Email: editor@ijerst.com or editor.ijerst@gmail.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Vol. 21, Issue 1, 2025

Drug Supply Chain Management using the Django Framework and Distributed Ledger Technology

[1] Malleboina Pravalika, [2] Chelimala Rohith Reddy, [3] P.R.Jyoshna, [4] Neelam Ashok Kumar, [5] Matla Ranadheer, [6] Mr.R.Manoj Kumar, [7] Dr.K.Eswaramoorthy,

[1,2,3,4,5]UG Scholar, Dept. of CS, Narsimha Reddy Engineering College, Maisammaguda, Kompally, Secunderabad, India.

[6] Assistant Professor, Dept. of CSE, Narsimha Reddy Engineering College, Maisammaguda, Kompally, Secunderabad, India.

[7] Professor, Dept. of EEE, Narsimha Reddy Engineering College, Maisammaguda, Kompally, Secunderabad, India.

Abstract—

By suggesting a DLT-based technique to ease and expedite the creation of such applications, this study offers a general-purpose solution to medication supply chain management. A system has been built using the Django Python framework, the Ethereum blockchain, and the web3.py library. It can be customized to fit most real-life supply chains and automatically generates the necessary applications, such as database schema, smart contracts, and apps, for specific domains like drug management and traceability. In order to demonstrate the efficacy of this method, a case study including a basic medication tracking system from the manufacturer to the hospital wards is detailed.

Substitute: Django, Drug Supply Chain, Smart Contracts, Permissioned Blockchain

I. INTRODUCTION

Current methods for managing the supply chains of pharmaceuticals and medical devices are tried and true. Nevertheless, at this time, we cannot promise that the data will remain unchangeable, even after software upgrades have been implemented. This might be due to unauthorized access to the database or other factors. At the moment, just a few of companies handle system management, and human administrators oversee both data and software programs; nevertheless, none of these factors is absolutely certain.

These systems cannot be made more efficient, reliable, or fast using blockchain technology. Nevertheless, there are various ways in which public and permissioned blockchain technology can enhance trust. For example: • A consensus system oversees the ledger, and any changes must be approved by all parties involved. • Data stored on the blockchain is immutable due to cryptography. • Smart contracts (SCs) on the blockchain are transparent and cannot be changed. • Cryptography is used to verify the identities of transaction senders, making signed transactions legally binding. • The system is transparent because recorded information can be accessible to everyone or selected actors. Medical supply management systems may greatly benefit from blockchain registrations, which increase their integrity. A blockchain-based methodology for a general-purpose approach to controlling the medication supply chain is proposed in this research to ease and enhance the efficiency of the production of such applications. Most of these supply chains may be configured using this system, which generates the database schema, SCs, and the applications that communicate with them automatically. The foundation of this system is a generic SC that is tailored to the particular system that needs management. It is further supported by applications that communicate with these SCs. Django with the web3.py Python module allowed us to automate the database creation and app management processes.

We provide a case study of a basic medication traceability system that goes from the manufacturer to the hospital wards to show how the method works. Since blockchains, even permissioned ones, are far slower than modern transactional systems and blockchain software development is more costly, we

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Vol. 21, Issue 1, 2025

restrict blockchain's application to its strongest suit: trust. All other calculations and duties are handled off-chain by a standard information system that uses a relational database and the Python programming language.
a) What this work contributes: to save the reader the trouble of beginning from square one every time they want to construct a blockchain application for medication management; to do this, it develops customizable, assembleable building blocks. Here are the key points of what we have accomplished.
• A model that incorporates trustworthy certifications and third-party interventions; transparent and unchangeable certificates of origin and quality; safe commerce and payments; and management of transformations, manufacturing/storage locations, shipping, and other related matters.
• An adaptable, straightforward model for managing individual pharmaceutical supply chain systems. The system, including the database, GUI, and SCs, can be automatically created from requirements written in a tabular language. There is a method to reduce implementation costs and build securely on top of an existing framework that has been through a rigorous security audit. To evaluate the strategy, a case study is utilized.

b) The paper's structure: What follows is the outline for the rest of the article. In Section II, we go over the background and any relevant work. Section III-A presents the proposed approach; Section III-B describes the entities involved; Section III-C describes the Django models; Section III-D describes the SCs deployed; and Section III-E describes the user interface (UI) necessary to interact with the system. Section IV presents a case study that demonstrates the technique via an examination of a basic producer-to-hospital-ward traceability system. Section V makes a few assertions.

II. BACKGROUND AND RELATED WORK

In the absence of a central authority figure, a distributed network of computers may execute an application programmatically known as a decentralized application (DApp). On the DLT network, a decentralized application (DApp) is both developed and utilized.
Because of this, it is decentralized, transparent, unchangeable, and deterministic. Separate programs, or SCs, are what make up a DApp and execute on every node in the distributed ledger technology (DLT). There are usually server components used to store data that cannot be stored on a DLT, and a graphical user interface (GUI) that lets users connect with the DApp. The usual architecture and components of a DApp are described in depth in [1]. Among the many areas that have embraced DApps, the one we'll be discussing today is the management and distribution of drugs via supply chains.
A drug management system's primary goals are as follows: • to record, in a secure and transparent manner, all events pertaining to the production and delivery of drugs; • to enable auditing and certification of all stages of production by supervisory authorities; • to record both manually and automatically by Internet of Things devices; • to track the locations and shipments of drugs, potentially including measurements of the storage and transport environments; • to record the current and previous owners of the products.

There is a tried and true mechanism in place for managing the supply chain for medical equipment and pharmaceuticals. Their efficiency, productivity, and resilience will not be enhanced by DLT technology. The arguments stated in the introduction show that it may, nevertheless, increase trust. Within this framework, academic articles discussing the integration of DLT into the pharmaceutical supply chain are starting to surface. More specifically, several studies have investigated the potential of blockchain technology to improve medication management and traceability systems since 2016. You may find up-to-date literature evaluations in [2] and [3]. As far as we are aware, our work is the first to suggest using a system that can be automatically generated from a description of the specific supply chain involved in tracking the provenance and ownership of drugs and preventing fraud. This was determined by comparing our work to other papers that deal with the use of DApps and blockchain for this purpose. Section A: Django's Foundation Introduced in 2005, Django is an open-source framework that helps developers build web applications. It's free and available to everybody [4]. To facilitate user interaction with data stored in a database, web applications display a collection of HTML pages. Defining the data that needs management is the first step in developing a Django application. To do this, the Django Object-Relational Mapping (ORM) libraries are used to create "Models," which are essentially database tables. The entries in these tables are then mapped to Python objects. Python functions or methods called "Views" are associated with the URLs of the HTML pages and are called when the browser reaches the related URL.

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Vol. 21, Issue 1, 2025

These Views communicate with the instances of the Model and carry out their operations. The Views are linked to HTML "Templates" that govern cycles and choices and communicate data with their Views using embedded instructions written in DTL (Django Template Language). Developers don't have to write code for administration capabilities given by Django, such as user management, model editing, and responsive behavior.

III. METHODOLOGY

A lot of the supply chains involved in medication management are rather similar. Raw ingredients, which are processed into packaged pharmaceuticals, usually enter the system first in the supply chain. Occasionally, the supply chain is bypassed entirely by packaged medications purchased from third-party vendors. Intermediate distributors or final consumers get these packaged medications. All shipments, whether made directly or via a forwarder, must be verified and traceable in accordance with strict quality standards to guarantee, for instance, the appropriate storage temperature of medications. A private customer purchasing medications from a pharmacy or a hospital ward usually constitutes the end user. Every one of these phases has the potential to host process-relevant events. In order to identify the key players, ideas, and entities involved in medication management production and distribution systems, as well as their connections and relevant events, we conducted a study using object-oriented design and analytic approaches. Here we lay out our strategy and demonstrate how the traceability system may be automatically developed, which cuts down on expenses and time. All of the software development was done using an agile methodology known as Agile BlockChain DApp Engineering (ABCDE) [5].

A. Architecture

Since Ethereum is the most popular and well-established distributed ledger technology (DLT), we used it, and more especially, its permissioned counterparts like Hyperledger Besu. Potentially, additional DLTs might be included into our method along the road. The design of the DApp system adheres to the principles outlined in [1]. The server system, developed in Python with Django, is the meat and potatoes of the system. Figure 1 is a UML deployment and component diagram that explains the architecture in detail, with the Django components highlighted. Main is where you'll find the Django engine and all of its Python modules. This package contains the Django Models and Views modules. They are shown in blue in the illustration beside the

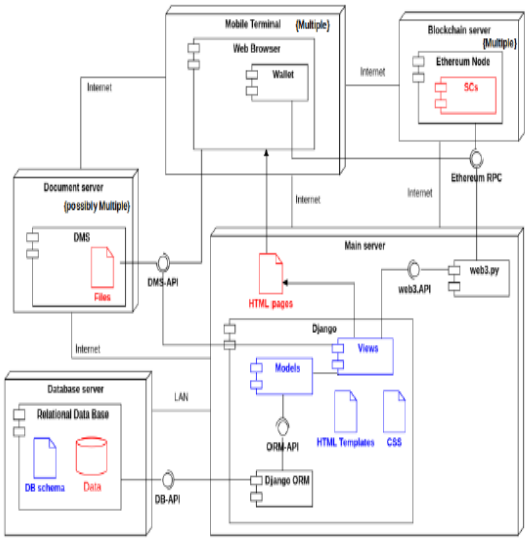

Fig. 1: The software architecture of the proposed system. In blue the components and artifacts common to all supply chains,in red the data that configure and manage a specific implementation.

Web page templates and style sheet files required to define the system's web pages. All systems use the same models to explain the data structure of the generic entities. The models are provided in the following section. Upon initial system startup, the schema for the relational database that Django uses to store this data is produced by the Database server. Information on the setup of the particular system is stored in a few tables. During system execution, these tables and others will store data. Python routines called Views provide graphical user interface capability; they make use of CSS and HTML templates to dynamically show information on the operators' mobile terminals and collect their feedback. These methods communicate with the Model's data, and a few Views can transmit blockchain transactions over the web3.py standard interface, as well as receive and send documents via the Document server on this example, Django uses its object relational mapping (ORM) component to read from and update the relational database, which is MySQL, stored on the database server. Authenticated users may access the documents (files) stored on the document server. Actually, several entities may control and maintain document servers, each of which can be recognized by its unique URL. There

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Vol. 21, Issue 1, 2025

are many different kinds of cloud storage, ranging from basic online file systems to repository services like Google Drive or Dropbox to full-fledged document management systems like Alfresco. For any entity that oversees the permissioned blockchain, there are several Blockchain servers that run Ethereum (or Hyperledger Besu) nodes. All of them include an identical replica of the system's SCs and the blockchain. The system may be accessed by mobile terminals using web browsers that use the HTML 5 language. For users that are part of the system's administration, their browser includes an Ethereum wallet plugin that allows them to save and submit transactions to the blockchain using the private key of their operator's address. Documents are uploaded to the document server and their links and hash signatures are recorded in the blockchain. Operators enter data into the system, which is then written to the database and the blockchain. The graphic displays this facts in red.

B. Objects of the system

We performed a thorough analysis of drug management systems through literature and interviews with pharmacy managers of local hospitals, and identified the relevant concepts of these systems. As in many other supply chains, there are organizations using the system, locations, various kinds of human and IoT actors. Each actor, individual or device, is identified by a unique address able to send transactions and queries to the blockchain. Other key concepts are of course the raw materials and semi-finished products used to manufacture a drug, and the final packaging of drugs. Most of these objects are associated with a SC which keeps track of the events linked to it. An event is something that occurs over time. We identified and managed the relevant events that can occur to the system, such as registration into the system of materials and products, transformation, packaging, shipment, purchase, certification of a product. Events are not associated to SCs, but are recorded in the ”storage” of the SC associated to the corresponding object. A more detailed analysis of these concepts, there called ”entities”, together with a full UML class diagram showing their relationships and attributes, can be found in [6]

TABLE I: Workflow of supply chain management generation.

Step	Description	Output
1	Analysis of the supply chain, with the identification of organizations, roles, operators, products, relevant events, etc.	A document describing the supply chain
2	Writing files with the description of the system	The .csv files with the needed description of the system, in tabular form, which are input to the generator
3	Starting a scratch copy of Django.	A Django project, or app, is created to manage the system, copying Model, View and Template files. The Views are associated with proper URLs. The DB is created.
4	Automatic generation of the population of the DB and of the SCs describing the system.	The generator parses the files and outputs: (i) SQL code to populate the DB with organizations, operators, product data, event data; (ii) Python code to create the SCs of the system. SQL and Python codes are run and the system is populated.
5	Customization of HTML Templates of the system with the styles chosen by the organizations involved.	The CSS files are customized and tested according to the wishes of the organizations involved.
6	Setup of the system by the system administrator, enabling operators and auditors and linking them with their blockchain address.	Operators and auditors receive a message to connect to the system and set their passwords.
7	The system is up and running.	Operators, auditors and possibly customers can connect to the system's URL using their mobile terminals, and start operations.

C. Models Definition

Our system generator takes in the supply chain configuration systems and generates two kinds of output: (i) SQL code to add supply chain data (organizations, operators, locations, product types, and events) to the database, and (ii) Python code to build the smart contracts that use the distributed ledger technology (DLT) component of the system. The HTML pages and designs may also be customized to suit the needs of individual clients. After that, the supply system is constructed by executing the produced Python and SQL code.

In the graphic, the code and data that are relevant are highlighted in red. The procedure that resulted in the creation and rollout of our system is shown in Table I.To help you organize and deal with your application's data, the Django framework offers Models, an abstraction layer. Everything about the data you're storing, including its fields and attributes, is part of the model. It is common practice to associate one database table with each model. For every model attribute, there is a corresponding table

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Vol. 21, Issue 1, 2025

column. With Django, you get a database access API and the accompanying database schema is created automatically. Python classes that inherit from django.db.models are called models. Model. What follows is an example of the Product class and model in Django. Aside from defining the ProductState enumeration, it also defines the product's property. The Django ForeignKey field lets you handle many-to-one connections in an easy, efficient, and transparent manner; it's used to implement the relationships between products, their events, and their owners or locations.

```
class Product(models.Model):
  Pr_State = [
     ("PR", "Produced"),
     ("PK", "Packed"),
     ("TR", "In Transit"),
     ("EG", "Expiring"),
     ("EX", "Expired"),
     ("US", "Used"),
  ]
  id = models.AutoField(primary_key=True)
  type = models.ForeignKey(ProductData)
  notes = models.CharField(max_length=256, null = True)
  organization = models.ForeignKey('Organization')
  location = models.ForeignKey('Location', null = True)
  events = models.ForeignKey('Event', null = True)
  quantity = models.FloatField()
  initialQuantity = models.FloatField(default = quantity)
  state = models.CharField(max_length=2, choices = Pr_State)
  bcaddress = models.CharField(max_length=64, null = True)
```

Django automatically creates the database schema in step 3 of Table I after the models are established. In step 4, when the Django project is started and initialized, our automatic generator reads the.csv files that describe the supply chain to be managed and generates the SQL code needed to load the system with the correct data.

D. Smart Contracts

System SCs are designed and generated in a similar fashion to the Django system. Every system uses the same design for the SCs. To keep things brief, we look up the whole analysis of this design in [6]. Every business, site, customer, and product has its own SC. All of the organization's users, locations, and products are linked to in the SC's mapping. Many raw ingredients to be processed into packaged medications or many finished drugs (e.g., lots manufactured and recorded on a certain day) might be represented by a Product. In the SC of every Product you'll find a mapping that stores all the events related to that Product that have been recorded by a certain User. Its physical location is also related to it. Upon application of a QR code to the finished product (the packaged medicine), data pertaining to its background may be retrieved. You may do this by going through each event in (backwards) registration order and extracting the data you need, such as the documents' URLs and their probable hash digest. You may also see which Organization made a certain product.

We optimize the code using the OpenZeppelin libraries [7], which are a toolkit for creating, updating, deploying, and interacting with Smart Contracts. Ownable is a contract module that every SC inherits from. It lets you provide the smart contract's owner address and regulate which transactions may only be executed from this address, giving you a rudimentary way to limit who can access the contract. We considered some recommended practices to prevent vulnerabilities, using the ABCDE technique. Specifically, we implemented security and gas saving best practices in a methodical manner [8].

We also execute a Python script that uses the web3.py package to construct the SCs for the particular supply chain by reading the.csv files during step 4 of Table I. The addresses kept in the database connect the SCs to the Django Views.

E. User Interface

The HTML5 pages are used to implement the user interface. A view is a component of the Django framework that encapsulates the logic that processes a user's request and returns the result. Python functions called views are used to process and return web requests and responses. This answer may take several forms, including images, XML documents, 404 errors, redirects, HTML page contents, and so forth. As mentioned earlier, the entities and events that make up a drug supply chain are widely used and may include the majority of manufacturing steps. As a result, the programs that enable them to enter, update (where permitted), and retrieve are likewise standard, and their user interface (UI) may be built automatically from the system description. To be more specific, the operator can create and edit events by dynamically generating the HTML code sent to the browser based on the views of our Django system and the related HTML templates after a specific production process's events have been defined with their data, constraints, and authorizations. The user interface (UI) design and look are obviously modifiable, but the data entry, after passing all necessary tests, does not need any further programming. As we go on to the following

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Vol. 21, Issue 1, 2025

area, Fig. 3 serves as an example of the system GUI. The view makes use of the Web3.py Python framework to facilitate interactions with the blockchain and its SCs. Web3.py enables reading of block data, signing and sending of transactions, deployment and interaction with contracts, and much more besides.

IV. CASE STUDY: A SIMPLE DRUG TRACEABILITY SYSTEM FROM A PRODUCER TO HOSPITAL WARDS

The goal of this case study is to demonstrate how a simple distributed ledger technology (DLT) medication traceability system can monitor the movement of drug lots from their point of origin (PharmaCompany1) to their final destination (DrugDistributor1) and finally to a hospital. Figure 2 depicts a simplified version of the supply chain. The two medications that the manufacturer makes—an antibiotic and a vaccine—need to be stored at temperatures below -30°C (-22°F). One such firm is DrugForwarder1, which transports various drug batches from one place to another. The forwarder's trucks, the distributor's warehouse, and the hospital pharmacy all have Internet of Things (IoT) temperature sensors that record the temperature and add it to the distributed ledger technology (DLT) at regular intervals. There are two wards in the hospital that will be receiving medicine batches via internal conferral, in addition to the pharmacy. The whole process is audited by a government entity, namely the FDA, which generates certifications on a periodic basis and also has them recorded on the DLT.

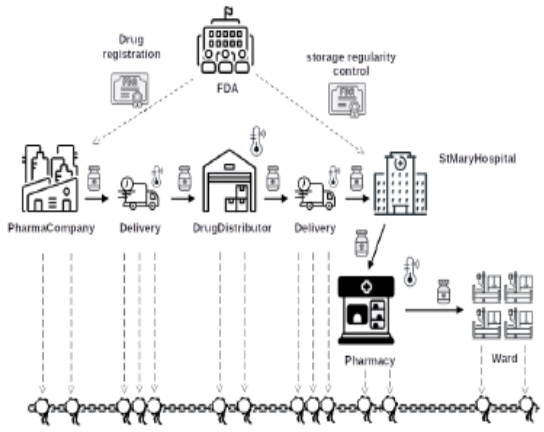

Fig. 2: A typical drug supply chain.

The supply chain is managed by personnel of all participating firms who also enter system events.

The Djiango Python framework was used to automatically construct the database, smart contracts, and GUI based on the supply chain specifications provided in.csv files. We just provide a small excerpt from these.csv files, which include the column names in the header, so that the report is brief. We have.csv files for a variety of things, including: • The actors' roster. The kind, name, and actor ID for every actor are defined. Take note of the last actor type; it stands for an Internet of Things device:

```
id,name,type
OP,Firm Operator,OP
CF,Drug Certifier,AU
AP,Apothecary,HE
TS,Temperature sensor,DV
```

A rundown of the groups taking part in the procedure. It should be noted that each hospital department or ward that receives supplies is listed as a separate entity.

```
id,name,type
MA1,PharmaCompany1,MF
FW1,DrugForwarder1,FW
FDA,Food&DrugAdministration,GA
HPH,StMaryHospPharmacy,PH
```

We recorded the following details for every product type that was part of the production process: id, name, description, type ("RW" for raw material, "DR" for packaged drug), and unit of measure:

```
id,name,description,type,unitOfMeas
DR1,CovVax,Covid Vaccine,DR,Dose
DR2,Lovoflox,Levofloxacin antibiotic,DR,Box
```

For the DLT to process transactions, it needs a list of approved operators as well as a list of Internet of Things devices that are able to do so. Each one has the DLT address that the related transactions are sent from. In this report, we just provide excerpts from the.csv files. The report is organized as follows: IoT devices file first, then human users file:

```
id,type,organiz.,model,SCaddress
TE1,DV,FW1,th-A34,0xA4bF310ca755e3C2B78...
```

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Vol. 21, Issue 1, 2025

```
% TE4,DV,DD1,th-A34,0xC63e181e305B34363DC...
% TE5m,DV,HPH,W0810,0x90903104617914931E2...
% . . .

% id,type,org.,name,surname,email,Scaddress
% UM1,OP,MA1,Joe,Ross,jros@phacmp.com,0x...
% AU1,CF,FDA,Ann,Zhang,annz@fda.gov,0x...
UF1,CW,FW1,Tina,Cao,caotn@drfwr.com,0x...
AP1,AP,HPH,Mary,Lee,mlee@stmhp.com,0x...
DR1,PH,HW1,Eva,Kant,ekant@stmhp.com,0x...
. . .
```

Also, a.csv file describes the kinds of events that may be recorded. Additional input files establish limitations, such as which types of actors may record particular events, among others, and they are entered into the system. The DApp generator took the input files and generated SQL statements to populate Django's database schema and the Solidity code for SC management. The next step was for the system administrator to build user profiles in Django and associate them with the approved operator profiles already in the database. This would enable the authorized operators to manage the DApp system using the Django framework. When users want to enter a new transaction, Django generates the DApp graphical user interface (GUI) on the fly. In Figure 3, we can see two captured images of the event input application, which is automatically produced. Following completion of these procedures, authorized operators and IoT devices might communicate transactions to the system, therefore controlling the medication supply chain, producing events, and so on.

Fig. 3: Recording Delivery and Conferral events.

V. CONCLUSIONS AND FUTURE WORKS

Crucial for ensuring compliance with manufacturing and storage regulations and the removal of contaminated drugs from the supply chain, traceability systems play a key role. Crucial when dealing with pricey narcotics, they may also attest that the substances are in the custody of their proper owners. Integrating IoT devices with DLT-based monitoring systems allows data transparency and identification of the origin of the packed medicine lot. It also allows the end-customer to control the quality and authenticity of the bought drug. Drug supply chain management is normally the purview of third parties; however, in DLT-based systems, this is superfluous.

We are unaware of any previous efforts to automate the creation of bespoke DApps for use in medication supply chain management. It was accomplished by using a state-of-the-art framework for developing web applications, such as Django, and making SCs that could be customized. Additionally, we demonstrated the strategy's efficacy via the development of a proof-of-concept system. When building and improving SCs on the Ethereum blockchain, be mindful that this flexible approach might lead to significant gas costs. We recommend a permissioned blockchain that is open to the public as a distributed ledger technology (DLT). Possible future developments include integration with other distributed ledger technologies (DLTs), explicit support for additional data used by drug auditing

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Vol. 21, Issue 1, 2025

organizations, and clear payment administration via a token or a stable coin tied to a traditional fiat currency.

REFERENCES

[1] L. Marchesi, M. Marchesi, R. Tonelli, and M. I. Lunesu, "A blockchain architecture for industrial applications," Blockchain: Research and Applications, p. 100088, 2022.

[2] A. Sharma, S. Kaur, and M. Singh, "A comprehensive review on blockchain and internet of things in healthcare," Trans. Emerging Telecomm. Technologies, vol. 32, no. 10, p. e4333, 2021.

[3] A. Ghadge, M. Bourlakis, S. Kamble, and S. Seuring, "Blockchain implementation in pharmaceutical supply chains: A review and conceptual framework," Int. J. Production Research, pp. 1–19, 2022.

[4] Django Software Foundation, "Django." [Online]. Available: https: //djangoproject.com

[5] L. Marchesi, M. Marchesi, and R. Tonelli, "Abcde–agile block chain dapp engineering," Blockchain: Research and Applications, vol. 1, no. 1-2, 2020.

[6] L. Marchesi, "Automatic generation of blockchain-based drug supply chain management system," in 6th WETSEB ICSE Workshop, ser. WETSEB '23. Melbourne, Australia: ACM, 2023.

[7] OpenZeppelin, "Openzeppelin: Contracts," 2020. [Online]. Available: https://github.com/OpenZeppelin/openzeppelin-contracts

[8] L. Marchesi, M. Marchesi, G. Destefanis, G. Barabino, and D. Tigano, "Design patterns for gas optimization in ethereum," in 2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE). IEEE, 2018, pp. 9–15.