['pid']

c, conn = connection()

sql='select amount from product where id="%s"'% \

(pid)

c.execute(sql);

amount=0;

result=c.fetchall();

count=0;

for cc in result:

count=1

amount=cc

if(count==1):

sql='select amount from usdtoken where sid="%s"'% \

(sender\_address)

c.execute(sql);

result1=c.fetchall();

count=0;

amm=0

count=0

for cc1 in result1:

count=1

amm=cc1

sql='select amount from usdtoken where sid="%s"'% \

(recipient\_address)

c.execute(sql);

result1=c.fetchall();

count=0;

amm1=0

count1=0

for cc1 in result1:

count1=1

amm1=cc1

if(count==1):

res = functools.reduce(lambda sub, ele: sub \* 10 + ele, amm)

res1 = functools.reduce(lambda sub, ele: sub \* 10 + ele, amount)

res2 = functools.reduce(lambda sub, ele: sub \* 10 + ele, amm1)

result=int(res1)-int(res)

result2=int(res2)+int(res)

sql1='update usdtoken set amount="%s" where sid="%s"' % \

(result,sender\_address)

c.execute(sql1)

conn.commit()

sql1='update usdtoken set amount="%s" where sid="%s"' % \

(result2,recipient\_address)

c.execute(sql1)

conn.commit()

sql1='insert into trans(sid,rid,pid) values("%s", "%s","%s")' % \

(sender\_address,recipient\_address,pid)

c.execute(sql1)

conn.commit()

conn.close()

value=res+" "+pid

transaction = Transaction(sender\_address, sender\_private\_key, recipient\_address, value)

response = {'transaction': transaction.to\_dict(), 'signature': transaction.sign\_transaction(),'pid': pid}

return jsonify(response), 200

@app.route('/contract/smart', methods=['POST'])

def smart\_contract():

sender\_address = request.form['sender\_address']

sender\_private\_key = request.form['sender\_private\_key']

recipient\_address = request.form['recipient\_address']

value = request.form['amount']

c, conn = connection()

sql1='insert into smartcontract(sid,key1,rid,pid,status) values("%s", "%s","%s","%s","%s")' % \

(sender\_address,sender\_private\_key,recipient\_address,value,"Deliverd")

c.execute(sql1)

conn.commit()

conn.close()

value=value+" Deliverd";

transaction = Transaction(sender\_address, sender\_private\_key, recipient\_address, value)

response = {'transaction': transaction.to\_dict(), 'signature': transaction.sign\_transaction()}

return jsonify(response), 200

@app.route('/generate/transaction1', methods=['POST'])

def generate\_transaction1():

sender\_address = request.form['sender\_address']

sender\_private\_key = request.form['sender\_private\_key']

product = request.form['name']

quantity = request.form['quan']

amount = request.form['amount']

value=product+","+str(quantity)+","+str(amount)

transaction = Transaction(sender\_address, sender\_private\_key, sender\_address, value)

c, conn = connection()

sql='select \* from register where master="%s" AND private="%s"'% \

(sender\_address,sender\_private\_key)

c.execute(sql);

result=c.fetchall();

count=0;

for cc in result:

count=1

if(count==1):

sql1='insert into product(sid,pkey,product,quantity,amount,token) values("%s", "%s","%s","%s","%s","%s")' % \

(sender\_address,sender\_private\_key,product,quantity,amount,transaction.sign\_transaction())

print(sql1)

c.execute(sql1)

conn.commit()

conn.close()

response = {'transaction': transaction.to\_dict(), 'signature': transaction.sign\_transaction()}

return jsonify(response), 200

@app.route('/generate/useradding', methods=['POST'])

def useradding():

random\_gen = Crypto.Random.new().read

private\_key = RSA.generate(1024, random\_gen)

public\_key = private\_key.publickey()

name = request.form['name']

mno = request.form['mno']

address = request.form['address']

c, conn = connection()

pri=binascii.hexlify(private\_key.exportKey(format='DER')).decode('ascii')

pub=binascii.hexlify(public\_key.exportKey(format='DER')).decode('ascii')

sql1='insert into register(name, mno,address,master,private) values("%s", "%s","%s","%s","%s")' % \

(name,mno,address,pub,pri)

c.execute(sql1)

conn.commit()

conn.close()

response = {

'private\_key': binascii.hexlify(private\_key.exportKey(format='DER')).decode('ascii'),

'public\_key': binascii.hexlify(public\_key.exportKey(format='DER')).decode('ascii')

}

return jsonify(response), 200

@app.route('/generate/transaction2', methods=['POST'])

def generate\_transaction2():

sender\_address = request.form['sender\_address']

sender\_private\_key = request.form['sender\_private\_key']

recipient\_address = sender\_address

value = request.form['amount']

transaction = Transaction(sender\_address, sender\_private\_key, recipient\_address, value)

c, conn = connection()

sql='select \* from register where master="%s" and private="%s"'% \

(sender\_address,sender\_private\_key)

c.execute(sql);

result=c.fetchall();

count=0;

for cc in result:

count=1

if(count==1):

sql='select amount from usdtoken where sid="%s" and pkey="%s"'% \

(sender\_address,sender\_private\_key)

c.execute(sql);

result1=c.fetchall();

count=0;

amm=0

for cc1 in result1:

count=1

amm=cc1

if(count==1):

res = functools.reduce(lambda sub, ele: sub \* 10 + ele, amm)

amount=int(res)+int(value)

sql1='insert into usdtoken(sid,pkey,amount,token) values("%s", "%s","%s","%s")' % \

(sender\_address,sender\_private\_key,amount,transaction.sign\_transaction())

c.execute(sql1)

conn.commit()

conn.close()

else:

sql1='insert into usdtoken(sid,pkey,amount,token) values("%s", "%s","%s","%s")' % \

(sender\_address,sender\_private\_key,value,transaction.sign\_transaction())

c.execute(sql1)

conn.commit()

conn.close()

response = {'transaction': transaction.to\_dict(), 'signature': transaction.sign\_transaction()}

return jsonify(response), 200

if \_\_name\_\_ == '\_\_main\_\_':

from argparse import ArgumentParser

parser = ArgumentParser()

parser.add\_argument('-p', '--port', default=8080, type=int, help='port to listen on')

args = parser.parse\_args()

port = args.port

app.run(host='127.0.0.1', port=port)

* 1. **Server-side coding**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>Blockchain Client</title>

<!-- Bootstrap core CSS -->

<link href="/static/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<link href="/static/css/custom.css" rel="stylesheet">

</head>

<body>

<!-- Navigation -->

<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">

<div class="container">

<a class="navbar-brand" href="#">Blockchain Client</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarResponsive">

<ul class="navbar-nav ml-auto">

<li class="nav-item active">

<a class="nav-link" href="/">Key Generator

<span class="sr-only">(current)</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/make/transaction1">Node Contract</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/make/transaction2">USD Node Contract</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/make/transaction">Payment Transaction</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/contract/private">Product Delivery</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/view/transactions">View Transaction</a>

</li>

</ul>

</div>

</div>

</nav>

<div class="container">

<div class="row">

<div class="col-lg-12 text-center">

<div class="card-body">

<h4 class="card-title"> PKI Generator</h4>

<p class="card-text">To generate Key for User</p>

</div>

</div>

</div>

</div>

<br>

<div class="container">

<form id="transaction\_form">

<div class="row">

<label class="col-sm-2">Master Key:</label>

<div class="col-sm-10">

<textarea id="public\_key" name="public\_key" rows="5" class="form-control" readonly></textarea>

</div>

</div>

<br>

<div class="row">

<label class="col-sm-2">Enrollment Key:</label>

<div class="col-sm-10">

<textarea id="private\_key" name="private\_key" rows="5" class="form-control" readonly></textarea>

</div>

</div>

<br>

<div class="row">

<label class="col-sm-2">Name:</label>

<div class="col-sm-10">

<input type="text" name="name" id="name" rows="2" class="form-control" required/>

</div>

</div>

<br>

<div class="row">

<label class="col-sm-2">Mobile No:</label>

<div class="col-sm-10">

<input type="tel" name="mno" id="mno" rows="2" class="form-control" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}" required/>

</div>

</div>

<br>

<div class="row">

<label class="col-sm-2">Address:</label>

<div class="col-sm-10">

<input type="text" name="address" id="address" rows="2" class="form-control" required/>

</div>

</div>

<br>

<div class="row">

<div class="col-lg-12 text-center">

<input type="button" id="generate\_transaction" class="btn btn-primary btn-lg" value="Central Authority" />

</div>

</div>

<br>

</form>

<div class="row" id="warning" style="display:none">

<div class="col-sm-12">

<div class="alert alert-danger" role="alert">

IMPORTANT

<ul>

<li>Save you private and public keys. These keys cannot be recovered!</li>

<li>Don't share your private key with anyone!</li>

</ul>

</div>

</div>

</div>

</div>

<!-- Bootstrap core JavaScript -->

<script src="/static/vendor/jquery/jquery.min.js"></script>

<script src="/static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<script>

$(function(){

$("#generate\_transaction1").click(function(){

$.ajax({

url: '/wallet/new',

type: 'GET',

success: function(response){

document.getElementById("private\_key").innerHTML = response['private\_key'];

document.getElementById("public\_key").innerHTML = response['public\_key'];

document.getElementById("warning").style.display = "block";

},

error: function(error){

console.log(error);

}

});

});

})

$(function () {

$("#generate\_transaction").click(function () {

$.ajax({

url: "/generate/useradding",

type: "POST",

dataType : 'json',

data: $('#transaction\_form').serialize(),

success: function(response){

alert("Sucessfully Created");

document.getElementById("private\_key").innerHTML = response['private\_key'];

document.getElementById("public\_key").innerHTML = response['public\_key'];

document.getElementById("warning").style.display = "block";

},

error: function(error){

console.log(error);

}

});

});

});

</script>

</body>

</html>

**make\_transaction.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>Blockchain Client</title>

<!-- Bootstrap core CSS -->

<link href="/static/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<link href="/static/css/custom.css" rel="stylesheet">

</head>

<body>

<!-- Navigation -->

<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">

<div class="container">

<a class="navbar-brand" href="#">Blockchain Client</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarResponsive">

<ul class="navbar-nav ml-auto">

<li class="nav-item active">

<a class="nav-link" href="/">Key Generator

<span class="sr-only">(current)</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/make/transaction1">Node Contract</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/make/transaction2">USD Node Contract</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/make/transaction">Payment Transaction</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/contract/private">Product Delivery</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/view/transactions">View Transaction</a>

</li>

</ul>

</div>

</div>

</nav>

<div class="container">

<div class="row">

<div class="col-lg-12">

<div class="card-body">

<h4 class="card-title">Send Coins</h4>

<p class="card-text">Enter transaction details and click on "Generate Transaction" button to generate your transaction</p>

</div>

</div>

</div>

</div>

<br>

<div class="container alert alert-secondary">

<form id="transaction\_form">

<div class="row">

<label class="col-sm-2">Sender Address:</label>

<div class="col-sm-10">

<input type="text" name="sender\_address" id="sender\_address" rows="2" class="form-control">

</div>

</div>

<br>

<div class="row">

<label class="col-sm-2">Sender Private Key:</label>

<div class="col-sm-10">

<input type="text" name="sender\_private\_key" id="sender\_private\_key" rows="2" class="form-control">

</div>

</div>

<br>

<div class="row">

<label class="col-sm-2">Recipient Address:</label>

<div class="col-sm-10">

<input type="text" name="recipient\_address" id="recipient\_address" rows="2" class="form-control">

</div>

</div>

<br>

<div class="row">

<label class="col-sm-2">Product ID:</label>

<div class="col-sm-10">

<input type="text" name="pid" id="pid" rows="2" class="form-control">

</div>

</div>

<br>

<div class="row">

<label class="col-sm-2">Amount to Send:</label>

<div class="col-sm-10">

<input type="text" name="amount" id="amount" rows="2" class="form-control">

</div>

</div>

<br>

<div class="row">

<div class="col-lg-12 text-center">

<input type="button" id="generate\_transaction" class="btn btn-primary btn-lg" value="Generate Transaction">

</div>

</div>

<br>

</form>

</div>

<!-- Modal -->

<div class="modal modal-alert fade" id="basicModal" tabindex="-1" role="dialog" aria-labelledby="basicModal" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<div class="modal-title col-md-10">Confirm transaction details, enter a blockchain node url and click on "Confirm Transaction" to finalize your transaction.</div>

<button type="button" class="close col-md-2" data-dismiss="modal" aria-hidden="true">&times;</button>

</div>

<div class="modal-body">

<form id="confirmation\_transaction\_form">

<div class="row">

<label class="col-sm-12">Sender Address:</label>

<div class="col-sm-12">

<input type="text" name="sender\_address" id="confirmation\_sender\_address" rows="2" class="form-control" readonly>

</div>

</div>

<div class="row">

<label class="col-sm-12">Recipient Address:</label>

<div class="col-sm-12">

<input type="text" name="recipient\_address" id="confirmation\_recipient\_address" rows="2" class="form-control" readonly>

</div>

</div>

<div class="row">

<label class="col-sm-12">Amount to Send:</label>

<div class="col-sm-12">

<input type="text" name="amount" id="confirmation\_amount" rows="2" class="form-control" readonly>

</div>

</div>

<div class="row">

<label class="col-sm-12">Product ID:</label>

<div class="col-sm-12">

<input type="text" name="pid" id="confirmation\_pid" rows="2" class="form-control" readonly>

</div>

</div>

<div class="row">

<label class="col-sm-12">Transaction Signature:</label>

<div class="col-sm-12">

<input type="text" name="signature" id="transaction\_signature" rows="2" class="form-control" readonly>

</div>

</div>

</form>

<div class="row">

<label class="col-sm-12">Blockchain Node URL:</label>

<div class="col-sm-12">

<input type="text" name="node\_url" id="node\_url" rows="2" class="form-control" value="http://127.0.0.1:5004">

</div>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-danger" data-dismiss="modal">Cancel</button>

<button type="button" id="button\_confirm\_transaction" class="btn btn-success">Confirm Transaction</button>

</div>

</div>

</div>

</div>

<!-- Alert Message for successful transaction -->

<div class="modal modal-alert fade" id="success\_transaction\_modal" tabindex="-1" role="dialog" aria-labelledby="basicModal" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>

</div>

<div class="modal-body">

<div class="alert alert-success" role="alert">

<h4 class="alert-heading">Successful Transaction!</h4>

<p>You successfully completed your transaction. It will be added to the next block.</p>

</div>

</div>

<div class="modal-footer">

<button type="button" id="button\_confirm\_transaction" class="btn btn-success" data-dismiss="modal">OK</button>

</div>

</div>

</div>

</div>

<!-- Bootstrap core JavaScript -->

<script src="/static/vendor/jquery/jquery.min.js"></script>

<script src="/static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<script>

$(function () {

$("#generate\_transaction").click(function () {

$.ajax({

url: "/generate/transaction",

type: "POST",

dataType : 'json',

data: $('#transaction\_form').serialize(),

success: function(response){

document.getElementById("confirmation\_sender\_address").value = response["transaction"]["sender\_address"];

document.getElementById("confirmation\_recipient\_address").value = response["transaction"]["recipient\_address"];

document.getElementById("confirmation\_amount").value = response["transaction"]["value"];

document.getElementById("confirmation\_pid").value = response["pid"];

document.getElementById("transaction\_signature").value = response["signature"];

$("#basicModal").modal('show');

},

error: function(error){

console.log(error);

}

});

});

});

$(function () {

$("#button\_confirm\_transaction").click(function () {

//console.log($('#confirmation\_transaction\_form').serialize());

$.ajax({

url: document.getElementById("node\_url").value + "/transactions/new",

type: "POST",

headers: {'Access-Control-Allow-Origin':'\*'},

dataType : 'json',

data: $('#confirmation\_transaction\_form').serialize(),

success: function(response){

//reset both forms

$("#transaction\_form")[0].reset();

$("#confirmation\_transaction\_form")[0].reset();

//clean text boxes

$("#sender\_address").val("");

$("#sender\_private\_key").val("");

$("#recipient\_address").val("");

$("#amount").val("");

$("#pid").val("");

$("#basicModal").modal('hide');

$("#success\_transaction\_modal").modal('show');

},

error: function(error){

console.log(error);

}

});

});

});

</script>

</body>

</html>

**make\_transaction1.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>Blockchain Client</title>

<!-- Bootstrap core CSS -->

<link href="/static/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<link href="/static/css/custom.css" rel="stylesheet">

</head>

<body>

<!-- Navigation -->

<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">

<div class="container">

<a class="navbar-brand" href="#">Blockchain Client</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarResponsive">

<ul class="navbar-nav ml-auto">

<li class="nav-item active">

<a class="nav-link" href="/">Key Generator

<span class="sr-only">(current)</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/make/transaction1">Node Contract</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/make/transaction2">USD Node Contract</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/make/transaction">Payment Transaction</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/view/transactions">Transactions</a>

</li>

</ul>

</div>

</div>

</nav>

<div class="container">

<div class="row">

<div class="col-lg-12">

<div class="card-body">

<h4 class="card-title">Add Product Goods</h4>

<p class="card-text">Enter Product details and click on "Generate Ledger" button to generate your Ledger Record</p>

</div>

</div>

</div>

</div>

<br>

<div class="container alert alert-secondary">

<form id="transaction\_form">

<div class="row">

<label class="col-sm-2">Sender Address:</label>

<div class="col-sm-10">

<input type="text" name="sender\_address" id="sender\_address" rows="2" class="form-control">

</div>

</div>

<br>

<div class="row">

<label class="col-sm-2">Sender Private Key:</label>

<div class="col-sm-10">

<input type="text" name="sender\_private\_key" id="sender\_private\_key" rows="2" class="form-control">

</div>

</div>

<br>

<div class="row">

<label class="col-sm-2">Product name:</label>

<div class="col-sm-10">

<input type="text" name="name" id="name" rows="2" class="form-control">

</div>

</div>

<br>

<div class="row">

<label class="col-sm-2">Quantity:</label>

<div class="col-sm-10">

<input type="text" name="quan" id="quan" rows="2" class="form-control">

</div>

</div>

<br>

<div class="row">

<label class="col-sm-2">Amount:</label>

<div class="col-sm-10">

<input type="text" name="amount" id="amount" rows="2" class="form-control">

</div>

</div>

<br>

<div class="row">

<div class="col-lg-12 text-center">

<input type="button" id="generate\_transaction" class="btn btn-primary btn-lg" value="Generate Ledger">

</div>

</div>

<br>

</form>

</div>

<!-- Modal -->

<div class="modal modal-alert fade" id="basicModal" tabindex="-1" role="dialog" aria-labelledby="basicModal" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<div class="modal-title col-md-10">Confirm transaction details, enter a blockchain node url and click on "Confirm Transaction" to finalize your transaction.</div>

<button type="button" class="close col-md-2" data-dismiss="modal" aria-hidden="true">&times;</button>

</div>

<div class="modal-body">

<form id="confirmation\_transaction\_form">

<div class="row">

<label class="col-sm-12">Sender Address:</label>

<div class="col-sm-12">

<input type="text" name="sender\_address" id="confirmation\_sender\_address" rows="2" class="form-control" readonly>

</div>

</div>

<div class="row">

<label class="col-sm-12">Recipient Address:</label>

<div class="col-sm-12">

<input type="text" name="recipient\_address" id="confirmation\_recipient\_address" rows="2" class="form-control" readonly>

</div>

</div>

<div class="row">

<label class="col-sm-12">Amount to Send:</label>

<div class="col-sm-12">

<input type="text" name="amount" id="confirmation\_amount" rows="2" class="form-control" readonly>

</div>

</div>

<div class="row">

<label class="col-sm-12">Transaction Signature:</label>

<div class="col-sm-12">

<input type="text" name="signature" id="transaction\_signature" rows="2" class="form-control" readonly>

</div>

</div>

</form>

<div class="row">

<label class="col-sm-12">Blockchain Node URL:</label>

<div class="col-sm-12">

<input type="text" name="node\_url" id="node\_url" rows="2" class="form-control" value="http://127.0.0.1:5001">

</div>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-danger" data-dismiss="modal">Cancel</button>

<button type="button" id="button\_confirm\_transaction" class="btn btn-success">Confirm Transaction</button>

</div>

</div>

</div>

</div>

<!-- Alert Message for successful transaction -->

<div class="modal modal-alert fade" id="success\_transaction\_modal" tabindex="-1" role="dialog" aria-labelledby="basicModal" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>

</div>

<div class="modal-body">

<div class="alert alert-success" role="alert">

<h4 class="alert-heading">Successful Transaction!</h4>

<p>You successfully completed your transaction. It will be added to the next block.</p>

</div>

</div>

<div class="modal-footer">

<button type="button" id="button\_confirm\_transaction" class="btn btn-success" data-dismiss="modal">OK</button>

</div>

</div>

</div>

</div>

<!-- Bootstrap core JavaScript -->

<script src="/static/vendor/jquery/jquery.min.js"></script>

<script src="/static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<script>

$(function () {

$("#generate\_transaction").click(function () {

$.ajax({

url: "/generate/transaction1",

type: "POST",

dataType : 'json',

data: $('#transaction\_form').serialize(),

success: function(response){

document.getElementById("confirmation\_sender\_address").value = response["transaction"]["sender\_address"];

document.getElementById("confirmation\_recipient\_address").value = response["transaction"]["recipient\_address"];

document.getElementById("confirmation\_amount").value = response["transaction"]["value"];

document.getElementById("transaction\_signature").value = response["signature"];

$("#basicModal").modal('show');

},

error: function(error){

console.log(error);

}

});

});

});

$(function () {

$("#button\_confirm\_transaction").click(function () {

//console.log($('#confirmation\_transaction\_form').serialize());

$.ajax({

url: document.getElementById("node\_url").value + "/transactions/new",

type: "POST",

headers: {'Access-Control-Allow-Origin':'\*'},

dataType : 'json',

data: $('#confirmation\_transaction\_form').serialize(),

success: function(response){

//reset both forms

$("#transaction\_form")[0].reset();

$("#confirmation\_transaction\_form")[0].reset();

//clean text boxes

$("#sender\_address").val("");

$("#sender\_private\_key").val("");

$("#recipient\_address").val("");

$("#amount").val("");

$("#basicModal").modal('hide');

$("#success\_transaction\_modal").modal('show');

},

error: function(error){

console.log(error);

}

});

});

});

</script>

</body>

</html>

**view\_transactions.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>Blockchain Client</title>

<!-- Bootstrap core CSS -->

<link href="/static/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<link href="/static/vendor/DataTables/css/datatables.min.css" rel="stylesheet">

<link href="/static/css/custom.css" rel="stylesheet">

</head>

<body>

<!-- Navigation -->

<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">

<div class="container">

<a class="navbar-brand" href="#">Blockchain Client</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarResponsive">

<ul class="navbar-nav ml-auto">

<li class="nav-item active">

<a class="nav-link" href="/">Key Generator

<span class="sr-only">(current)</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/make/transaction1">Node Contract</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/make/transaction2">USD Node Contract</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/make/transaction">Payment Transaction</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/contract/private">Product Delivery</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/view/transactions">View Transaction</a>

</li>

</ul>

</div>

</div>

</nav>

<!-- Blockchain Node URL Selector -->

<div class="container">

<div class="row">

<div class="col-lg-12">

<div class="card-body">

<h4 class="card-title">View Transactions</h4>

<p class="card-text">Enter a blockchain node URL and click on "View Transactions" button to check all transactions</p>

</div>

</div>

</div>

</div>

<div class="container alert alert-secondary">

<div class="row">

<label class="col-sm-2">Node URL:</label>

<div class="col-sm-10">

<textarea id="node\_url" rows="1" class="form-control">http://127.0.0.1:5000</textarea>

</div>

</div>

<br>

<div class="row">

<div class="col-lg-12 text-center">

<input type="button" id="view\_transactions" class="btn btn-primary btn-lg" value="View Transactions">

</div>

</div>

</div>

<br>

<!-- Blockchain Transactions Table -->

<div class="container">

<table id="transactions\_table" class="table table-striped table-bordered" cellspacing="0" width="100%">

</table>

</div>

<!-- Bootstrap core JavaScript -->

<script src="/static/vendor/jquery/jquery.min.js"></script>

<script src="/static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<script type="text/javascript" src="/static/vendor/DataTables/js/datatables.min.js"></script>

<script src="/static/vendor/DataTables/js/ellipsis.js"></script>

<script>

$(function(){

$('#view\_transactions').click(function(){

$.ajax({

url: document.getElementById("node\_url").value + "/chain",

type: 'GET',

success: function(response){

console.log(response);

//Generate Transactions Table

var transactions = [];

count = 1;

for (i = 1; i < response.length; i++) {

for (j = 0; j < response["chain"][i]["transactions"].length; j++) {

//format date

var options = { year: "numeric", month: "short", day: "numeric", hour: "2-digit", minute: "2-digit", second: "2-digit" };

var date = new Date(response["chain"][i]["timestamp"] \* 1000);

var formattedDateTime = date.toLocaleTimeString("en-us", options);

transaction = [count,

response["chain"][i]["transactions"][j]["recipient\_address"],

response["chain"][i]["transactions"][j]["sender\_address"],

response["chain"][i]["transactions"][j]["value"],

formattedDateTime,

response["chain"][i]["block\_number"]];

transactions.push(transaction);

count += 1;

};

};

// Restrict a column to 10 characters, do split words

$('#transactions\_table').dataTable( {

data: transactions,

columns: [{ title: "#" },

{ title: "Recipient Address"},

{ title: "Sender Address"},

{ title: "Value"},

{ title: "Timestamp"},

{ title: "Block"}],

columnDefs: [ {targets: [1,2,3,4,5], render: $.fn.dataTable.render.ellipsis( 25 )}]

} );

},

error: function(error){

console.log(error);

}

});

});

})

</script>

</body>

</html>

**dbconnect.py**

import pymysql

pymysql.install\_as\_MySQLdb()

import MySQLdb

def connection():

conn = MySQLdb.connect(host="localhost",

user = "root",

passwd = "root",

db = "privacyecommerce")

c = conn.cursor()

return c, conn

**CHAPTER 7**

**SYSTEM TESTING**

* 1. **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 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/o system configuration. Unit tests ensure that each unique path of a business process performs accurately the documented specifications and contains clearly defined inputs and expected results

**7.2 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.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications,

e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**7.3.Test Cases & Reports / Performance Analysis**

**Unit Testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | ACTION | INPUT | EXPECTED OUTPUT | ACTUAL OUTPUT | TEST RESULT |
| 1. | Enter the name, mobile number and address as seller’s details | Name: Alice  Mobile number:  9876543210  Address: Chennai | Successfully created pop up message | Successfully created pop up message | Passed |
| 2. | Enter the name, mobile number and address as buyer’s details | Name: Bob  Mobile number:  6549873012  Address: Bangalore | Successfully created pop up message | Successfully created pop up message | Passed |

**7.3.1.SELLER AND BUYER - REGISTRATION VALIDATION PAGE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | ACTION | INPUT | EXPECTED OUTPUT | ACTUAL OUTPUT | TEST RESULT |
| 1. | After entering seller’s details hashcode is produced | Name: Alice  Mobile number:  9876543210  Address: Chennai | Hashcode is produced | Hashcode is produced | Passed |
| 2. | After entering buyer’s details hashcode is produced | Name: Bob  Mobile number:  6549873012  Address: Bangalore | Hashcode is produced | Hashcode is produced | Passed |

**7.3.2.SELLER AND BUYER -HASHCODE GENERATON**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | ACTION | INPUT | EXPECTED OUTPUT | ACTUAL OUTPUT | TEST RESULT |
| 1. | Buyer - The transaction that we performed during product buying can be viewed using transaction | Node URL: http://127.0.0.1:5000 | http://127.0.0.1:5000 URL’s transaction is displayed | http://127.0.0.1:5000 URL’s transaction is displayed | Passed |
| 2. | Seller - The transaction that we performed during product selling can be viewed using transaction | Node URL: http://127.0.0.1:5000 | http://127.0.0.1:5000 URL’s transaction is displayed | http://127.0.0.1:5000 URL’s transaction is displayed | Passed |

**7.3.3.SELLER AND BUYER -VIEW TRANSACTION**

**Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

### Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**CHAPTER 8**

**CONCLUSION**

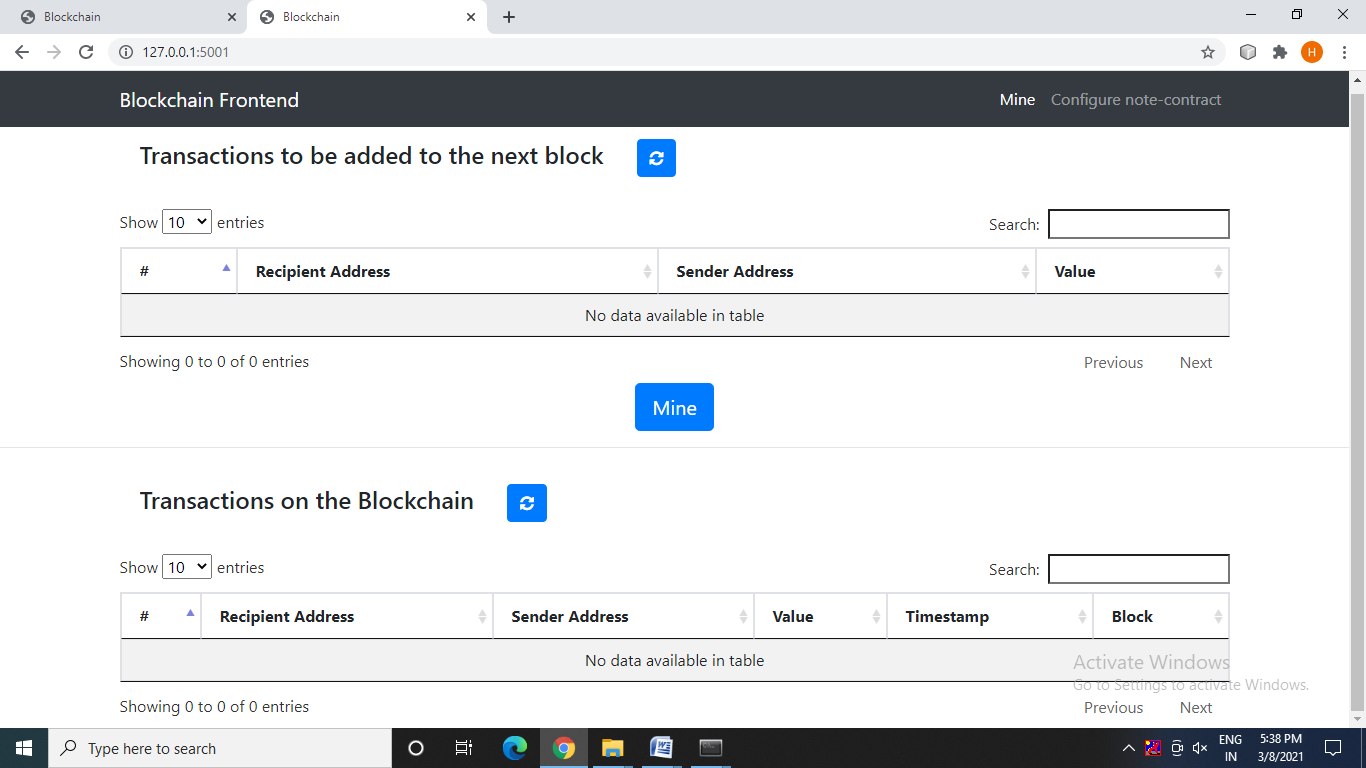
# 8.1 Conclusion and Future Enhancements

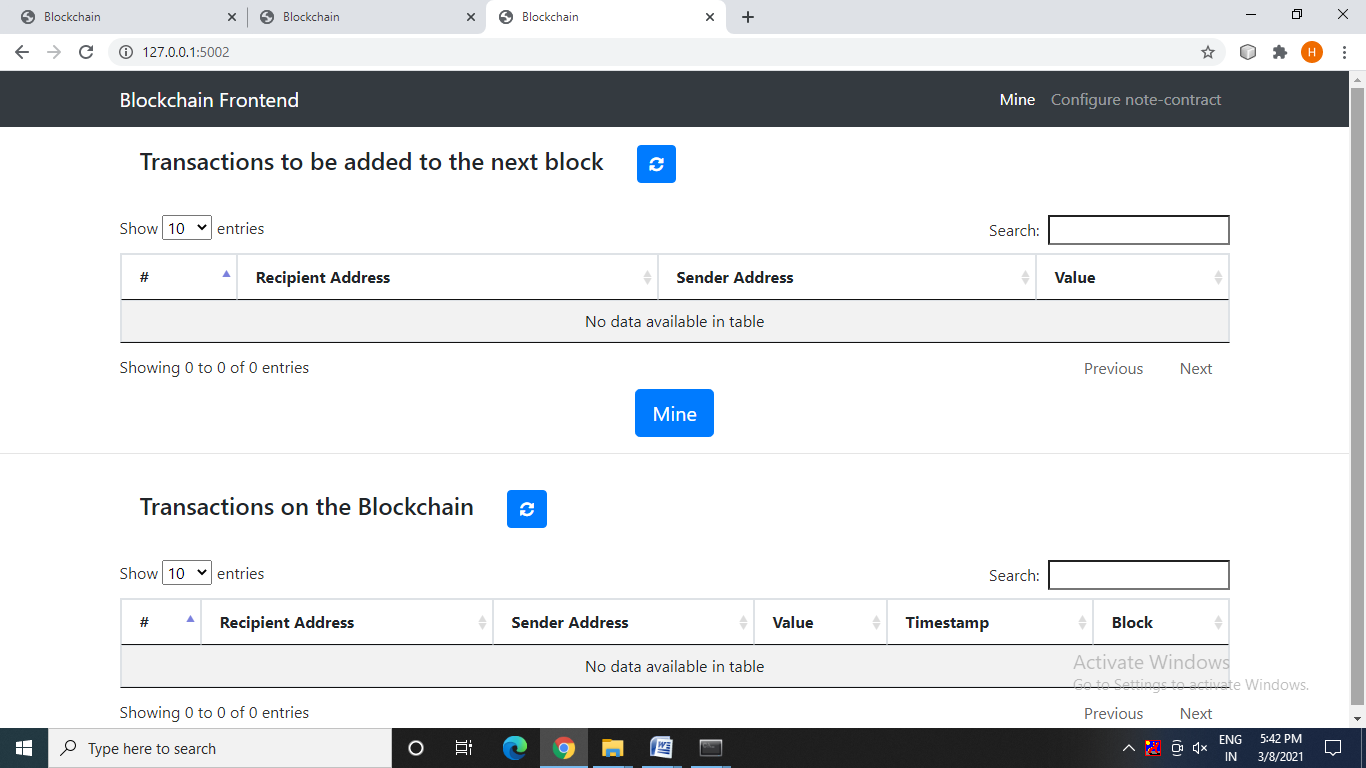
We design a privacy-preserving model for E-commerce systems based on the blockchain technology. In order to protect users’ identities and guarantee proof of ownership, we employ a zero-knowledge proof algorithm called zk-SNARKs. The algorithm allows one party (the prover) prove to another party (the veriﬁer) that a given statement is true, without conveying any information apart from the fact that the statement is indeed true by zero knowledge proof. To protect users’ addresses, we index the address by a hash string and record it on the logistics ledger. To protect phone numbers, we use delivery boxes and intelligent logistics centers that work with IoT. Different from common practice, buyers do not need to provide a phone number for SMS notiﬁcation. Finally, We build implementations of the architecture using two existing blockchain application platforms. Performance analysis of the blockchain platforms provided insights into the models feasibility and further considerations for deploying a usable implementation. In the future, we will implement our logistics chain on blockchain platforms such as IOTA, which is mainly used in the ﬁeld of IoT, and perform further simulations on the hardware through the Raspberry Pi.

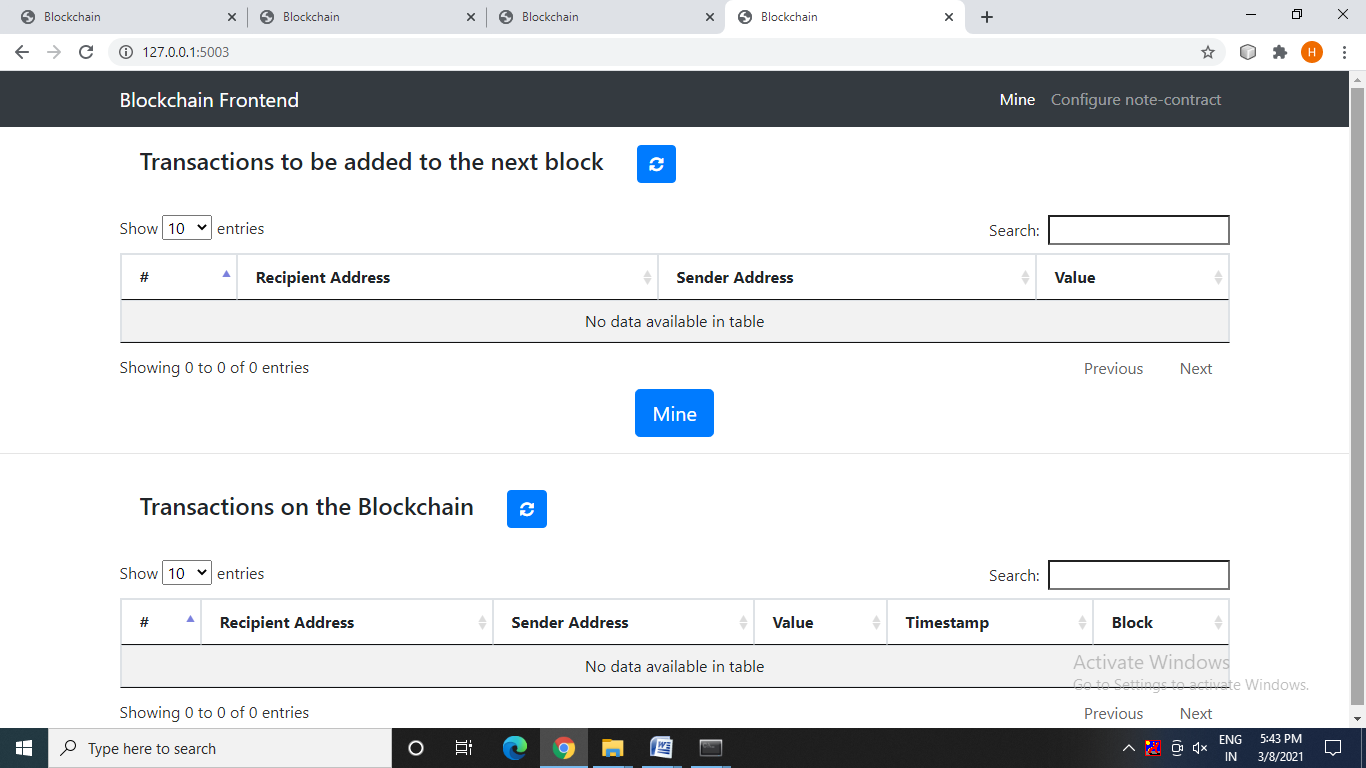
**APPENDICES**

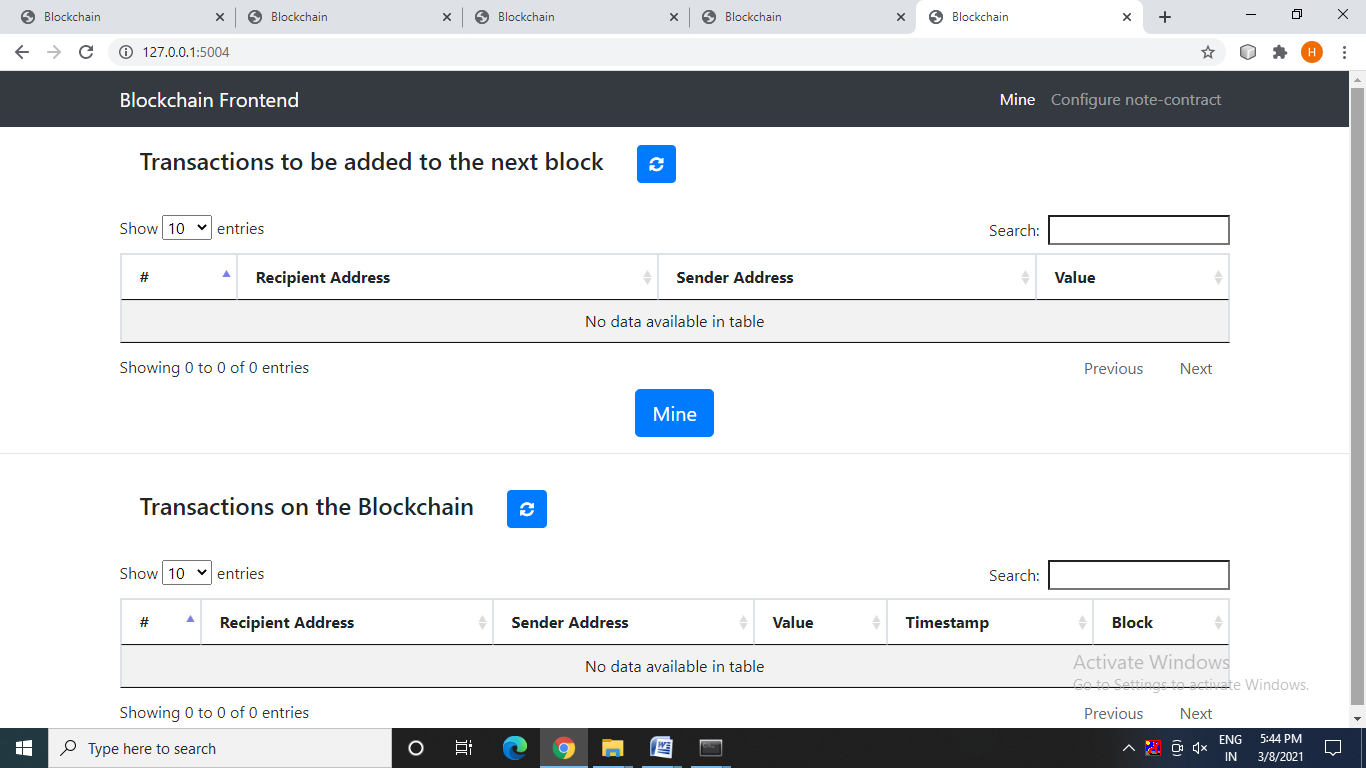
**A.1 Sample Screenshots:**

### 



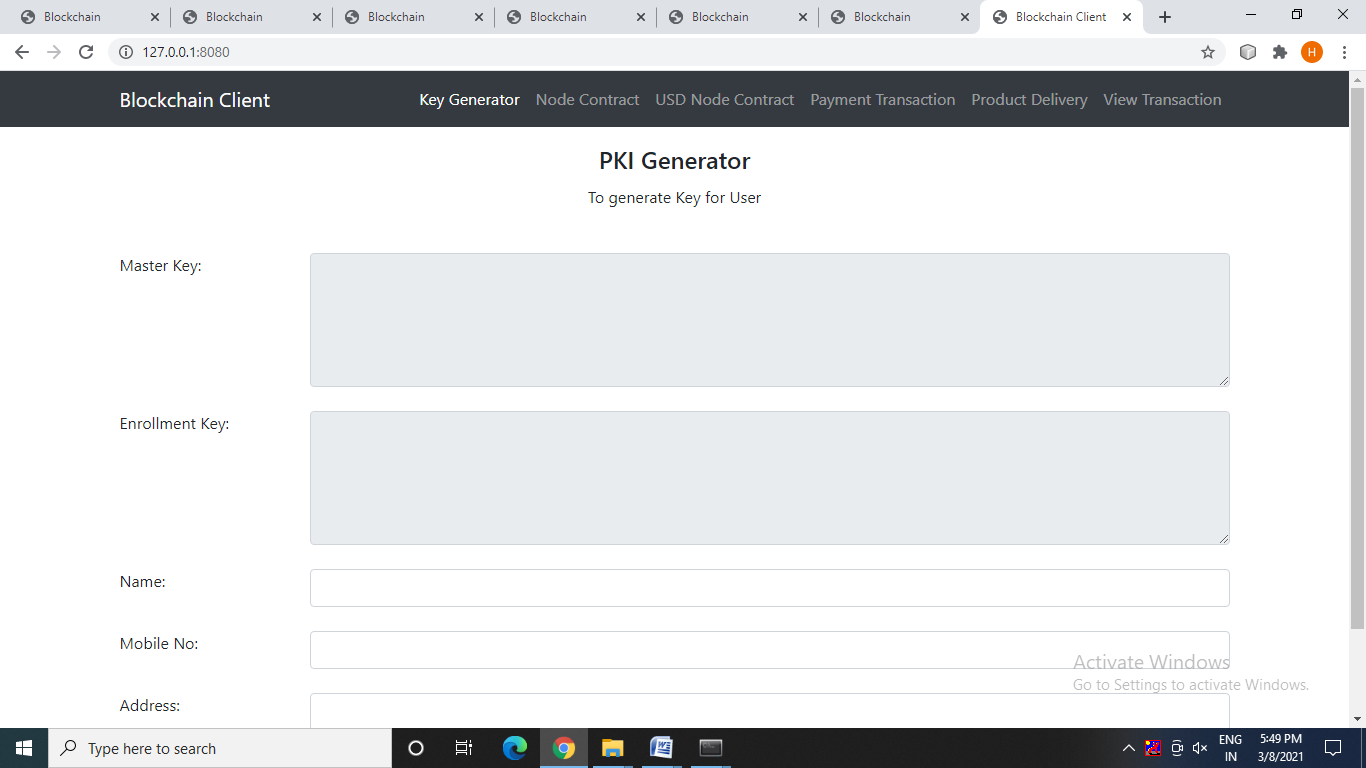


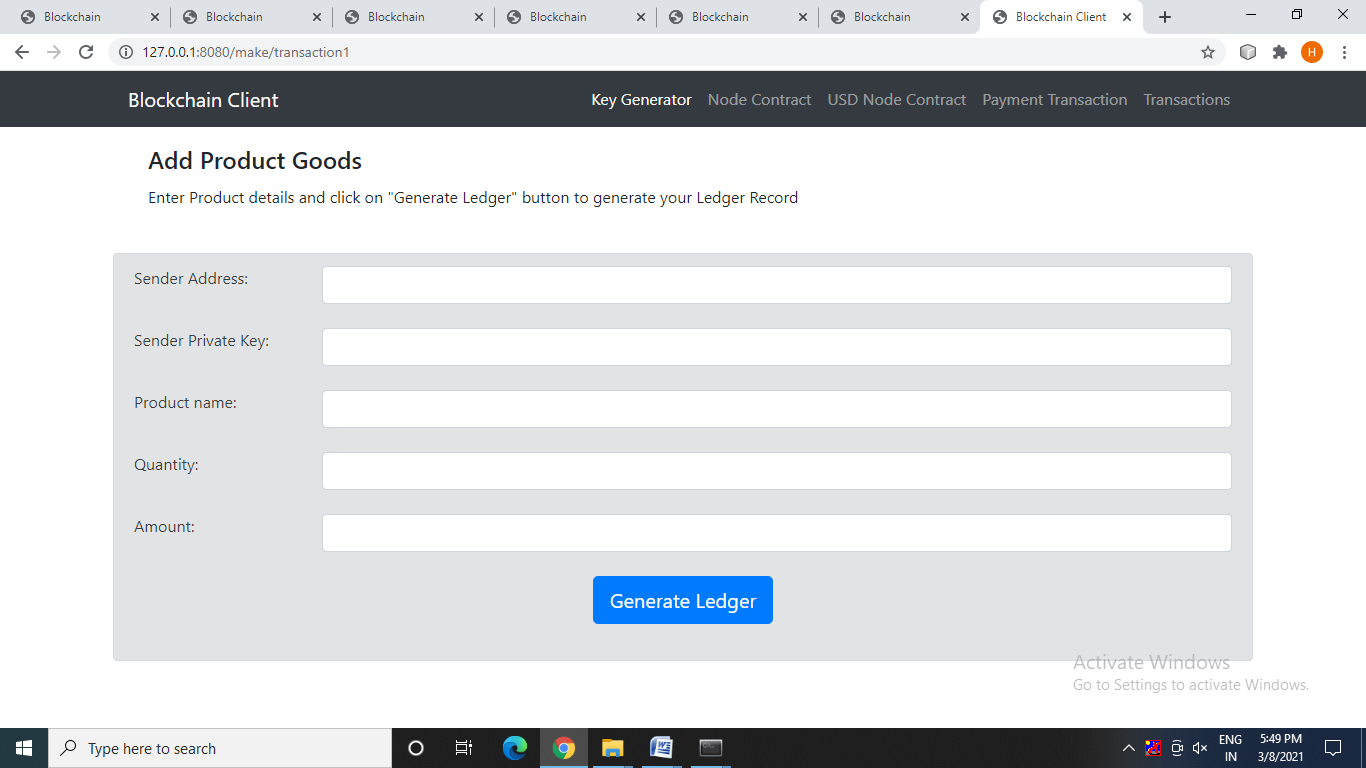


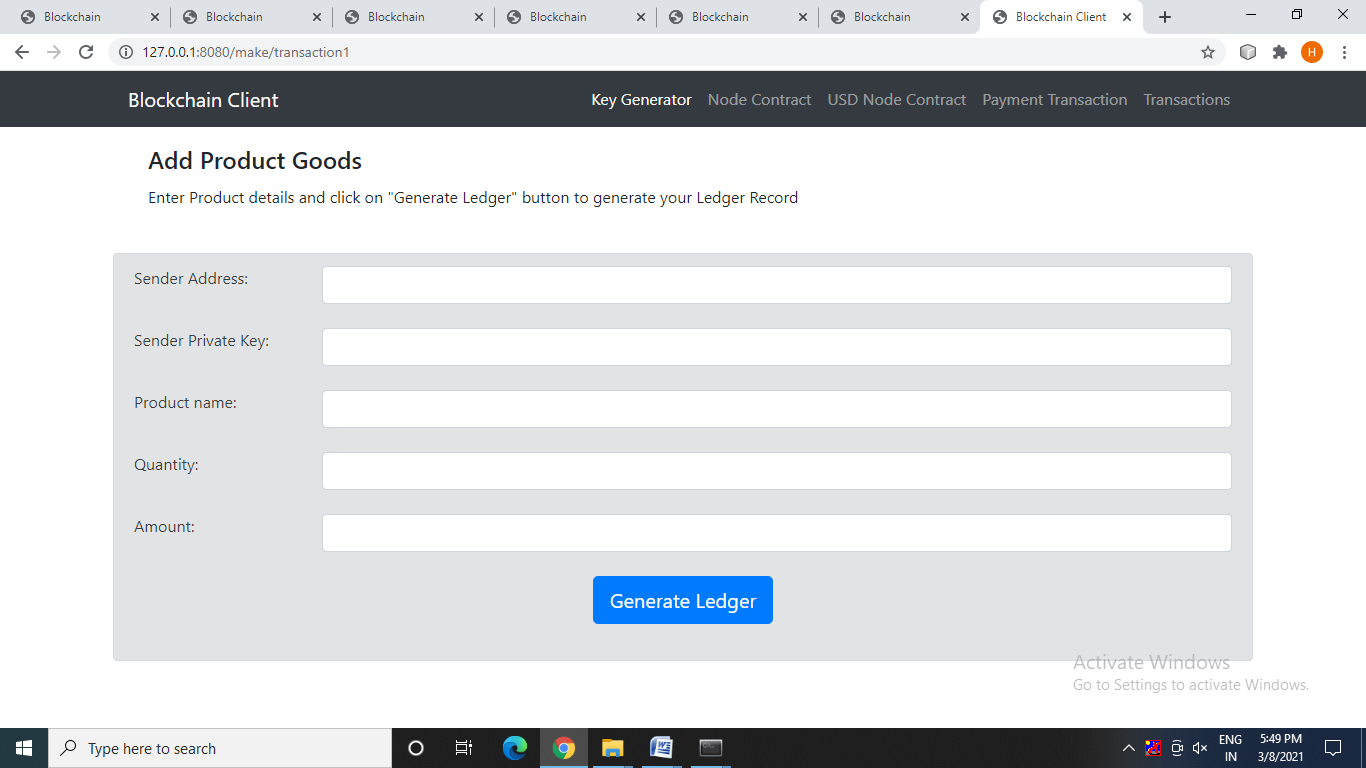


****

**USER:**

****

****

****

**A.2.Publications**

Mrs.C.Vijayalakshmi, Aruna M, Harini T M, Krithikaa K **“OPTIMIZATION OF**

**E-COMMERCE PLATFORM BASED ON BLOCKCHAIN TECHNOLOGY”**, International Journal for Research in Applied Science & Engineering Technology (IJRASET), ISSN:2321-9653, Volume.9, Issue VI, pp.1334-1347, June 2021

**DOI Details**: <https://doi.org/10.22214/ijraset.2021.35262>

**REFERENCES**

[1]M. Niranjanamurthy and D. D. Chahar, “The study of e-commerce security issues and solutions,” International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, no. 7, 2013.

[2]J. Y. Tsai, S. Egelman, L. Cranor, and A. Acquisti, “The effect of online privacy information on purchasing behavior: An experimental study,”Information Systems Research,Vol.22,No.2,June 2011.

[3]B. Berendt, O. G¨unther, and S. Spiekermann, “Privacy in e-commerce: stated preferences vs. actual behavior,” Communications of the ACM, vol. 48, no. 4, pp. 101–106, 2005.

[4]L. Xiong and L. Liu, “A reputation-based trust model for peer-to-peer e-commerce communities,” in E-Commerce, 2003. CEC 2003. IEEE International Conference on. IEEE, 2003, pp. 275–284.

[5]D. Vandervort, “Challenges and opportunities associated with a bitcoinbased transaction rating system,” in International Conference on Financial Cryptography and Data Security. Springer, 2014, pp. 33–42.

[6]Eli Ben-Sasson, Alessandro Chiesa, Christina Garman and Matthew Green “Zerocash: Decentralized anonymous payments from bitcoin,” in Security and Privacy (SP), 2014 IEEE Symposium on. IEEE, 2014, pp. 459–474.

[7]Rosario Gennaro, Steven Goldfeder and Arvind Narayanan “Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security,” in International Conference on Applied Cryptography and Network Security. Springer, 2016, pp. 156–174.

[8]D. Ron and A. Shamir, “Quantitative analysis of the full bitcoin transaction graph,” in International Conference on Financial Cryptography and Data Security. Springer, 2013, pp. 6–24.

[9]N. Hackius and M. Petersen, “Blockchain in logistics and supply chain: trick or treat?” A chapter in Digitalization in Supply Chain Management and Logistics: Smart and Digital Solutions for an Industry 4.0 Environment, 2017, pp 3-18 from Hamburg University of Technology (TUHH), Institute of Business Logistics and General Management.

[10]Hemantkumar P. Bulsara, Pratiksinh S. Vaghela."Blockchain Technology for E-commerce Industry" in International Journal of advanced Science and Technology,Vol.29,No.5,2020.

[11]Shazia Yasin,Khalid Haseeb,Rashid Jalal Qureshi,"Cryptography Based E-Commerce Security: A Review" in International Journal of Computer Science Issues,9(2),2012.

[12]Uriel Feige,Amos Fiat,Adi Shamir,"Zero-knowledge proofs of identity",in Journal of Cryptology,Springer,1,77-94(1988).

[13]Weiqin Zou,David Lo,Pavneet Singh Kochhar,Xuan-Bach D. Le,Xin Xia,Yang Feng,Zhenyu Chen,Baowen Xu, "Smart Contract Development: Challenges and Opportunities"published in IEEE Transactions on Software Engineering,24 September 2019.

[14]Paul J.Taylor,TooskaDargahi,AliDehghantanha,Reza M.Parizi,Kim-Kwang RaymondChood"A systematic literature review of blockchain cyber security" in Digital Communication and Networks,Volume 6, Issue 2, May 2020, Pages 147-156.