



**IMPLEMENTASI SMART CONTRACT PADA PLATFORM  
CROWDFUNDING BERBASIS BLOCKCHAIN ETHEREUM  
MENGUNAKAN ALGORITMA KONSENSUS PROOF OF STAKE**

**SKRIPSI**

Diajukan sebagai salah satu syarat penelitian untuk memenuhi kelulusan program studi S1 Teknologi Informasi Fakultas Ilmu Komputer Universitas Jember

Oleh:

**Muhammad Amanda Maulana Malik Ibrahim**

**212410102035**

**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET,  
DAN TEKNOLOGI  
UNIVERSITAS JEMBER  
FAKULTAS ILMU KOMPUTER  
PROGRAM STUDI S1 TEKNOLOGI INFORMASI  
2025**

## **PERSEMBAHAN**

Puji Syukur kehadiran Allah SWT yang telah melimpahkan Rahman serta hidayah-Nya sehingga penulis dapat menyelesaikan skripsi ini dengan penuh kesabaran dan kerendahan hati yang luar biasa. Keberhasilan dalam penyelesaian skripsi ini tentunya tidak lepas dari bantuan berbagai pihak. Oleh karena itu penulis menyampaikan terima kasih kepada:

1. Allah SWT yang senantiasa memerikan rahmat serta hidayah-Nya dan memberikan kemudahan kepada penulis dalam menyelesaikan skripsi
2. Ayahanda Muhammad Bahrur Razi dan Ibunda Siti Khumairoh yang selalu memerikan dukungan penulis berupa moral maupun materil yang tak terhingga, serta doa yang tiada putusnya yang selalu diberikan kepada penulis sehingga penulis mampu menyelesaikan studi sarjana hingga selesai.
3. Bapak Diksy Media Firmansyah S.Kom., M.Kom dan Bapak Yanuar Nurdiansyah S.T., M.Cs selaku dosen pembimbing yang telah mendukung, memeberi nasehat dan membantu penulis untuk menyelesaikan dan menyempurnakan skripsi ini.
4. Keluarga besar Bapak Alm. H. Moh. Amin dan Ibu Hj. Siti Susmiyah yang selalu memberikan doa serta dukungan dalam proses Menyusun skripsi.
5. Kepada pemilik NIM 1034, Terima kasih telah menjadi bagian dari perjalanan yang penuh rintangan ini. Berkontribusi banyak dalam proses penyusunan skripsi ini baik waktu, tenaga, doa dan dukungan yang telah di berikan.
6. Terimakasih kepada teman-teman Fakultas Ilmu Komputer Universitas Jember dan penghuni indekost putra Dwi Nandana 1 yang telah menemani penulis selama di masa perkuliahan, pertarungan di land of dawn, dan pertempuran di war classic maupun war liga, hingga sampai skripsi ini selesai.

### **MOTTO**

*“Sesungguhnya beserta kesulitan ada kemudahan, apabila engkau telah selsai (dengan suatu kebaikan), teruslah bekerja keras (untuk kebaikan yang lain), dan hanya kepada Tuhanmu kamu berharap”*

(Q.S Al-Insyirah: 5 -8)

### PERNYATAAN ORISINALITAS

Saya yang bertanda tangan di bawah ini:

Nama: Muhammad Amanda Maulana Malik Ibrahim

NIM: 212410102035

Menyatakan dengan sesungguhnya karya ilmiah yang berjudul “Implementasi *Smart Contract* Pada Platform *Crowdfunding* Berbasis Blockchain Ethereum Menggunakan Algoritma Konsensus Proof of Stake” adalah benar-benar hasil karya sendiri, kecuali jika dalam pengutipan substansi yang disebutkan sumbernya, belum pernah diajukan pada institusi manapun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika di kemudian hari pernyataan ini tidak benar.

Jember, XX XXXX 2025

Yang menyatakan,

M Amanda Maulana Malik I

212410102035

## HALAMAN PERSETUJUAN

Skripsi berjudul *Implementasi Smart Contract Pada Platform Crowdfunding Berbasis Blockchain Ethereum Menggunakan Algoritma Konsensus Proof of Stake* telah diuji oleh Fakultas Ilmu Komputer Universitas Jember pada:

Hari :

Tanggal :

Tempat : Fakultas Ilmu Komputer Universitas Jember

Pembimbing Tanda Tangan

### 1. Pembimbing Utama

Nama : Diksy Media Firmansyah, S.Kom., M.Kom. (.....)

NIP : 199112132023211015

### 2. Pembimbing Anggota

Nama : Yanuar Nurdiansyah, S.T., M.Cs. (.....)

NIP : 198201012010121004

### Penguji

#### 1. Penguji Utama

Nama : Prof. Drs. Antonius C. Prihandoko, M.App.Sc. Ph.D. (.....)

NIP : 196909281993021001

#### 2. Penguji Anggota

Nama : Mohammad Zarkasi, S.Kom., M.Kom (.....)

NIP : 199011112019031018

### **ABSTRAK**

Permasalahan – metode – tujuan penelitian – manfaatnya – hasil penelitian -kesimpulan  
-kekurangna dan kelebihan

### **ABSTRACT**

## **RINGKASAN**

## PRAKATA

Puji Syukur kehadiran Allah SWT yang telah melimpahkan rahmat serta hidayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul **“Implementasi Smart Contract Pada Platform Crowdfunding Berbasis Blockchain Ethereum Menggunakan Algoritma Konsensus Proof of Stake”** dengan penuh kerendahan hati dan kesabaran yang luar biasa.

Keberhasilan dalam penulisan skripsi ini tentunya tidak lepas dari berbagai pihak. Oleh karena itu penulis menyampaikan terima kasih kepada:

1. Prof. Drs. Antonius Cahya Prihandoko, M.App.Sc. Ph.D. Selaku Dekan Fakultas Ilmu Komputer Universitas Jember.
2. Priza Pandunata S.Kom., M.Sc. Selaku ketua program studi S1 Teknologi Informasi
3. Yanuar Nurdiansyah, S.T., M.Cs. Selaku dosen wali penulis.
4. Diksy Media Firmansyah. S.Kom., M.Kom, dan Yanuar Nurdiansyah, S.T., M.Cs. yang telah membimbing, mendukung, dan memberi nasihat sehingga bisa menyelesaikan skripsi.
5. Prof. Drs. Antonius Cahya Prihandoko, M.App.Sc. Ph.D. dan Mohammad Zarkasi. S.Kom., M.Kom. Selaku penguji satu dan penguji dua skripsi penulis
6. Para dosen, staff, dan seluruh civitas akademika FASILKOM UNEJ yang telah memberikan semangat, dukungan dan fasilitas selama perkuliahan.

Jember, XX XXXX 2025

M Amanda Maulana Malik I

212410102035



## DAFTAR ISI

PERSEMBAHAN .....	i
MOTTO .....	ii
PERNYATAAN ORISINALITAS .....	iii
HALAMAN PERSETUJUAN .....	iv
ABSTRAK .....	v
RINGKASAN .....	vi
PRAKATA .....	vii
DAFTAR ISI .....	viii
DAFTAR TABEL .....	x
DAFTAR GAMBAR .....	xi
DAFTAR KODE .....	xii
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	3
1.3 Tujuan Penelitian .....	3
1.4 Manfaat Penelitian .....	3
1.5 Batasan Masalah .....	4
BAB II TINJAUAN PUSTAKA .....	5
2.1 Penelitian Terdahulu .....	5
2.2 Blockchain .....	6
2.3 Smart Contract .....	7
2.4 Konsensus Proof of Stake .....	8
2.5 Kriptografi .....	9
2.6 Ethereum .....	9
2.7 Ethereum Sepolia Testnet .....	9
2.7 Decentralized Apps (Dapps) .....	10
2.8 Thirdweb .....	11
2.9 Hardhat JS .....	11
2.10 Mythril .....	11
BAB III METODOLOGI PENELITIAN .....	12
3.1 Jenis Penelitian .....	12

3.2 Objek Penelitian .....	12
3.3 Tahapan Penelitian.....	12
3.3.1 Identifikasi dan Perumusan Masalah.....	12
3.3.2 Perancangan Sistem.....	13
3.3.3 Pembuatan Desain Blockchain.....	15
3.3.4 Pemodelan Smart Contract.....	16
3.3.5 Implementasi Smart Contract dan Web3 .....	21
3.3.6 Pengujian .....	21
BAB IV HASIL DAN PEMBAHASAN.....	23
4.1 Implementasi Smart Contract dan Dapps .....	23
4.2 Implementasi Web3 .....	24
4.3 Uji Fungsionalitas.....	24
4.4 Uji Non-fungsionalitas .....	28
BAB V KESIMPULAN DAN SARAN.....	35
DAFTAR PUSTAKA .....	36

## DAFTAR TABEL

Tabel 2.1 Penelitian terdahulu.....	6
Tabel 2.2 Perbandingan centralized app dan decentralized app.....	10
Tabel 3.1 Kebutuhan fungsional dan non-fungsional sistem .....	14
Tabel 3.2 Tools dan teknologi yang digunakan.....	14
Tabel 3.3 Nama dan keterangan kontrak.....	16
Tabel 3.4 Keterangan struktur campaign .....	17
Tabel 3.5 Keterangan kontrak campaign.....	18
Tabel 3. 6 Keterangan kontrak staking token dan reward token .....	19
Tabel 3.7 Keterangan kontrak token drop .....	19
Tabel 3.8 Keterangan kontrak staking.....	20
Tabel 4.1 Daftar test case untuk masing-masing kontrak .....	25

## DAFTAR GAMBAR

Gambar 2.1 Struktur blockchain .....	7
Gambar 2.2 Alur algoritma proof of stake .....	8
Gambar 2.3 Perbedaan arsitektur jaringan testnet Ethereum .....	10
Gambar 3.1 Tahapan penelitian.....	12
Gambar 3.2 Alur sistem transaksi crowdfunding.....	15
Gambar 3.3 Alur sistem staking .....	15
Gambar 3. 4 Desain arsitektur layer blockchain .....	16
Gambar 3.5 Class diagram kontrak crowdfunding .....	17
Gambar 3.6 Class diagram staking token dan reward token.....	18
Gambar 3.7 Class diagram kontrak token drop.....	19
Gambar 3.8 Class diagram kontrak staking .....	20
Gambar 4.1 Membuat proyek pada EVM .....	23
Gambar 4.2 Deploy smart contract ke EVM.....	23
Gambar 4.3 Hasil pengujian kontrak crowdfunding dengan Hardhat.....	25
Gambar 4.4 Hasil pengujian kontrak crowdfunding dengan Hardhat.....	26
Gambar 4.5 Hasil pengujian kontrak staking dan reward token dengan Hardhat. ....	26
Gambar 4.6 Hasil pengujian kontrak token drop dengan Hardhat.....	27
Gambar 4.7 Hasil pengujian kontrak staking dengan Hardhat .....	27
Gambar 4.8 Hasil pengujian kontrak crowdfunding dengan Mythril .....	28
Gambar 4.10 Hasil pengujian kembali smart contract crowdfunding.....	30
Gambar 4.11 Hasil pengujian kontrak staking token dengan Mythril .....	30
Gambar 4.12 Hasil pengujian kontrak reward token dengan Mythril.....	30
Gambar 4.13 Hasil pengujian kontrak token drop dengan Mythril .....	31
Gambar 4.14 Hasil pengujian kembali kontrak token drop .....	33
Gambar 4.15 Hasil pengujian kontrak staking dengan Mythril .....	33
Gambar 4.16 Interface utama sepolia etherscan .....	33
Gambar 4.17 Interface sepolia etherscan detail transaksi .....	34

## DAFTAR KODE

Kode 4.1 Perbaikan smart contract crowdfunding.....	29
Kode 4.2 Perbaikan smart contract token drop .....	32

## DAFTAR LAMPIRAN

Lampiran 1. Wireframe dan interface halaman dashboard .....	38
Lampiran 2. Wireframe dan interface halaman detail kampanye.....	38
Lampiran 3. Wirefram dan interface halaman membuat kampanye .....	38
Lampiran 4. Wireframe dan interface halaman riwayat donasi .....	38
Lampiran 5. Wireframe dan interface halaman klaim staking token .....	39
Lampiran 6. Wireframe dan interface halaman staking .....	39
Lampiran 7. Kode smart contract crowdfunding .....	39
Lampiran 8. Kode smart contract staking token .....	40
Lampiran 9. Kode smart contract reward token.....	41
Lampiran 10. Kode smart contract token drop .....	43
Lampiran 11. Kode smart contract staking .....	43

## **BAB I**

### **PENDAHULUAN**

#### **1.1 Latar Belakang**

**Commented [AM1]:** Perbaiki kepenulisan kutipan

Crowdfunding berasal dari kata "crowd" yang berarti keramaian atau sekelompok orang, dan "funding" yang berarti pengumpulan dana. Crowdfunding adalah metode pengumpulan dana dari banyak orang yang memiliki minat terhadap suatu kegiatan atau bisnis. Crowdfunding memberikan peluang bagi kreator untuk mendapatkan dana dari masyarakat yang tertarik dengan gagasan mereka dan menciptakan potensi untuk mempercepat pengembangan bisnis atau kegiatan yang sedang berjalan (Aufila et al., 2024).

Sistem crowdfunding tradisional sudah memberikan fungsional dengan baik, namun masih terdapat beberapa tantangan dalam sistem terpusat, terutama terkait dengan kepercayaan dan keamanan. Funder sering kali kesulitan menentukan apakah kampanye pada platform crowdfunding benar-benar sah, funder juga bisa saja dihadapkan pada kasus penipuan, kampanye yang tidak memenuhi syarat sebagai penerima dana, dan risiko penyalahgunaan dana. Selain itu, peran perantara pihak ketiga dalam proses ini menyebabkan biaya operasional yang tinggi (Chatkar et al., 2023).

Salah satu solusi untuk mengatasi hal ini adalah dengan menerapkan teknologi blockchain. Blockchain, sebagai teknologi desentralisasi, menawarkan tingkat keamanan dan transparansi yang tinggi. Teknologi ini berfungsi sebagai sistem pencatatan yang terdistribusi dan terdesentralisasi, di mana data disimpan dalam blok-blok yang saling terhubung dan terenkripsi serta terverifikasi oleh banyak node. Proses verifikasi ini memastikan bahwa data hampir mustahil dimanipulasi tanpa persetujuan mayoritas jaringan. Dengan demikian, blockchain menjadi solusi potensial untuk meningkatkan kepercayaan, keamanan dan transparansi dalam sistem penggalangan dana (Hasan et al., 2024).

Lingkup teknologi blockchain terdapat smart contract yang bersifat objektif untuk menentukan secara spesifik bagaimana proses transaksi akan dikelola dan

tindakan apa yang akan diambil ketika suatu peristiwa terjadi. Seluruh data dan transaksi dapat divalidasi secara otomatis, dan dieksekusi oleh kode dalam jaringan blockchain sebagai perjanjian digital, Kontrak ini disimpan dalam blockchain dan didistribusikan ke semua node, sehingga tidak bisa diubah, selayaknya data pada blockchain yang bersifat permanen dan transparan (Hermawan et al., 2023).

Teknologi blockchain sebagai fundamental di balik mata uang kripto ini telah merevolusi cara pengelolaan dan pemahaman terhadap aset digital (Julio et al., 2024). Namun, seiring dengan berkembangnya teknologi, muncul kekhawatiran terkait dampak dari teknologi kripto itu sendiri, salah satu faktor utama yang memengaruhi besarnya dampak lingkungan adalah konsensus yang digunakan. Sebagai contoh Proof of Work (PoW) yang digunakan oleh Bitcoin yang diketahui membutuhkan daya komputasi yang besar, sehingga mengakibatkan peningkatan konsumsi energi, emisi karbon, hingga limbah elektronik. Di sisi lain, algoritma Proo of Stake (PoS) dinilai jauh lebih ramah lingkungan karena tidak memerlukan persaingan komputasi, Dengan demikian, peralihan ke sistem PoS dapat menjadi langkah strategis dalam mengurangi jejak karbon dan meningkatkan keberlanjutan ekosistem cryptocurrency secara keseluruhan (Mendl et al., 2022).

**Commented [AM2]:** Kelebihan Pos daripada Pow

Penelitian terdahulu yang dilakukan oleh Baihaqsani (2023) dengan mengimplementasikan smart contract blockchain pada sistem klaim asuransi. Penelitian ini mendapat hasil bahwa blockchain dapat meningkatkan transparansi, keamanan dan efisiensi pada proses klaim, serta mencegah manipulasi data melalui smart contract menggunakan algoritma Proof of Work (PoW). Penelitian lainnya yang dilakukan oleh Sahputra (2019) dengan mengimplementasikan smart contract pada sistem voting. Penelitian ini berhasil memastikan integritas data yang sudah dienkripsi dan disimpan pada jaringan blokchcain.

Beberapa penelitian terdahulu telah membuktikan bahwa permasalahan pada data yang bersifat tetap dan transaksional seperti data asuransi dan voting terdapat pada transparansi, dan integritas data yang semua permasalahan tersebut bisa diselesaikan menggunakan teknologi blockchain. Berdasarkan latar belakang tersebut, penulis mengusulkan untuk mengimplementasikan smart contract



blockchain pada sistem crowdfunding. Diharapkan penggalang dana dapat lebih mudah membangun kepercayaan dengan donator melalui transparansi penuh dalam transaksi, sementara penerima dana akan mendapatkan proses pencairan yang lebih cepat dan aman tanpa keterlibatan pihak ketiga yang kompleks.

### 1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas maka didapatkan rumusan masalah dalam penelitian adalah sebagai berikut:

1. Bagaimana proses perancangan dan implementasi smart contract untuk crowdfunding berbasis blockchain dengan memperhatikan aspek keamanan dan efisiensi dalam consensus Proof of Stake (PoS)?
2. Bagaimana hasil pengujian keamanan transaksi dalam smart contract tersebut ketika digunakan pada platform crowdfunding berbasis blockchain Ethereum?

### 1.3 Tujuan Penelitian

Dengan masalah yang sudah disebutkan, tujuan dari penelitian ini yaitu:

1. Menghasilkan smart contract yang dapat menangani proses crowdfunding dan menjamin keamanan data transaksi pada aplikasi berbasis blockchain Ethereum menggunakan konsensus Proof of Stake (PoS)
2. Menganalisis hasil pengujian smart contract untuk memastikan keamanan dan integritas data transaksi pada aplikasi berbasis blockchain Ethereum menggunakan konsensus Proof of Stake (PoS)

### 1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat antara lain:

- a. Bagi Akademisi dan peneliti

Menambah wawasan dan pengetahuan mengenai penerapan teknologi blockchain dalam sisi keamanan data pada platform crowdfunding menggunakan algoritma Proof of Stake, serta membantu memberikan referensi penelitian lebih lanjut mengenai penerapan teknologi blockchain.

**Commented [AM3]:** Perbaiki rumusan masalah

1. perbaiki kalimat aman dan efisien agar artinya tidak bias
2. fokus pada hasil dari pengujian, bukan mekanismenya lagi karena pada penelitian ini tidak mencari tahu mekanisme pengujiannya

b. Bagi Funder dan Fundraiser

Penelitian ini diharapkan memberikan wawasan yang bermanfaat bagi pemberi dana (funder) dan penerima dana (fundraiser) mengenai penerapan teknologi blockchain pada crowdfunding, sehingga funder dapat memastikan dana yang mereka berikan diterima dan dikelola dengan baik dan tidak disalahgunakan. Sedangkan fundraiser dapat membangun kepercayaan yang lebih melalui sistem yang lebih transparan dan terdesentralisasi

### **1.5 Batasan Masalah**

Dalam penelitian ini, penulis memberikan beberapa Batasan masalah untuk menghindari terjadi penyimpangan selama proses penelitian dan penulisan. Beberapa Batasan masalah antara lain:

1. Penelitian tidak membahas aspek off-chain dari platform crowdfunding, meliputi manajemen pengguna dan UI/UX aplikasi.
2. Analisis smart contract mencakup aspek keamanan, fungsionalitas dan non-fungsionalitas tanpa membandingkan dengan mekanisme crowdfunding konvensional.
3. Pengujian smart contract crowdfunding dilakukan pada jaringan testnet Sepolia Ethereum, tanpa implementasi di mainnet Ethereum.

## BAB II

### TINJAUAN PUSTAKA

#### 2.1 Penelitian Terdahulu

Beberapa penelitian terdahulu yang berkaitan dengan penelitian penulis ini akan disajikan pada tabel 2.1 sebagai berikut:

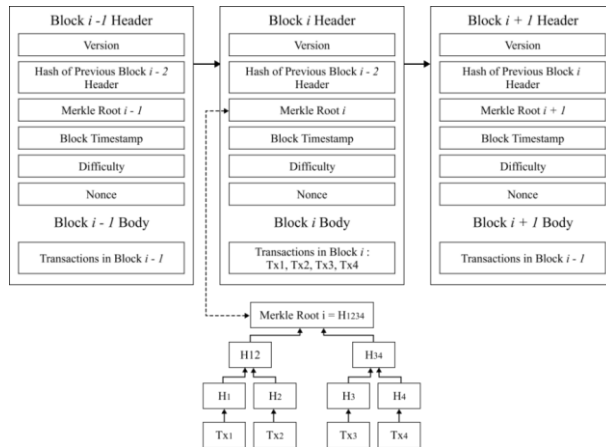
No	Judul	Hasil	Gap	Kontribusi
1	Implementasi IPFS Untuk Mengurangi Gas Fee Smart Contract Ethereum Pada Aplikasi Penggalangan Dana.  (Hutomo Sakti Kartiko, 2023)	Menjelaskan proses bagaimana acara optimalisasi Gas Fee dan memori di blockchain menggunakan Interplanetary File System (IPFS) pada Proof of Work. Hasil dari penelitian ini menunjukkan penggunaan IPFS dapat menurunkan biaya Gas Fee rata-rata sebesar 94.39%, meskipun kecepatan transaksi sedikit meningkat sebesar 13.55%.	Menggunakan consensus Proof of Work, berfokus pada implementasi IPFS untuk efisiensi Gas Fee dan memori. Penelitian ini juga berfokus pada sistem yang spesifik yakni penggalangan dana dengan IPFS.	Memberikan wawasan mengenai consensus Proof of Stake yang boros sumber daya, maka diterapkanlah IPFS untuk efisiensi Gas Fee sebagai penyimpanan terdesentralisasi, serta memberikan hasil efisiensi Gas Fee pada Proof of Work
2	Implementasi Teknologi Blockchain Dengan Sistem Smart Contract Pada Klaim Asuransi.  (Baihaqsani, 2023)	menjelaskan proses implementasi blockchain pada sistem klaim asuransi dengan consensus Proof of Work. Hasil penelitian ini menunjukkan bahwa smart contract blockchain dapat menjamin integritas dan ketetapan data yang berjalan diatas Ethereum	Berfokus pada implementasi blockchain pada klaim asuransi. Penelitian menggunakan consensus Proof of Work, dan mempelajari bagaimana blockchain dapat menjamin integritas data pada klaim asuransi	Memberikan wawasan tentang implementasi blockchain pada klaim asuransi dan memberikan gambaran tentang penggunaan blockchain pada suatu organisasi yang memiliki regulasi.
3	The Privacy Protection Mechanism of Hyperledger Fabric and Its Application In Supply Chain Finance.  (Chaoqun Ma, 2019)	Menjelaskan bahwa blockchain mulai digunakan di bidang keuangan, seperti supply chain yang membutuhkan privasi khusus. Hasil dari penelitian ini menunjukkan bahwa model Hyperledger	fokus pada penerapan model Hyperledger Fabric. Membahas mekanisme keamanan pada platform permissioned (membutuhkan izin)	Memberikan wawasan tentang implementasi blockchain pada supply chain finance dan memberikan penggunaan blockchain pada

		Fabric mampu mengakomodasi kebutuhan keamanan jaringan blockchain pada supply chain.		organisasi yang memiliki regulasi.
4	Blockchain Without Waste: Proof-of-Stake (Saleh,2020)	Penelitian ini menunjukkan bahwa Proof of Stake dapat menciptakan blockchain permissionless yang efektif tanpa mengandalkn konsumsi yang tinggi dengan memberlakukan ambang batas minim bagi validator atau menetapkan jadwal pemberian imbalan blok atas pembaruan blockchain dengan blok-blok baru.	Berfokus mempelajari dan mengembangkan Proof of Stake. Penelitian menjelaskan beberapa desain jaringan blockchain menggunakan algoritma Proof of Stake.	Memberikan wawasan penggunaan teknologi blockchain melalui pendekatan algoritma consensus Proof of Stake. Memberikan wawasan desain jaringan pada algoritma Proof of Stake yang bisa dikembangkan

Tabel 2.1 Penelitian terdahulu

## 2.2 Blockchain

Blockchain adalah teknologi terdesentralisasi yang memungkinkan penyimpanan data secara aman, transparan, dan tidak dapat diubah melalui jaringan yang tersebar atau terdesentralisasi. Dalam blockchain, data disimpan dalam buku besar terdistribusi. Teknologi blockchain menyediakan integritas dan ketersediaan yang memungkinkan pengguna dalam jaringan blockchain untuk menulis, membaca, dan memverifikasi transaksi yang tercatat didalamnya. Namun, teknologi ini tidak memungkinkan penghapusan atau modifikasi terhadap transaksi dan informasi lain yang tersimpan di buku besar tersebut. Sistem blockchain didukung dan diamankan oleh elemen-elemen kriptografi dan enkripsi, seperti tanda tangan digital, fungsi hash, dan sebagainya. Elemen-elemen ini menjamin bahwa transaksi yang dicatat dilindung integritasnya, diverifikasi keasliannya, dan tidak dapat diubah (Huaqun Guo, 2022)



Gambar 2.1 Struktur blockchain

Pada struktur blockchain di pisah menjadi dua bagian yaitu block header dan block body, block header berisi mengenai metadata block yang terdiri dari beberapa bagian yakni:

Bagian Header Block:

- Version: Versi blockchain
- Hash of Previous Block: Hash dari header block sebelumnya
- Merkle Root: Hash dari semua transaksi dalam block
- Timestamp: Waktu pembuatan block
- Difficulty: Tingkat kesulitan untuk menambang di block ini Nonce: Nilai yang diubah selama proses penambangan untuk menemukan hash yang sesuai

Sedangkan pada bagian block body berisi semua data transaksi yang sudah terenkripsi di dalam block tersebut.

### 2.3 Smart Contract

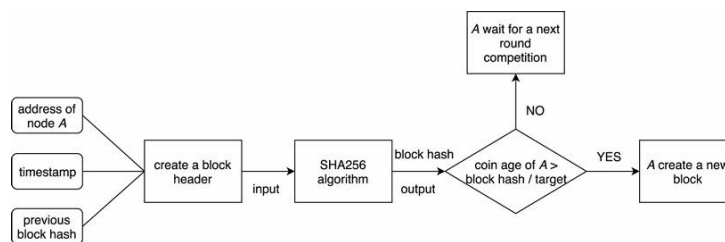
Smart contract adalah program yang berjalan di atas blockchain yang secara otomatis mengeksekusi, mengelola, atau mendokumentasikan peristiwa sesuai dengan syarat dan ketentuan yang telah diprogramkan. Smart contract

memungkinkan pelaksanaan kode yang objektif dalam mengatur proses secara otomatis. Salah satu tujuan utama pengembangan smart contract di Ethereum adalah mengatasi keterbatasan Bitcoin. Smart contract menawarkan otonomi, efisiensi, akurasi, serta penghematan biaya, tanpa memerlukan perantara. Selain itu, smart contract tidak harus melibatkan dua pihak atau bersifat mengikat secara hukum (Huaqun Guo, 2022).

#### 2.4 Konsensus Proof of Stake

Algoritma konsensus adalah algoritma yang berfungsi sebagai mekanisme utama pada jaringan blockchain yang bertugas untuk mencapai kesepakatan pada smart contract di dalam jaringan blockchain. Konsensus berjalan dengan memanfaatkan sumberdaya mayoritas pengguna untuk menjaga integritas sistem. Konsensus memastikan bahwa transaksi yang dicatat di blockchain tidak berbahaya, tidak duplikat dan tidak double spending. Mekanisme konsensus menjadi inti dari seluruh kegiatan di blockchain,

Algoritma konsensus proof of Stake (PoS) dikembangkan untuk mengatasi konsumsi daya yang besar pada Proof of Work (PoW). Dalam Proof of Stake pengguna tidak perlu memecahkan masalah matematika untuk mencapai konsensus (nonce), melainkan cukup menggunakan beberapa cryptocurrency untuk di taruhkan (Staking). Pencipta blok baru dipilih secara acak dari uang pengguna yang telah melakukan staking. Sistem ini mengurangi pemborosan sumber daya PoW dan mendorong pemegang koin untuk meningkatkan waktu penyimpanan, yang meningkatkan keamanan blockchain. (Jannah Yusoff, 2022)



Gambar 2.2 Alur algoritma proof of stake

## 2.5 Kriptografi

Kriptografi adalah seni dan ilmu yang digunakan untuk menyembunyikan pesan, dengan tujuan menjaga kerahasiaan informasi. Pesan yang belum dienkripsi disebut plaintext, sedangkan setelah proses enkripsi, pesan tersebut menjadi ciphertext. Kriptografi mempelajari dan menerapkan teknik komunikasi yang aman dari ancaman pihak ketiga.

Komunikasi yang aman memastikan bahwa pesan atau data yang ditransfer antara dua pihak tidak dapat diakses oleh pihak lawan. Lawan dimaksud entitas yang berusaha mendapatkan informasi berharga dan merusak prinsip-prinsip keamanan. Kerahasiaan data, integritas data, autentikasi, dan non-repudiasi. Kekuatan kriptografi diukur dari waktu dan sumber daya yang diperlukan untuk mengembalikan plaintext. Hasil dari kriptografi yang kuat adalah ciphertext yang sangat sulit dipecahkan tanpa alat dekripsi yang tepat, bahkan dengan semua kekuatan komputasi yang ada saat ini (Archana B U, 2023).

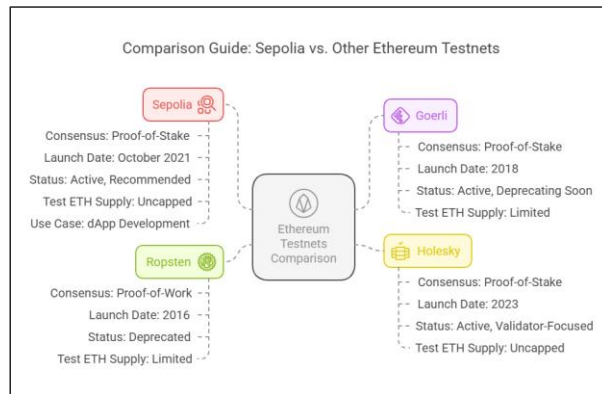
## 2.6 Ethereum

Ethereum adalah cryptocurrency terbesar kedua berdasarkan kapitalisasi pasar. Platform ini merupakan sistem komputasi terdistribusi opensource yang dirancang untuk mendukung smart contract, memungkinkan pengembang membangun aplikasi terdesentralisasi dengan mudah dengan teknologi blockchain. Berbeda dengan Bitcoin yang hanya mencatat transaksi antar alamat, blockchain Ethereum juga menyimpan alamat dengan kode yang dapat dijalankan oleh Ethereum Virtual Machine (EVM) (Dzulfikar & Susanto, 2020).

## 2.7 Ethereum Sepolia Testnet

Ethereum Sepolia Testnet adalah jaringan publik yang dirancang untuk mereplikasi fungsionalitas dari Mainnet Ethereum. Pada jaringan testnet para pengembang dapat menguji smart contract, Dapps atau proyek blockchain dengan menggunakan token testnet (Sepolia ETH), yang digunakan khusus pada jaringan tersebut. Sepolia beralih ke mekanisme konsensus Proof-of-Stake setelah Ethereum melakukan *The Merge* pada tahun 2022 (lumyna.io, 2025). Berikut ini adalah perbandingan antara Sepolia Ethereum testnet dengan testnet lainnya.

**Commented [AM4]:** Penambahan Pustaka tentang testnet sepolia dan alasan menggunakan sepolia eth



Gambar 2.3 Perbedaan arsitektur jaringan testnet Ethereum (lumyna.io. 2025)

## 2.7 Decentralized Apps (Dapps)

Decentralized App atau Dapps adalah program berbasis blockchain yang berjalan pada infrastruktur jaringan peer-to-peer, terdesentralisasi dan tidak terpusat pada suatu pihak manapun. Jaringan peer-to-peer adalah jaringan di mana dua pengguna berinteraksi atau berbagi informasi tanpa intervensi pihak pusat. Dalam Dapps semua data transaksi dilindungi dengan kriptografi. Dapps umumnya menggunakan uang kripto internal sebagai pendorong utama ekosistemnya.. Berikut ini adalah beberapa perbedaan dari centralized app (Apps) dan decentralized app (Dapps) (Raza et al., 2024).

Centralized App	Decentralized App
Apps beroperasi menggunakan sebuah arsitektur <i>client-server</i>	Dapps menggunakan smart contract untuk berinteraksi dengan <i>client</i>
Database pengguna dapat diakses dan terhubung pada backend server	Menggunakan jaringan peer-to-peer pada backend
Membutuhkan arahan dari superadmin (pusat) untuk menetapkan <i>role</i> dan <i>permission</i>	Tidak ada arahan dari pihak superadmin (pusat)
Kerentanan terhadap keamanan dan privasi	Aman, Tidak dapat diubah dan dapat diatur secara mandiri

Tabel 2.2 Perbandingan centralized app dan decentralized app



## 2.8 Thirdweb

Thirdweb merupakan sebuah tools dan utility yang cukup lengkap untuk membantu para developer Web3/Dapps untuk berinteraksi dengan berbagai jaringan blockchain, seperti Ethereum yang langsung dari proyek Javascript. Tools ini menyediakan integrasi fitur blockchain ke aplikasi front-end menjadi lebih mudah untuk digunakan para pengembang (Kolomojets & Kynash, 2023).

## 2.9 Hardhat JS

Hardhat adalah framework berbasis Node.js yang dikembangkan oleh Nomic Foundation yang memiliki banyak fitur, seperti dukungan pengembangan, pengujian menggunakan Javascript dan juga proses deploy smart contract. Hardhat memiliki kelebihan yang sangat fleksibel dan bisa dikonfigurasi sesuai dengan kebutuhan para pengembang, sehingga mereka bisa mengatur berbagai bagian dari siklus pengembangan proyek. (Helms & McGahon, 2023).

## 2.10 Mythril

Mythril adalah salah satu framework paling matang untuk analisis smart contract di blockchain yang dirancang khusus untuk Ethereum Virtual Machine (EVM). Mythril menggunakan teknik symbolic execution, yaitu teknik yang mengevaluasi perilaku program dengan menggunakan input simbolik (bukan nilai tetap) untuk menelusuri semua kemungkinan eksekusi kerentanan dalam smart contract. Mythril lebih unggul dibandingkan dengan analisis statis ringan (lightweight static analysis), Mythril cenderung menghasilkan lebih sedikit alert, tetapi mampu menunjukkan kemungkinan nyata dari serangan yang teridentifikasi (Bonomi et al., 2023).

## BAB III

### METODOLOGI PENELITIAN

#### 3.1 Jenis Penelitian

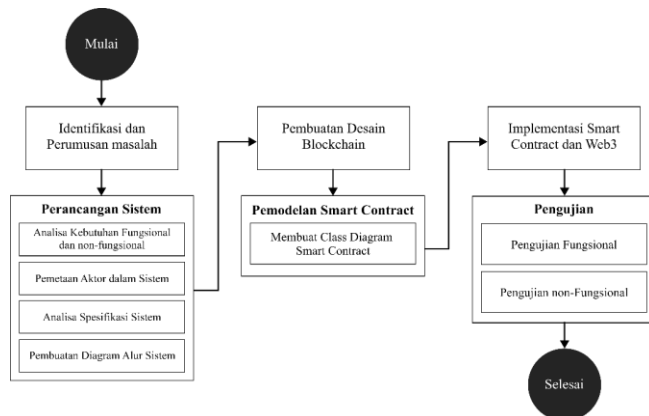
Metode dalam penelitian ini merupakan penelitian pengembangan (RnD). Penelitian pengembangan diawali dengan proses analisis terhadap permasalahan yang diidentifikasi dan diatasi dengan mengembangkan sebuah model atau produk. Produk atau model yang dikembangkan dirancang untuk menjawab permasalahan yang ada (Maruwu, 2024).

#### 3.2 Objek Penelitian

Objek dari penelitian ini adalah smart contract yang nantinya akan di deploy pada jaringan blockchain dan konsensus Proof of Stake. Smart contract yang dihasilkan kemudian akan digunakan untuk menangani proses transaksi pada platform crowdfunding.

#### 3.3 Tahapan Penelitian

Pada bagian ini berisi tahapan yang dilakukan peneliti selama penelitian berlangsung, berikut beberapa tahapan penelitian ditunjukkan pada gambar G.5.



Gambar 3.1 Tahapan penelitian

##### 3.3.1 Identifikasi dan Perumusan Masalah

Identifikasi masalah dilakukan melalui pencarian berita, kejadian di lingkungan sekitar dan di internet untuk mengetahui permasalahan yang terjadi. Berdasarkan beberapa kasus yang ditemukan, seperti dugaan penyelewengan dana oleh Yayasan Rumah Penghafal Qur'an (RFPS) pada tahun 2022, pengelola dana terduga hanya menyalurkan sebagian kecil dana donasi berkisar antara Rp 300.000 (tiga ratus ribu rupiah) hingga Rp 50.000.000 (lima puluh juta rupiah) dari total donasi RP 1,6 miliar (jawapos.com, 2022). Terdapat juga kasus lembaga filantropi Aksi Cepat Tanggap (ACT) yang memotong dana donasi secara tidak transparan untuk kepentingan operasional dan kepentingan pribadi pengelola dana seperti membeli rumah dan perabotnya, serta menerima gaji dan fasilitas lainnya dengan nilai ratusan juta rupiah (tempo.co, 2022).

Hal tersebut menimbulkan krisis kepercayaan pada pengelola dana pihak ketiga, sehingga para donator kesulitan untuk memilih proyek yang sah (Gada, 2021). Sistem penggalangan dana yang tersentralisasi menyebabkan ketergantungan pada pihak ketiga untuk evaluasi dan pengelolaan risiko, sehingga mengurangi desentralisasi dan meningkatkan kemungkinan kesalahan atau kolusi dalam proyek. Ketergantungan pihak ke tiga juga menyebabkan biaya layanan yang tinggi (Xu, 2023).

### **3.3.2 Perancangan Sistem**

Pada tahap ini akan dilakukan proses perancangan sistem berbasis blockchain Ethereum. Perancangan sistem dimulai dengan melakukan analisa kebutuhan fungsional dan kebutuhan non-fungsional, analisa pemetaan aktor dalam sistem, analisa spesifikasi sistem dan pembuatan alur sistem. Analisa kebutuhan fungsional dan non-fungsional meliputi identifikasi fitur utama yang harus dimiliki oleh sistem agar dapat beroperasi sesuai dengan tujuan yang telah di tentukan dengan melakukan observasi pada platform penggalangan online sejenis yang sudah ada.

Kebutuhan fungsional mencakup proses pengiriman Ether dari pengirim ke penerima melalui smart contract, pencatatan transaksi crowdfunding dan proses staking pada blockchain, serta mekanisme pembagian dan penarikan reward.

Sementara itu, kebutuhan non-fungsional mencakup aspek teknis seperti terdesentralisasi sistem, transparansi transaksi, dan interaksi smart contract dengan user dan jaringan blockchain. Berdasarkan observasi didapatkan kebutuhan fungsional dan non-fungsional sistem dengan penyesuaian sebagai berikut:

<b>Kebutuhan fungsional</b>	<b>Kebutuhan non-fungsional</b>
User dapat mengirim ether ke smart contract	Menggunakan smart contract untuk berinteraksi dengan blockchain
Smart contract mencatat transaksi donasi pada blockchain	Sistem dibuat terdesentralisasi atau tanpa perantara pihak ketiga
User dapat mengambil staking token untuk melakukan staking	Smart contract mencatat transaksi crowdfunding dan staking pada blockchain
User dapat melakukan penarikan reward setelah melakukan staking	Transaksi pada blockchain dapat di lacak transparasinya

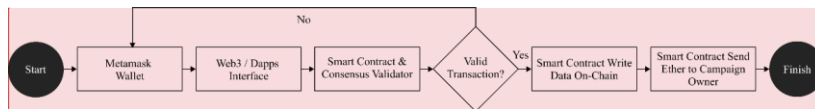
Tabel 3.1 Kebutuhan fungsional dan non-fungsional sistem

Analisa pemetaan aktor dalam sistem mencakup identifikasi aktor yang terlibat pada penggunaan sistem crowdfunding, meliputi campaign owner, donator/backer/funder, smart contract blockchain dan web3 interface. Analisa spesifikasi sistem meliputi teknologi dan tools apa saja yang nantinya akan digunakan untuk membangun aplikasi terdesentralisasi, meliputi bahasa pemrograman yang digunakan untuk smart contract dan frontend, jenis blockchain dan konsensus yang digunakan, library yang digunakan untuk integrasi, dan platform Ethereum Virtual Machine (EVM). Berikut ini adalah hasil analisa spesifikasi sistem dan tools yang digunakan.

<b>Komponen</b>	<b>Tools dan Teknologi</b>
Blockchain	Ethereum (Sepolia Ethereum Testnet)
Konsensus	Ethereum (Proof of Stake)
Bahasa Smart Contract	Solidity
Provider EVM	Remix, Thirdweb
Library Web3	Ethers JS, Thirdweb Provider
Frontend	React JS
Backend	Ethereum (Sepolia Ethereum Testnet)

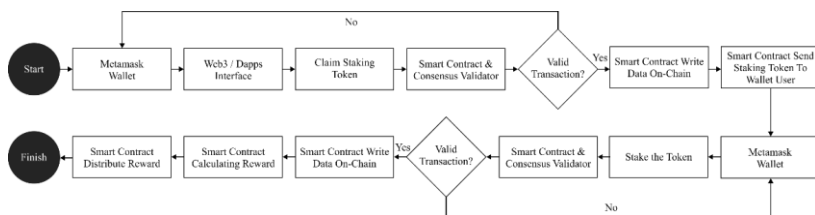
Tabel 3.2 Tools dan teknologi yang digunakan

Pada sistem yang akan dibuat, pemilik kampanye akan membuka kampanye pada aplikasi yang nantinya bisa menerima donasi dari donator. Para donator akan mengirim Ether dari wallet kripto mereka lewat interface aplikasi, Ether yang didonasikan tidak langsung dikirim ke wallet pemilik kampanye, namun akan di terima oleh smart contract untuk divalidasi dan dicatat pada blockchain. Ether yang didonasikan secara otomatis akan dikirim oleh smart contract ke pemilik kampanye jika sudah divalidasi. User bisa melakukan klaim staking token untuk menaruh token mereka dan nantinya akan mendapat reward. Berikut ini adalah alur transaksi pada proses crowdfunding dan alur pada proses staking.



Gambar 3.2 Alur sistem transaksi crowdfunding

**Commented [AM5]:** Penambahan alur transaksi crowdfunding dan staking

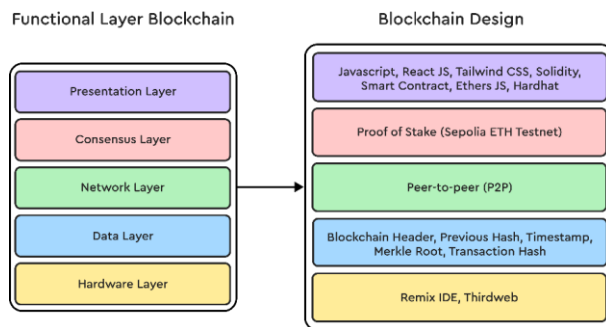


Gambar 3.3 Alur sistem staking

### 3.3.3 Pembuatan Desain Blockchain

Pembuatan desain blockchain dilakukan berdasarkan analisa spesifikasi sistem dan jaringan blockchain yang dipilih, dengan mempertimbangkan setiap layer dalam arsitektur blockchain, konsensus dan infrastruktur jaringan. Setiap layer memiliki peran masing-masing, seperti Presentation Layer yang bertanggung jawab atas tampilan antarmuka, serta interaksi dengan smart contract melalui library Ethers JS. Consensus Layer mengadopsi Proof of Stake (PoS) pada jaringan blockchain Sepolia ETH untuk memvalidasi transaksi, sementara Network Layer menggunakan mode Peer-to-Peer (P2P) untuk komunikasi antar node dalam jaringan, dan Data Layer yang

memastikan keamanan dan validitas data transaksi yang disimpan dengan menerapkan Transaction Hash, Hash Algorithm dan pencatatan data transaksi. Berikut ini adalah desain blockchain yang akan digunakan berdasarkan layer fungsional pada blockchain.



Gambar 3. 4 Desain arsitektur layer blockchain

### 3.3.4 Pemodelan Smart Contract

Pada tahap ini dilakukan pemodelan smart contract yang akan digunakan nantinya. Pemodelan dilakukan dengan menentukan kontrak apa saja yang akan digunakan pada aplikasi crowdfunding. Berdasarkan alur sistem yang sudah dibuat didapatkan smart contract yang akan digunakan sebagai berikut.

Kontrak	Keterangan
Crowdfunding Contract	Kontrak yang digunakan untuk menangani proses penggalangan dana dan penyaluran dana dari donator ke pemilik kampanye
Token Drop Contract	Kontrak yang digunakan untuk proses klaim staking token yang nantinya digunakan pada proses staking
Staking Token dan Reward Token	Kontrak yang digunakan untuk membuat token staking dan token reward
Staking Contract	Kontrak yang digunakan untuk menangani proses staking, perhitungan reward dan distribusi reward

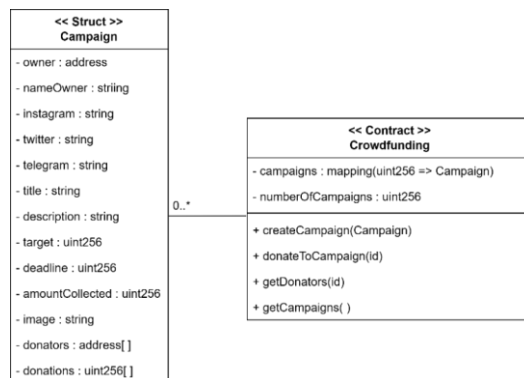
Tabel 3.3 Nama dan keterangan kontrak

Untuk mempermudah proses implementasi, pemodelan smart contract juga akan menggunakan Class Diagram untuk menetapkan atribut dan method dari setiap kontrak. Pada tahap ini juga dilakukan pembuatan proses staking. Staking

adalah bagian yang paling populer dari konsensus Proof of Stake (PoS), dimana para user bisa menaruh token mereka dalam periode waktu tertentu dan akan mendapat reward berdasarkan token yang mereka staking. Pemodelan spesifikasi smart contract dan class diagram adalah sebagai berikut:

1. Kontrak Crowdfunding
  - a. Membuat dan menyimpan data kampanye
  - b. Mengambil data kampanye
  - c. Mengambil data donator dan ether
  - d. Mengirim ether dari donator ke pemilik kampanye

Detail dari struktur data kontrak crowdfunding terdapat pada gambar 3.5 dan keterangan terdapat pada tabel 3.4 dan tabel 3.5.



Gambar 3.5 Class diagram kontrak crowdfunding

Nama	Struct Campaign
Dekripsi	Mendefinisikan struktur dari data campaign
Abtribut	Owner, nameOwner, instagram, twitter, telegram, title, description, target, deadline, amountCollected, image, donators, donations

Tabel 3.4 Keterangan struktur campaign

Nama	Crowdfunding Contract
Dekripsi	Kontrak yang mengatur transaksi pada crowdfunding
Atribut	campaigns, numberOfCampaign
Method	createCampaign(Campaign) Method untuk membuat campaign dengan parameter struct campaign

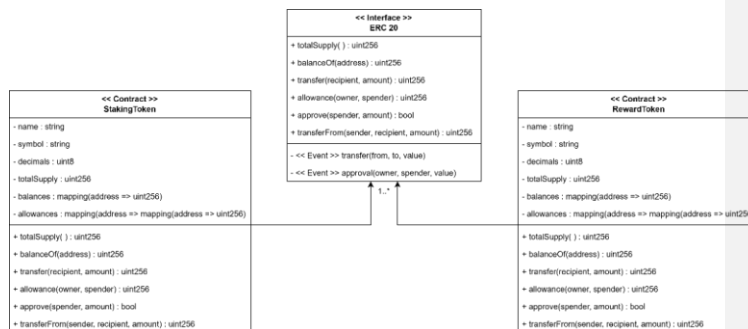
	donateToCampaign(id)	Method untuk mengirim ether ke campaign tertentu
	getDonators(id)	Method untuk mengambil data donator dan jumlah ether yang mereka donasikan
	getCampaigns()	Method untuk mengambil data semua campaign untuk di tampilkan

Tabel 3.5 Keterangan kontrak campaign

## 2. Kontrak Staking Token dan Reward Token

- Melakukan persetujuan alamat
- Mengirim ether ke alamat yang disetujui

Detail dari struktur kontrak staking token dan reward token terletak pada gambar 3.6 dan keterangan terdapat pada tabel 3.6.



Gambar 3.6 Class diagram staking token dan reward token

Nama	Staking Token dan Reward Token Contract	
Dekripsi	Kontrak yang digunakan untuk membuat token staking dan reward yang sesuai dengan standard interface dari ERC20	
Atribut	name, symbol, decimals, totalSupply, balances, allowances	
Method	totalSupply()	Method untuk inisiasi total jumlah suplai token
	balanceOf(address)	Method untuk melihat saldo dari alamat wallet tertentu
	transfer(recipient, amount)	Method untuk melakukan transfer ke alamat wallet dengan jumlah tertentu
	allowance(owner, spender)	Method untuk melakukan persetujuan pihak ketiga menggunakan token pengguna
	approve(spender, amount)	Method untuk melakukan persetujuan ke alamat pengirim dengan jumlah token tertentu



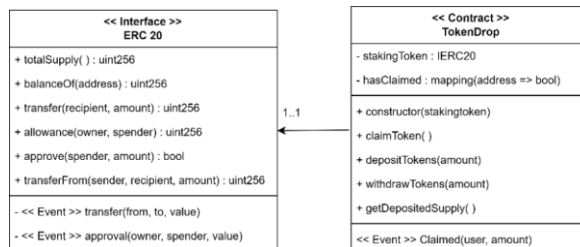
	transferFrom(sender, recipient, amount)	Method untuk memindahkan token dari alamat pengirim ke penerima dengan jumlah tertentu
--	---	--

Tabel 3. 6 Keterangan kontrak staking token dan reward token

## 3. Kontrak Token Drop

- a. Menerima deposit token
- b. Mengirim token ke alamat tertentu

Detail dari struktur token drop terletak pada gambar 3.7 dan keterangan terletak pada tabel 3.7.



Gambar 3.7 Class diagram kontrak token drop

Nama	Token Drop Contract	
Dekripsi	Kontrak yang mengatur proses klaim token staking	
Atribut	stakingToken, hasClaimed	
Method	constructor(stakingToken)	Method untuk mengonosoaso alamat dari staking token Ketika kontrak pertama kali dideploy
	claimToken()	Method untuk melakukan claim
	depositToken(amount)	Method untuk deposit jumlah staking token tertentu oleh smart contract deployer
	withdrawTokens(amount)	Method untuk mengambil kembali token dengan jumlah tertentu oleh smart contract deployer
	getDepositedSupply()	Method untuk mengambil data jumlah token yang sudah masuk pada kontrak token drop

Tabel 3.7 Keterangan kontrak token drop

## 4. Kontrak Staking

- a. Menerima staking token
- b. Melakukan kalkulaais reward token

c. Mengirim reward token

Detail dari struktur kontrak staking terdapat pada gambar 3.8 dan keterangan terdapat pada tabel 3.8.



Gambar 3.8 Class diagram kontrak staking

Nama	Staking Contract	
Dekripsi	Kontrak yang mengatur proses staking, kalkulasi dan pembagian reward staking	
Atribut	rewardToken, stakingToken, rewardTokenBalance, stakingTokenBalance, timeUnit, rewardNumerator, rewardDenominator, stakeBalance, stakeTimestamp	
Method	constructor()	Method yang pertama kali dijalankan Ketika deploy smart contract
	stake(uint256)	Method untuk melakukan proses staking
	unstake(uint256)	Method untuk membatalkan proses staking
	claimRewards()	Method untuk klaim reward setelah melakukan staking
	depositRewardToken(uint256)	Method untuk deposit reward token oleh smart contract deployer
	withdrawRewardToken(uint256)	Method untuk mengambil kembali token reward yang telah di deposit
	calculateReward(address)	Method untuk menghitung jumlah reward
	getstakeInfo(address)	Method untuk melihat info staking

Tabel 3.8 Keterangan kontrak staking

### 3.3.5 Implementasi Smart Contract dan Web3

Pada tahap ini dilakukan implementasi desain dan kerangka yang sudah dibuat pada tahap sebelumnya, dan diimplementasikan dalam bentuk sistem dan kode. Implementasi terdapat dua fase yaitu implementasi smart contract dan implementasi Web3 atau Dapps.

#### a. Implementasi Smart Contract

Implementasi smart contract dilakukan berdasarkan model kontrak yang sudah dilakukan pada tahap sebelumnya, peneliti akan mengimplementasikan semua attribute dan method yang ada pada class diagram yang sudah dibuat. Implementasi smart contract dilakukan menggunakan Remix IDE untuk memudahkan proses debugging secara langsung pada Ethereum Virtual Machine (EVM). Setelah proses debugging selesai, smart contract akan di deploy ke jaringan testnet Sepolia ETH menggunakan platform Thirdweb untuk selanjutnya akan diintegrasikan dengan aplikasi.

#### b. Implementasi Web3 atau Dapps

Implementasi Web3 atau Dapps dilakukan setelah proses deploy smart contract, pada fase ini akan dimulai dengan membuat user interface menggunakan bahasa pembangun website dan akan diintegrasikan menggunakan library dari Thirdweb dan Ethers JS. Implementasi melibatkan extension dari browser yaitu metamask wallet untuk bisa berinteraksi dengan aplikasi berbasis blockchain.

### 3.3.6 Pengujian

Pada tahap ini akan dilakukan pengujian smart contract, pengujian bertujuan untuk mengetahui kelebihan dan kelemahan dari smart contract dan sistem yang telah dibuat. Pengujian akan dibagi menjadi dua fase yaitu uji fungsionalitas dan non-fungsionalitas.

#### a. Uji fungsionalitas

Uji Fungsionalitas dilakukan pada smart contract menggunakan Hardhat JS, Hardhat JS digunakan untuk menguji apakah smart contract yang dirancang bisa

berjalan sesuai dengan kebutuhan atau tidak. Pengujian ini juga dilakukan pada aplikasi untuk menguji integrasi antara smart contract dengan user interface.

b. Uji non-fungsionalitas

Uji fungsionalitas dilakukan dengan memeriksa potensi celah keamanan pada smart contract dengan menggunakan Mythril, penggunaan tools untuk mendeteksi kerentanan pada smart contract meningkatkan akurasi dan efisiensi ketika testing (Khan & Namin, 2024). serta melakukan uji smart contract secara On-chain pada Sepolia explorer untuk dilihat data yang disimpan serta melakukan dekripsi untuk menguji transparansi data yang disimpan pada blockchain.

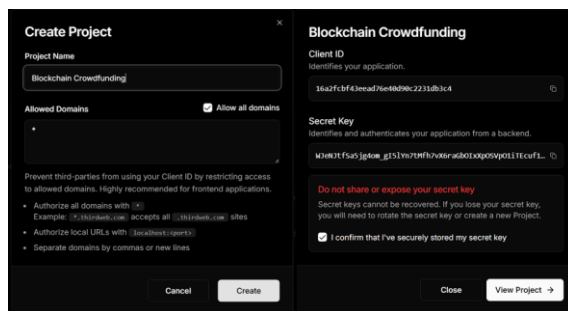
## BAB IV

### HASIL DAN PEMBAHASAN

#### 4.1 Implementasi Smart Contract dan Dapps

Berdasarkan model smart contract yang telah dibuat maka didapatkan hasil beberapa kontrak meliputi kontrak crowdfunding, kontrak staking dan reward token, kontrak token drop dan kontrak staking lengkap dengan method dan atribut masing-masing kontrak. Hasil dari implementasi masing-masing kontrak terdapat pada lampiran. Setelah masing-masing kontrak telah diimplementasi, tahap selanjutnya adalah melakukan deploy pada EVM menggunakan Thirdweb, Thirdweb digunakan sebagai EVM untuk membantu integrasi anantara smart contract dengan aplikasi. Untuk mengintegrasikan sistem maka dilakukan langkah-langkah berikut:

1. Membuat proyek di Thirdweb



Gambar 4.1 Membuat proyek pada EVM

2. Deploy smart contract



Gambar 4.2 Deploy smart contract ke EVM

Ketika proses deploy berhasil, sistem akan menampilkan tautan url pada terminal. Tautan tersebut digunakan untuk melakukan konfigurasi masing-masing kontrak untuk memilih chain atau jaringan blockchain dimana kontrak akan di deploy.

#### 4.2 Implementasi Web3

Kontrak yang sudah dideploy pada jaringan Sepolia Ethereum lewat Ethereum Virtual Machine menggunakan thirdweb akan dilakukan integrasi dengan aplikasi. Aplikasi dikembangkan menggunakan bahasa Javascript dengan menggunakan library React JS. Interface aplikasi terdiri dari beberapa halaman seperti pada lampiran 1 sampai lampiran 6.

#### 4.3 Uji Fungsionalitas

Setelah dilakukan implementasi smart contract akan di uji dari segi fungsionalitasnya, pengujian bermaksud untuk memastikan bahwa semua fungsi yang dituliskan pada smart contract berjalan dengan baik. Pengujian dilakukan pada masing-masing kontrak menggunakan Hardhat JS. Test case untuk masing-masing smart contract untuk uji fungsional terdapat pada tabel 4.1 dibawah ini.

Kontrak	No	Test Case
Crowdfunding Contract	1	Kontrak dapat membuat kampanye
	2	Kontrak dapat melarang kampanye dengan deadline masalalu
	3	Kontrak dapat melakukan donasi ke kampanye aktif
	4	Kontrak dapat melarang donasi setelah lewat deadline
	5	Kontrak dapat menyimpan data donator dan jumlah donasi
	6	Kontrak dapat melihat seluruh kampanye
Staking Token dan Reward Token	1	Kontrak dapat mendeploy token dengan suplai tertentu
	2	Kontrak dapat mentransfer token antar akun
	3	Kontrak dapat mencatat transfer yang berhasil
	4	Kontrak dapat membatalkan transfer jika tidak cukup token
	5	Kontrak dapat menyetujui akun lain untuk transfer
	6	Kontrak dapat mentransfer token yang sudah di setujui

	7	Kontrak dapat membatalkan transfer akun yang disetujui jika token kurang
	8	Kontrak dapat mencatat persetujuan yang berhasil
Token Drop Contract	1	Kontrak dapat menetapkan alamat reward token
	2	Kontrak dapat mengizinkan user klaim token
	3	Kontrak hanya mengizinkan owner smart contract saja untuk deposit token
	4	Kontrak hanya mengizinkan owner smart contract saja untuk menarik token
	5	Kontrak dapat memperbarui suplai token setelah deposit dan penarikan token
Staking Contract	1	Kontrak dapat menetapkan alamat owner, alamat token staking dan alamat token reward
	2	Kontrak dapat mengizinkan user untuk staking, memastikan jumlah token staking adalah satu, dan melakukan update timestamp
	3	Kontrak dapat mengizinkan user untuk staking, memastikan jumlah token yang di-unstaking tidak lebih dari satu, dan melarang user untuk unstaking jika tidak melakukan staking
	4	Kontrak dapat mengkalkulasi jumlah reward, mengizinkan hanya owner yang bisa menarik token reward, dan melarang user menarik reward lebih dari jumlah reward yang di kalkulasi
	5	Kontrak dapat menampilkan data staking

Tabel 4.1 Daftar test case untuk masing-masing kontrak

Hasil pengujian fungsional smart kontrak crowdfunding menggunakan Hardhat JS dapat dilihat pada gambar 4.3

```

RYZEN5@DESKTOP-8PA208E D:\...\test main npx hardhat test .\01_Crowdfunding.test.js
Compiled 2 Solidity files successfully (evm target: paris).

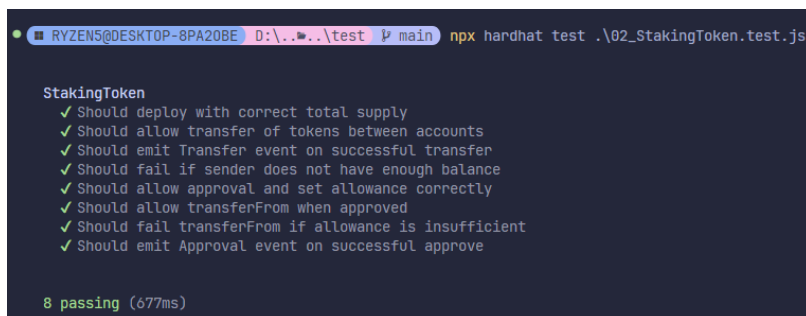
Crowdfunding
  ✓ Should create a campaign successfully (57ms)
  ✓ Should not allow create campaign with past deadline (45ms)
  ✓ Should allow donations to an active campaign
  ✓ Should not allow donations after campaign deadline
  ✓ Should store donator address and donation amount
  ✓ Should retrieve all campaigns

6 passing (3s)

```

Gambar 4.3 Hasil pengujian kontrak crowdfunding dengan Hardhat

Dari proses testing fungsional menggunakan Hardhat JS, didapatkan hasil dari testing smart contract crowdfunding yang berhasil memenuhi test case. Selanjutnya, hasil penguian fungsional kontrak staking token dan reward token terdapat pada gambar 4.4 dan gambar 4.5.



```

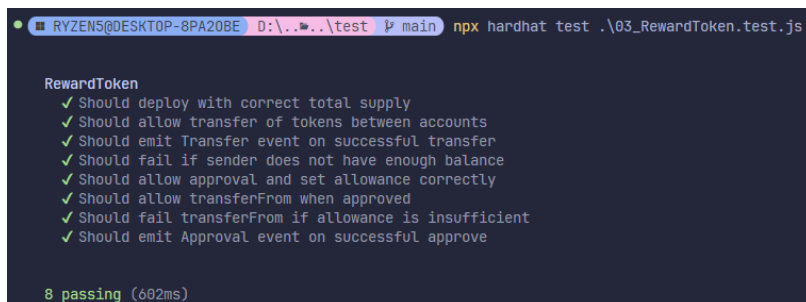
RYZEN5@DESKTOP-8PA20BE D:\...\test main npx hardhat test .\02_StakingToken.test.js

StakingToken
  ✓ Should deploy with correct total supply
  ✓ Should allow transfer of tokens between accounts
  ✓ Should emit Transfer event on successful transfer
  ✓ Should fail if sender does not have enough balance
  ✓ Should allow approval and set allowance correctly
  ✓ Should allow transferFrom when approved
  ✓ Should fail transferFrom if allowance is insufficient
  ✓ Should emit Approval event on successful approve

8 passing (677ms)

```

Gambar 4.4 Hasil pengujian kontrak crowdfunding dengan Hardhat



```

RYZEN5@DESKTOP-8PA20BE D:\...\test main npx hardhat test .\03_RewardToken.test.js

RewardToken
  ✓ Should deploy with correct total supply
  ✓ Should allow transfer of tokens between accounts
  ✓ Should emit Transfer event on successful transfer
  ✓ Should fail if sender does not have enough balance
  ✓ Should allow approval and set allowance correctly
  ✓ Should allow transferFrom when approved
  ✓ Should fail transferFrom if allowance is insufficient
  ✓ Should emit Approval event on successful approve

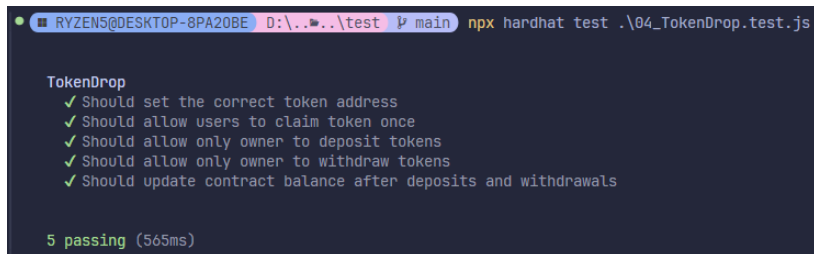
8 passing (682ms)

```

Gambar 4.5 Hasil pengujian kontrak staking dan reward token dengan Hardhat

Dari proses testing fungsional menggunakan Hardhat JS, didapatkan hasil dari testing smart contract staking dan reward token yang berhasil dan memenuhi test case. Selanjutnya, hasil penguian fungsional kontrak token drop terdapat pada gambar 4.6.





```

RYZEN5@DESKTOP-8PA20BE D:\...\test main npx hardhat test .\04_TokenDrop.test.js

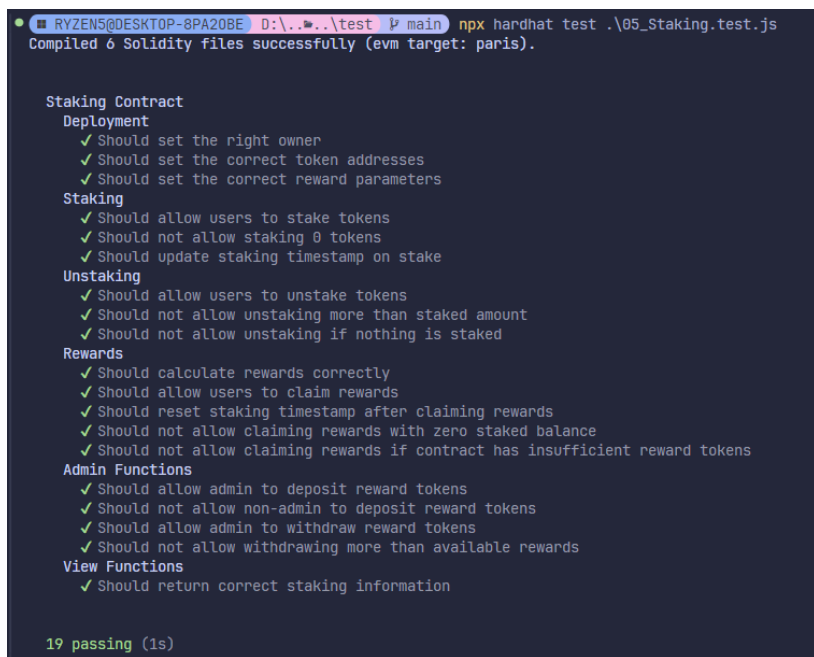
TokenDrop
  ✓ Should set the correct token address
  ✓ Should allow users to claim token once
  ✓ Should allow only owner to deposit tokens
  ✓ Should allow only owner to withdraw tokens
  ✓ Should update contract balance after deposits and withdrawals

5 passing (565ms)

```

Gambar 4.6 Hasil pengujian kontrak token drop dengan Hardhat

Dari proses testing fungsional menggunakan Hardhat JS, didapatkan hasil dari testing smart contract token drop yang berhasil dan memenuhi test case. Selanjutnya, hasil pengujian fungsional kontrak staking terdapat pada gambar 4.7.



```

RYZEN5@DESKTOP-8PA20BE D:\...\test main npx hardhat test .\05_Staking.test.js
Compiled 6 Solidity files successfully (evm target: paris).

Staking Contract
  Deployment
    ✓ Should set the right owner
    ✓ Should set the correct token addresses
    ✓ Should set the correct reward parameters
  Staking
    ✓ Should allow users to stake tokens
    ✓ Should not allow staking 0 tokens
    ✓ Should update staking timestamp on stake
  Unstaking
    ✓ Should allow users to unstake tokens
    ✓ Should not allow unstaking more than staked amount
    ✓ Should not allow unstaking if nothing is staked
  Rewards
    ✓ Should calculate rewards correctly
    ✓ Should allow users to claim rewards
    ✓ Should reset staking timestamp after claiming rewards
    ✓ Should not allow claiming rewards with zero staked balance
    ✓ Should not allow claiming rewards if contract has insufficient reward tokens
  Admin Functions
    ✓ Should allow admin to deposit reward tokens
    ✓ Should not allow non-admin to deposit reward tokens
    ✓ Should allow admin to withdraw reward tokens
    ✓ Should not allow withdrawing more than available rewards
  View Functions
    ✓ Should return correct staking information

19 passing (1s)

```

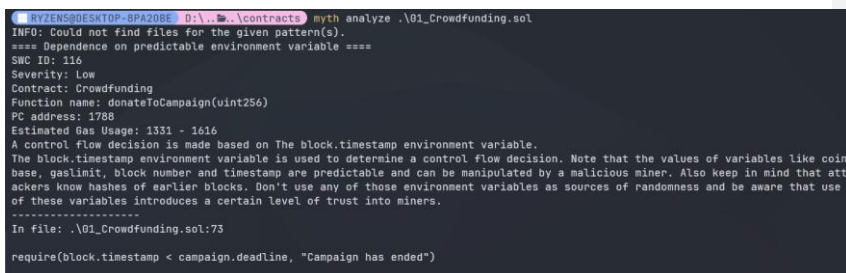
Gambar 4.7 Hasil pengujian kontrak staking dengan Hardhat

Dari proses testing fungsional menggunakan Hardhat JS, didapatkan hasil dari pengujian smart contract staking yang berhasil dan memenuhi test case. Dari

masing-masing kontrak yang sudah dilakukan testing secara fungsional mendapatkan hasil bahwa smart contract yang dibangun dapat beroperasi sesuai dengan fungsi yang sudah dituliskan dalam smart contract terkait. Hal ini dapat dilihat dari hasil testing menggunakan Hardhat dan hasil yang ditampilkan Hardhat JS memenuhi semua test case yang ada.

#### 4.4 Uji Non-fungsionalitas

Pengujian non-fungsionalitas dilakukan untuk mendeteksi potensi celah keamanan smart contract, memastikan integritas data dan availability data. Mythril digunakan sebagai tools untuk mendeteksi potensi celah keamanan. Sedangkan sepoliaetherscan digunakan untuk memastikan integritas dan availability data yang disimpan pada jaringan blockchain. Hasil pengujian menggunakan Mythril pada kontrak crowdfunding terdapat pada gambar 4.8 dibawah ini.



```

myth analyze .\01_Crowdfunding.sol
INFO: Could not find files for the given pattern(s).
==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Crowdfunding
Function name: donateToCampaign(uint256)
PC address: 1788
Estimated Gas Usage: 1331 - 1616
A control flow decision is made based on The block.timestamp environment variable.
The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coin
base, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that att
ackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use
of these variables introduces a certain level of trust into miners.
-----
In file: .\01_Crowdfunding.sol:73
require(block.timestamp < campaign.deadline, "Campaign has ended")

```

Gambar 4.8 Hasil pengujian kontrak crowdfunding dengan Mythril

Pada hasil testing kontrak Crowdfunding ditemukan tingkat keparahan kode yang rendah, masalah ditemukan pada fungsi `donateToCampaign(uint256)`. Berdasarkan pada deskripsi program, fungsi tersebut menggunakan `block.timestamp` untuk menentukan apakah kampanye masih terbuka atau sudah tutup. Nilai `block.timestamp` adalah nilai yang dapat diprediksi dan dapat dimanipulasi oleh miner dengan memajukan timestap beberapa detik. Hal ini akan memperngaruhi logika kontrak seperti membuat donasi masih bisa dilakukan meskipun sudah melewati batas waktu.

Untuk menangani kerentanan pada kode, dilakukan perbaikan kode pada smart contract pada beberapa bagian dengan menambahkan validasi pada bagian

fungsi `donateToCampaign`. Perbaikan pada smart contract terletak pada kode 4.1 dibawah ini.

```

...
event CampaignStatusChanged(uint256 indexed campaignId, bool isActive);

function donateToCampaign(uint256 _id) public payable {
    uint256 amount = msg.value;
    Campaign storage campaign = campaigns[_id];

    require(campaign.isActive, "Campaign is not active");

    if (block.timestamp >= campaign.deadline) {
        campaign.isActive = false;
        emit CampaignStatusChanged(_id, false);
        revert("Campaign has ended");
    }

    require(amount > 0, "Donation must be greater than zero");

    campaign.donators.push(msg.sender);
    campaign.donations.push(amount);

    campaign.amountCollected += amount;

    emit DonationReceived(_id, msg.sender, amount);

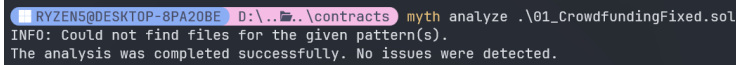
    (bool sent,) = payable(campaign.owner).call{value: amount}("");
    require(sent, "Failed to send donation to campaign owner");
}
...

```

Kode 4.1 Perbaikan smart contract crowdfunding

Pada fungsi `donateToCampaign` ditambahkan validasi yang lebih aman untuk memvalidasi deadline dari kampanye. Validasi tidak langsung menggunakan block timestamp, namun menggunakan variable `isActive` dan menambahkan event `CampaignStatusChanged` untuk mencatat perubahan pada nilai variabel `isActive`.

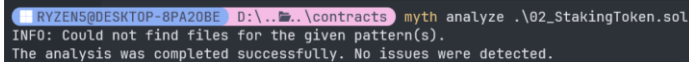
Setelah ditambahkan kode untuk validasi pada fungsi `donateToCampaign`, selanjutnya akan dilakukan Analisa ulang menggunakan Mythril. Testing dilakukan untuk memastikan apakah kode smart contract yang sudah di perbaiki masih memiliki celah keamanan atau tidak. Hasil pengujian ulang smart contract crowdfunding menggunakan Mythril terdapat pada gambar 4.10 dibawah ini.



```
RYZEN5@DESKTOP-8PA20BE D:\..\..\contracts myth analyze .\01_CrowdfundingFixed.sol
INFO: Could not find files for the given pattern(s).
The analysis was completed successfully. No issues were detected.
```

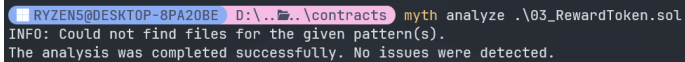
Gambar 4.9 Hasil pengujian kembali smart contract crowdfunding

Pengujian ulang kontrak crowdfunding sudah tidak ditemukan kerentanan dan dilanjutkan dengan pengujian kontrak lainnya. Hasil pengujian menggunakan Mythril pada kontrak staking token dan reward token terdapat pada gambar 4.11 dan gambar 4.12.



```
RYZEN5@DESKTOP-8PA20BE D:\..\..\contracts myth analyze .\02_StakingToken.sol
INFO: Could not find files for the given pattern(s).
The analysis was completed successfully. No issues were detected.
```

Gambar 4.10 Hasil pengujian kontrak staking token dengan Mythril



```
RYZEN5@DESKTOP-8PA20BE D:\..\..\contracts myth analyze .\03_RewardToken.sol
INFO: Could not find files for the given pattern(s).
The analysis was completed successfully. No issues were detected.
```

Gambar 4.11 Hasil pengujian kontrak reward token dengan Mythril

Pada hasil testing kontrak Staking Token dan Reward Token tidak ditemukan keparahan atau kerentanan kode pada proses analisa menggunakan Mythril. Kemudian pada hasil pengujian menggunakan Mythril pada kontrak Token Drop terdapat pada gambar 4.13.

```

RYZENS@DESKTOP-8PA20BE D:\...\.contracts myth analyze .\04_TokenDrop.sol
INFO: Could not find files for the given pattern(s).
==== External Call To User-Supplied Address ====
SWC ID: 107
Severity: Low
Contract: TokenDrop
Function name: withdrawTokens(uint256)
PC address: 524
Estimated Gas Usage: 4352 - 39859
A call to a user-supplied address is executed.
An external message call to an address specified by the caller is executed. Note that the callee account
might contain arbitrary code and could re-enter any function within this contract. Reentering the contr
act in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are
executed after this call and/or reentrancy guards are in place.
-----
In file: .\04_TokenDrop.sol:146

stakingToken.transfer(msg.sender, _amount)

==== External Call To User-Supplied Address ====
SWC ID: 107
Severity: Low
Contract: TokenDrop
Function name: depositTokens(uint256)
PC address: 1523
Estimated Gas Usage: 4568 - 40170
A call to a user-supplied address is executed.
An external message call to an address specified by the caller is executed. Note that the callee account
might contain arbitrary code and could re-enter any function within this contract. Reentering the contr
act in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are
executed after this call and/or reentrancy guards are in place.
-----
In file: .\04_TokenDrop.sol:141

stakingToken.transferFrom(msg.sender, address(this), _amount)

==== External Call To User-Supplied Address ====
SWC ID: 107
Severity: Low
Contract: TokenDrop
Function name: claimToken()
PC address: 940
Estimated Gas Usage: 11619 - 68373
A call to a user-supplied address is executed.
An external message call to an address specified by the caller is executed. Note that the callee account
might contain arbitrary code and could re-enter any function within this contract. Reentering the contr
act in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are
executed after this call and/or reentrancy guards are in place.
-----
In file: .\04_TokenDrop.sol:134

stakingToken.transfer(msg.sender, 1 * (10 ** 18))

```

Gambar 4.12 Hasil pengujian kontrak token drop dengan Mythril

Hasil pengujian smart contract Token Drop menampilkan beberapa kerentanan atau keparahan kode. Kerentanan terdapat pada tiga fungsi yakni fungsi `withdrawToken`, fungsi `claimToken` dan fungsi `depositTokens`. Pada ketiga fungsi terdapat ancaman yang sama, berdasarkan pada deskripsi program, ketiga fungsi tersebut rentan terhadap potensi celah keamanan yakni *Reentrancy Attack*. *Reentrancy Attack* adalah sebuah mekanisme (cari jurnal aja lek ku). Untuk mengatasi kerentanan ini, smart contract perlu diperbaiki pada ketiga fungsi tersebut. Hasil perbaikan dari kontrak token drop terdapat pada kode 4.2.

```

...
modifier nonReentrant() {
    require(!_locked, "ReentrancyGuard: reentrant call");
    _locked = true;
    _;
    _locked = false;
}

function claimToken() external nonReentrant {
    require(!hasClaimed[msg.sender], "You have already claimed");
    hasClaimed[msg.sender] = true;
    uint256 claimAmount = 1 * (10 ** 18);

    require(stakingToken.balanceOf(address(this)) >= claimAmount, "Insufficient
balance");

    emit Claimed(msg.sender, claimAmount);
    bool success = stakingToken.transfer(msg.sender, claimAmount);
    require(success, "Token transfer failed");
}

function depositTokens(uint256 _amount) external onlyOwner nonReentrant {
    require(_amount > 0, "Amount must be greater than zero");

    uint256 balanceBefore = stakingToken.balanceOf(address(this));

    bool success = stakingToken.transferFrom(msg.sender, address(this), _amount);
    require(success, "Token transfer failed");

    uint256 balanceAfter = stakingToken.balanceOf(address(this));
    require(balanceAfter >= balanceBefore + _amount, "Token deposit verification
failed");

    emit TokensDeposited(msg.sender, _amount);
}

function withdrawTokens(uint256 _amount) external onlyOwner nonReentrant {
    require(_amount > 0, "Amount must be greater than zero");
    require(stakingToken.balanceOf(address(this)) >= _amount, "Insufficient balance");

    emit TokensWithdrawn(msg.sender, _amount);

    bool success = stakingToken.transfer(msg.sender, _amount);
    require(success, "Token transfer failed");
}
...

```

Kode 4.2 Perbaikan smart contract token drop

Perbaikan fungsi claimToken, depositToken dan withdrawToken dilakukan dengan menambahkan modifier nonreentrant untuk mencegah pemanggilan fungsi kembali, selain itu pada masing-masing kontrak ditambahkan validasi untuk memastikan keamanan dan tidak melakukan eksekusi fungsi secara langsung. Setelah dilakukan perbaikan maka smart contract akan diuji kembali. Hasil pengujian kembali smart contract token drop sudah tidak lagi ditemukan keparahan atau kerentanan yang ditampilkan pada gambar 4.14.

```
RYZEN5@DESKTOP-8PA20BE D:\...\contracts myth analyze .\04_TokenDropFixed.sol
INFO: Could not find files for the given pattern(s).
The analysis was completed successfully. No issues were detected.
```

Gambar 4.13 Hasil pengujian kembali kontrak token drop

Hasil pengujian menggunakan Mythril pada kontrak staking menunjukkan kontrak staking tidak ditemukan rekentanan. Hasil analisis menggunakan Mythril pada kontrak staking terdapat pada gambar 4.15 di bawah ini.

```
RYZEN5@DESKTOP-8PA20BE D:\...\contracts myth analyze .\05_Staking.sol
INFO: Could not find files for the given pattern(s).
The analysis was completed successfully. No issues were detected.
```

Gambar 4.14 Hasil pengujian kontrak staking dengan Mythril

Pengujian juga dilakukan secara on-chain dengan melakukan tracking pada jaringan blockchain secara langsung pada explorer. Proses pengujian untuk memastikan integritas data dengan mengunjungi [sepolia.etherscan.com](https://sepolia.etherscan.com) untuk melihat seluruh transaksi, seluruh transaksi akan di tampilkan pada halaman blockchain eksplorers yang ditampilkan pada gambar 4.16.

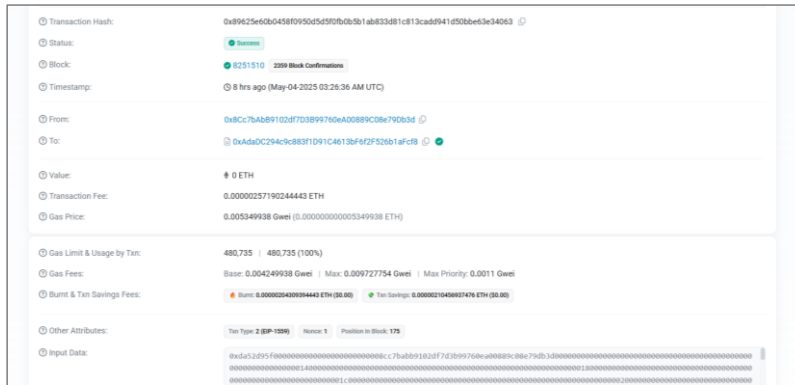
Overview		More Info		Multichain Info	
ETH BALANCE # 0.150296798609304187 ETH		TRANSACTIONS SENT Latest: 8 hrs ago ⌵ First: 10 days ago ⌵		N/A	
		FUNDED BY 0xd1b27c9...C6434EE57 ⌵ at tx 0xd9f1c3e8c2...			

Transaction Hash	Method	Block	Age	From	To	Amount	Txn Fee
0xb9625ef0b0...	0xabi00f	8251510	8 hrs ago	0x8C7bAB8...08e790b3d	0xAd8C294...26b1afCf8	0 ETH	0.00000257
0xd9f996c3ea13...	Transfer	8251488	8 hrs ago	0x150995e8...158Ac7ad	0x8C7bAB8...08e790b3d	0.05 ETH	0.00000009
0x3c743eadf1...	Transfer	8217687	5 days ago	0x5F829CE...3D36C033	0x8C7bAB8...08e790b3d	0.05 ETH	0.00021637
0xb10537ee6...	0xabi00f	8183302	10 days ago	0x8C7bAB8...08e790b3d	Contract Creation	0 ETH	0.00070962

Gambar 4.15 Interface utama sepolia etherscan

Pada [sepoliaetherscan](https://sepolia.etherscan.com) ditampilkan data seluruh transaksi yang pernah dilakukan oleh wallet terkait. Data-data tersebut disimpan pada jaringan blockchain Ethereum testnet Sepolia. Selanjutnya dilakukan pengecekan detail transaksi dari transaction hash. Sepoliaetherscan akan menampilkan detail data dari transaksi seperti pada gambar 4.17.



Gambar 4.16 Interface sepolia etherscan detail transaksi

Pada sepoliaetherscan ditampilkan detail data transaksi dan juga menampilkan data yang disimpan pada blockchain dalam bentuk yang sudah dienkripsi. Data tersebut akan dilihat bentuk aslinya untuk memastikan availability dan transparansi data yang disimpan, kemudian akan dibandingkan dengan data pada komputer lokal sebelum disimpan pada blockchain. Hasil dekripsi data pada blockchain dan data asli pada computer lokal adalah seperti pada gambar 4.15 dan 5.16 sebagai berikut.

**Gambar 4.15** Hasil dekripsi data pada blockchain

**Gambar 4.16** Perbandingan data pada yang disimpan dan sebelum disimpan



## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Berdasarkan hasil dari analisis dan penelitian yang sudah dilakukan dapat diambil kesimpulan sebagai berikut.

1. Proses perancangan dan implementasi smart contract sistem crowdfunding dilakukan melalui beberapa tahapan, dimulai dari identifikasi masalah, perancangan alur sistem dan aktor yang terlibat, hingga pemetaan struktur blockchain agar memperjelas arsitekturnya. Pemodelan smart contract disusun berdasarkan kebutuhan fungsional dan mekanisme staking dalam consensus Proof of Stake (PoS). yang dipilih karena lebih efisien dibanding dengan Proof of Work (PoW). Aspek keamanan juga diperhatikan melalui pengujian smart contract melalui uji fungsionalitas, penggunaan tools, serta pengujian integritas data secara On-chain.
1. Hasil pengujian keamanan transaksi dalam smart contract crowdfunding menunjukkan bahwa smart contract bisa menjalankan tugasnya secara fungsional dan aman yang dibuktikan pada pengujian fungsional dan pengujian menggunakan tools mythrill. Blockchain mampu menjaga keamanan data yang disimpan dengan melakukan enkripsi, transparansi data juga bisa dilihat dengan langsung mengunjungi blockchain eksplorernya yaitu sepolia etherscan, serta hasil perbandingan data pada blockchain dan komputer lokal menunjukkan bahwa data yang disimpan nilainya tetap dan tidak bisa diubah.

## DAFTAR PUSTAKA

- Aditiya Hermawan, D. P. (2023). Pemanfaatan Smart Contract dalam Transformasi Supply Chain melalui Teknologi Blockchain. *Jurnal Edukasi dan Penelitian Informatika (JEPIN)*, 1-2.
- Archana B U, V. N. (2023). Overview of Cryptography. *Data Analytics and Artificial Intelligence*, 2.
- Arunmozhi Manimuthu, R. S. (2019). A literature review on Bitcoin: Transformation of crypto currency into a global phenomenon. *IEEE Engineering Management Review*, 1-2.
- Aufila, I. Z., Musfiroh, M. F., & Hinawati, T. (2024). Securities Crowdfunding Sebagai Instrumen Pembiayaan Usaha. *Journal of Management, Economics, and Entrepreneur Mikro Kecil Menengah (Umkem)*, 3-4.
- Baihaqsani, A. K. (2023). Implementasi Teknologi Blockchain Dengan Sistem Smart Contract Pada Klaim Asuransi. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIK)*, 5.
- Budi Sahputra, T. D. (2024). Pemanfaatan Teknologi ETH Blockchain Untuk Aplikasi E-Voting Dengan Memanfaatkan Server Lokal. *Jurnal Ilmiah Teknik Informatika dan Sistem Informasi (JUTISI)*, 31-32.
- Chaoqun Ma, X. K. (2019). The Privacy Protection Mechanism of Hyperledger Fabric and Its Application in Supply Chain Finance. *Springer Opern*, 9.
- Chatkar, H. V., Singh, H. G., Sonavane, A. S., Singh, S., & Pulgam, N. (2023). Crowdfunding using Blockchain. *International Conference on Emerging Trends in Engineering & Technology - Signal and Information Processing (ICETET - SIP)*, 1-2.
- Dzulfikar, F., & Susanto, A. (2020). Implementation of Smart Contracts Ethereum Blockchain in Web-Based Electronic Voting. *TRANSFORMTIKA*, 58.
- Eleazer Gottlieb Julio Sumampouw, I. S. (2024). ANALISIS VERIFIKASI PROOF OF STAKE (POS) NFT DENGAN TEKNOLOGI SMART CONTRACT. *Jurnal Pendidikan Teknologi Informasi dan Komunikasi (EduTIK)*, 15-16.
- Gada, S. (2021). Blockchain-Based Crowdfunding: A Trust Building Model. *International Conference on Artificial Intelligence and Machine Vision (AIMV)* (pp. 3-4). Mumbai: IEEE.
- Hasan, S. A., Al-Zahra, W. N., & Auralia, A. S. (2024). Implementasi Teknologi Blockchain dalam Pengamanan Sistem Keuangan pada Perguruan Tinggi. *Jurnal Manajemen, Pendidikan dan Teknologi Informasi (MENTARI)*, 12.

- Huaqun Guo, X. Y. (2022). A survey on blockchain technology and its security. *Blockchain: Research and Applications*, 1.
- Hutomo Sakti Kartiko, T. R. (2023). Implementasi IPFS untuk Mengurangi Gas Fee Smart Contract Ethereum pada Aplikasi Penggalangan Dana. *Jurnal Edukasi dan Penelitian Informatika (JEPIN)*, 8.
- Jannah Yusoff, Z. M. (2022). A Review: Consensus Algorithms on Blockchain . *Journal of Computer and Communications*, 2022, 10, 37-50, 41 - 42.
- jawapos.com. (2022, 12 14). *hukum dan kriminal*. Retrieved from radarsemarang.jawapos.com: <https://radarsemarang.jawapos.com/hukum-dan-kriminal/721404033/diduga-gelapkan-dana-kemanusiaan-yayasan-rfps-dipolisikan>
- Liu, J. (2023). Digital Signature and Hash Algorithms Used in Bitcoin and Ethereum . *Third International Conference on Machine Learning and Computer Application (ICMLCA 2022)*, (pp. 15 - 18). SPIE.
- Lumyna. (2025, June 11). *Ethereum Sepolia Testnet a Beginners Guide*. Retrieved from lumyna.io: <https://lumyna.io/ethereum-sepolia-testnet-a-beginners-guide/>
- Maruwu, M. (2024). Metode Penelitian dan Pengembangan (R&D): Konsep, Jenis, Tahapan dan Kelebian. *Jurnal Ilmiah Profesi Pendidikan*, 1120 - 1230.
- Mendl, M., Doan, M. H., & Sassen, R. (2022). The Environtmental Impact of Cryptocurrencies Using Proof of Work and Proof of Stake Consensus Algorithms. *Journal o Environtmental Management*, 10.
- Raza, H., Ali, R., Iqbal, J., & Awais, M. (2024). Secure Room-Sharing Decentralized App Development on Ethereum Blockchain Using Smart Contract. *Journal of Informatic and Web Engineering*, 148 - 150.
- Saleh, F. (2020). Blockchain Without Waste: Proof-of-Stake. *SSRN Electronic Journal*, 21.
- tempo.co. (2022, 8 4). *hukum*. Retrieved from www.tempo.co: <https://www.tempo.co/hukum/ppatk-temukan-176-yayasan-filantropi-mirip-act-yang-selewengkan-uang-sumbangan-312917>
- Xu, Y. (2023). A decentralized Trust Management Mechanism for Crowdfunding. *Information Sciences*, 2-6.
- Zulfiqar Ali Khan, A. S. (2024). A Survey of Vulnerability Detection Techniques by Smart Contract Tools. *IEEE Access*, Volume 12, 70906.

## LAMPIRAN

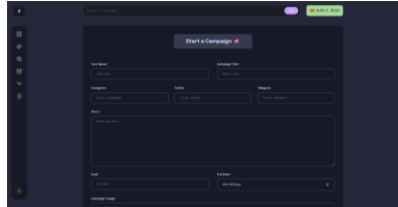
Lampiran 1. Wireframe dan interface halaman dashboard



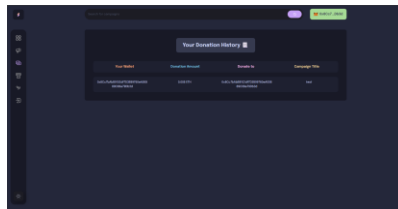
Lampiran 2. Wireframe dan interface halaman detail kampanye



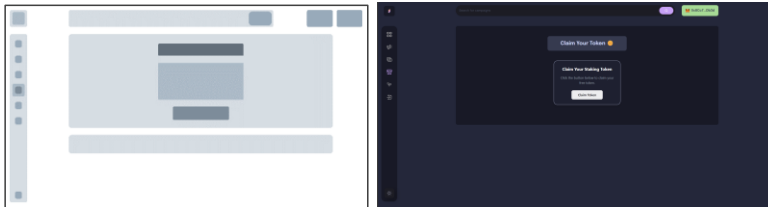
Lampiran 3. Wirefram dan interface halaman membuat kampanye



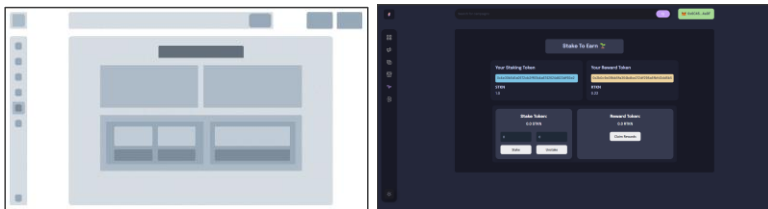
Lampiran 4. Wireframe dan interface halaman riwayat donasi



### Lampiran 5. Wireframe dan interface halaman klaim staking token



### Lampiran 6. Wireframe dan interface halaman staking



### Lampiran 7. Kode smart contract crowdfunding

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

contract Crowdfunding {
    struct Campaign {
        address owner;
        string nameOwner;
        string instagram;
        string twitter;
        string telegram;
        string title;
        string description;
        uint256 target;
        uint256 deadline;
        uint256 amountCollected;
        string image;
        address[] donators;
        uint256[] donations;
    }

    mapping(uint256 => Campaign) public campaigns;
    uint256 public numberOfCampaigns = 0;

    function createCampaign(
        address _owner,
        string memory _nameOwner,
        string memory _instagram,
        string memory _twitter,
        string memory _telegram,
        string memory _title,
        string memory _description,
        uint256 _target,
        uint256 _deadline,
        string memory _image
    ) public returns (uint256) {
        require(_deadline > block.timestamp, "The deadline should be a date in the future");
    }
}
```

```

    Campaign storage campaign = campaigns[numberOfCampaigns];
    campaign.owner = _owner;
    campaign.nameOwner = _nameOwner;
    campaign.instagram = _instagram;
    campaign.twitter = _twitter;
    campaign.telegram = _telegram;
    campaign.title = _title;
    campaign.description = _description;
    campaign.target = _target;
    campaign.deadline = _deadline;
    campaign.amountCollected = 0;
    campaign.image = _image;

    numberOfCampaigns++;

    return numberOfCampaigns - 1;
}

function donateToCampaign(uint256 _id) public payable {
    uint256 amount = msg.value;
    Campaign storage campaign = campaigns[_id];

    require(block.timestamp < campaign.deadline, "Campaign has ended");
    require(amount > 0, "Donation must be greater than zero");

    campaign.donators.push(msg.sender);
    campaign.donations.push(amount);
    campaign.amountCollected += amount;

    (bool sent,) = payable(campaign.owner).call{value: amount}("");
    require(sent, "Failed to send donation to campaign owner");
}

function getDonators(uint256 _id) public view returns (address[] memory, uint256[]
memory) {
    return (campaigns[_id].donators, campaigns[_id].donations);
}

function getCampaigns() public view returns (Campaign[] memory) {
    Campaign[] memory allCampaigns = new Campaign[](numberOfCampaigns);

    for (uint256 i = 0; i < numberOfCampaigns; i++) {
        Campaign storage item = campaigns[i];
        allCampaigns[i] = item;
    }

    return allCampaigns;
}
}

```

## Lampiran 8. Kode smart contract staking token

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

// interface ERC20
interface IERC20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint256 amount) external
    returns (bool);

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

```

```

//contract staking token inherit interface erc20
contract StakingToken is IERC20 {
    string public name = "STAKING TOKEN";
    string public symbol = "STKN";
    uint8 public decimals = 18;
    uint256 private _totalSupply = 1000000000000 ;

    mapping(address => uint256) private _balances;
    mapping(address => mapping(address => uint256)) private _allowances;

    constructor() {
        _totalSupply = _totalSupply * (10 ** uint256(decimals));
        _balances[msg.sender] = _totalSupply;
        emit Transfer(address(0), msg.sender, _totalSupply);
    }

    function totalSupply() external view override returns (uint256) {
        return _totalSupply;
    }

    function balanceOf(address account) external view override returns (uint256) {
        return _balances[account];
    }

    function allowance(address owner, address spender) external view override returns
(uint256) {
        return _allowances[owner][spender];
    }

    function approve(address spender, uint256 amount) external override returns (bool) {
        _allowances[msg.sender][spender] = amount;
        emit Approval(msg.sender, spender, amount);
        return true;
    }

    function transfer(address recipient, uint256 amount) external override returns (bool) {
        require(_balances[msg.sender] >= amount, "ERC20: transfer amount exceeds balance");
        _balances[msg.sender] -= amount;
        _balances[recipient] += amount;
        emit Transfer(msg.sender, recipient, amount);
        return true;
    }

    function transferFrom(address sender, address recipient, uint256 amount) external
override returns (bool) {
        require(_balances[sender] >= amount, "ERC20: transfer amount exceeds balance");
        require(_allowances[sender][msg.sender] >= amount, "ERC20: transfer amount exceeds
allowance");

        _balances[sender] -= amount;
        _balances[recipient] += amount;
        _allowances[sender][msg.sender] -= amount;

        emit Transfer(sender, recipient, amount);
        return true;
    }
}

```

#### Lampiran 9. Kode smart contract reward token

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

// interface erc20
interface IERC20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
}

```

```

function transfer(address recipient, uint256 amount) external returns (bool);
function allowance(address owner, address spender) external view returns (uint256);
function approve(address spender, uint256 amount) external returns (bool);
function transferFrom(address sender, address recipient, uint256 amount) external
returns (bool);

event Transfer(address indexed from, address indexed to, uint256 value);
event Approval(address indexed owner, address indexed spender, uint256 value);
}

//contract staking token inherit interface erc20
contract RewardToken is IERC20 {
    string public name = "REWARD TOKEN";
    string public symbol = "RTKN";
    uint8 public decimals = 18;
    uint256 private _totalSupply = 1000000000000000000;

    mapping(address => uint256) private _balances;
    mapping(address => mapping(address => uint256)) private _allowances;

    constructor() {
        _totalSupply = _totalSupply * (10 ** uint256(decimals));
        _balances[msg.sender] = _totalSupply;
        emit Transfer(address(0), msg.sender, _totalSupply);
    }

    function totalSupply() external view override returns (uint256) {
        return _totalSupply;
    }

    function balanceOf(address account) external view override returns (uint256) {
        return _balances[account];
    }

    function allowance(address owner, address spender) external view override returns
(uint256) {
        return _allowances[owner][spender];
    }

    function approve(address spender, uint256 amount) external override returns (bool) {
        _allowances[msg.sender][spender] = amount;
        emit Approval(msg.sender, spender, amount);
        return true;
    }

    function transfer(address recipient, uint256 amount) external override returns (bool) {
        require(_balances[msg.sender] >= amount, "ERC20: transfer amount exceeds balance");
        _balances[msg.sender] -= amount;
        _balances[recipient] += amount;
        emit Transfer(msg.sender, recipient, amount);
        return true;
    }

    function transferFrom(address sender, address recipient, uint256 amount) external
override returns (bool) {
        require(_balances[sender] >= amount, "ERC20: transfer amount exceeds balance");
        require(_allowances[sender][msg.sender] >= amount, "ERC20: transfer amount exceeds
allowance");

        _balances[sender] -= amount;
        _balances[recipient] += amount;
        _allowances[sender][msg.sender] -= amount;

        emit Transfer(sender, recipient, amount);
        return true;
    }
}

```



## Lampiran 10. Kode smart contract token drop

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

import "@openzeppelin/contracts/extension/Ownable.sol";

interface IERC20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint256 amount) external
    returns (bool);

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

contract TokenDrop is Ownable {
    IERC20 public stakingToken;
    mapping(address => bool) public hasClaimed; // Menyimpan status klaim token

    event Claimed(address indexed user, uint256 amount);

    constructor(address _stakingToken) {
        stakingToken = IERC20(_stakingToken);
    }

    // Fungsi untuk klaim staking token
    function claimToken() external {
        require(!hasClaimed[msg.sender], "You have already claimed");

        hasClaimed[msg.sender] = true;
        stakingToken.transfer(msg.sender, 1 * (10 ** 18));

        emit Claimed(msg.sender, 1 * (10 ** 18));
    }

    function depositTokens(uint256 _amount) external onlyOwner {
        stakingToken.transferFrom(msg.sender, address(this), _amount);
    }

    function withdrawTokens(uint256 _amount) external onlyOwner {
        stakingToken.transfer(msg.sender, _amount);
    }

    function getDepositedSupply() external view returns (uint256) {
        return stakingToken.balanceOf(address(this));
    }
}
```

## Lampiran 11. Kode smart contract staking

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

contract Staking {
    address public rewardToken;
    address public stakingToken;
    address public admin;
    uint256 public rewardTokenBalance;
    uint256 public stakingTokenBalance;
    uint80 public timeUnit;
    uint256 public rewardRatioNumerator;
    uint256 public rewardRatioDenominator;
```

```

mapping(address => uint256) public stakedBalances;
mapping(address => uint256) public stakingTimestamps;

event Staked(address indexed user, uint256 amount);
event Unstaked(address indexed user, uint256 amount);
event RewardClaimed(address indexed user, uint256 amount);
event RewardDeposited(uint256 amount);
event RewardWithdrawn(uint256 amount);

modifier onlyAdmin() {
    require(msg.sender == admin, "Not authorized");
    _;
}

constructor(
    address _rewardToken,
    address _stakingToken,
    uint80 _timeUnit,
    uint256 _rewardRatioNumerator,
    uint256 _rewardRatioDenominator
) {
    admin = msg.sender;
    rewardToken = _rewardToken;
    stakingToken = _stakingToken;
    timeUnit = _timeUnit;
    rewardRatioNumerator = _rewardRatioNumerator;
    rewardRatioDenominator = _rewardRatioDenominator;
}

function stake(uint256 _amount) external {
    require(_amount > 0, "Cannot stake 0");
    stakedBalances[msg.sender] += _amount;
    stakingTimestamps[msg.sender] = block.timestamp;
    stakingTokenBalance += _amount;
    emit Staked(msg.sender, _amount);
}

function unstake(uint256 _amount) external {
    require(stakedBalances[msg.sender] >= _amount, "Insufficient staked balance");
    stakedBalances[msg.sender] -= _amount;
    stakingTokenBalance -= _amount;
    emit Unstaked(msg.sender, _amount);
}

function claimRewards() external {
    require(stakedBalances[msg.sender] > 0, "No staked tokens");
    uint256 reward = calculateReward(msg.sender);
    require(reward <= rewardTokenBalance, "Not enough rewards");
    rewardTokenBalance -= reward;
    stakingTimestamps[msg.sender] = block.timestamp;
    emit RewardClaimed(msg.sender, reward);
}

function depositRewardTokens(uint256 _amount) external onlyAdmin {
    require(_amount > 0, "Amount must be greater than zero");
    rewardTokenBalance += _amount;
    emit RewardDeposited(_amount);
}

function withdrawRewardTokens(uint256 _amount) external onlyAdmin {
    require(_amount <= rewardTokenBalance, "Not enough reward tokens");
    rewardTokenBalance -= _amount;
    emit RewardWithdrawn(_amount);
}

function calculateReward(address _user) internal view returns (uint256) {
    uint256 stakingDuration = block.timestamp - stakingTimestamps[_user];
    return (stakedBalances[_user] * stakingDuration * rewardRatioNumerator) /

```

```
        (timeUnit * rewardRatioDenominator);  
    }  
  
    function getStakeInfo(address _user) external view returns (uint256 stakedAmount,  
uint256 rewardAmount) {  
        stakedAmount = stakedBalances[_user];  
        rewardAmount = calculateReward(_user);  
    }  
}
```