```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Drug{
    address public owner;

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can perform this
action");
        _;
    }

    struct Drug {
        string drugName;
        string manufacturer;
        uint256 manufacturingDate;
        address trackingHistory;
    }

    mapping(uint256 => Drug) public drugs;
    uint256 public drugCount;

    event DrugManufactured(uint256 indexed drugId, string drugName, string
manufacturer, uint256 manufacturingDate);
    event DrugTransferred(uint256 indexed drugId, address indexed from,
address indexed to, uint256 transferDate);

    function manufactureDrug(uint256 drugId, string memory _drugName, string
memory _manufacturer, uint256 _manufacturingDate) external onlyOwner {

        address initialHistory;
        initialHistory = owner;

        drugs[drugId] = Drug(_drugName, _manufacturer, _manufacturingDate,
initialHistory);
        drugCount++;

        emit DrugManufactured(drugId, _drugName, _manufacturer,
_manufacturingDate);
    }

    function transferDrugOwnership(uint256 _drugId, address _to) external {
        require(_to != address(0), "Invalid address");
        require(_to != drugs[_drugId].trackingHistory, "Already owned by the
new address");

        address from = drugs[_drugId].trackingHistory;
        drugs[_drugId].trackingHistory = _to;
        emit DrugTransferred(_drugId, from, _to, block.timestamp);
    }

    function getDrugDetails(uint256 _drugId) external view returns (string 
memory, string memory, uint256, address) {

        Drug memory drug = drugs[_drugId];
        return (drug.drugName, drug.manufacturer, drug.manufacturingDate, 
drug.trackingHistory);
    }
}
```