DAFTAR PUSTAKA

Abbas, Qalab E, dan Jang Sung-Bong. (2019) A Survey of Blockchain and Its Applications. International Conference on Artivicial Intelligence in Information and Communication (ICAIIC).

Annisya, Emy, H. Implementasi Teknologi Block*chain Proof of Work* Pada Penelusuran Supply Chain Produk Komputer". Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), Vol. 1, No.1, pp. 446-455, 2021.

Belleflamme, Paul, Thomas Lambert, dan Armin Schwienbacher. (2013). Crowdfunding: Tapping The Right Crowd. Journal of Business Venturing: 25.

Bentley, D. (2023, Desember) The Insidious Problem of Counterfeiting in Healthcare.https://www.raconteur.net/legal/intellectual-property/counterfeiting-healthcare.

Dzaky, A (2019). Sistem Verifikasi Dokumen Hasil Investigasi Forensik Digital Berbasis Teknologi Block*chain*. Unversitas Islam Indonesian Yogyakarta.

Ethereum Community. "Ethereum Whitepaper." Ethereum Web site. 5 Juni 2020. https://ethereum.org/whitepaper (diakses Juni 25, 2020).

Fakhri, Dinan, dan Kusprasapta Mutijarsa. (2018). Secure IoT Communication using Blockchain Technology." International Symposium on Electronics and Smart Devices (ISESD).

Gao, Weichao, William G. Hatcher, dan Wei Yu. (2018). A Survey of Blockchain: Techniques, Applications, and Challanges. 2018 27th International Conference on Computer Communication and Networks (ICCCN). Hangzhou, China: IEEE.

Gardiner, Ross. (2018). Free Trade: Composable Smart Contracts. Dissertation, Bristol.

Hileman, Garrick, dan Michel Rauchs. (2017). Global Blockchain Benchmarking Study.

Ihza, M. (2022). Sistem Penilaian Tempat Wisata Di Kabupaten Kediri Berbasis Web Menggunakan Block*chain* Ethereum". Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Islam, Mohammad Rabiul, Imad Fakhri Al-Shaikhli, Rizal Mohd Nor, dan Kabir Sardar Mohammad. (2018). Cryptocurrency vs Fiat Currency: Architecture, Algorithm, Cashflow & Ledger Technology on Emerging Economy. 2018 International Conference on Information and Communication Technology for the Muslim World (ICT4M).

Jani, Shailak. (2020) "Smart Contracts: Building Blocks for Digital Transformation Thesis, New-Delhi, India.

Kasireddy, Preethi. (2023, Desember). How does Ethereum work, anyway?" Medium Croporation Web site. 27 September 2017. https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369.

Kaushik, Akanksha, Archana Choudhary, Chinmay Ektare, dan Deepti Thomas. (2017). Blockchain – Literature Survey. 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). India.

Kemenkes Republik Indonesia (2023, November). Aplikasi Info Alat Kesehatan & PKRT. https://infoalkes.kemkes.go.id/.

Kim, Kiyun. (2023, Desember). Modified Merkle Patricia Trie — How Ethereum saves a state. Medium Corporation Web site. 26 Juni 2018. https://medium.com/codechain/modified-merkle-patricia-trie-how-ethereum-saves-a-state-e6d7555078dd .

Lin, Iuon-Chang, dan Tzu-Chun Liao. (2017). A Survey of Blockchain Security Issues and Challenges. International Journal of Network Security, Vol.19.

Liza, W., Thomas S., & Chrisdityra L. (2021) Implementasi Algoritma Konsensus Proof-of-Work dalam Block*chain* terhadap Rekam Medis". Jurnal Pekomnas, Vol. 7, No.1, pp. 41-42

Nakamoto, Satoshi. "Bitcoin: A Peer-to-peer Electronic Cash System." Bitcoin Organization. https://nakamotoinstitute.org/bitcoin/.

Rennock, Michael J.W., Alan Cohn, dan Jared R. Butcher. (2023, Desember)BlockChain Technology. https://www.steptoe.com/images/content/1/7/v3/171269/LIT-FebMar18-Feature-Blockchain.pdf.

Sarmah, Simanta Shekhar. “Understanding Blockchain Technology.” 2018.

Sayeed, Sarwar, Hector Marco-Gisbert, dan Tom Caira. “Smart Contract: Attacks and Protections.” IEEE Access, 2020.

Senay, G. A., Haya, H. R., Khaled, S., & Raja, J. NFT-Based Traceability and Ownership Management of Medical Devices”. IEEE Access, vol. 10, pp. 126394– 126410, 2022.

Shrivas, Mahendra Kumar, dan Thomas Yeboah. (2018). The Disruptive Blockchain:

Types, Platforms and Applications.” 5th Texila World Conference for Scholars (TWCS).

Singla, Vinayak, Indra Kumar Malav, Jaspreet Kaur, dan Sumit Kalra. “Develop Leave Application using Blockchain Smart Contract.” 11th International Conference on Communication Systems & Networks (COMSNETS). 2019.

Szabo, Nick. (2023, Desember) Smart Contracts: Building Blocks for Digital Markets.https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html.

Taş, Ruhi, dan Ömer Özgür Tanrıöver. (2019). Building A Decentralized Application on the Ethereum Blockchain. 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT). Ankara, Turkey: IEEE.

Vujičić, Dejan, Dijana Jagodić, dan Siniša Ranđić. (2018). Blockchain Technology, Bitcoin, and Ethereum: A Brief Overview. 17th International Symposium INFOTEH-JAHORINA (INFOTEH). Čačak, Serbia.

Walaa, A., Andrej, S., and Raja, J. Block*chain*-Based Decentralized Digital Manufacturing and Supply for COVID-19 Medical Devices and Supplies. IEEE Access, vol. 9, pp. 137923– 137938, 2021.

DAFTAR LAMPIRAN

Lampiran 1. Source Code *Smart* Contract (SupplycChain.sol

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.0;
pragma experimental ABIEncoderV2;

import './RawAlkes.sol';

contract SupplyChain {
    address public immutable owner;

    constructor()  {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(owner == msg.sender, "Not the owner");
        _;
    }

    enum Roles { NoRole, Manufaktur, Distributor, Kemenkes,
RumahSakit, Customer }

    struct UserData {
        bytes32 name;
        bytes32 email;
        bytes32 noTelp;
        Roles role;
        address userAddr;
    }

    struct SupplyChainAlkesDetails {
        bytes32 namaAlkes;
        bytes32 deskripsiAlkes;
        bytes32 klasifikasiAlkes;
        bytes32 tipeAlkes;
        bytes32 kelasAlkes;
        bytes32 kelasResiko;
        bytes32 noIzinEdar;
    }
```

```
    mapping(address => UserData) public userInfo;

    event UserRegister(address indexed _address, bytes32 name);
    event RequestEvent(address indexed distributor, address
indexed manufaktur, address alkesAddr, uint indexed timestamp);
    event IzinEvent(address indexed distributor, address indexed
kemenkes, address alkesAddr, uint indexed timestamp);
    event RsEvent(address rs, address indexed distributor,
address alkesAddr, uint indexed timestamp);

    function requestProduct(address distributor, address
manufaktur, address packageAddr) public {
        emit RequestEvent(distributor, manufaktur, packageAddr,
block.timestamp);
    }

    function izinRequest(address distributor, address kemenkes,
address alkesAddr) public {
        emit IzinEvent(distributor, kemenkes, alkesAddr,
block.timestamp);
    }

    function reqAlkesRs(address rs, address distributor, address
alkesAddr) public {
        emit RsEvent(rs, distributor, alkesAddr,
block.timestamp);
    }

    function registerUser(
        bytes32 name,
        bytes32 email,
        bytes32 noTelp,
        uint8 role,
        address userAddr
    ) public onlyOwner {
        userInfo[userAddr] = UserData({
            name: name,
            email: email,
            noTelp: noTelp,
            role: Roles(role),
            userAddr: userAddr
        });

        emit UserRegister(userAddr, name);
    }
```

```
function changeUserRole(uint8 role, address addr) public
onlyOwner {
        userInfo[addr].role = Roles(role);
    }

    function getUserInfo(address addr) public view returns
(UserData memory) {
        return userInfo[addr];
    }

    mapping(address => address[]) public dataAlkesManufaktur;

    function createAlkesManufaktur(
        SupplyChainAlkesDetails memory alkesDetails,
        address distributorAddr,
        address kemenkesAddr,
        address rumahSakitAddr
    ) public returns (address) {
        bytes20 alkesId =
bytes20(sha256(abi.encodePacked(msg.sender, block.timestamp)));
        RawAlkes alkes = new RawAlkes(
            msg.sender,
            address(alkesId),
            RawAlkes.AlkesDetails({
                namaAlkes: alkesDetails.namaAlkes,
                deskripsiAlkes: alkesDetails.deskripsiAlkes,
                klasifikasiAlkes: alkesDetails.klasifikasiAlkes,
                tipeAlkes: alkesDetails.tipeAlkes,
                kelasAlkes: alkesDetails.kelasAlkes,
                kelasResiko: alkesDetails.kelasResiko,
                noIzinEdar: alkesDetails.noIzinEdar
            }),
            distributorAddr,
            kemenkesAddr,
            rumahSakitAddr
        );

        dataAlkesManufaktur[msg.sender].push(address(alkes));
        return address(alkes);
    }

    function getNoOfPackagesOfSupplier() public view returns
(uint) {
        return dataAlkesManufaktur[msg.sender].length;
    }
```

```
    function getAllPackages() public view returns (address[]
memory) {
        return dataAlkesManufaktur[msg.sender];
    }


       function getAllPackagesData()
        public
        view
        returns (address[] memory retAddress, bytes32[] memory
retNamaAlkes, bytes32[] memory retKlasifikasi)
    {
        uint len = dataAlkesManufaktur[msg.sender].length;
        retAddress = new address[](len);
        retNamaAlkes = new bytes32[](len);
        retKlasifikasi = new bytes32[](len);

        for (uint i = 0; i < len; i++) {
            address alkesAddr =
dataAlkesManufaktur[msg.sender][i];
            RawAlkes alkesNew = RawAlkes(alkesAddr);

            retAddress[i] = alkesAddr;
            retNamaAlkes[i] = alkesNew.getNamaAlkes();
            retKlasifikasi[i] = alkesNew.getKlasifikasiALkes();
        }

        return (retAddress, retNamaAlkes, retKlasifikasi);
    }
}
```

Lampiran 2. Source Code *Smart* Contract (RawAlkes.sol)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
pragma experimental ABIEncoderV2;

import "./Transactions.sol";

contract RawAlkes {
    address public immutable owner;

    enum PackageStatus {
        AtManufacturer,
        AtDistributor,
        AtKemenkes,
        AtRumahSakit
    }

    struct AlkesDetails {
        bytes32 namaAlkes;
        bytes32 deskripsiAlkes;
        bytes32 klasifikasiAlkes;
        bytes32 tipeAlkes;
        bytes32 kelasAlkes;
        bytes32 kelasResiko;
        bytes32 noIzinEdar;
    }

    address public immutable productid;
    address public distributor;
    address public immutable manufakturer;
    address public kemenkes;
    address public rumahSakit;

    AlkesDetails public alkesDetails;
    PackageStatus public status;
    address public txnContractAddress;

    constructor(
        address _creatorAddr,
        address _productid,
        AlkesDetails memory _alkesDetails,
        address _distributorAddr,
        address _kemenkesAddr,
        address _rumahSakitAddr
    ) {
```

```
        require(
            _creatorAddr != address(0) &&
                _productid != address(0) &&
                _distributorAddr != address(0) &&
                _kemenkesAddr != address(0) &&
                _rumahSakitAddr != address(0)
        );

        owner = _creatorAddr;
        productid = _productid;
        alkesDetails = _alkesDetails;
        manufakturer = _creatorAddr;
        distributor = _distributorAddr;
        kemenkes = _kemenkesAddr;
        rumahSakit = _rumahSakitAddr;
        status = PackageStatus.AtManufacturer;
        initializeTxnContract(_distributorAddr);
    }

    function initializeTxnContract(address distributorAddr)
internal {
        Transactions txnContract = new
Transactions(distributorAddr);
        txnContractAddress = address(txnContract);
    }
    function getRawAlkes()
        public
        view
        returns (
            address,
            AlkesDetails memory,
            address,
            address,
            address,
            address,
            address
        )
    {
        return (
            productid,
            alkesDetails,
            manufakturer,
            distributor,
            kemenkes,
            rumahSakit,
            txnContractAddress
        );
    }
```

```
    function getNamaAlkes() public view returns (bytes32) {
        return alkesDetails.namaAlkes;
    }

    function getKlasifikasiALkes() public view returns (bytes32)
{
        return alkesDetails.klasifikasiAlkes;
    }

    function getRawAlkesStatus() public view returns (uint) {
        return uint(status);
    }

    function updatedistributorAddress(address addr) public {
        require(addr != address(0), "Invalid address");

        distributor = addr;
        status = PackageStatus.AtDistributor;
    }

    function distributorToKemenkes(address addr) public {
        require(addr != address(0), "Invalid address");

        kemenkes = addr;
    }

    function izinEdarApprove(bytes32 nomor, address addr) public
{
        alkesDetails.noIzinEdar = nomor;
        require(addr != address(0), "Invalid address");
        kemenkes = addr;
        status = PackageStatus.AtKemenkes;
    }

    function distributorToRs(address addr) public {
        require(addr != address(0), "Invalid address");
        rumahSakit = addr;
        status = PackageStatus.AtRumahSakit;
    }
}
```

Lampiran 3. Source Code *Smart* Contract (Transactions.sol)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
pragma experimental ABIEncoderV2;

contract Transactions {
    address public immutable creator;
    struct Txns {
        bytes32 txnHash;
        address fromAddr;
        address toAddr;
        bytes32 prevTxn;
        uint timestamp;
    }
    mapping(uint => Txns) public transactions;
    uint public txnCount = 0;

    event TxnCreated(
        bytes32 _txnHash,
        address _from,
        address _to,
        bytes32 _prev,
        uint _timestamp
    );

    constructor(address _creator) {
        require(_creator != address(0));
        creator = _creator;
    }

{

    function createTxnEntry(
        bytes32 txnHash,
        address from,
        address to,
        bytes32 prev
    )


    public {
        uint blockNumber = block.number;
        if (txnCount == 0) {
            transactions[txnCount] = Txns(
                txnHash,
                from,
```

```
                to,
                prev,
                blockNumber
            );
        } else {
            require(
                transactions[txnCount - 1].txnHash == prev,
                "Transaction error occurred!"
            );
            transactions[txnCount] = Txns(
                txnHash,
                from,
                to,
                prev,
                blockNumber
            );
        }
        txnCount += 1;
        emit TxnCreated(txnHash, from, to, prev, blockNumber);
    }

    function getAllTransactions() public view returns (Txns[]
memory) {
        uint len = txnCount;
        Txns[] memory ret = new Txns[](len);
        for (uint i = 0; i < len; i++) {
            ret[i] = transactions[i];
        }
        return ret;
    }
}
```

Lampiran 4. Full Source Code

Full Source Code
fadhil-khusnul/alkes-*blockchain* (github.com)