**DETECTION OF FAKE PRODUCTS USING BLOCKCHAIN TECHNOLOGY**

**CHAPTER 1**

* 1. **Abstract of the project:**

The manufacturing and marketing of counterfeit or duplicate products and goods leads to consequential financial, health and safety threat to end users. It also impacts on the economic growth of original manufacturers and businesses through revenue loss, product defamation, downtime, replacement expenses, forcing brands to spend money fighting counterfeits, trust among business partners can also be at risk, stealing sales etc. To overcome these crucial effects of counterfeiting, a blockchain based system is used in identification of original products and detects duplicate products to ensure the identification of original goods. In this work, with massive emerging trends in wireless technology, QR (Quick Response) codes and barcodes provides a robust technique to cut down the practice of counterfeiting the products. The fake products are detected using camera scanner, where QR or barcode of the product is linked to a block chain in order to store product details and guaranteed unique code of each product as blocks in the database. If the code matches, the notification will be sent to the customer indicating the authenticity of the product and else if it does not match, a notification will be sent to customer that product is fake or counterfeited as well as to manufacturer about the place of purchase if customer accepts the request made by the application. This approach ensures that consumers won’t completely rely on merchants to determine if products are original or forged.

**INTRODUCTION**

The global development of the product always comes with risk factors such as counterfeiting and duplication which in turn can affect the company name, reputation, revenue and customer satisfaction. The trading and marketing of counterfeit products is growing in humongous. It affects the sales, reputation, and profits of the companies and also do poses a fatal threat for the unsuspecting buyers. In order to ensure the identification and traceability of false goods or products throughout the supply chain and to combat this phenomenon, a fully functional blockchain system is proposed. Companies only need to pay very low transaction fees and they no longer need to worry about the possibility of delivering counterfeit products to end-users. Because of counterfeit or fake products manufacturers face the biggest problems and huge losses in sense of brand damage as well as revenue loss. To find the originality of the product blockchain technology can be used. Blockchain is an arrangement of recorded information that makes it difficult or impossible to modify or hack the framework. Blockchain technology is a distributed, decentralized, and digital ledger that stores transaction information as blocks in databases which is connected with chains. Each block in the chain contains multiple transactions, and every time a new transaction occurs on the blockchain, a record of that transaction is added to every participant’s record. The decentralized database managed by the number of participants is known as Distributed Ledger Technology (DLT). Blockchain is a type of DLT in which transactions are recorded with an immutable cryptographic signature called a hash. Blockchain technology helps to solve the problem of counterfeiting a product. Blockchain technology is more secure compared to other technology. Once the product is stored on the network hash code is generated for that product and it is possible to maintain all transaction records of the product and its current owner as a chain will be created for that product transactions. It will store all the transaction records as blocks in the blockchain. In the proposed system we are assigning a generated QR code or barcode to a particular product created by manufacturer along with all the details of the product and the end customer can scan that QR code to get all information about that product. After scanning the QR code or barcode the user can identify whether the product is real or fake

**1.2 Objectives:**

* Counterfeiting of products leads to significant financial, health, and safety risks for end-users. It also impacts the economic growth of original manufacturers and businesses through revenue loss, product defamation, downtime, replacement expenses, and loss of trust among business partners.
* To overcome these issues, a blockchain-based system is proposed that utilizes QR codes or barcodes to identify original products and detect duplicates.
* The system works by linking the QR code or barcode of a product to a blockchain, which stores the product details and a unique code for each product as blocks in the database.
* When a customer scans the QR code or barcode using a camera scanner, the system checks if the code matches the one stored in the blockchain. If it matches, the customer is notified that the product is authentic. If it doesn't match, the customer is notified that the product is fake or counterfeit, and the manufacturer is also informed about the location of purchase.

**1.3 Problem statement:**

a. Revenue loss for original manufacturers and businesses

b. Product defamation and damage to brand reputation

c. Downtime and increased replacement expenses

d. Expenses incurred by brands to fight counterfeiting

e. Loss of trust among business partners

**1.4 Scope of the project:**

a. Integrate the system with QR codes or barcodes for product identification

b. Establish a blockchain network to store product details and unique codes

c. Implement mechanisms to link product codes to the blockchain

**CHAPTER 2**

**LITERATURE REVIEW**

**2.1 Introduction:**

**2.3 LITERATURE SURVEY:**

# **2.3.1 Fake consumer Logo detection using deep neural networks integrating word embedding’s and emotion mining, 2020**

# **Author*:*** Michal Munk, Aliaksandr Barushka, Petr Hajek

# **Methodology:**

Fake consumer Logo detection has attracted much interest in recent years owing to the increasing number of Internet purchases. Existing approaches to detect fake consumer Logos use the Logo content, product and Logoer information and other features to detect fake Logos. However, as shown in recent studies, the semantic meaning of Logos might be particularly important for text classification. In addition, the emotions hidden in the Logos may represent another potential indicator of fake content. To improve the performance of fake Logo detection, here we propose two neural network models that integrate traditional bag-of-words as well as the word context and consumer emotions. Specifically, the models learn document-level representation by using three sets of features: (1) *n*-grams, (2) word embeddings and (3) various lexicon-based emotion indicators. Such a high-dimensional feature representation is used to classify fake Logos into four domains. To demonstrate the effectiveness of the presented detection systems, we compare their classification performance with several state-of-the-art methods for fake Logo detection. The proposed systems perform well on all datasets, irrespective of their sentiment polarity and product category.

**Advantage:**

* Time consumption is low.

**Disadvantage*:***

* Preventive actions can be taken against Logos in the aforementioned platforms.

# **2.3.2Fake online Logos: Literature Logo, synthesis, and directions for future research, 2020**

# **Author:** [YuanyuanWu](https://www.sciencedirect.com/science/article/abs/pii/S016792362030035X" \l "!)[ab](https://www.sciencedirect.com/science/article/abs/pii/S016792362030035X" \l "!)[Eric,W.T.Ngai](https://www.sciencedirect.com/science/article/abs/pii/S016792362030035X" \l "!)[b](https://www.sciencedirect.com/science/article/abs/pii/S016792362030035X" \l "!), [PengkunWuabc](https://www.sciencedirect.com/science/article/abs/pii/S016792362030035X#!),[ChongWua](https://www.sciencedirect.com/science/article/abs/pii/S016792362030035X#!)

# **Methodology:**

Fake online Logos in e-commerce significantly affect online consumers, merchants, and, as a result, market efficiency. Despite scholarly efforts to examine fake Logos, there still lacks a survey that can systematically analyze and summarize its antecedents and consequences. This study proposes an antecedent–consequence–intervention conceptual framework to develop an initial research agenda for investigating fake Logos. Based on a Logo of the extant literature on this issue, we identify 20 future research questions and suggest 18 propositions. Notably, research on fake Logos is often limited by lack of high-quality datasets. To alleviate this problem, we comprehensively compile and summarize the existing fake Logos-related public datasets. We conclude by presenting the theoretical and practical implications of the current research.

**Advantage**:

* The antecedent–consequence–intervention (ACI) conceptual framework is proposed.
* A Logo of fake online Logos is presented.
* Twenty future research questions are identified.

**Disadvantage:**

* The process is implemented without removing unwanted data.

# **2.3.3 One Methodology for Spam Logo Detection Based on Logo Coherence Metrics, 2005**

**Author:** Xinkai Yang

**Methodology:**

In this paper, we propose an iterative computation framework to detect spam Logos based on coherent examination. We first define some Logos' coherent metrics to analyze Logo coherence in the granularity of sentence. Then the framework and its evaluation process are discussed in details. Until today, there are a few attempts have been made on Logo spam detection which is a challenging and under exploration area. In this paper, we propose a general framework to detect spam Logos based on coherent examination. We first discuss some assumptions of spam Logos, and then we define some Logo coherent metrics. The iterative computation framework is also provided. Our proposed model tries to analyze Logo coherence in the granularity of sentence. We define some metrics to investigate the coherence between sentiment words and other related words based on the flow smoothness between sentences: word transition probability and word concurrence probability. Because we try to identify spam Logos on the semantic level, our proposed model can reveal more important clues of spam Logos based on their word pattern. This work provides a novel viewpoint for spam Logo detection and more potential approaches could be explored in the future.

**Advantage*:***

* The Logo ratings are clustered into unfairly high ratings and unfairly low ratings by using third party ratings products.
* The sentence concurrence measure for two consecutive sentences can be defined by using the highest word concurrence measure.

# **2.3.4 New Avenues in Opinion Mining and Sentiment Analysis, 2013**

# **Author**: Erik Cambria, Björn Schuller, Yunqing Xia, Catherine Havasi,

# **Methodology:**

Others’ opinions can be crucial when it’s time to make a decision or choose among multiple options. When those choices involve valuable resources (for example, spending time and money to buy products or services) people often rely on their peers’ past experiences. Until recently, the main sources of information were friends and specialized magazine or websites. Now, the “social web” provides new tools to efficiently create and share ideas with everyone connected to the World Wide Web. Forums, blogs, social networks, and content-sharing services help people share useful information. This information is unstructured, however, and because it’s produced for human consumption, it’s not something that’s “machine process able.” Capturing public opinion about social events, political movements, company strategies, marketing campaigns, and product preferences is garnering increasing interest from the scientific community (for the exciting open challenges), and from the business world (for the remarkable marketing fallouts and for possible financial market prediction). The resulting emerging fields are opinion mining and sentiment analysis.

**Advantage*:***

* The results of a topic-based search engine. However, several studies suggest that managing these two tasks jointly might benefit overall performance.

**Disadvantage:**

* Training time is high.
* Prediction is not accurate.

# **2.3.5 Sentiment analysis using product Logo data, 2015**

# **Author:** Xing Fang\* and Justin Zhan

**Methodology**:

Sentiment analysis or opinion mining is one of the major tasks of NLP (Natural Language Processing). Sentiment analysis has gain much attention in recent years. In this paper, we aim to tackle the problem of sentiment polarity categorization, which is one of the fundamental problems of sentiment analysis. A general process for sentiment polarity categorization is proposed with detailed process descriptions. Data used in this study are online product Logos collected from Amazon.com. Experiments for both sentence-level categorization and Logo-level categorization are performed with promising outcomes. At last, we also give insight into our future work on sentiment analysis. Sentiment is an attitude, thought, or judgment prompted by feeling. Sentiment analysis, which is also known as opinion mining, studies people’s sentiments towards certain entities. Internet is a resourceful place with respect to sentiment information. From a user’s perspective, people are able to post their own content through various social media, such as forums, micro-blogs, or online social networking sites.

**Advantage**:

* With the help of the ROC curves, it is clear to see that all three models performed quite well for testing data that have high posterior probability.

**Disadvantage:**

Low accuracy

**EXISTING SYSTEM**

* In existing system, Reading product reviews before buying the product becomes a habit [3], especially for potential customers. For a company, the positive reviews from customers can generate significant financial benefits for the business, which can be taken as input for decisions related to product design and what services are provided to customers. Related to the financial benefits gained as a result of the positive reviews about the product or service from the customer, the fraudsters tried to play the existing system by writing fake reviews and providing an assessment that is not fair to promote or discredit a product or service. Automatic detection of spammers is a very important task but still lacks research. Unlike other types of spam, such as web spam or email spam, spam on a review is far more difficult to detect. The main reason is that spammers can easily disguise themselves. Thus, it is difficult for users to recognize

**DISADVANTAGES**

* Doesn’t Efficient for handling large volume of data.
* Theoretical Limits
* Incorrect Classification Results.
* Less Prediction Accuracy.

**PROPOSED SYSTEM**

* 1. To protect brand value and duplication threats by developing fake product detection system using blockchain technology. 2. To secure and authenticate the product details which helps in identification and traceability of the specific product throughout the supply chain. 3. All product details are secured and stored in QR code or product ID which helps in identification which is stored in tamper proof blocks of blockchain for further security. 4. Manufacturer can add product details and system generates QR code which can be used by retailers and distributors for tracing and even by consumers to ensure purchasing of original products.
* It is more effective of performance analysis.

**ADVANTAGES**

* High performance.
* Provide accurate prediction results.
* It avoid sparsity problems.
* Reduces the information Loss and the bias of the inference due to the multiple estimates.

**SYSTEM ARCHITECTURE**

Enc set

Blockchain

Pre-processed data

Dataset

Dec set

Prediction

Classification

**FLOW DIAGRAM**

Clean Dataset

Select Dataset

Count Vectorizer

Classification

Prediction

Feature Selection

**USE CASE DIAGRAM**

USER

**ER DIAGRAM**

**COUNT VECTORIZER**

**FEATURE SELECTION**

**DATA SELECTION & LOAD**

**CLASSIFICATION**

**RESULT**

**GENERATION**

**SEQUENCE DIAGRAM**

Clean

Prediction

Count Vectorizer

Model Selection

Classification

Select

Select dataset

 Load dataset

Start

Result Generation

**TESTING OF PRODUCT**

**Testing of Product**

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system.  System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved.  The candidate system is subject to variety of tests-on-line response, Volume Street, recovery and security and usability test.  A series of tests are performed before the system is ready for the user acceptance testing.  Any engineered product can be tested in one of the following ways.  Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational.  Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

**UNIT TESTING**

Unit testing is the testing of each module and the integration of the overall system is done.  Unit testing becomes verification efforts on the smallest unit of software design in the module.  This is also known as ‘module testing’.  The modules of the system are tested separately.  This testing is carried out during the programming itself.  In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module.  There are some validation checks for the fields.  For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included.  It is very easy to find error and debug the system.

**INTEGRATION TESTING**

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function.  Integrated testing is systematic testing that can be done with sample data.  The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

1. Top-down integration testing.
2. Bottom-up integration testing.

**WHITE BOX TESTING**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases.  Using the white box testing methods, we derived test cases that guarantee that all independent paths within a module have been exercised at least once.

**BLACK BOX TESTING**

* Black box testing is done to find incorrect or missing function
* Interface error
* Errors in external database access
* Performance errors
* Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’.  It tests the external behaviour of the system.  Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

**VALIDATION TESTING**

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many, but a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer.

# **USER ACCEPTANCE TESTING**

User acceptance of the system is the key factor for the success of the system.  The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

# **OUTPUT TESTING**

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format.  The output displayed or generated by the system under consideration.  Here the output format is considered in two ways.  One is screen and the other is printed format.  The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs.  For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

**System Implementation**

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended users and the operation of the system. The people are not sure that the software is meant to make their job easier.

* The active user must be aware of the benefits of using the system
* Their confidence in the software built up
* Proper guidance is impaired to the user so that he is comfortable in using the application

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not running on the server, the actual processes will not take place.

**User Training**

To achieve the objectives and benefits expected from the proposed system it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for education and training is more and more important. Education is complementary to training. It brings life to formal training by explaining the background to the resources for them. Education involves creating the right atmosphere and motivating user staff. Education information can make training more interesting and more understandable.

**Training on the Application Software**

After providing the necessary basic training on the computer awareness, the users will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design, type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the data entered. This training may be different across different user groups and across different levels of hierarchy.

**Operational Documentation**

Once the implementation plan is decided, it is essential that the user of the system is made familiar and comfortable with the environment. A documentation providing the whole operations of the system is being developed. Useful tips and guidance is given inside the application itself to the user. The system is developed user friendly so that the user can work the system from the tips given in the application itself.

**System Maintenance**

The maintenance phase of the software cycle is the time in which software performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is to make adaptable to the changes in the system environment. There may be social, technical and other environmental changes, which affect a system which is being implemented. Software product enhancements may involve providing new functional capabilities, improving user displays and mode of interaction, upgrading the performance characteristics of the system. So only thru proper system maintenance procedures, the system can be adapted to cope up with these changes. Software maintenance is of course, far more than “finding mistakes”.

**Corrective Maintenance**

The first maintenance activity occurs because it is unreasonable to assume that software testing will uncover all latent errors in a large software system. During the use of any large program, errors will occur and be reported to the developer. The process that includes the diagnosis and correction of one or more errors is called Corrective Maintenance.

**Adaptive Maintenance**

The second activity that contributes to a definition of maintenance occurs because of the rapid change that is encountered in every aspect of computing. Therefore Adaptive maintenance termed as an activity that modifies software properly with a changing environment is both necessary & common place.

**Perceptive Maintenance**

The third activity that may be applied to a definition of maintenance occurs when a software package is successful. As the software is used, recommendations for new capabilities, modifications to existing functions, and general enhancement are received from users. To satisfy requests in this category, Perceptive maintenance is performed. This activity accounts for the majority of all efforts expended on software maintenance.

**Preventive Maintenance**

The fourth maintenance activity occurs when software is changed to improve future maintainability or reliability, or to provide a better basis for future enhancements. Often called preventive maintenance, this activity is characterized by reverse engineering and re-engineering techniques.

**Types of Software Testing**

**Ad-hoc testing**

This type of software testing is very informal and unstructured and can be performed by any stakeholder with no reference to any test case or test design documents. The person performing Ad-hoc testing has a good understanding of the domain and workflows of the application to try to find defects and break the software. Ad-hoc testing is intended to find defects that were not found by existing test cases.

**Acceptance Testing**

Acceptance testing is a formal type of software testing that is performed by end user when the features have been delivered by developers. The aim of this testing is to check if the software confirms to their business needs and to the requirements provided earlier. Acceptance tests are normally documented at the beginning of the sprint (in agile) and is a means for testers and developers to work towards a common understanding and shared business domain knowledge.

**Accessibility Testing**

In accessibility testing, the aim of the testing is to determine if the contents of the website can be easily accessed by disable people. Various checks such as colour and contrast (for colour blind people), font size for visually impaired, clear and concise text that is easy to read and understand.

**Agile Testing**

Agile Testing is a type of software testing that accommodates agile software development approach and practices. In an Agile development environment, testing is an integral part of software development and is done along with coding. Agile testing allows incremental and iterative coding and testing.

**API Testing**

API testing is a type of testing that is similar to unit testing. Each of the Software APIs are tested as per API specification. API testing is mostly done by testing team unless APIs to be tested or complex and needs extensive coding. API testing requires understanding both API functionality and possessing good coding skills.

**Automated testing**

This is a testing approach that makes use of testing tools and/or programming to run the test cases using software or custom developed test utilities. Most of the automated tools provided capture and playback facility, however there are tools that require writing extensive scripting or programming to automate test cases.

**All Pairs testing**

Also known as Pair wise testing, is a black box testing approach and a testing method where in for each input is tested in pairs of inputs, which helps to test software works as expected with all possible input combinations.

**Beta Testing**

This is a formal type of software testing that is carried out by end customers before releasing or handing over software to end users. Successful completion of Beta testing means customer acceptance of the software.

**Black Box testing**

Black box testing is a software testing method where in testers are not required to know coding or internal structure of the software. Black box testing method relies on testing software with various inputs and validating results against expected output.

**Backward Compatibility Testing**

Type of software testing performed to check newer version of the software can work successfully installed over previous version of the software and newer version of the software works as fine with table structure, data structures, files that were created by previous version of the software.

**Boundary Value Testing (BVT)**

Boundary Value Testing is a testing technique that is based on concept “error aggregates at boundaries”. In this testing technique, testing is done extensively to check for defects at boundary conditions. If a field accepts value 1 to 100 then testing is done for values 0, 1, 2, 99, 100 and 101.

**Big Bang Integration testing**

This is one of the integration testing approaches, in Big Bang integration testing all or all most all of the modules are developed and then coupled together.

**Bottom up Integration testing**

Bottom up integration testing is an integration testing approach where in testing starts with smaller pieces or sub systems of the software till all the way up covering entire software system. Bottom up integration testing begins with smaller portion of the software and eventually scale up in terms of size, complexity and completeness.

**Branch Testing**

Is a white box testing method for designing test cases to test code for every branching condition? Branch testing method is applied during unit testing.

**Browser compatibility Testing**

It is one of the sub types of testing of compatibility testing performed by testing team. Browser compatibility testing is performed for web applications with combination of different browsers and operating systems.

**Compatibility testing**

Compatibility testing is one of the test types performed by testing team. Compatibility testing checks if the software can be run on different hardware, operating system, bandwidth, databases, web servers, application servers, hardware peripherals, emulators, different configuration, processor, different browsers and different versions of the browsers etc.

**Component Testing**

This type of software testing is performed by developers. Component testing is carried out after completing unit testing. Component testing involves testing a group of units as code together as a whole rather than testing individual functions, methods.

**Condition Coverage Testing**

Condition coverage testing is a testing technique used during unit testing, where in developer tests for all the condition statements like if, if else, case etc., in the code being unit tested.

**Dynamic Testing**

Testing can be performed as Static Testing and Dynamic testing, Dynamic testing is a testing approach where-in testing can be done only by executing code or software are classified as Dynamic Testing. Unit testing, Functional testing, regression testing, performance testing etc.

**Decision Coverage Testing**

Is a testing technique that is used in Unit testing, objective of decision coverage testing is to expertise and validate each and every decisions made in the code e.g. if, if else, case statements.

**End-to-end Testing**

End to end testing is performed by testing team, focus of end to end testing is to test end to end flows e.g. right from order creation till reporting or order creation till item return etc. and checking. End to end testing is usually focused mimicking real life scenarios and usage. End to end testing involves testing information flow across applications.

**Exploratory Testing**

Exploratory testing is an informal type of testing conducted to learn the software at the same time looking for errors or application behaviour that seems non-obvious. Exploratory testing is usually done by testers but can be done by other stake holders as well like Business Analysts, developers, end users etc. who are interested in learning functions of the software and at the same time looking for errors or behaviour is seems non-obvious.

**Equivalence Partitioning**

Equivalence partitioning is also known as Equivalence Class Partitioning is a software testing technique and not a type of testing by itself. Equivalence partitioning technique is used in black box and grey box testing types. Equivalence partitioning classifies test data into Equivalence classes as positive Equivalence classes and negative Equivalence classes, such classification ensures both positive and negative conditions are tested.

**Functional Testing**

Functional testing is a formal type of testing performed by testers. Functional testing focuses on testing software against design document, Use cases and requirements document. Functional testing is a black box type of testing and does not require internal working of the software unlike white box testing.

**Fuzz Testing**

Fuzz testing or fuzzing is a software testing technique that involves testing with unexpected or random inputs. Software is monitored for failures or error messages that are presented due to the input errors.

**GUI (Graphical User Interface) testing**

This type of software testing is aimed at testing the software GUI (Graphical User Interface) of the software meets the requirements as mentioned in the GUI mock-ups and Detailed designed documents. For e.g. checking the length and capacity of the input fields provided on the form, type of input field provided, e.g. some of the form fields can be displayed as dropdown box or a set of radio buttons. So GUI testing ensures GUI elements of the software are as per approved GUI mock-ups, detailed design documents and functional requirements. Most of the functional test automation tools work on GUI capture and playback capabilities. This makes script recording faster at the same time increases the effort on script maintenance.

**Glass box Testing**

Glass box testing is another name for White box testing. Glass box testing is a testing method that involves testing individual statements, functions etc., Unit testing is one of the Glass box testing methods.

**Gorilla Testing**

This type of software testing is done by software testing team, has a scary name though? Objective of Gorilla Testing is to exercise one or few functionality thoroughly or exhaustively by having multiple people test the same functionality.

**Happy Path Testing**

Also known as Golden path testing, this type of testing focuses on selective execution of tests that do not exercise the software for negative or error conditions.

**Integration Testing**

Integration testing also known as met in short, in one of the important types of software testing. Once the individual units or components are tested by developers as working then testing team will run tests that will test the connectivity among these units/component or multiple units/components. There are different approaches for Integration testing namely, Top-down integration testing, Bottom-up integration testing and a combination of these two known as Sand witch testing.

**Interface Testing**

Software provides support for one or more interfaces like “Graphical user interface”, “Command Line Interface” or “Application programming interface” to interact with its users or other software. Interfaces serves as medium for software to accept input from user and provide result. Approach for interface testing depends on the type of the interface being testing like GUI or API or CLI.

**Internationalization Testing**

Internationalization testing is a type of testing that is performed by software testing team to check the extent to which software can support Internationalization i.e., usage of different languages, different character sets, double byte characters etc., For e.g.: Gmail, is a web application that is used by people all over work with different languages, single by or multi byte character sets.

**Keyword-driven Testing**

Keyword driver testing is more of an automated software testing approach than a type of testing itself. Keyword driven testing is known as action driven testing or table driven testing.

**Load Testing**

Load testing is a type of non-functional testing; load testing is done to check the behaviour of the software under normal and over peak load conditions. Load testing is usually performed using automated testing tools. Load testing intends to find bottlenecks or issues that prevent software from performing as intended at its peak workloads.

**Localization Testing**

Localization testing a type of software testing performed by software testers, in this type of testing, software is expected to adapt to a particular locale, it should support a particular locale/language in terms of display, accepting input in that particular locale, display, font, date time, currency etc., related to a particular locale. For e.g. many web applications allow choice of locale like English, French, German or Japanese. So once locale is defined or set in the configuration of software, software is expected to work as expected with a set language/locale.

**Negative Testing**

This type of software testing approach, which calls out the “attitude to break”, these are functional and non-functional tests that are intended to break the software by entering incorrect data like incorrect date, time or string or upload binary file when text files supposed to be upload or enter huge text string for input fields etc. It is also a positive test for an error condition.

**Non-functional testing**

Software are built to fulfil functional and non-functional requirements, non-functional requirements like performance, usability, localization etc., There are many types of testing like compatibility testing, compliance testing, localization testing, usability testing, volume testing etc., that are carried out for checking non-functional requirements.

**Pair Testing**

**It** is a software testing technique that can be done by software testers, developers or Business analysts (BA). As the name suggests, two people are paired together, one to test and other to monitor and record test results. Pair testing can also be performed in combination of tester-developer, tester-business analyst or developer-business analyst combination. Combining testers and developers in pair testing helps to detect defects faster, identify root cause, fix and test the fix.

**Performance Testing**

**It** is a type of software testing and part of performance engineering that is performed to check some of the quality attributes of software like Stability, reliability, availability. Performance testing is carried out by performance engineering team. Unlike Functional testing, Performance testing is done to check non-functional requirements. Performance testing checks how well software works in anticipated and peak workloads. There are different variations or sub types of performance like load testing, stress testing, volume testing, soak testing and configuration testing.

**Penetration Testing**

**It** is a type of security testing, also known as pen test in short. Penetration testing is done to tests how secure software and its environments (Hardware, Operating system and network) are when subject to attack by an external or internal intruder. Intruder can be a human/hacker or malicious programs. Pen test uses methods to forcibly intrude (by brute force attack) or by using a weakness (vulnerability) to gain access to a software or data or hardware with an intent to expose ways to steal, manipulate or corrupt data, software files or configuration. Penetration Testing is a way of ethical hacking, an experienced Penetration tester will use the same methods and tools that a hacker would use but the intention of Penetration tester is to identify vulnerability and get them fixed before a real hacker or malicious program exploits it.

**Regression Testing**

**It** is a type of software testing that is carried out by software testers as functional regression tests and developers as Unit regression tests. Objective of regression tests are to find defects that got introduced to defect fix (is) or introduction of new feature(s). Regression tests are ideal candidate for automation.

**Retesting**

**It** is a type of retesting that is carried out by software testers as a part of defect fix verification. For e.g. a tester is verifying a defect fix and let us say that there are 3 test cases failed due to this defect. Once tester verifies defect fix as resolved, test will retest or test the same functionality again by executing the test cases that were failed earlier.

**Risk based Testing**

It is a type of software testing and a different approach towards testing a software. In Risk based testing, requirements and functionality of software to be tested are prioritized as Critical, High, Medium and low. In this approach, all critical and high priority tests are tested and them followed by Medium. Low priority or low risk functionality are tested at the end or may not base on the time available for testing.

**Smoke testing**

**It** is a type of testing that is carried out by software testers to check if the new build provided by development team is stable enough i.e., major functionality is working as expected in order to carry out further or detailed testing. Smoke testing is intended to find “show stopper” defects that can prevent testers from testing the application in detail. Smoke testing carried out for a build is also known as build verification test.

**Security Testing**

**It** is a type of software testing carried out by specialized team of software testers. Objective of security testing is to secure the software is to external or internal threats from humans and malicious programs. Security testing basically checks, how good is software’s authorization mechanism, how strong is authentication, how software maintains confidentiality of the data, how does the software maintain integrity of the data, what is the availability of the software in an event of an attack on the software by hackers and malicious programs is for Security testing requires good knowledge of application, technology, networking, security testing tools. With increasing number of web applications necessarily of security testing has increased to a greater extent.

**Sanity Testing**

**It** is a type of testing that is carried out mostly by testers and in some projects by developers as well. Sanity testing is a quick evaluation of the software, environment, network, external systems are up & running, software environment as a whole is stable enough to proceed with extensive testing. Sanity tests are narrow and most of the time sanity tests are not documented.

**Scalability Testing**

**It** is a non-functional test intended to test one of the software quality attributes i.e. “Scalability”. Scalability test is not focused on just one or few functionality of the software instead performance of software as a whole. Scalability testing is usually done by performance engineering team. Objective of scalability testing is to test the ability of the software to scale up with increased users, increased transactions, increase in database size etc., It is not necessary that software’s performance increases with increase in hardware configuration, scalability tests helps to find out how much more workload the software can support with expanding user base, transactions, data storage etc.,

**Stability Testing**

**It** is a non-functional test intended to test one of the software quality attributes i.e. “Stability”. Stability testing focuses on testing how stable software is when it is subject to loads at acceptable levels, peak loads, loads generated in spikes, with more volumes of data to be processed. Scalability testing will involve performing different types of performance tests like load testing, stress testing, spike testing, soak testing, spike testing etc…

**Static Testing** is a form of testing where in approaches like reviews, walkthroughs are employed to evaluate the correctness of the deliverable. In static testing software code is not executed instead it is reviewed for syntax, commenting, naming convention, size of the functions and methods etc. Static testing usually has check lists against which deliverables are evaluated. Static testing can be applied for requirements, designs, and test cases by using approaches like reviews or walkthroughs.

**Stress Testing** is a type of performance testing, in which software is subjected to peak loads and even to a break point to observe how the software would behave at breakpoint. Stress testing also tests the behaviour of the software with insufficient resources like CPU, Memory, Network bandwidth, Disk space etc. Stress testing enables to check some of the quality attributes like robustness and reliability.

**SYSTEM REQUIREMENTS**

**Software Requirements**

* O/S : Windows 7.
* Language : Python
* Front End : Anaconda Navigator – Spyder Notebook

**Hardware Requirements**

* System : Pentium IV 2.4 GHz
* Hard Disk : 200 GB
* Mouse : Logitech.
* Keyboard : 110 keys enhanced
* Ram : 4GB

**MODULES**

* Data Selection and Loading
* Data Preprocessing
* Splitting Dataset into Train and Test Data
* Classification
* Prediction
* Result Generation

**DATA SELECTION AND LOADING**

* Data selection is the process of determining the appropriate data type and source, as well as suitable instruments to collect data.
* Data selection precedes the actual practice of data collection and it is the process where data relevant to the analysis is decided and retrieved from the data collection.
* In this project, the Fake product dataset is used for detecting Fake type prediction.

**DATA PREPROCESSING**

* The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.
* **Missing Data:**   
  This situation arises when some data is missing in the data. It can be handled in various ways.
  + - Ignore the tuples:   
      This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.
    - Fill the Missing values:   
      There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.
* **Encoding Categorical data**: That categorical data is defined as variables with a finite set of label values. That most machine learning algorithms require numerical input and output variables. That an integer and one hot encoding is used to convert categorical data to integer data.
* **Count Vectorizer:** Scikit-learn's CountVectorizer is used to convert a collection of text documents to a vector of term/token **counts**. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.

**BLOCKCHAIN**

Blockchain is a crucial component of the proposed solution for product authentication and anti-counterfeiting. Here's how blockchain can be leveraged in this project:

Distributed Ledger:

The blockchain network serves as a distributed, decentralized ledger to store product information and unique codes.

This ensures the integrity and immutability of the product records, as they are recorded across multiple nodes in the network.

Transparency and Traceability:

The blockchain provides transparency, as all transactions and product records are visible to authorized parties.

This allows for better traceability of products throughout the supply chain, enabling the identification of the source of counterfeit or duplicate goods.

Secure Data Storage:

The blockchain's cryptographic nature ensures the security and confidentiality of the stored product data.

Each product record is stored as a block, with a unique hash linked to the previous block, making it difficult to tamper with the data.

Decentralized Authentication:

The blockchain-based system empowers consumers to independently verify the authenticity of products, without relying solely on the merchant's claims.

The decentralized nature of the blockchain eliminates the need for a centralized authority, providing a more trustworthy and transparent authentication process.

Automated Notifications:

Smart contracts deployed on the blockchain can be used to trigger automated notifications to customers and manufacturers when a product is identified as counterfeit or duplicate.

This ensures timely and reliable communication about product authenticity.

Scalability and Adaptability:

The blockchain network can be designed to scale as the system grows, accommodating an increasing number of products and users.

The modular nature of blockchain technology allows for the integration of additional features and functionality as the project evolves.

**PREDICTION**

Predictive analytics algorithms try to achieve the lowest error possible by either using “boosting” or “bagging”.

**Accuracy** − Accuracy of classifier refers to the ability of classifier. It predict the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.

**Speed** − Refers to the computational cost in generating and using the classifier or predictor.

**Robustness** − It refers to the ability of classifier or predictor to make correct predictions from given noisy data.

**Scalability** − Scalability refers to the ability to construct the classifier or predictor efficiently; given large amount of data.

**Interpretability** − It refers to what extent the classifier or predictor understands.

**RESULT GENERATION**

The Final Result will get generated based on the overall classification and prediction. The performance of this proposed approach is evaluated using some measures like,

* Accuracy

**Accuracy** of classifier refers to the ability of classifier. It predicts the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.

AC=

* Precision

**Precision** is defined as the number of true positives divided by the number of true positives plus the number of false positives.

Precision=

* Recall

**Recall** is the number of correct results divided by the number of results that should have been returned. In binary classification, recall is called sensitivity. It can be viewed as the probability that a relevant document is retrieved by the query.

* ROC

**ROC** curves are frequently used to show in a graphical way the connection/trade-off between clinical sensitivity and specificity for every possible cut-off for a test or a combination of tests. In addition the area under the ROC curve gives an idea about the benefit of using the test(s) in question.

* Confusion matrix

A **confusion**  **matrix** is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

**SOFTWARE DESCRIPTION**

**Python**

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

## **Features of Python**

### **Simple**

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

### **Easy to Learn**

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

### **Free and Open Source**

Python is an example of a FLOSS (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

### **High-level Language**

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

### **Portable**

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, and # -\*- coding: utf-8 -\*-

z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like [Kivy](http://kivy.org) to create games for your computer and for iPhone, iPad, and Android.

### **Interpreted**

This requires a bit of explanation.

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called byte codes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

### **Object Oriented**

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

### **Extensible**

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

### **Embeddable**

You can embed Python within your C/C++ programs to give scripting capabilities for your program's users.

### **Extensive Libraries**

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the Batteries Included philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the [Python Package Index](http://pypi.python.org/pypi).

**FEASIBILITY STUDY**

The feasibility study is carried out to test whether the proposed system is worth being implemented. The proposed system will be selected if it is best enough in meeting the performance requirements.

The feasibility carried out mainly in three sections namely.

**•** Economic Feasibility

• Technical Feasibility

• Behavioural Feasibility

**Economic Feasibility**

Economic analysis is the most frequently used method for evaluating effectiveness of the proposed system. More commonly known as cost benefit analysis. This procedure determines the benefits and saving that are expected from the system of the proposed system. The hardware in system department if sufficient for system development.

**Technical Feasibility**

This study centre around the system’s department hardware, software and to what extend it can support the proposed system department is having the required hardware and software there is no question of increasing the cost of implementing the proposed system. The criteria, the proposed system is technically feasible and the proposed system can be developed with the existing facility.

**Behavioural Feasibility**

People are inherently resistant to change and need sufficient amount of training, which would result in lot of expenditure for the organization. The proposed system can generate reports with day-to-day information immediately at the user’s request, instead of getting a report, which doesn’t contain much detail.

**3.4 Hardware and software requirements:**

**3.4.1 Hardware requirements:**

* System : Pentium IV 2.4 GHz
* Hard Disk : 200 GB
* Ram : 4GB

**3.4.2 Software requirements:**

* O/S : Windows 7.
* Language : Python
* Front End : Anaconda Navigator – Spyder

**3.5 External Interface Requirements:**

**3.5.1 User Interfaces:**

A user interface specification (UI specification) is a document that captures the details of the software user interface into a written document. The specification covers all possible actions that an end user may perform and all visual, auditory and other interaction elements. A user interface (UI) is the space where interactions between humans and machines occur.

In our process, the user choose the input by using tkinter and load or read the input by using panda’s package.

**3.5.2 Hardware Interfaces:**

A hardware interface specifies the plugs, sockets, cables and electrical signals that pass through each line between the CPU and a peripheral device or communications network.

In our process, there is no connection between software and hardware. Here we are used the hardware devices such as mouse, keyboard.

**3.5.3 Software Interfaces:**

Software interfaces (programming interfaces) are the languages, codes and messages that programs use to communicate with each other and to the hardware. Examples are the Windows, Mac and Linux operating systems, SMTP email, IP network protocols and the software drivers that activate the peripheral devices.

In our process, we are using PYTHON language to detect the different types of botnet attacks. Here we are running our process in SPYDER IDE from Anaconda Navigator.

**3.5.4 Communication Interfaces:**

Communications interfaces: wireless or wired technologies are used to connect devices to one another, the Internet, remote servers, etc. From: Internet of Things. In our process, the input dataset as DFAKE PRODUCT dataset. For each of the IoT devices we trained and optimized on 2/3 of its benign data (i.e., the training set of each device). This was done to capture normal network traffic patterns. The test data of each device comprised of the remaining 1/3 of benign data plus all the malicious data. On each test set we applied the respective trained Search Similarity based for prediction as an anomaly detector. The detection of anomalies (i.e., the cyber-attacks launched from each of the above IoT devices) concluded with 100% TPR.

**3.5.4 Feasibility Analysis of the Requirements:**

In our process, the feasibility analysis is, the tool which we are used as Spyder IDE from Anaconda Navigator. The programming language which we are used as Python.

**CONCLUSION**

The research paper is the only smart contract based blockchain system that proposes a fully functional anti-counterfeiting application. Companies will start to adopt it at a very fast pace due to its useful features. Blockchain has been found as a great tool for the purpose of fake product identification and elimination in supply chain management. The proposed system willallow user to easily identify and gather information about the product that they want to check or verify by paying a very low transaction fee, users of this application will no longer need to be worried about the possibility of purchasing an unsusceptible counterfeited product. Manufacturers can use the application to store relevant information on product details inblockchain which can be accessible to anyone in the application network. The total amount of sales that is sold by the distributor, retailers and manufacturers and the number of products currently left in the warehouse are transparent to manufacturing head because of serialized QR code. The consumers can make use of all provided features by the application to immediately perform manufacturer-side verification. The system provides identity verification by using digitalized QR code. This will help users to make better choice in the market and also allow them to trust the seller and the manufacturer. They don’t have to rely on a third party to verify the authenticity of the product which will help in smooth and risk-free experience for them. There are no other means to decrypt the private key of the key owner unless the key owner accidentally leaks the key

**SAMPLE CODE**

**from flask import Flask, render\_template, request, redirect, url\_for, session, flash**

**from index import BlockChain**

**import json**

**import smtplib as smtp**

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

**app.secret\_key = "alkdjfalkdjf"**

**import smtplib**

**def send\_email():**

**# Email configuration**

**smtp\_server = 'smtp.gmail.com'**

**smtp\_port = 465 # For SSL**

**sender\_email = 'sathyakumar17112022@gmail.com'**

**receiver\_email = 'mahalakshmitg21@gmail.com'**

**password = 'ncfplwzacztjnxyp' # Your email password**

**# Create a secure SSL connection to the SMTP server**

**connection = smtplib.SMTP\_SSL(smtp\_server, smtp\_port)**

**try:**

**# Login to the email server**

**connection.login(sender\_email, password)**

**# Construct the email message**

**subject = 'Fake Product Alert'**

**body = 'Dear recipient,\n\nThis is to inform you about a fake product.\n\nBest regards,\nSender'**

**message = f'Subject: {subject}\n\n{body}'**

**# Send the email**

**connection.sendmail(sender\_email, receiver\_email, message)**

**print("Email sent successfully!")**

**except Exception as e:**

**print(f"Error sending email: {e}")**

**finally:**

**# Close the connection**

**connection.close()**

**# Test the send\_email function**

**send\_email()**

**@app.route("/")**

**def home():**

**if session.get("user"):**

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

**else:**

**flash("Please login to access Verifier")**

**return redirect(url\_for('login'))**

**@app.route("/login", methods=["POST", "GET"])**

**def login():**

**if request.method == "POST":**

**user = request.form["username"]**

**pswd = request.form["password"]**

**if user == "Admin":**

**if pswd == "password":**

**session["user"] = "Admin"**

**#return redirect(url\_for("admin"))**

**return redirect(url\_for("home"))**

**elif user == "Nike":**

**if pswd == "password":**

**session["user"] = "Nike"**

**#return redirect(url\_for("shoes"))**

**return redirect(url\_for("check"))**

**elif user == "Musigny":**

**if pswd == "password":**

**session["user"] = "Musigny"**

**#return redirect(url\_for("dress"))**

**return redirect(url\_for("check"))**

**elif user == "Lupin":**

**if pswd == "password":**

**session["user"] = "Lupin"**

**#return redirect(url\_for("medicine"))**

**return redirect(url\_for("check"))**

**elif user == "Kisan":**

**if pswd == "password":**

**session["user"] = "Kisan"**

**#return redirect(url\_for("fertilizer"))**

**return redirect(url\_for("check"))**

**else:**

**flash("Invalid Login details")**

**return redirect(url\_for('login'))**

**else:**

**return render\_template('login.html')**

**@app.route("/verify/<kid>", methods=["GET"])**

**def verify(kid):**

**return render\_template('verify.html', keyId=kid)**

**@app.route("/verify", methods=["POST"])**

**def success():**

**post\_data = request.form["keyId"]**

**with open('./NODES/N1/blockchain.json', 'r') as bfile:**

**n1\_data = str(bfile.read())**

**with open('./NODES/N2/blockchain.json', 'r') as bfile:**

**n2\_data = str(bfile.read())**

**with open('./NODES/N3/blockchain.json', 'r') as bfile:**

**n3\_data = str(bfile.read())**

**with open('./NODES/N4/blockchain.json', 'r') as bfile:**

**n4\_data = str(bfile.read())**

**pd = str(post\_data)**

**if (pd in n1\_data) and (pd in n2\_data) and (pd in n3\_data) and (pd in n4\_data):**

**with open('./NODES/N1/blockchain.json', 'r') as bfile:**

**for x in bfile:**

**if pd in x:**

**a = json.loads(x)["data"]**

**b = a.replace("'", "\"")**

**data = json.loads(b)**

**product\_brand = data["Manufacturer"]**

**product\_name = data["ProductName"]**

**product\_batch = data["ProductBatch"]**

**manuf\_date = data["ProductManufacturedDate"]**

**expiry\_date = data["ProductExpiryDate"]**

**product\_id = data["ProductId"]**

**product\_price = data["ProductPrice"]**

**product\_size = data["ProductSize"]**

**product\_type = data["ProductType"]**

**return render\_template('success.html', brand=product\_brand, name=product\_name, batch=product\_batch, manfdate=manuf\_date, exprydate=expiry\_date, id=product\_id, price=product\_price, size=product\_size, type=product\_type)**

**else:**

**# Call send\_email function from mail.py**

**send\_email()**

**return render\_template('fraud.html', message="Fake Product")**

**@app.route("/addproduct", methods=["POST", "GET"])**

**def addproduct():**

**if request.method == "POST":**

**brand = request.form["brand"]**

**name = request.form["name"]**

**batch = request.form["batch"]**

**pid = request.form["id"]**

**manfdate = request.form["manfdate"]**

**exprydate= request.form["exprydate"]**

**price = request.form["price"]**

**size = request.form["size"]**

**ptype = request.form["type"]**

**print(brand, name, batch, manfdate, exprydate, pid, price, size, ptype)**

**bc = BlockChain()**

**bc.addProduct(brand, name, batch, manfdate, exprydate, pid, price, size, ptype)**

**flash("Product added successfully to the Blockchain")**

**# return render\_template('home.html')**

**return redirect(url\_for('home'))**

**else:**

**# return render\_template('home.html')**

**return redirect(url\_for('home'))**

**@app.route("/admin")**

**def admin():**

**if session["user"] == "Admin":**

**return render\_template('admin.html')**

**else:**

**return redirect(url\_for('login'))**

**@app.route("/verifyNodes")**

**def verifyNodes():**

**bc = BlockChain()**

**isBV = bc.isBlockchainValid()**

**if isBV:**

**flash("All Nodes of Blockchain are valid")**

**return redirect(url\_for('admin'))**

**else:**

**flash("Blockchain Nodes are not valid")**

**return redirect(url\_for('admin'))**

**@app.route("/medicine")**

**def medicine():**

**return render\_template('MedicinePage.html')**

**@app.route("/fertilizer")**

**def fertilizer():**

**return render\_template('FertilizersPage.html')**

**@app.route("/shoes")**

**def shoes():**

**return render\_template('ShoesPage.html')**

**#check**

**@app.route("/check")**

**def check():**

**return render\_template('check.html')**

**@app.route("/dress")**

**def dress():**

**return render\_template('dressPage.html')**

**@app.route("/logout")**

**def logout():**

**session["user"] = ""**

**return redirect(url\_for('login'))**

**if \_\_name\_\_ == "\_\_main\_\_":**

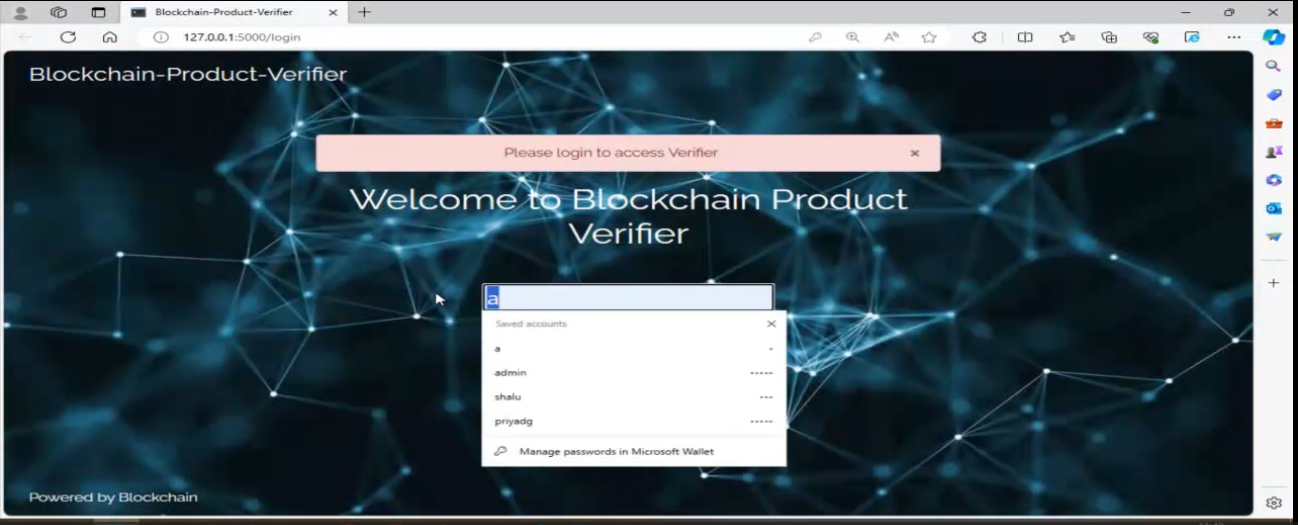
**app.run(debug=True)**

**session["user"] = ""**

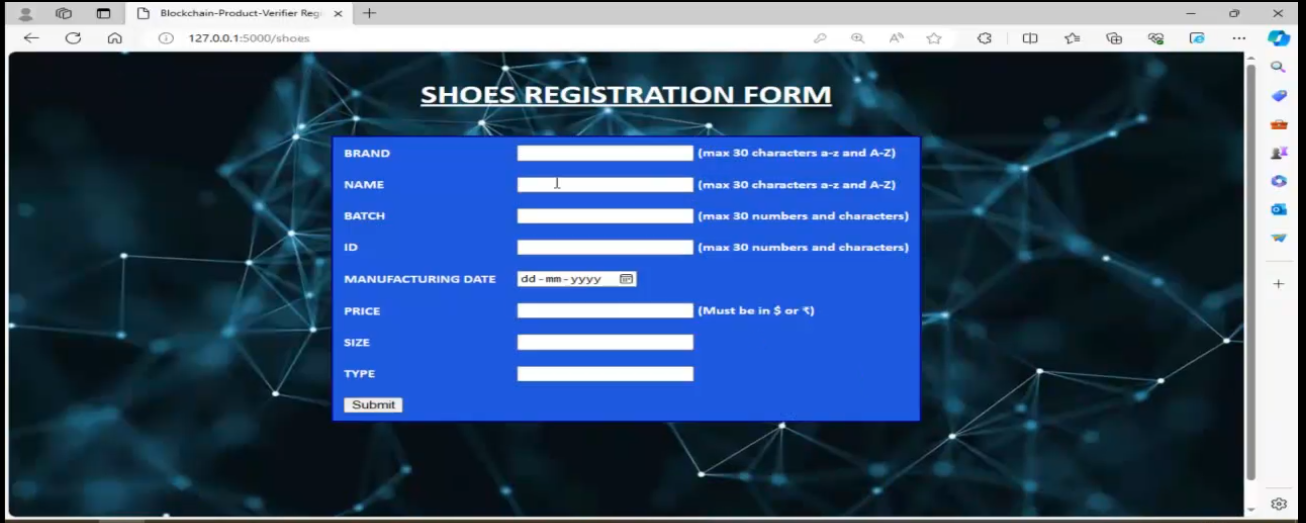
**SAMPLE SCREENSHOT**

**SCREENSHOT**

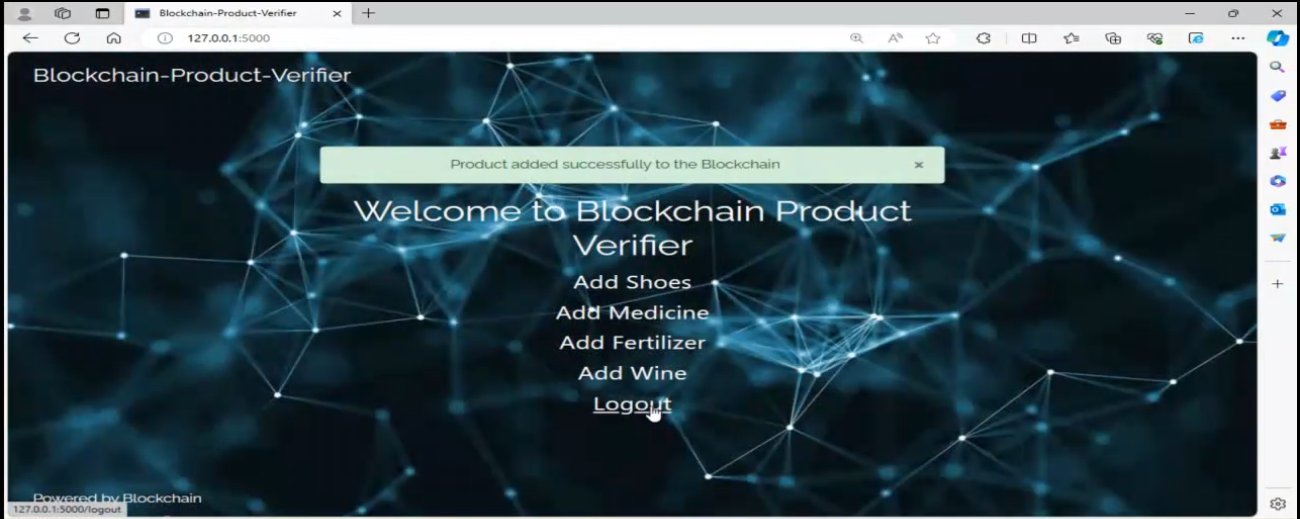
**SAMPLE SCREENSHOT**



**Figure 1 Home page admin on Fake product Blockchain**



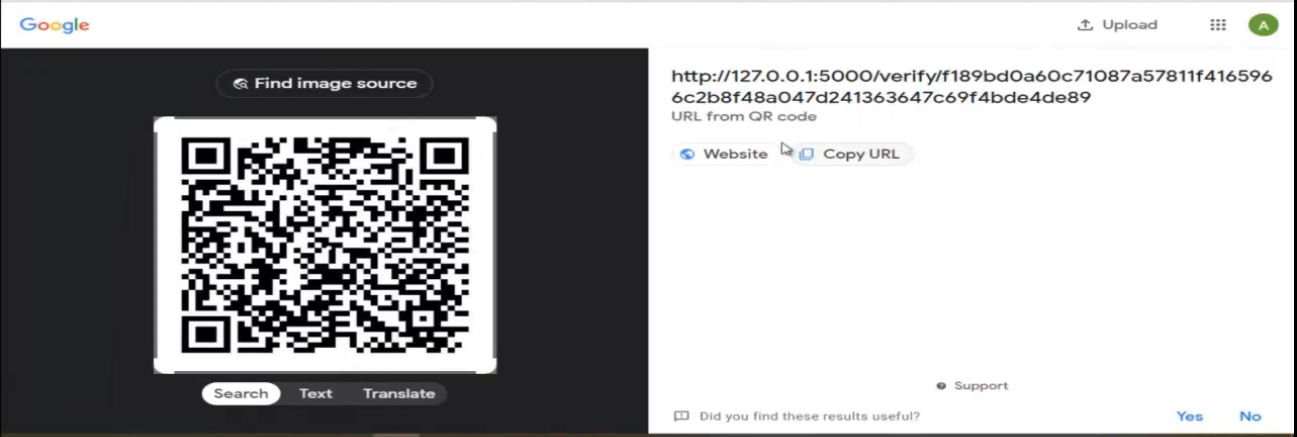
**Figure 2 Shoes Register Form on Fake product Blockchain**



**Figure 3 Verified on Fake product Blockchain**



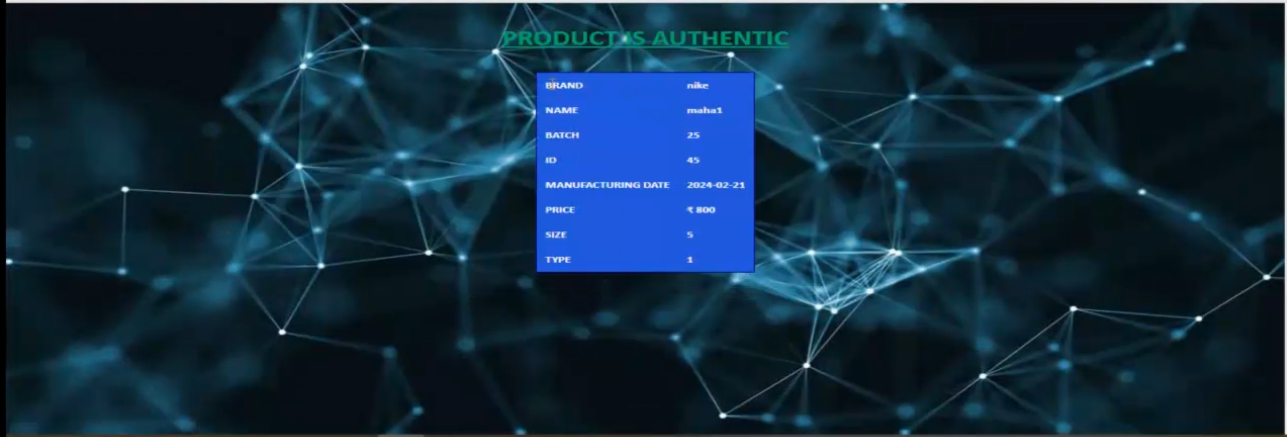
**Figure 4 QR Code Generated on Fake product Blockchain**



**Figure 5 Scanning based QR Code Generated on Fake product Blockchain**



**Figure 6 Verified based hash data on Fake product Blockchain**



**Figure 7 Verified based product on Fake product Blockchain**



**Figure 8 Verified based Fake product Blockchain on QR code**



**Figure 9 Verified based Fake product**

**1.5 References:**

* [1] Pathan A-SK, Azad S, Khan R, et al. Security mechanisms and data access protocols in innovative wireless networks. London: Sage; 2018.
* [2] Yong-xiong Z, Liang-ming W, Lu-xia Y. A network attack discovery algorithm based on unbalanced sampling vehicle evolution strategy for intrusion detection. Int J Comput Appl. 2017: 1–9.
* [3] Zargar ST, Joshi J, Tipper D. A survey of defense mechanisms against distributed denial of service (DFAKE PRODUCT) flooding attacks. IEEE Commun Surv Tutorials. 2013;15(4):2046–2069.
* [4] Toledo AL, Wang X. Robust detection of MAC layer denial-of-service attacks in CSMA/CA wireless networks. IEEE Trans Inf Forensics Secure. 2008;3(3):347–358.
* [5] Guo Y, Ten CW, Hu S, et al. Modeling distributed denial of service attack in advanced metering infrastructure. 2015 IEEE power & energy society innovative smart grid technologies conference (ISGT); 2015. p. 1–
* A. A. Khan, M. H. Rehmani, and M. Reisslein, ‘‘Cognitive radio for smart grids: Survey of architectures, spectrum sensing mechanisms, and networking protocols,’’ IEEE Commun. Surveys Tuts., vol. 18, no. 1, pp. 860–898, 1st Quart., 2016.
* Y. Lin, X. Zhu, Z. Zheng, Z. Dou, and R. Zhou, ‘‘The individual identification method of wireless device based on dimensionality reduction,’’ J. Supercomput., vol. 75, no. 6, pp. 3010–3027, Jun. 2019.
* T. Liu, Y. Guan, and Y. Lin, ‘‘Research on modulation recognition with ensemble learning,’’ EURASIP J. Wireless Commun. Netw., vol. 2017, no. 1, p. 179, 2017.
* Y. Tu, Y. Lin, J. Wang, and J.-U. Kim, ‘‘Semi-supervised learning with generative adversarial networks on digital signal modulation classification,’’ Comput. Mater. Continua, vol. 55, no. 2, pp. 243–254, 2018.
* C. Shi, Z. Dou, Y. Lin, and W. Li, ‘‘Dynamic threshold-setting for RFpowered cognitive radio networks in non-Gaussian noise,’’ EURASIP J. Wireless Commun. Netw., vol. 2017, no. 1, p. 192, Nov. 2017.
* Z. Zhang, X. Guo, and Y. Lin, ‘‘Trust management method of D2D communication based on RF fingerprint identification,’’ IEEE Access, vol. 6, pp. 66082–66087, 2018.
* H. Wang, J. Li, L. Guo, Z. Dou, Y. Lin, and R. Zhou, ‘‘cFractal complexitybased feature extraction algorithm of communication signals,’’ Fractals, vol. 25, no. 4, pp. 1740008-1–1740008-3, Jun. 2017.
* J. Zhang, S. Chen, X. Mu, and L. Hanzo, ‘‘Evolutionary-algorithm-assisted joint channel estimation and turbo multiuser detection/decoding for OFDM/SDMA,’’ IEEE Trans. Veh. Technol., vol. 63, no. 3, pp. 1204–1222, Mar. 2014.
* J. Zhang, S. Chen, X. Guo, J. Shi, and L. Hanzo, ‘‘Boosting fronthaul capacity: Global optimization of power sharing for centralized radio access network,’’ IEEE Trans. Veh. Technol., vol. 68, no. 2, pp. 1916–1929, Feb. 2019.
* J. Wen, Z. Zhou, Z. Liu, M.-J. Lai, and X. Tang, ‘‘Sharp sufficient conditions for stable recovery of block sparse signals by block orthogonal matching pursuit,’’ Appl. Comput. Harmon. Anal., vol. 47, pp. 948–974, Nov. 2019.
* J. Wen and X.-W. Chang, ‘‘On the KZ reduction,’’ IEEE Trans. Inf. Theory, vol. 65, no. 3, pp. 1921–1935, Mar. 2019.
* Z. Dou, C. Shi, Y. Lin, and W. Li, ‘‘Modeling of non-gaussian colored noise and application in cr multi-sensor networks,’’ EURASIP J. Wireless Commun. Netw., vol. 2017, no. 1, p. 192, Nov. 2017.
* M. Liu, J. Zhang, Y. Lin, Z. Wu, B. Shang, and F. Gong, ‘‘Carrier frequency estimation of time-frequency overlapped MASK signals for underlay cognitive radio network,’’ IEEE Access, vol. 7, pp. 58277–58285, 2019.
* Y. Lin, Y. Li, X. Yin, and Z. Dou, ‘‘Multisensor fault diagnosis modeling based on the evidence theory,’’ IEEE Trans. Rel., vol. 67, no. 2, pp. 513–521, Jun. 2018.
* H. Wang, L. Guo, Z. Dou, and Y. Lin, ‘‘A new method of cognitive signal recognition based on hybrid information entropy and DS evidence theory,’’ Mobile Netw. Appl., vol. 23, no. 4, pp. 677–685, Aug. 2018.
* W. Wei and J. M. Mendel, ‘‘Maximum-likelihood classification for digital amplitude-phase modulations,’’ IEEE Trans. Commun., vol. 48, no. 2, pp. 189–193, Feb. 2000.
* A. Swami and B. M. Sadler, ‘‘Hierarchical digital modulation classification using cumulants,’’ IEEE Trans. Commun., vol. 48, no. 3, pp. 416–429, Mar. 2000.
* N. Mahmoudi and E. Duman, ‘‘Detecting credit card fraud by modified Fisher discriminant analysis,’’ Expert Syst. Appl., vol. 42, no. 5, pp. 2510–2516, Apr. 2015.
* L. Wu, C. Shen, and A. van den Hengel, ‘‘Deep linear discriminant analysis on Fisher networks: A hybrid architecture for person re-identification,’’ Pattern Recognit., vol. 65, pp. 238–250, May 2017.
* S. Srivastava and M. R. Gupta, ‘‘Distribution-based Bayesian minimum expected risk for discriminant analysis,’’ in Proc. IEEE Int. Symp. Inf. Theory, Jul. 2006, vol. 1, no. 1, pp. 2294–2298.