Blockchain Final Project Documentation

Team: Rawan Mohamed, Salma Montasser, Fouad Hossam, Omar Rayyan

QAHWA Supply Chain

TABLE OF CONTENTS

USE CASE

Problem

QAHWA's supply chain involves numerous processes that require continuous management and control to maintain a high-quality service and product delivery. Several problems arise from this supply chain including transparency, automation, and trust between buyers and sellers.

1. **Trust and Transparency:** In traditional supply chains, there is often a lack of trust between buyers and sellers, and transparency is limited. Buyers might be unsure if they will receive the product after payment, and sellers might worry about receiving payment after delivery.
2. **Automated Workflow:** Manual processes in supply chain management can lead to errors, delays, and inefficiencies. Tracking the status of orders manually can be inaccurate.
3. **Disagreement Resolution:** Resolving disagreements between buyers and sellers can be complex and time-consuming.
4. **Secure Payments:** Ensuring secure and accurate payments is a significant challenge. Buyers might be reluctant to pay upfront without guarantees, and sellers need assurance that they will be paid.
5. **Tracking and Accountability:** Keeping track of orders and their status can be difficult, especially for businesses with a high volume of transactions.
6. **Efficient Returns and Refunds:** Managing returns and refunds can be hectic between buyers and sellers, with the potential for delays and disagreements.

Solution

1. **Trust and Transparency:** The smart contract enforces a transparent and automated process where each step (order placement, payment, delivery, return) is recorded on the blockchain. This immutable record ensures both parties can trust the process.
2. **Automated Workflow:** The smart contract automates the state transitions of an order (from creation to payment, delivery, and return). This reduces the need for manual intervention and speeds up the process.
3. **Disagreement Resolution:** By maintaining a clear and immutable history of each order's state transitions, the smart contract provides a transparent basis for resolving arguments.
4. **Secure Payments:** The smart contract ensures that payment is only made when specific conditions are met (e.g., an order is created and then marked as paid). Payments are securely handled within the contract, reducing the risk of fraud.
5. **Tracking and Accountability:** The smart contract provides functions for tracking the status and history of each order, ensuring accountability and easy monitoring.
6. **Efficient Returns and Refunds:** The smart contract includes a clear process for handling returns and issuing refunds, with state transitions and timestamp tracking to ensure fairness and efficiency.

SMART CONTRACT FUNCTIONALITY

Smart Contract Breakdown

The smart contract we designed identifies and manages various states in the supply chain process of QAHWA. Below is a breakdown of the supply chain process as defined in the contract.

1. Order Creation

- Function: placeOrder
- Description: A buyer places an order for a product by specifying the product name and price. The order is initialized with the state Created.
- Event: OrderCreated
- State Transition: None (initial state is Created).

2. Payment

- Function: makePayment
- Description: The buyer makes a payment for the order. The payment amount must match the price of the product. Upon successful payment, the order state transitions to Paid.
- Event: OrderPaid
- State Transition: From Created to Paid.

3. Delivery Confirmation

- Function: confirmDelivery
- Description: The buyer confirms the delivery of the product. This action changes the order state to Delivered.
- Event: OrderDelivered
- State Transition: From Paid to Delivered.

4. Return Request

- Function: requestReturn
- Description: The buyer requests a return of the delivered product. This changes the order state to Returned.
- Event: OrderReturned
- State Transition: From Delivered to Returned.

5. Refund Processing

- Function: processRefund
- Description: The seller processes the refund for the returned order. The refund amount is transferred back to the buyer, and the order state reverts to Created.
- Event: OrderCreated
- State Transition: From Returned to Created.

6. Order Tracking

- Function: trackOrder
- Description: Allows any user to view the history of state changes for a specific order.
- Output: Returns an array of timestamps indicating when state transitions occurred.

State Variables

- owner: The address of the contract deployer.
- orderCounter: A counter for tracking the number of orders.
- orders: A mapping from order IDs to Order structs.
- orderHistory: A mapping from order IDs to an array of timestamps representing the history of the order.

Enum

- SupplyChainState: Represents the state of an order. Possible values are:
    - Created
    - Paid
    - Delivered
    - Returned

Struct

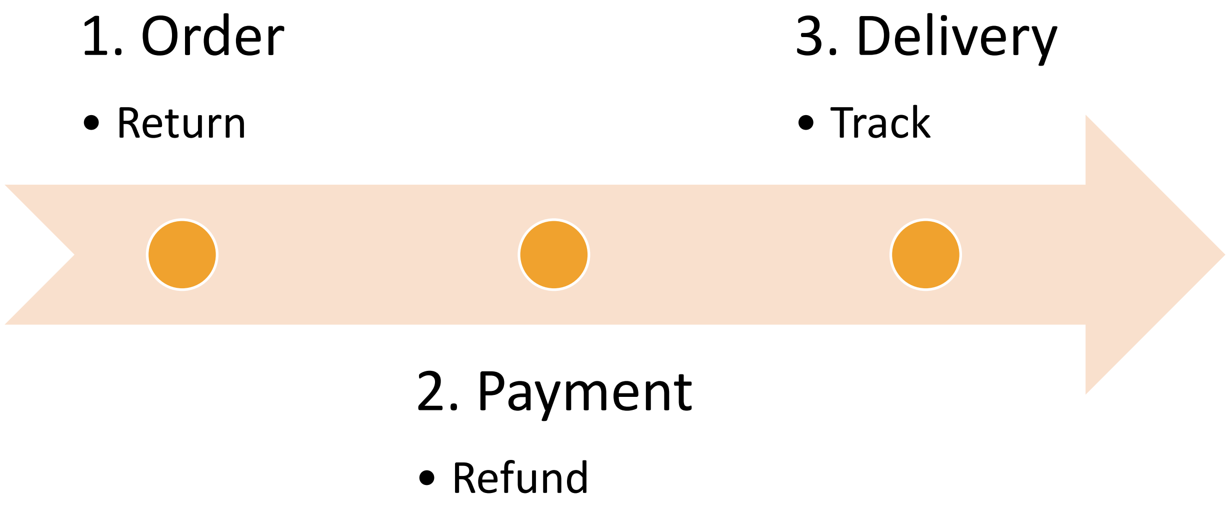
- Order: Represents an order with the following fields:
    - productName: The name of the product.
    - price: The price of the product.
    - buyer: The address of the buyer.
    - seller: The address of the seller.
    - state: The current state of the order.
    - timestamp: The last updated timestamp of the order.

Events

- OrderCreated(uint256 orderID, uint256 timestamp): Emitted when a new order is created.
- OrderPaid(uint256 orderID, uint256 timestamp): Emitted when an order is paid.
- OrderDelivered(uint256 orderID, uint256 timestamp): Emitted when an order is delivered.
- OrderReturned(uint256 orderID, uint256 timestamp): Emitted when an order is returned.

QAHWA Supply Chain Flowchart

The smart contract we designed identifies and manages various states in the supply chain process of QAHWA. Below is a breakdown of the supply chain process as defined in the contract.



Smart Contract Error Handling

1. Input Validation:
   Ensures that all function inputs are valid (e.g., non-empty product names, valid order IDs, correct payment amounts).
   Such as:

```
require(bytes(_productName). length > 0, "Product name cannot be empty");
```

2. State Validation:
   Ensures that functions are only executed when the order is in the correct state (e.g., an order must be in the Created state to be paid).
   Such as:

```
require(orders[_orderID]. state == SupplyChainState.Created,
"Invalid order state");
```

3. Authorization Checks:
   Ensures that only authorized users can perform certain actions (e.g., only the buyer can confirm delivery, only the seller can process a refund).
   Such as:

```
require(msg.sender == orders[_orderID].buyer, "Unauthorized user");
```

4. General Safety Checks:
   Validates sender addresses to prevent zero address interactions and ensures order IDs are within the valid range.
   Such as:

```
require(msg.sender != address(0), "Invalid address");
```

Testing

Test Procedures

To ensure the functionality, reliability, and correctness of the Supply Chain smart contract, we conducted a series of tests using the Hardhat development environment.

Tools and Setup

Hardhat: A development environment for Ethereum software.
Ethers.js: A library for interacting with the Ethereum blockchain and its ecosystem.
Chai: An assertion library for Node.js used in our tests to assert conditions.

Test Script

We used JavaScript test script, which is executed in the Hardhat environment. The script covers functions of the Supply Chain contract to ensure it behaves as expected.

Test Steps

1. Setup and Deployment:
- Deploy the Supply Chain contract before each test case to ensure a fresh contract state.
- Initialize required addresses (owner, buyer, seller).
2. Test Cases:
- Deployment
- Order placement
- Payment
- Delivery Confirmation
- Return Request
- Refund Processing
- Order Tracking

Deployment

Deployment Instructions

1. **Initialize Hardhat Project**: npx hardhat
2. **Compile the Contract**: npx hardhat compile
3. **Run the Deployment Script:** npx hardhat run scripts/deploy.js –network
4. **Verify Deployment:** output will be as follow:

```
Deploying contracts with the
account:0x32aA051AAE0DDC8F8aDCd1e3B764A282F67C92fC
Account balance: 3192153000000000
SupplyChain contract deployed to
address:0x2d4d77730F37c0541B5edDdFdFA597d6Ca5Bb9c7
```