Studi kasus = toko jual beli

1. Smart Contract

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";

contract MyToken is ERC20Burnable {
    receive() external payable{}
    fallback() external payable{}
    address public owner;

    struct Product{
        uint price;
        uint amount;
        string name;
    }

    mapping(uint => Product) public products;

    constructor(uint256 _supply,string memory name, string memory symbol) ERC20("MyToken","TOKEN"){
        _mint(msg.sender, _supply*1000000000000000000);
        owner = msg.sender;
        string memory _name = name;
        string memory _symbol = symbol;
        products[1] = Product(20*10000**18, 10, "Apple");
    }

    mapping (address => uint256) balances;
    mapping (address => mapping (address => uint256)) allowed;
```

Smart Contract dibuat dengan ERC20, pada smart contract terdapat struktu produk dan pada saat awal di deploy sudah terdapat satu buah product untuk dibeli yakno apple, lalu untuk memudahkan dibuat mapping product

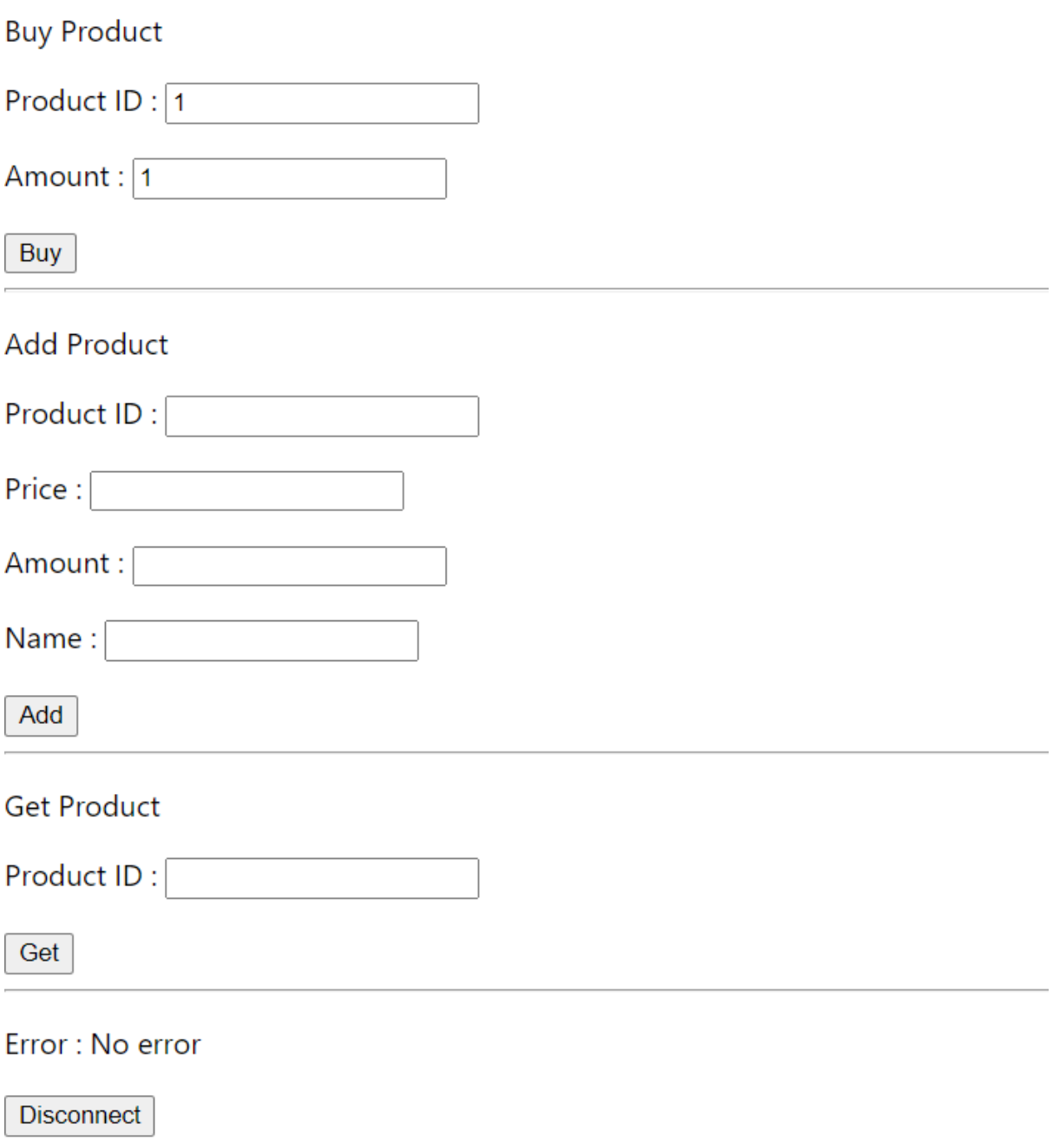
```
function buyProduct(uint id, uint amount) public payable{
    require(products[id].amount >= amount);
    require(products[id].price * amount <= msg.value);
    products[id].amount -= amount;
    payable(msg.sender).transfer(products[id].price * amount);
    _burn(msg.sender, products[id].price * amount);
}

function addProduct(uint id, uint price, uint amount, string memory name) public{
    require(products[id].amount == 0);
    products[id] = Product(price, amount, name);
}

function getProduct(uint id) public view returns (uint price, uint amount, string memory name){
    return (products[id].price, products[id].amount, products[id].name);
}

function getBalance(address _owner) public view returns (uint256 balance){
    return balances[_owner];
}
```

Setelahnya terdapat fungsi buyProduct dengan memilih id product yang terdapat pada mapping dan juga jumlah amount yang akan dibayar. Selain itu juga terdapat fungsi untuk meng-add product, getProduct dan getBalance.



Untuk tampilan dapps dibuat sederhana, terdapat form untuk setiap fungsi yang dibuat. Untuk form masih menggunakan parameter hardcode untuk setiap onClick

```
<p>Product ID : <input type="text"/></p>
<p>Amount : <input type="text"/></p>
<button onClick={()=>{
  buyProduct({id:1,amount:1});
}}>Buy</button>
<hr/>      You, 56 minutes ago • Uncommitted chan
<p>Add Product</p>
<p>Product ID : <input type="text"/></p>
<p>Price : <input type="text"/></p>
<p>Amount : <input type="text"/></p>
<p>Name : <input type="text"/></p>
<button onClick={()=>{
  addProduct({id:1,price:1,amount:1,name:"test"});
}}>Add</button>
<hr/>
<p>Get Product</p>
<p>Product ID : <input type="text"/></p>
<button onClick={()=>{
  getProduct({id:1});
}}>Get</button>
```

Setiap terjadi onClick maka akan menjalankan fungsi dari useState yang telah dibuat.