Automating Attended Home Deliveries with Smart Contracts: A Blockchain-based Solution for E-commerce Logistics

Kadim Lahcen Nadime, Jamal Benhra, Rajaa Benabbou, and Salma Mouatassim

1LARILE, ENSEM, University Hassan II of Casablanca, Morocco

Abstract. The rapid growth of e-commerce has placed considerable strain on traditional logistics systems, prompting a need for innovative solutions to optimize delivery processes and enhance customer satisfaction. This research paper presents a decentralized crowdsourced delivery application that leverages blockchain technology and smart contracts to address the challenges faced by centralized logistics models. The proposed system allows e-commerce companies to outsource product deliveries to carriers from a diverse pool, offering greater flexibility and cost-effectiveness while also enhancing transparency and trust among all parties involved. Built on the Ethereum blockchain, the application manages both the delivery and return processes, generating verifiable proofs of delivery (PoD) and proofs of return (PoR) for each transaction. The paper provides a comprehensive analysis of the system architecture and the implementation of the application using smart contracts. Furthermore, it explores the potential impact of the proposed system on e-commerce companies, carriers, and customers, and identifies challenges and future directions for research and development in this field. The findings of this study contribute to the ongoing discourse on the transformative potential of blockchain technology and crowdsourcing in the e-commerce logistics industry, offering valuable insights into the design and real-world application of a decentralized delivery system.

1 Introduction

In recent years, the e-commerce industry has experienced exponential growth, fundamentally changing the way consumers shop for goods and services [1]. This rapid expansion has placed a significant strain on traditional delivery logistics, which often struggle to keep up with the increasing demand for fast, efficient, and reliable product deliveries [2]. As a result, e-commerce companies are constantly seeking innovative solutions to improve their delivery processes and enhance customer satisfaction.

One area of interest is the potential of leveraging decentralized systems and crowdsourcing [3] to optimize delivery logistics. By decentralizing the delivery process, e-commerce companies can tap into a vast pool of carriers and offer more flexible, cost-effective, and timely delivery services to their customers. In addition, a decentralized system

* Corresponding author: kadim.nadime-etu@etu.univh2c.ma

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).
can help enhance transparency and trust among all parties involved, including e-commerce companies, carriers, and customers.

The current centralized delivery systems operated by major logistics companies often face challenges related to scalability, efficiency, and cost-effectiveness [4].

As e-commerce continues to grow, these issues become more pronounced, resulting in increased delivery times, higher costs, and reduced customer satisfaction. Managing crowdsourced deliveries poses substantial risks when relying on centralized systems [5], largely due to the inherent difficulties in ensuring accountability and integrity and controlling the costs within the supply chain. Moreover, centralized systems lack transparency and can suffer from bottlenecks, as all decisions are made by a single controlling entity, making them susceptible to errors, fraud, and disputes.

Blockchain technology, with its trustless nature and decentralization, presents a viable solution for managing crowdsourced deliveries in e-commerce companies. As a decentralized ledger, blockchain enables a transparent, immutable, and traceable record of all transactions and interactions. This facilitates unparalleled visibility across the supply chain, effectively mitigating the risks associated with centralized systems [6]. Coupled with the use of smart contracts, blockchain further enhances the viability of crowdsourcing models.

A decentralized crowdsourced delivery application can address these issues by distributing the delivery process among a diverse pool of carriers. This not only helps to alleviate the strain on centralized systems but also provides e-commerce companies with greater flexibility in selecting the most suitable carrier for each delivery. Through the use of blockchain technology and smart contracts, the application can provide verifiable proofs of delivery (PoD) [7] and proofs of return (PoR), allowing customers and e-commerce companies to track the delivery process in real-time and ensure that their transactions are secure and reliable.

Given these potential benefits, there is a pressing need for a decentralized crowdsourced delivery application that can revolutionize the e-commerce logistics landscape and offer a more efficient, cost-effective, and transparent alternative to traditional delivery systems.

This research paper aims to present a decentralized crowdsourced delivery application that allows e-commerce companies to outsource product deliveries to carriers from a crowd. By employing a smart contract-based system built on the Ethereum Blockchain, this application addresses both the delivery and return processes, ultimately generating proofs of delivery (PoD) and proofs of return (PoR) to ensure transparency and trust.

The proposed application specifically addresses two vital aspects of decentralization: the decentralization of information and financial flow within the supply chain. On the informational front, our system decentralizes the traceability of multiple facets such as carrier history, product information, proofs of delivery (PoD), and proofs of return (PoR). This granularity in decentralizing information ensures not only heightened transparency but also fortifies the integrity of the entire logistics process. As for financial decentralization, our application employs smart contracts to manage the payment to carriers autonomously, thereby reducing reliance on a single financial entity and streamlining the payment process. It is imperative to clarify that the scope of this research does not encompass the payment mechanisms from the customer; our focus remains solely on the decentralization of information and carrier payments within the supply chain. This dual-pronged approach to decentralization provides a comprehensive solution to the challenges currently faced by e-commerce logistics, notably in the realm of urban and last-mile deliveries.

This research contributes to the field by developing a decentralized crowdsourced delivery application that leverages the capabilities of blockchain technology and smart contracts. The application aims to:

- Alleviate the scalability and efficiency issues associated with centralized delivery systems by distributing the delivery process among a diverse pool of carriers.
Enhance transparency and trust through the generation of verifiable proofs of delivery (PoD) and proofs of return (PoR), thereby mitigating risks and vulnerabilities.

Provide e-commerce companies with the flexibility to select the most suitable carriers for each delivery, contributing to cost-effectiveness and timeliness.

Utilize smart contracts to automate aspects of the delivery and return processes, ensuring a secure and reliable transactional framework.

This article is organized as follows: Section 2 provides a summary of the relevant literature and current state of the art. Section 3 outlines the methodology used to design the proposed solution. Section 4 presents the system deployment results. In Section 5, we discuss the implications of the results and identify benefits and potential limitations of the approach. Finally, Section 6 concludes the article by providing an overview of the study's findings.

2 Literature Review

This review will shed light on the existing body of knowledge concerning crowdsourced delivery, the potential advantages it poses over traditional models, the existing discourse on optimizing delivery routes within this context, and the somewhat sparse discussion on decentralized crowdsourcing.

Crowdsourced delivery as a logistics model has gained traction due to the exponential growth of e-commerce and the inherent challenges it brings to traditional logistics systems [8,9]. The review of literature acknowledges that such models serve to distribute the delivery workload among a wide array of carriers, thus mitigating the strain on centralized logistics systems [10]. This innovative approach to delivery service provision, as per the findings by [11,12], has been recognized for its potential to streamline delivery times, lower operational costs, and bolster customer satisfaction. It also underscores the importance of operational aspects such as detour length, parking behaviour, and daily traffic variations.

Several researchers have focused on optimizing delivery routes in the context of crowdsourced delivery [13,14]; they typically do not account for the unique opportunities and challenges presented by decentralized crowdsourcing models. The literature scarcely explores the decentralized crowdsourcing models, leaving a considerable gap in the research that assesses its viability.

An extensive systematic literature review investigating blockchain's real-world applications in various supply chain management fields has pointed towards a conspicuous absence of research focusing on blockchain technology in the crowdsourced last mile [15]. Research regarding decentralized crowdsourcing models is still in its nascent stages. Few studies have touched upon decentralized crowdsourcing for last-mile deliveries:

A study by [16] delved into the challenges faced by knowledge-intensive crowdsourcing platforms, identifying fragmentation of expertise, lack of trust between task providers and crowdsourcing participants, and limited learning from past experiences as core problems. The authors posit a blockchain-based reference architecture to counter these issues, aiming to streamline transactions between knowledge supply and demand.

The study conducted by [17], investigates decentralized crowd logistics and its associated challenges such as user trust, data safety, and the quality of tracking services. They propose an innovative e-commerce crowd logistics conceptual model underpinned by blockchain technology, enhancing the flexibility of the crowd logistics system. This model envisages a trading platform where transactions occur between buyers, sellers, and couriers. Notably, it employs smart contracts for automated settlements and uses the InterPlanetary File System (IPFS) for storing and sharing product detail. The model encourages all entities to act honestly by ensuring collateral is deposited by each party, which in turn boosts trust, security, and traceability. The use of blockchain technology in this model supports transparency and

E3S Web of Conferences 469, 00026 (2023)
immutability, with smart contracts helping track the complete order process, thus enhancing overall process transparency.

PackChain [18] presents a unique blockchain-based solution that addresses several last-mile delivery issues. PackChain employs Ethereum blockchain to establish a crowdsourcing platform for last-mile delivery management, boasting autonomous and transparent processes. Its approach significantly reduces costs and enhances transparency by managing user information, accepting offers, verifying transactions, and handling payments. Furthermore, the framework provides a proof of delivery as an arbitration mechanism, which is crucial in establishing trust between involved parties. However, despite its numerous strengths, one limitation of this approach is its lack of provisions for product traceability. This is an area that our proposed solution aims to address. We propose leveraging the Ethereum-based token standard ERC-721 [19] to generate a digital asset equivalent for each product, enabling the tracking of product ownership throughout its journey. By integrating this functionality into the existing model, we can augment the benefits of the PackChain platform, enhancing both transparency and accountability in last-mile delivery processes.

According to [20] the concept of crowdsourced last mile has garnered interest primarily among frequent and community-oriented e-shoppers who prefer home delivery and are positive towards innovations and sustainability enhancements. In understanding this, the proposed solution can strategically be tailored to meet these consumer expectations, thereby enhancing the efficiency and effectiveness of the last-mile logistics operations.

3 Detailed overview of the solution architecture

The proposed decentralized crowdsourcing supply chain model involves e-commerce companies, carriers, customers, and a smart contract. It operates in two echelons: between the company and carrier, and between the carrier and customer. The smart contract facilitates transparent and secure transactions, including product generation, order creation, bidding, and proof of delivery and return. A competitive bidding system ensures cost-effectiveness and service quality. The system criteria include registration, transparent bidding, secure payments, and traceability. The smart contract is implemented on the Ethereum platform using Solidity, with the support of OpenZeppelin and the ERC721 standard for standardization and security.

3.1 Supply Chain Echelons

The proposed supply chain solution in the paper is based on a decentralized crowdsourcing model enabled by Blockchain and Smart Contracts. Figure 1 illustrates the operational workflow of the proposed decentralized crowdsourcing model. At its core, the model encapsulates three primary entities: E-commerce companies, carriers, and customers.

These supply chain entities form two primary echelons. This two-echelon structure represents a streamlined and efficient approach to decentralized logistics, offering a simplified yet robust framework for crowdsourcing in supply chain management. It also facilitates local service delivery and provides an effective model for urban environments where quick, low-cost delivery services are in high demand.

At the heart of the system lies the Decentralised Ledger, which records transactions and ensures transparency and security. Linked to this ledger are Smart Contracts, automating and verifying contractual obligations without intermediaries.
3.1.1 Echelon One: Between the Company and Carrier

This is the first stage of the supply chain process where the e-commerce company and carrier interact. The company, after being registered by the smart contract owner, generates products as ERC-721 digital assets in the smart contract. It also creates new delivery orders, which are open for carriers to bid on.

Carriers register themselves in the smart contract using an app and then bid on these available delivery orders. The bid includes their compensation proposal. The company operator reviews these bids along with the carriers' history and chooses the most suitable one. Once a bid is accepted, the carrier picks up the product from the company, marking the transition to the next echelon.

For product returns, the company operator reviews return orders initiated by customers and selects a bid for the return order. The carrier then picks up the product from the customer and delivers it back to the company.

3.1.2 Echelon Two: Between the Carrier and Customer

This is the second and final stage of the supply chain process, involving interactions between the carrier and the customer. Once the carrier picks up the product from the company, they deliver it to the customer. The proof of delivery is generated through the smart contract, ensuring that the product was physically delivered to the customer.

In case of returns, the customer initiates a return request through the smart contract. The carrier whose bid was accepted by the company then picks up the product from the customer and delivers it back to the company. A proof of return is generated at this stage through the smart contract.

The smart contract acts as the technological enabler and mediator for the transactions within these two echelons. It governs the interactions between the company, carrier, and customer, providing transparency, security, and traceability. Functions within the smart contract allow for the generation of products as ERC-721 tokens, creation and bidding on orders, transfer of products between entities, and generation of proof of delivery and return events.
3.2 Order/Offer system:

Central to the efficient operation of the proposed decentralized supply chain model is a competitive and dynamic bidding system that bridges the gap between company orders and carrier offers (Figure 2). The bidding mechanism encourages open competition amongst carriers, promoting quality service provision and cost-effectiveness.

![Fig. 2. The offer and order system between companies and carriers.](image)

Figure 3 illustrates the bidding mechanism. In the proposed system, once a company generates a delivery or return order, it is made visible to all registered carriers, who can then submit their bids. Each bid includes not only the proposed cost for delivering the order, but also the carrier's service history. This dual-factor proposal fosters competition on the grounds of both service cost and delivery quality [21]. Carriers are thus incentivized to maintain a perfect service history, as it strengthens their competitiveness in the bidding process. A carrier with a flawless track record could potentially be chosen over another with a lower bid but a less favourable service history. This approach naturally encourages carriers to strive for excellence in their service delivery, knowing that their historical performance directly influences their chance of winning future bids.

![Fig. 3. Simplified Class Diagram for Bidding and Creating Offers and Orders.](image)

Moreover, the binding nature of smart contracts in the system further guarantees the execution of the agreed-upon service. Once a company accepts a bid, the carrier is contractually bound to fulfil the delivery as per the conditions laid out in the smart contract. In the unlikely event of a service failure, the system's traceability feature enables the precise identification of underlying issues, which in turn facilitates their resolution. Thus, the
proposed mechanism creates a trustless environment where companies can be confident in the delivery process, despite not directly knowing or having previous trust relations with the delivery agent. The combination of competitive bidding, transparent service histories, and binding smart contracts ensures reliable delivery, eliminating the need for traditional trust-building measures.

3.3 Detailed Breakdown of the Decentralized Crowdsourcing Supply Chain Framework

3.3.1 Registration

a. E-commerce Companies: The process begins with the registration of the e-commerce companies by the owner of the smart contract. This step marks the companies' entry into the system and enables them to generate products and initiate delivery orders.

b. Carriers: Independent logistic service providers, or carriers, register themselves through an application interfacing with the smart contract. Upon registration, carriers gain access to the available delivery orders placed by the e-commerce companies.

c. Customers: The end-users or customers are registered in the smart contract via the e-commerce platform they use for product orders. This registration allows their product orders to be tracked and efficient communication throughout the delivery process.

3.3.2 Product Generation and Delivery Order Creation

Once registered, e-commerce companies generate the products they plan to sell as ERC-721 digital assets in the smart contract. Simultaneously, they create new delivery orders corresponding to these digital assets, thereby creating demand for the services of the registered carriers.

3.3.3 Bidding Process

Carriers participate in a competitive bidding process, submitting their bids on available delivery orders. These bids include the proposed compensation for the delivery service they provide.

3.3.4 Bid Evaluation and Acceptance

E-commerce companies review and evaluate the bids based on the proposed price and the historical performance of the carriers. Following evaluation, companies accept the most cost-efficient and reliable bid, leading to an agreement for product delivery.

3.3.5 Product Pickup and Delivery

a. Pickup: The carrier with the accepted bid picks up the product from the e-commerce company.

b. Delivery: The carrier delivers the product to the customer. The delivery process includes a critical physical verification step: the receiver (in this case, the customer) confirms receipt by scanning a uniquely generated QR code. This QR code, linked with the smart contract,
verifies that both parties (the carrier and the customer) were physically present at the same location during the delivery.

c. **Proof of Delivery and Payment:** Scanning the QR code generates a verifiable digital signature, effectively creating a tamper-proof Proof of Delivery (PoD) within the smart contract. This PoD validates the physical delivery of the product to the customer. Once the PoD is generated, the smart contract releases the agreed compensation from the e-commerce company to the carrier (Figure 4).

![Simplified Sequence Diagram for Normal Delivery Operations (Downstream)](image-url)

**Fig. 4.** Simplified Sequence Diagram for Normal Delivery Operations (Downstream).

### 3.3.6 Return Process (if applicable)

**a. Initiation:** If the customer wishes to return a product, they initiate a return request via the smart contract, which automatically generates a return order.

**b. Bidding on Return Order:** Carriers bid on the return order, similar to the initial delivery process.

**c. Acceptance of Return Bid:** The e-commerce company evaluates and accepts the return and chooses a suitable bid for the return order.

**d. Product Pickup and Return Delivery:** The selected carrier picks up the product from the customer and returns it to the company. Similar to the delivery process, the receiver (here, the e-commerce company) confirms receipt by scanning a uniquely generated QR code, which verifies the physical presence of both parties at the return location.

**e. Proof of Return and Payment:** The scanned QR code creates a verifiable digital signature, producing a tamper-proof Proof of Return within the smart contract. Upon
successful generation of this Proof of Return, the smart contract releases the carrier's compensation for the return delivery (Figure 5).

Fig. 5. Simplified Sequence Diagram for Return Delivery Operations (Upstream).

3.4 Smart Contract Implementation

Our proposed decentralized supply chain model utilizes a smart contract, a self-executing contract with agreement terms coded directly into it, offering a way to execute trusted transactions without a centralized authority. We have adopted the Ethereum platform for our smart contract due to its established reputation, vast user base, integrated cryptocurrency Ether (ETH), and extensive developer resources. We use Solidity, Ethereum's native statically-typed programming language, to write our smart contracts, owing to its native support for contractual clauses, widespread acceptance, and robust documentation. The use of Ethereum and Solidity ensures our smart contract is secure and reliable, allowing companies, carriers, and customers to engage securely within our decentralized model. The smart contract, CrowdsourcedDelivery, leverages the standardization, security, efficiency, and flexibility provided by OpenZeppelin and the ERC721 standard. OpenZeppelin is a library of reusable, secure Solidity-based smart contracts, while ERC721 is a standard used for non-fungible tokens (NFTs), enabling the representation of unique product attributes and orders within our system.
3.4.1 Contract Overview: Functions, Objects, and Events

Table 1 serves as a quick reference to understand the functionality and usage of each function in the contract. Whether it's adding companies, creating orders, initiating deliveries, or processing returns, the functions table provides a clear overview of the contract's capabilities.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addCompany()</td>
<td>Adds a company with the specified ID and name to the contract's companies mapping.</td>
</tr>
<tr>
<td>addCarrier()</td>
<td>Adds a carrier with the specified full name and vehicle type to the contract's carriers mapping.</td>
</tr>
<tr>
<td>addCustomer()</td>
<td>Adds a customer with the specified full name and physical address to the contract's customers mapping.</td>
</tr>
<tr>
<td>addProduct()</td>
<td>Adds a product to the contract's products mapping and assigns it to the specified company.</td>
</tr>
<tr>
<td>createOffer()</td>
<td>Creates an offer from a carrier with the specified city zone, capacity, and payment.</td>
</tr>
<tr>
<td>createOrder()</td>
<td>Creates an order from a company with the specified product, customer, order type, pickup location, drop-off destination, vehicle needed, and payment.</td>
</tr>
<tr>
<td>chooseOffer()</td>
<td>Allows a carrier to bid on an order with the specified ID, offer ID, and payment.</td>
</tr>
<tr>
<td>getOrder()</td>
<td>Retrieves the details of an order with the specified ID.</td>
</tr>
<tr>
<td>getOffer()</td>
<td>Retrieves the details of an offer with the specified ID.</td>
</tr>
<tr>
<td>getProduct()</td>
<td>Retrieves the details of a product with the specified ID.</td>
</tr>
<tr>
<td>getBid()</td>
<td>Retrieves the details of a bid with the specified ID.</td>
</tr>
<tr>
<td>initiateDelivery()</td>
<td>Initiates the delivery process for an order with the specified ID.</td>
</tr>
<tr>
<td>deliverProduct()</td>
<td>Marks an order as delivered and transfers the product ownership to the customer.</td>
</tr>
<tr>
<td>initiateReturn()</td>
<td>Initiates the return process for a product with the specified ID.</td>
</tr>
<tr>
<td>acceptReturn()</td>
<td>Allows a company to accept a return for an order with the specified ID and bid ID.</td>
</tr>
<tr>
<td>returnProduct()</td>
<td>Marks an order as returned and transfers the product ownership back to the company.</td>
</tr>
</tbody>
</table>
confirmReturn() Confirms the return of an order and completes the payment process.

Table 2 highlights the various objects used in the Solidity contract and their purposes. These objects, such as Company, Carrier, Customer, Order, Offer, Bid, and Product, are essential building blocks of the contract's data structure. Each object has specific attributes that store relevant information, allowing the contract to track and manage companies, carriers, customers, orders, offers, bids, and products.

**Table 2.** Objects Summary Table.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>Represents a company and stores its ID and name.</td>
</tr>
<tr>
<td>Carrier</td>
<td>Represents a carrier and stores its ID, full name, vehicle type, assigned delivery, and delivery history.</td>
</tr>
<tr>
<td>Customer</td>
<td>Represents a customer and stores its ID, full name, physical address, and order history.</td>
</tr>
<tr>
<td>Order</td>
<td>Represents an order and stores its ID, product ID, company ID, customer ID, order type, pickup location, drop-off destination, vehicle needed, payment, and chosen offer ID.</td>
</tr>
<tr>
<td>Offer</td>
<td>Represents an offer made by a carrier and stores its ID, carrier ID, city zone, capacity, and payment.</td>
</tr>
<tr>
<td>Bid</td>
<td>Represents a bid made by a carrier on an order and stores its ID, carrier ID, order ID, offer ID, and payment.</td>
</tr>
<tr>
<td>Product</td>
<td>Represents a product and stores its ID, owner ID, name, price, description, and status.</td>
</tr>
<tr>
<td>ProductStatus</td>
<td>An enumeration representing the status of a product. It has the following possible values: Idle, AwaitingPickup, AwaitingDelivery, Delivered, AwaitingReturn, and Returned. Used to track the status of a product.</td>
</tr>
</tbody>
</table>

Table 3 provides a clear overview of the events emitted by the contract. These events capture important moments and state changes, such as adding companies or carriers, creating orders, matching offers, delivering products, processing payments, initiating returns, and more. By emitting events, the contract provides a log of significant occurrences, enabling external systems to listen and react to these events.

**Table 3.** Events Summary Table.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompanyAdded</td>
<td>Triggered when a new company is added to the contract. It emits the company's address and name.</td>
</tr>
</tbody>
</table>
CarrierAdded | Triggered when a new carrier is added to the contract. It emits the carrier's address, full name, and vehicle type.
---|---
CustomerAdded | Triggered when a new customer is added to the contract. It emits the customer's address, full name, and physical address.
OrderEvent | Triggered when a new order is created. It emits the order's ID, product ID, and company ID.
OfferEvent | Triggered when a new offer is created by a carrier. It emits the offer's ID and carrier's address.
OrderOfferMatched | Triggered when an offer is chosen for an order. It emits the order's ID and the chosen offer's ID.
ProofOfDelivery | Triggered when a product is delivered. It emits the order's ID and the delivered product's ID.
PaymentEvent | Triggered when a payment is made for an order. It emits the order's ID, the product's ID, and the payment amount.
ReturnRequest | Triggered when a return is initiated by a customer. It emits the order's ID, the product's ID, and the customer's address.
ProofOfReturn | Triggered when a product is returned. It emits the order's ID and the returned product's ID.

4 Results

In this study, the deployment of the smart contract was successfully executed, and subsequent testing was conducted to evaluate the operational functionality of the code. The development and unit testing of the smart contract were performed using the Truffle environment, a development environment, testing framework, and asset pipeline for Ethereum. Simultaneously, Ganache was utilized to simulate a local blockchain, facilitating the preliminary deployment and testing of the Crowdsourced Delivery contract.

The choice of Ethereum for deploying our smart contract was influenced by its relative network stability, especially when compared to newer platforms like Solana, which despite its potential, has experienced significant outages making it less reliable. Although Ethereum has had its share of outages, they have been minimal and swiftly resolved. The Ethereum ecosystem also provides extensive tooling, like the Truffle suite, which was a cornerstone in the development process; it furnished a sophisticated development environment and provided an integrated testing framework. This ensured that the smart contract was both functionally robust and free from vulnerabilities.

As corroborated by Figure 6, the deployment was seamlessly achieved with the absence of any discernible errors. Key attributes of the contract, including the total gas consumed and the unique contract address, can also be seen in the figure. One of the inherent aspects of deploying smart contracts on the blockchain is their association with gas fees. These fees, intrinsically variable, depend on the level of network activity and the computational resources demanded by the deployment.
**Fig. 6.** Successful deployment of the smart contract in the Truffle environment.

Figure 7 showcases the Ganache blockchain simulation used for local testing of the contract. Ganache provides the ability to perform intricate and in-depth testing by emulating the blockchain’s behaviour. Here, we see a snapshot of the transaction blocks generated during the contract’s execution.

**Fig. 7.** Simulation of the local blockchain using Ganache.

Unit testing was then carried out to verify that all functions operated as expected. A variety of scenarios were created to ensure that each function handled both typical and edge cases properly. We also tested the ability of the contract to interact appropriately with the Ethereum network, including accurate transaction processing and correct usage of gas. The results of these tests, displayed in Figure 8, indicate a successful execution of all function calls, implying the contract behaves as expected in all tested scenarios. The passing of all unit tests establishes confidence in the reliability and robustness of the developed smart contract.
5 Discussion and perspectives

The results presented underscore the value of employing smart contracts in the management of roles in crowdsourcing within a last-mile logistics context. Successful deployment and comprehensive testing of our solution highlight the potential utility of this technology in overcoming many of the challenges associated with last-mile delivery logistics. Within the context of the crowdsourced last-mile delivery process, our system provides a comprehensive set of guarantees aimed at augmenting service standards. The immutable and transparent nature of the blockchain technology ensures the traceability of all transactions, a feature that becomes significantly vital when multiple independent entities or individuals (crowdsourcing participants) are involved in the delivery process.

By fostering a transparent environment, the blockchain can provide all participants in the crowdsourced delivery network with equal access to the same information. This transparent approach reduces misunderstandings and discrepancies, thus contributing to higher standards of accuracy and efficiency. Each transaction related to the delivery process, from pick-up to drop-off, is meticulously scrutinized, authenticated, and logged on the blockchain. This is particularly beneficial in the crowd context, as it allows for accountability and traceability of actions, even when performed by different participants. Hence, regardless of the number of entities involved in the delivery process, the combination of blockchain's immutability and transparency enables a seamless, efficient, and reliable last-mile delivery system in a crowdsourcing model.

The role of carrier history in a crowdsourced delivery model is crucial to instilling trust and ensuring reliability. By utilizing the immutable characteristics of the blockchain, our system maintains accurate and reliable carrier histories. Every delivery made by a carrier is permanently recorded on the blockchain, bolstering transparency and accountability. This feature is pivotal in augmenting the efficacy of last-mile delivery, as it enables customers and service providers to choose carriers based on their proven track records, thereby ensuring superior service. This is particularly important when it comes to managing fraud which is a significant challenge in crowdsourced delivery systems, with instances of false delivery reports, tampering of carrier history, and manipulation of payment systems being prevalent.

While our decentralized system is engineered to introduce robust countermeasures against fraudulent activities, it is essential to note that the true robustness of the system will only be fully understood when tested in real-world scenarios. The inherent transparency and immutability of blockchain offer strong safeguards against tampering with transaction records or fabricating delivery reports. However, real-world complexities and unexpected events can pose challenges that are not readily accounted for in a controlled setting. Similarly, while automated smart contracts aim to minimize manual errors and potential exploitation, their effectiveness and resilience need to be evaluated under actual operational conditions.
Consequently, although the system is designed to significantly mitigate the risk of fraudulent activities, further empirical studies are warrant to assess its robustness and reliability in real-world supply chain logistics.

Moving forward, several avenues for future research and development can be identified to further refine and expand the application. First, the user interface of the application remains a critical component for its adoption and usability. An intuitive, user-friendly interface will be instrumental in facilitating seamless interactions among all participants in the supply chain, including e-commerce companies, carriers, and customers. Second, the smart contracts governing the transactions could be further optimized for efficiency and responsiveness, addressing any limitations observed during real-world testing.

The application could also be extended to include more roles and functionalities, such as integrating crowdsourced storage solutions. This would involve pooling storage space providers from a crowd, thereby offering a more versatile and comprehensive supply chain solution. Such an extension would not only add another layer of decentralization but also provide additional avenues for optimizing costs and improving efficiency.

6 Conclusion

The boom in online shopping has shown that we need better ways to deliver packages, especially the last leg of the journey to customers’ doorsteps. Traditional delivery methods often face challenges such as delays, lack of clarity, inefficiency, opacity, and disjointed control. In response, a novel model has been explored, utilizing blockchain technology and smart contracts to address these inefficiencies.

The primary objective of this research was to explore the feasibility and potential benefits of employing a decentralized, blockchain-based system for crowdsourced delivery. The study introduces a decentralized crowdsourced delivery system, leveraging blockchain and smart contracts, to enhance attended home deliveries. Customers place orders on an e-commerce platform, which are then broadcasted to carriers via a crowdsourcing application. The smart contract facilitates binding delivery agreements, automates validation, and enables immediate payments upon completion.

The study revealed significant improvements in traceability, transparency, accuracy, and security, primarily owing to the use of Blockchain and smart contracts. The proposed solution enables e-commerce companies to outsource product deliveries to a crowd-based pool of carriers, which enhances flexibility and cost-effectiveness. The system leverages Ethereum Blockchain for its architecture, and smart contracts manage the delivery and return processes. This results in the generation of verifiable proofs of delivery (PoD) and proofs of return (PoR), improving traceability and transparency and fostering trust among all stakeholders. These results serve as a key contribution to the ongoing discourse on the transformative potential of blockchain technology and crowdsourcing in the e-commerce logistics industry.

Acknowledgement

This work was conducted as part of the MILEX project and greatly benefited from the unwavering support of the Hassan II University of Casablanca, Ecole Centrale Casablanca, and Mohammed VI Polytechnic University. The collaborative and constructive involvement of these institutions played a pivotal role in the successful realization of this work, and we express our profound gratitude for their invaluable contributions throughout the research process. Additionally, we wish to extend our sincere appreciation to the referees, whose insightful feedback and constructive comments significantly enriched the quality and rigor of this study.
References


