TUGAS BESAR
IF4035 Blockchain
Semester I 2024/2025

Kelompok A

Fakhri Muhammad Mahendra	13521045
Kenneth Ezekiel Suprantoni	13521089
Fakhri Putra Mahardika	18221080

INSTITUT TEKNOLOGI BANDUNG

2024

Daftar Isi

Daftar Gambar

Daftar Tabel

I. Problem Statement

Sistem pelayanan kesehatan di Indonesia, khususnya yang melibatkan asuransi BPJS, masih menghadapi berbagai masalah yang dapat dimanfaatkan oleh pihak-pihak yang tidak bertanggung jawab, berpotensi mengarah pada praktik korupsi. Salah satu masalah utama adalah kurangnya transparansi dalam proses-proses krusial seperti pembayaran dan klaim asuransi. Transaksi yang terjadi antara penyedia layanan kesehatan dan peserta BPJS tidak dapat diakses atau diawasi secara langsung oleh pengguna atau publik, membuka celah bagi manipulasi dan penyalahgunaan. Selain itu, proses pencatatan rekam medis elektronik dan transaksi klaim yang dilakukan selama ini tidak aman dan sering kali lambat, menyebabkan keterlambatan pelayanan dan potensi kesalahan dalam penanganan data medis pasien.

Tantangan ini menuntut solusi yang tidak hanya meningkatkan tingkat transparansi dalam sistem asuransi, tetapi juga menyederhanakan dan mempercepat proses transaksi serta pencatatan data medis secara digital. Diperlukan sebuah sistem yang dapat memberikan transparansi yang lebih baik bagi semua pihak yang terlibat dan memastikan bahwa transaksi dilakukan secara efisien, aman, dan tepat waktu. Dengan solusi yang lebih transparan dan efisien, diharapkan dapat mengurangi potensi penyalahgunaan dan meningkatkan kepercayaan publik terhadap sistem asuransi kesehatan.

II. Rancangan Solusi

II.1. Solusi

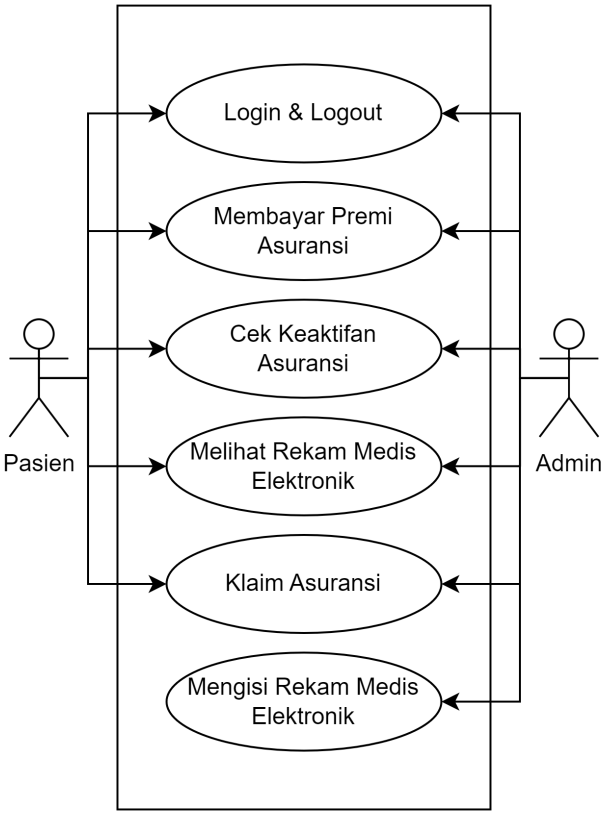
Solusi dari permasalahan tersebut adalah sebuah sistem pelayanan kesehatan yang terintegrasi dengan Blockchain untuk melakukan mendaftarkan rekam medis elektronik, melakukan pembayaran asuransi, dan melakukan klaim asuransi. Solusi ini dapat memanfaatkan Blockchain untuk transparansi dan juga keamanannya. Beberapa keuntungan utama dari solusi ini adalah:

1. Pemrosesan klaim asuransi yang lebih cepat, terstandarisasi, terotomasi, dan transparan
2. Meningkatkan keamanan, integritas, dan manajemen data rekam medis
3. Pencegahan penipuan dan korupsi dengan sistem distribusi token yang baik dan transparansi sistem
4. Interoperabilitas antar penyedia layanan kesehatan

Beberapa fitur yang akan dikembangkan di dalam solusi ini adalah:

1. Pembayaran premi asuransi
2. Pengecekan premi asuransi
3. Penambahan rekam medis elektronik
4. Pengecekan dan pembayaran hasil pemeriksaan
5. Klaim asuransi untuk pembayaran hasil pemeriksaan
6. Pembayaran menggunakan Medical Token

II.2. Use Case Diagram



Gambar II.1. Use Case Diagram

Use Case Diagram ini memiliki dua user, yaitu user dengan role Pasien, dan user dengan role Admin. Admin merupakan user dengan *elevated privilege* untuk melakukan pengisian rekam medis elektronik untuk pasien. Selain itu, kedua role dapat melakukan pembayaran premi asuransi pribadi, cek keaktifan asuransi untuk bulan dan tahun tertentu, melihat daftar rekam medis pribadi, dan juga melakukan klaim asuransi untuk rekam medis.

III. Rancangan Teknis

III.1. Tech Stack

Solusi ini dibagi menjadi tiga komponen utama, yaitu komponen Smart Contract atau Blockchain, komponen client application, dan komponen oracle. Smart Contract dibangun di atas Blockchain Platform Ethereum dengan bahasa pemrograman Solidity. Justifikasi dari pemilihan tersebut didasarkan pada:

1. Keterbukaan dan Transparansi Blockchain Ethereum, sehingga data dapat bersifat publik dan terdesentralisasi, sehingga meningkatkan keamanan untuk pembayaran klaim asuransi.
2. Interoperabilitas dan Integrasi Blockchain Ethereum yang lebih tinggi, sehingga memudahkan pengembangan seperti pengembangan token dengan memanfaatkan ERC-20 dan juga interoperabilitas dengan oracle yang lebih terintegrasi.
3. Kontrol Akses yang lebih terbuka pada Blockchain Ethereum jika dibandingkan dengan Hyperledger Fabric lebih baik karena seharusnya seluruh pengguna dapat melakukan pendaftaran dan verifikasi, sehingga tidak perlu mengontrol akun wallet, yang menjadikan kompleksitas sistem lebih sederhana dan keamanan akun yang lebih tinggi untuk pengguna.
4. Familiaritas penggunaan bahasa pemrograman Solidity yang lebih tinggi dibandingkan bahasa lainnya seperti Hyperledger Fabric yang dimiliki oleh tim kami, sehingga akan memudahkan dalam pengembangan karena lebih menguasai ekosistem pengembangannya.

Komponen Smart Contract dibangun menggunakan bahasa pemrograman Solidity dengan bantuan kakas hardhat, ethers, dan openzeppelin-contracts, yang membantu untuk melakukan *deployment* Smart Contract di atas Blockchain Ethereum lokal menggunakan skrip yang ditulis dengan bahasa TypeScript. Komponen client application dibangun menggunakan TypeScript, dengan bantuan kakas seperti Vite untuk *local development tools*, Metamask, cryptojs, ethers, web3js untuk integrasi dengan web3, dan Chakra-UI untuk memudahkan pembangunan komponen *front-end page*. Oracle dibangun menggunakan skrip JavaScript, yang dijalankan menjadi sebuah skrip lokal dengan bantuan kakas nodeJS dan juga kakas web3 untuk berinteraksi dengan Smart Contract dari Oracle yang akan menyimpan data dari oracle di dalam Blockchain.

III.2. High Level Design

Komponen yang digunakan dalam solusi ini dibagi menjadi tiga komponen utama, yaitu Client, Smart Contract, dan Oracle. Client dan Smart Contract

juga dibagi menjadi komponen-komponen lebih kecil yang saling berinteraksi dengan satu sama lain, berikut daftar komponen yang ada di dalam setiap komponen utama:

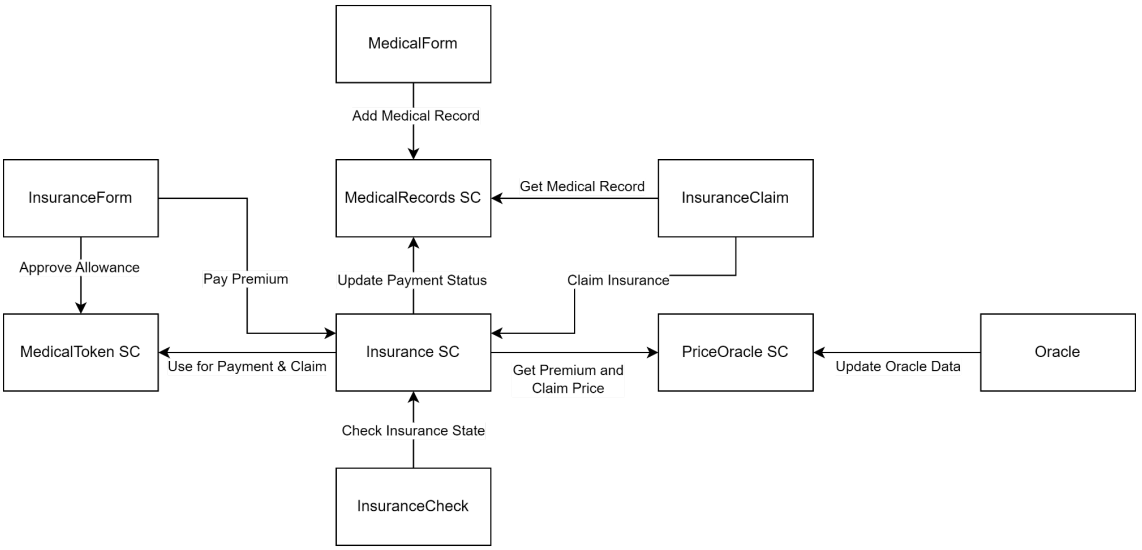
1. Client
    a. InsuranceCheck, Komponen untuk melihat dan menampilkan kondisi keaktifan dari asuransi pada bulan dan tahun tertentu, berinteraksi dengan Insurance SC
    b. InsuranceClaim, Komponen untuk mengambil rekam medis dari MedicalRecords SC lalu menampilkannya ke pengguna untuk melakukan Claim dengan memanggil Insurance SC
    c. InsuranceForm, Komponen untuk melakukan pembayaran premi dengan berinteraksi dengan Insurance SC dan MedicalToken SC untuk melakukan pembayaran dengan MedicalToken
    d. MedicalForm, Komponen untuk menambahkan rekam medis dengan berinteraksi dengan MedicalRecords SC
2. Smart Contract
    a. Insurance, Smart Contract Insurance yang akan mencatat data keaktifan asuransi untuk setiap pengguna di setiap bulan dan tahun, melakukan pembayaran premi, dan melakukan klaim asuransi, berinteraksi dengan MedicalToken untuk transaksi, MedicalRecords untuk *update status* pembayaran, dan Price Oracle untuk melihat data harga premi
    b. MedicalRecords, Smart Contract MedicalRecords yang akan mencatat data rekam medis untuk setiap pengguna, dan mencatat *status* pembayaran dari setiap pengecekan medis tersebut
    c. MedicalToken, Smart Contract MedicalToken untuk melakukan *minting* dan juga transaksi dari pembayaran dan klaim asuransi
    d. PriceOracle, Smart Contract untuk berinteraksi dengan Oracle lalu mendapatkan datanya dan menyimpannya ke dalam Blockchain untuk digunakan oleh Smart Contract lainnya
3. Oracle
    a. App, Backend Client yang akan mengupdate harga premi dan harga klaim pada Smart Contract PriceOracle secara berkala dengan menggunakan data dari API external (dalam kasus ini menggunakan data dummy).

Terdapat juga beberapa komponen pembantu seperti:

a. ABI, Application Binary Interface dari Smart Contract yang digunakan oleh Client untuk mengetahui interface dari Smart Contract tersebut
b. Web3Context, Komponen untuk memberikan context web3 yang terpadu di dalam seluruh Client Application

c. eth-app, Komponen wrapper untuk pemanggilan fungsi-fungsi dari Smart Contract saat dipanggil di dalam Client Application
d. Deployment Scripts, Skrip dalam TypeScript untuk melakukan deployment dari setiap Smart Contract

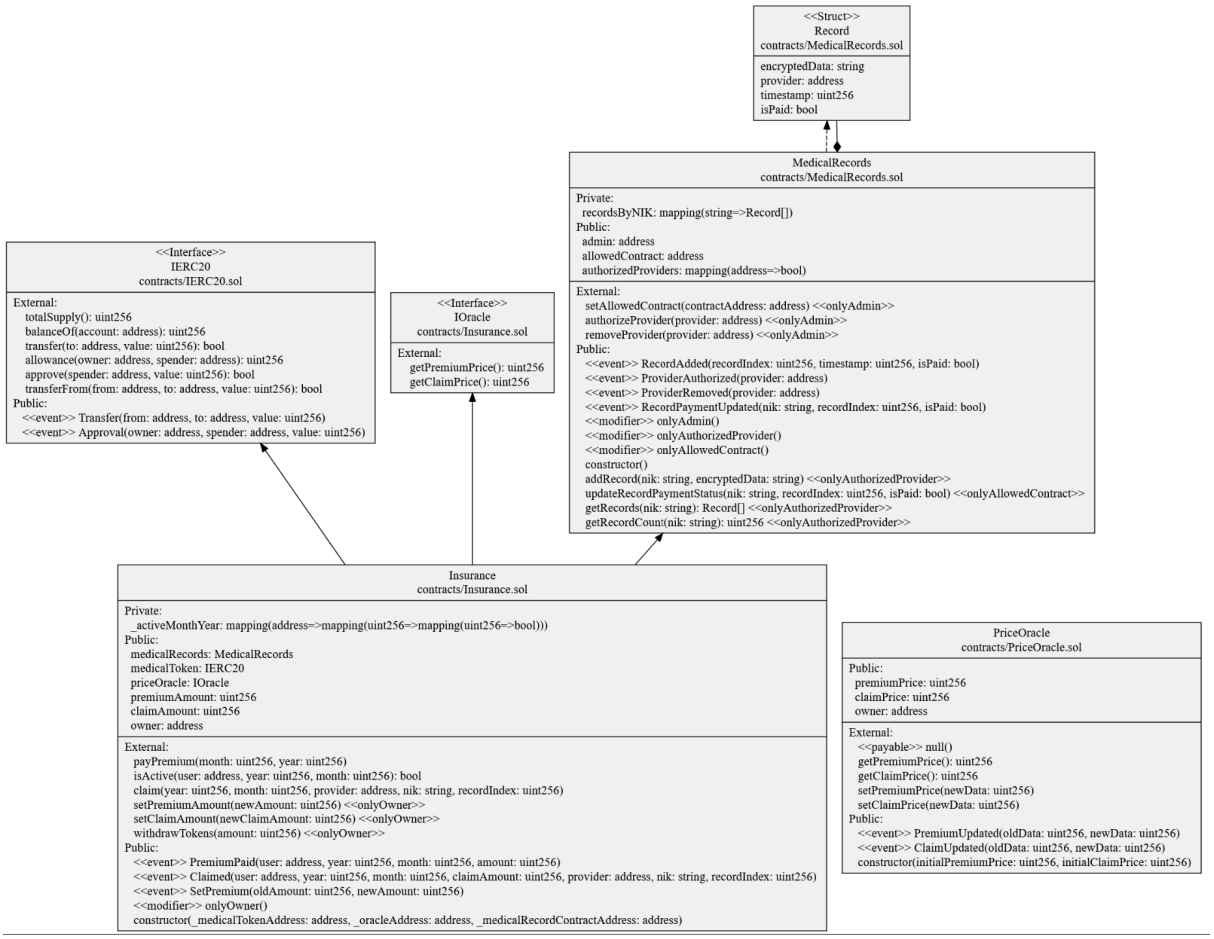
Interaksi antar komponen dapat lebih mudah dijelaskan menggunakan diagram interaksi antar komponen:



Gambar III.1. Diagram interaksi antar komponen

IV. Implementasi

IV.1. Class Diagram



Gambar IV.1. Class Diagram

Tabel IV.1. Insurance.sol Class Diagram

Insurance contracts/Insurance.sol
Private: _activeMonthYear: mapping(uint64=>mapping(uint256=>mapping(uint256=>bool))) Public: medicalRecords: MedicalRecords medicalToken: IERC20 priceOracle: IOracle premiumAmount: uint256 claimAmount: uint256 owner: address
External: payPremium(nik: uint64, month: uint256, year: uint256) isActive(nik: uint64, year: uint256, month: uint256): bool

claim(year: uint256, month: uint256, provider: address, nik: uint64, recordIndex: uint16) setPremiumAmount(newAmount: uint256) <<onlyOwner>> setClaimAmount(newClaimAmount: uint256) <<onlyOwner>> withdrawTokens(amount: uint256) <<onlyOwner>> Public: <<event>> PremiumPaid(user: address, year: uint256, month: uint256, amount: uint256) <<event>> Claimed(user: address, year: uint256, month: uint256, claimAmount: uint256, provider: address, nik: uint64, recordIndex: uint16) <<event>> SetPremium(oldAmount: uint256, newAmount: uint256) <<modifier>> onlyOwner() constructor(_medicalTokenAddress: address, _oracleAddress: address, _medicalRecordContractAddress: address)

Tabel IV.2. MedicalRecords.sol Class Diagram

MedicalRecords contracts/MedicalRecords.sol
Private: recordsByNIK: mapping(uint64=>Record[]) Public: admin: address allowedContract: address authorizedProviders: mapping(address=>bool)
External: setAllowedContract(contractAddress: address) <<onlyAdmin>> authorizeProvider(provider: address) <<onlyAdmin>> removeProvider(provider: address) <<onlyAdmin>> Public: <<event>> RecordAdded(recordIndex: uint16, timestamp: uint256, isPaid: bool) <<event>> ProviderAuthorized(provider: address) <<event>> ProviderRemoved(provider: address) <<event>> RecordPaymentUpdated(nik:uint64, recordIndex: uint16, isPaid: bool) <<modifier>> onlyAdmin() <<modifier>> onlyAuthorizedProvider() <<modifier>> onlyAllowedContract() constructor() addRecord(nik:uint64, encryptedData: string) <<onlyAuthorizedProvider>> updateRecordPaymentStatus(nik:uint64, recordIndex: uint16, isPaid: bool) <<onlyAllowedContract>> getRecords(nik:uint64): Record[] <<onlyAuthorizedProvider>> getRecordCount(nik:uint64): uint256 <<onlyAuthorizedProvider>>

Tabel IV.3. Price Oracle.sol Class Diagram

PriceOracle contracts/PriceOracle.sol
Public: premiumPrice: uint256 claimPrice: uint256 owner: address
External: <<payable>> null() getPremiumPrice(): uint256 getClaimPrice(): uint256

```
        setPremiumPrice(newData: uint256)
        setClaimPrice(newData: uint256)
Public:
        <<event>> PremiumUpdated(oldData: uint256, newData: uint256)
        <<event>> ClaimUpdated(oldData: uint256, newData: uint256)
        constructor(initialPremiumPrice: uint256, initialClaimPrice: uint256)
```

IV.2. Implementasi Oracle

Implementasi Oracle dibagi menjadi dua komponen utama, External Oracle yang mengakses External Data Source, dalam hal ini kami simulasikan menggunakan static data, dan juga PriceOracle Smart Contract untuk mengambil atau menerima data dari External Oracle dan menyimpannya di dalam Blockchain untuk diakses oleh Smart Contract lain. Berikut merupakan penjelasan lebih detail dari implementasi Oracle:

1. Penggunaan Oracle: Oracle digunakan untuk mengambil data harga premi dan juga harga klaim dari sumber eksternal dan secara dinamis.
2. Struktur data: Oracle mengembalikan data berupa BigInteger, dengan struktur '[BigInt, BigInt]' untuk '[Harga Premi, Harga Klaim]'.
3. Mekanisme pengambilan data: Mekanisme yang digunakan adalah mekanisme push dari Oracle ke PriceOracle Smart Contract, sehingga setiap interval sekian detik atau setiap ada update baru yang dilakukan pada harga, Oracle akan memanggil Smart Contract PriceOracle untuk melakukan update harga yang ada di dalam Blockchain.
4. Skenario buruk yang mungkin terjadi adalah manipulasi dari data dari Oracle karena tidak menggunakan konsensus dan juga diset oleh entitas eksternal. Hal ini tidak bisa dihindari karena naluri dari sistem dimana hanya pemerintah yang dapat menentukan dan melakukan update pada harga, menggunakan Skrip tersebut, yang memiliki credentials untuk mengakses PriceOracle Smart Contract, karena Smart Contract tersebut dibatasi aksesnya hanya oleh Owner. Sehingga skenario buruknya adalah jika pemerintah kebocoran credentials tersebut, dan solusinya hanyalah melakukan deployment ulang dan menonaktifkan PriceOracle Smart Contract.

IV.3. Design Pattern

IV.3.1 Check-Effect Interaction Pattern

Check effect digunakan pada fungsi claim dan payPremium pada smart contract Insurance.sol. Pattern in digunakan untuk mengurangi attack surface dari kontrak dengan niat buruk yang berusaha membajak alur kontrol setelah pemanggilan eksternal. Pattern ini digunakan dengan membuat perubahan state variable sebelum interaksi external.

IV.3.1 Push Interaction Pattern

Push kepada subscriber digunakan untuk melakukan update dari harga premi dan harga klaim pada PriceOracle Smart Contract, dimana setiap interval dan atau setiap terjadi update harga, hasil update tersebut dapat dipropagasikan dengan baik oleh Oracle ke PriceOracle Smart Contract untuk disimpan di dalam Blockchain.

IV.4. Optimasi

Terdapat beberapa optimasi yang dilakukan pada sistem blockchain ini. Optimasi berupa optimasi gas dan optimasi keamanan. Berikut merupakan optimasi yang dilakukan:

1. Menggunakan modifier 'external' daripada 'public' apabila fungsi tidak digunakan secara internal. Hal ini dilakukan untuk menghemat harga gas.
2. Mengganti string panjang dengan error bawaan yang sudah diberikan solidity (cth: 'OnlyAdmin()'). Hal ini dilakukan untuk menghemat harga gas.
3. Mengganti tipe data string pada nik, menjadi uint64 untuk menghemat harga gas.
4. Menggunakan Oracle Interface (IOracle) sehingga harga premi dan klaim bisa diset secara dinamis oleh sumber eksternal sehingga lebih fleksibel, tanpa perlu melakukan deployment ulang kontrak.
5. Memisahkan kontrak Insurance, MedicalRecords, dan PriceOracle untuk separation of concerns yang lebih baik sehingga sistem lebih modular dan lebih mudah dimaintain.
6. Mapping efisien menggunakan ‘_activeMonthYear’ untuk mencatat keaktifan asuransi dari setiap user untuk setiap bulan dan tahun, sehingga tidak perlu melakukan iterasi pada dataset yang besar.
7. Menggunakan access control dengan modifier ‘onlyOwner’ untuk memastikan bahwa fungsi tersebut hanya dapat dipanggil oleh pemilik kontra

Daftar Anggota

NIM	Nama	Pembagian Tugas
13521045	Fakhri Muhammad Mahendra	Smart Contract Medical Records Laporan Demo
13521089	Kenneth Ezekiel Suprantoni	Smart Contract Insurance Client application Oracle Laporan
18221080	Fakhri Putra Mahardika	Medical Token Laporan