DECODE: JURNAL PENDIDIKAN TEKNOLOGI INFORMASI
P-ISSN: 2775-2984 | E-ISSN: 2775-1813

Vol. 4, No 3, November 2024, Hal: 775-786
Doi: http://dx.doi.org/10.51454/decode.v4i3.630
https://journal.umkendari.ac.id/index.php/decode

Implementasi *Smart Contract* dalam Aplikasi Website Digitalisasi Realisasi Anggaran OPEX

Aida Lestari[1*], Alfa Ryano Yohannis[1]

[1]Program Studi Informatika, Universitas Pradita, Indonesia.

Artikel Info

Kata Kunci:

Blockchain;
Operational Expenditure;
OPEX;
Smart Contract.

Keywords:

Blockchain;
Operational Expenditure;
(OPEX);
Smart Contract.

Riwayat Artikel:

Submitted: 14 Juni 2024
Accepted: 25 Juli 2024
Published: 2 September 2024

Abstrak: Pemalsuan data merupakan masalah serius yang dapat merugikan berbagai pihak. Dalam era digital, teknologi *blockchain* menawarkan solusi dengan menyimpan data secara permanen dan aman. Penelitian ini bertujuan mengembangkan sistem berbasis *smart contract* pada platform *blockchain Ethereum* untuk digitalisasi realisasi anggaran *Operational Expenditure* (OPEX) perusahaan, menggunakan metode SDLC *Waterfall*. Metode ini dipilih karena dapat menyelesaikan tantangan waktu pengembangan dan tidak melibatkan respons pengguna pada tahap awal. Hasil penelitian menunjukkan bahwa sistem ini mampu mengotomatisasi proses persetujuan anggaran, mengurangi waktu dan biaya, serta meningkatkan transparansi dan akuntabilitas. Hasil pengujian Unit Testing menunjukkan bahwa sistem berjalan sesuai harapan tanpa kesalahan. Pengujian gas rata-rata menunjukkan penggunaan 106.375,3 *gas* dengan biaya sekitar 0,00208 ETH atau Rp 139.129. Pada *end-to-end testing* menggunakan skenario proses pembelian server IBM, dimulai dari *Budgeting Staff Controller* yang menginput harga Rp 150.000.000 untuk pembelian *IBM Power System S822LC Rack Server*. Setelah pengajuan realisasi diajukan, kepala departemen melakukan pengecekan dan persetujuan, yang berlanjut melalui beberapa tahap hingga seluruh pengajuan disetujui. Implementasi teknologi ini diharapkan dapat meningkatkan efisiensi, transparansi, dan keamanan dalam pengelolaan anggaran di perusahaan.

Abstract: *Data falsification is a serious issue that can harm various parties. In the digital age, blockchain technology offers a solution by storing data permanently and securely. This research aims to develop a smart contract-based system on the Ethereum blockchain platform for the digitization of Operational Expenditure (OPEX) budget realization for companies, using the SDLC Waterfall method. This method was chosen because it can address development time challenges and does not involve user responses in the early stages. The research results show that the system is capable of automating the budget approval process, reducing time and costs, and increasing transparency and accountability. Unit Testing results demonstrate that the system operates as expected without errors. Average gas testing indicates a usage of 106,375.3 gas with a cost of approximately 0.00208 ETH or IDR 139,129. End-to-end testing using the IBM server purchase process scenario begins with the Budgeting Staff Controller entering the price of IDR 150,000,000 for the IBM Power System S822LC Rack Server purchase. After the realization request is submitted, the department head checks and approves it, which continues through several stages until the entire request is approved. The implementation of this technology is expected to enhance efficiency, transparency, and security in budget management in the company.*

Corresponding Author:
Aida Lestari
Email: aida.lestari@student.pradita.ac.id

PENDAHULUAN

Pemalsuan data adalah tindakan yang dapat merugikan berbagai pihak karena pihak ketiga mungkin memanfaatkan data yang dipalsukan untuk perdagangan atau penggunaan pribadi(Rahardja, 2022). Oleh karena itu, diperlukan teknologi yang mampu menyimpan data secara permanen dengan tingkat keamanan modern sehingga data tersebut tidak bisa diubah dan hanya bisa dilihat saja (Watini et al., 2022). *Accounting fraud* merupakan fenomena yang banyak terjadi dalam perusahaan. Salah satu contoh kasus *accounting fraud* yang terjadi dalam skandal keuangan Wirecard pada tahun 2020, merupakan contoh yang melibatkan perusahaan teknologi keuangan Jerman yang dikenal sebagai Wirecard AG (Luthfiyyah & Dewayanto, 2023).

Dalam era digital yang terus berkembang, adopsi teknologi canggih menjadi sangat penting bagi perusahaan. Salah satu teknologi yang menawarkan manfaat besar adalah *blockchain*, khususnya melalui penerapan *smart contract*. Teknologi ini memungkinkan eksekusi otomatis perjanjian tanpa perlu pihak ketiga, meningkatkan transparansi dan efisiensi dalam manajemen data (Sunarya, 2022). *Ethereum* adalah platform *blockchain* pertama untuk mengembangkan *smart contract*. Platform ini mendukung *smart contract* yang canggih dan dapat disesuaikan dengan bantuan mesin virtual yang lengkap, yang disebut *Ethereum Virtual Machine* (EVM) (Khan et al., 2021). Penerapan *smart contract* menggunakan *Ethereum* pada platform *blockchain* menawarkan potensi besar untuk meningkatkan pengelolaan anggaran *Operational Expenditure* (OPEX) perusahaan.

Saat ini, banyak perusahaan menghadapi tantangan dalam mengelola anggaran OPEX secara efisien dan transparan. Proses tradisional yang mengandalkan dokumentasi kertas dan persetujuan manual sering kali memakan waktu, rawan kesalahan, dan rentan terhadap manipulasi. Hal ini dapat menyebabkan kurangnya akuntabilitas dalam pengelolaan anggaran. Oleh karena itu, diperlukan solusi inovatif yang dapat mengotomatisasi proses ini dan memastikan integritas data.

Penelitian oleh Asma Khatoon (2020) dalam *"A Blockchain-Based Smart Contract System for Healthcare Management"* menunjukkan bagaimana prinsip desentralisasi menggunakan teknologi *blockchain* dapat meningkatkan manajemen catatan medis. Sistem ini memperkenalkan *smart contract* untuk meningkatkan auditabilitas, interoperabilitas, dan aksesibilitas, serta menjamin privasi dan keamanan data. Selain itu, penelitian ini menunjukkan potensi pengurangan biaya transaksi dan penyederhanaan prosedur dalam manajemen data kesehatan (K., 2020).

Ronald Simanjuntak (2020) dalam penelitiannya *"Benefits of the Opex Pro Application in Online Project Monitoring and Evaluation at PT. XYZ"* menganalisis kontribusi aplikasi OPEX Pro dalam memudahkan monitoring dan evaluasi proyek konstruksi. Aplikasi ini memberikan manfaat berupa kemudahan pelaporan, pemantauan proyek yang efektif, dan kemampuan membuat keputusan cepat, yang semuanya dapat meningkatkan kinerja proyek (Ronald Simanjuntak & Nahdi, 2020).

Ada juga penelitian dari Ahmadisheykhsarmast (2020) yang merupakan penelitian tentang sistem kontrak pintar SMTSEC menggunakan *blockchain Ethereum*, yang dipublikasikan dalam *"A smart contract system for security of payment of construction contracts"*, menunjukkan bagaimana *Ethereum* dapat digunakan untuk mengamankan pembayaran dalam proyek konstruksi. Sistem ini mengunci dana di awal bulan dan mentransfernya secara otomatis setelah persetujuan pemberi kerja, meningkatkan efisiensi dan menghilangkan masalah pembayaran (Ahmadisheykhsarmast & Sonmez, 2020).

Penelitian ini mengusulkan implementasi *smart contract* dalam aplikasi website untuk digitalisasi realisasi anggaran OPEX menggunakan platform *blockchain Ethereum*. Setiap transaksi dan persetujuan akan dicatat dalam *blockchain*, memastikan data tidak dapat diubah dan dapat dilacak dengan mudah. Dengan *smart contract*, proses persetujuan anggaran dapat diotomatisasi, mengurangi waktu dan biaya, serta meningkatkan transparansi dan akuntabilitas.

Penelitian ini menawarkan inovasi dalam bentuk aplikasi website yang memanfaatkan *blockchain* dan *smart contract* untuk digitalisasi realisasi anggaran OPEX. Ini merupakan pendekatan baru dalam manajemen anggaran perusahaan. Keunikan solusi ini adalah kemampuan mengotomatisasi seluruh

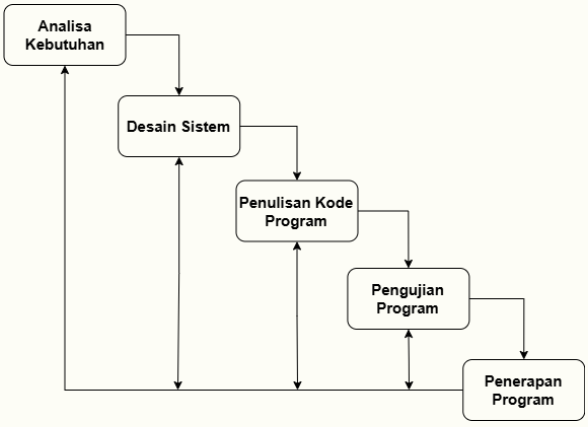
proses persetujuan anggaran dengan jaminan keamanan dan integritas data dari teknologi *blockchain*. Aplikasi ini juga berfungsi sebagai alat dokumentasi untuk audit dan pelaporan keuangan.

Tujuan utama dari penelitian ini adalah mengembangkan sistem yang meningkatkan efisiensi dan transparansi pengelolaan anggaran OPEX *melalui smart contract*. Penelitian ini diharapkan dapat mengurangi waktu dan biaya dalam proses persetujuan anggaran, mengurangi risiko kesalahan dan manipulasi data, serta menyediakan dokumentasi yang dapat diandalkan untuk audit dan pelaporan. Penelitian ini diharapkan memberikan kontribusi nyata dalam peningkatan praktik manajemen keuangan di PT XYZ.

METODE

Metode penelitian yang digunakan untuk mengembangkan sistem ini terdiri atas beberapa tahapan. Tahapan tersebut dimulai dari proses pengumpulan data melalui observasi secara langsung dan wawancara dengan user. Kemudian proses pengembangkan sistem mengacu dari SDLC (*Software Development Life Cycle*) dengan model *Waterfall*. Metode SDLC (*Software Development Life Cycle*) merupakan proses pembuatan dan perubahan sistem, serta model dan metodologi yang digunakan dalam pengembangan rekayasa perangkat lunak. Proses ini dilakukan dari awal hingga akhir (Tri et al., n.d.). Model *waterfall* adalah model klasik yang bersifat sistematis dan berurutan dalam pengembangan perangkat lunak. Model ini disebut harus menyelesaikan setiap tahap sebelum dapat melanjutkan ke tahap berikutnya, dan prosesnya berjalan secara berurutan (Mailasari, 2019). Keuntungan menggunakan metode *waterfall* meliputi alur kerja yang jelas, dokumentasi yang baik, penghematan biaya, dan cocok untuk pengembangan perangkat lunak skala besar. Namun, kelemahan metode ini adalah membutuhkan tim yang solid, kurang fleksibel, sulit mendapatkan gambaran sistem yang jelas, dan memerlukan waktu lebih lama (Suherni, 2023).

Metodologi *Waterfall* mengikuti pendekatan yang bersifat berurutan dan linear, setiap fase harus diselesaikan sebelum dapat melanjutkan ke fase berikutnya. Sifat yang terstruktur ini membuat proses pengembangan dapat diprediksi, dapat diselesaikan dengan jumlah tim yang kecil, dan garis waktu proyek yang sederhana (Cholifah et al., 2018). Penelitian ini menggunakan metode *Waterfall* karena dengan metode ini dapat menyelesaikan salah satu tantangan dalam mengembangkan sistem ini, yaitu terbatasnya waktu pengembangan. Selain itu, metode ini juga dipilih karena tim pengembang yang terlibat berjumlah kecil, sehingga pendekatan yang terstruktur dan linear dari metode Waterfall sangat cocok untuk memastikan setiap tahap pengembangan dapat diselesaikan secara efisien tanpa melibatkan banyak anggota tim. Hal ini didukung oleh sifat dari metode *Waterfall* yang belum melibatkan respon dari pengguna. Model *Waterfall* yang akan digunakan pada penelitian ini dapat dilihat pada gambar berikut:



Gambar 1. Model SDLC *Waterfall* (Wijaya & Susanto, 2021)

Berikut uraian mengenai model SLDC Waterfall pada Gambar 1, sebagai berikut:

1. Analisis kebutuhan yang dilakukan untuk merancang sistem ini adalah dengan menggunakan data primer dan data sekunder. Data primer berupa observasi bisnis proses dan wawancara dengan pengguna. Data sekunder berupa dokumen-dokumen yang terlibat dalam aplikasi. Hasilnya adalah

mempelajari penerapan blockchain dalam sistem ini sehingga setiap transaksi dapat disimpan permanen dan transparan. Pengguna sistem terbagi menjadi 2, yaitu: *Budgeting Controller Staff* dan *Approvers*. *Budgeting Controller Staff* dapat melakukan pengajuan dan merevisi realisasi, sedangkan *Approvers* yaitu setiap superior yang dapat melakukan *approval* dan *reject* dari realisasi yang ada.

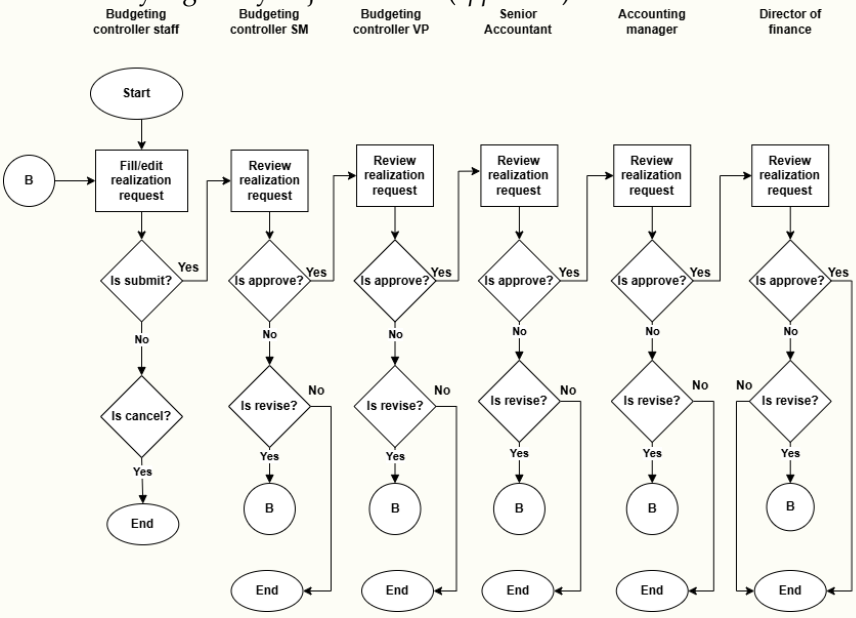
2. Desain sistem yang dibuat berdasarkan tahapan analisis kebutuhan sebelumnya. Dalam tahapan ini melakukan perancangan *database*, membuat *flowchart* aplikasi, dan membuat *use case diagram*. Diagram-diagram ini berperan sebagai gambaran dalam proses pembuatan sistem.
3. Penulisan kode program merupakan tahapan menuliskan kode program berdasarkan rancangan yang telah dibuat sebelumnya agar aplikasi dapat berjalan. Sistem dibuat dengan menggunakan teknologi NestJs, Prisma ORM (*Object-Relational Mapping*), database PostgreSQL. Remix IDE digunakan untuk menulis dan menguji *smart contract*, yang kemudian dihubungkan dengan backend menggunakan Ethers.js, sehingga dapat terjadi interaksi antara aplikasi dan *smart contract* di jaringan *Ethereum*.
4. Pengujian program dilakukan setelah program selesai dibuat. Jenis pengujian sistem yang dilakukan dalam penelitian ini adalah dengan menggunakan metode *unit testing*, pengujian *gas*, dan *end to end testing*. Pengujian unit testing menggunakan *plugin solidity unit testing* dari *remix.* Pengujian gas dilakukan melalui *remix* dengan mencatat penggunaan gas dalam satu transaksi dan kemudian menghitung berdasarkan yang diperoleh. *End to end testing* dilakukan dengan membuat skenario pengajuan realisasi anggaran, yang kemudian diajukan untuk disetujui oleh pihak-pihak yang berwenang.
5. Penerapan program dilakukan setelah sistem memasuki tahapan *production*. Sistem dipindahkan dari proses *development* ke tahap *production* dan akan diserahkan kepada pengguna, untuk melakukan latihan dengan sistem (Al-Saqqa et al., 2020). Selanjutnya akan diterapkan pemeliharaan sistem secara berkala, perbaikan sistem, penambahan fitur baru yang sesuai dengan kebutuhan pengguna, serta meningkatkan kinerja dan keamanan sistem.

HASIL DAN PEMBAHASAN

Perancangan system

a. Flowchart

Flowchart berfungsi sebagai cerminan visual dari program komputer menggunakan simbol-simbol *flowcharting* untuk menggambarkan langkah-langkah pemecahan masalah unit yang spesifik (A.B. Chaudhuri, 2020). Seperti terlihat pada Gambar 2, terdapat 6 aktor yang dibagi menjadi yang mengajukan realisasi dan yang menyetujui realisasi (*approvers*).

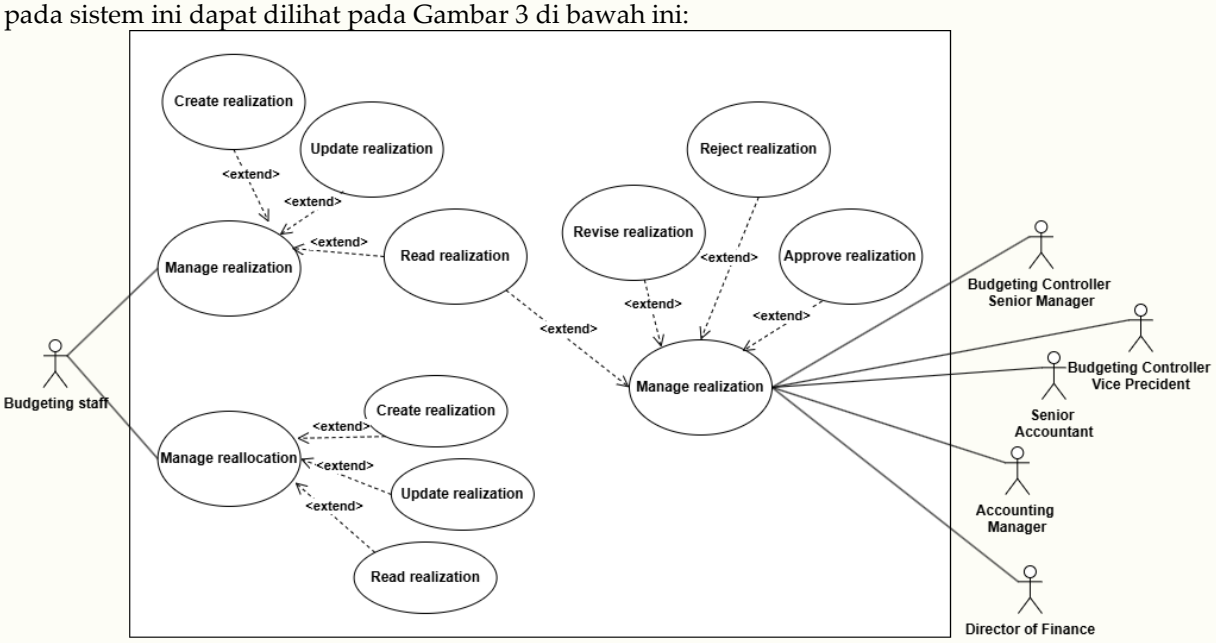


Gambar 2. *Flowchart* Sistem

Flowchart tersebut menggambarkan alur kerja dari sistem yang dibangun. Di mana *budgeting controller staff* melakukan pengajuan realisasi, dan akan melewati proses *approval* sebanyak 5 tahap oleh *superior* agar pengajuan realisasi disetujui seluruhnya. Jika para *superior* menilai perlunya dilakukan revisi dari pengajuan realisasi, maka akan dikembalikan lagi kepada *budgeting controller staff* yang terkait untuk dilakukan perbaikan.

b. Use Case Diagram

Use case diagram merupakan diagram yang menggambarkan hubungan antara *user* sebuah sistem dengan sistem tersebut tentang bagaimana sistem tersebut akan digunakan. *Use case diagram* terdiri atas aktor dan interaksi yang dapat dilakukannya. Aktor ini dapat berupa manusia, perangkat keras, sistem lain, atau entitas yang berinteraksi dengan sistem (Kurniawan & Syarifuddin, 2020). *Use case diagram* pada sistem ini dapat dilihat pada Gambar 3 di bawah ini:



Gambar 3. *Use Case Diagram*

Peran *budgeting staff* dapat melakukan pengajuan realisasi dan realokasi anggaran. Sedangkan peran *budgeting controller senior manager, budgeting controller vice precident, senior accountant, accounting manager, dan director of finance (approvers)* dapat menyetujui realisasi. *Approvers* dapat melakukan koneksi dengan *blockchain* untuk memberikan persetujuan terhadap pengajuan realisasi.

Smart Contract

Smart contract adalah kode yang dapat dieksekusi di atas *blockchain* untuk memfasilitasi, mengeksekusi, dan menegakkan perjanjian antara pihak-pihak yang tidak saling percaya tanpa memerlukan pihak ketiga yang terpercaya. *Smart contract* memungkinkan otomatisasi jaringan dan transformasi kontrak kertas menjadi kontrak digital. Dibandingkan dengan kontrak tradisional, *smart contract* memungkinkan pengguna untuk mengkodekan perjanjian dan hubungan kepercayaan mereka, menyediakan transaksi otomatis tanpa pengawasan dari otoritas pusat (Khan et al., 2021). Pada sistem yang dibangun, *smart contract* berfungsi untuk memberikan *approval* persetujuan kepada realisasi yang disetujui, seperti yang dapat dilihat dari Gambar 4 di bawah ini:

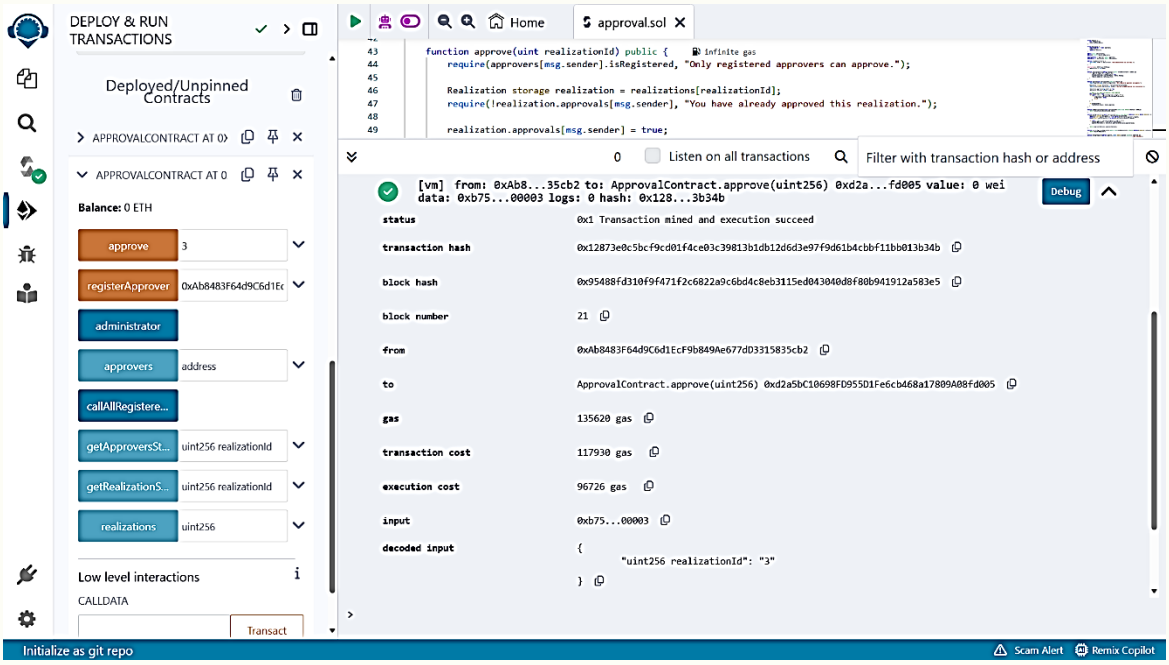
```
function approve(uint realizationId) public {
    require(approvers[msg.sender].isRegistered, "Only registered approvers can approve.");
    Realization storage realization = realizations[realizationId];
    require(!realization.approvals[msg.sender], "You have already approved this realization.");
    realization.approvals[msg.sender] = true;
    realization.approvers.push(msg.sender);
    // Check if all registered approvers have approved
    bool allApproved = true;
    for (uint i = 0; i < registeredApprovers.length; i++) {
      if (!realization.approvals[registeredApprovers[i]]) {
        allApproved = false;
        break;
      }
    }
    if (allApproved) {
      realization.status = Status.Approved;
    }
  }
```

Gambar 4. Potongan kode *smart contract*

Fungsi *approve* (uint *realizationId*) adalah fungsi publik yang digunakan untuk memberikan persetujuan pada realisasi tertentu. Pertama, fungsi ini memverifikasi bahwa pengirim transaksi (msg.sender) adalah seorang *approver* yang terdaftar. Jika *approver* tidak terdaftar, maka transaksi akan gagal dengan pesan "*Only registered approvers can approve*". Kemudian fungsi mengambil data realisasi yang sesuai dengan *realizationId* dari *array realizations*. Selanjutnya, fungsi memastikan bahwa pengirim belum memberikan persetujuan pada realisasi ini sebelumnya; jika sudah, transaksi akan gagal dengan pesan "*You have already approved this realization*". Jika belum, status persetujuan pengirim diatur menjadi *true* dan pengirim ditambahkan ke daftar *approver*s untuk realisasi tersebut. Setelah itu, fungsi memeriksa apakah semua *approver* yang terdaftar telah memberikan persetujuan mereka dengan mengiterasi melalui daftar *registeredApprovers*. Jika ada satu *approver* yang belum menyetujui, variabel *allApproved* diatur menjadi *false* dan perulangan dihentikan. Jika semua *approver* telah memberikan persetujuan mereka, status realisasi diubah menjadi *Approved*.

Fungsi *approve* memastikan bahwa hanya *approver* yang terdaftar yang dapat memberikan persetujuan untuk realisasi. Fungsi ini juga mencegah *approver* memberikan persetujuan lebih dari sekali untuk realisasi yang sama. Setelah semua *approver* memberikan persetujuan mereka, status realisasi diubah menjadi *Approved*. Fungsi ini sangat penting dalam sistem persetujuan realsasi anggaran, di mana setiap realisasi harus mendapatkan persetujuan dari semua pihak yang berwenang sebelum dapat dilanjutkan ke tahap selanjutnya.

Berikut adalah *output* ketika melakukan approval dari suatu realisasi.



Gambar 5. Hasil Smart Contract

Transaksi Ethereum ini berhasil ditambang dan dieksekusi, yang ditandai dengan status 0x1. Transaksi ini memiliki *hash* unik 0x74313b75af70b116de9d39b3f86c62611fbd8115c7ed94b6aaaad6d5ebb8fbd0 dan termasuk dalam blok dengan *hash* 0x95d0f8f4d755124214b38e636f11ac427e6a5116ed6094a7732c78f55e14a488 pada nomor blok 10. Pengirim transaksi adalah alamat 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 dan transaksi ini ditujukan ke *ApprovalContract* pada fungsi *approve* dengan alamat 0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47. Transaksi ini disediakan gas ebesar 135645, dengan biaya transaksi aktual sebesar 117952 gas dan biaya eksekusi sebesar 96748 gas. Data *input* mentah dari transaksi ini adalah 0xb75...00001, yang setelah diterjemahkan menunjukkan bahwa transaksi tersebut memanggil fungsi *approve* dengan memberi *input* realizationId bernilai 1.

Transaction hash ini nantinya akan menjadi acuan ketika melakukan proses auditing. *Transaction hash* menyimpan informasi penting mengenai status realisasi yang telah disetujui oleh pihak yang berwenang. Setiap transaksi yang tercatat dalam *blockchain* memiliki *hash* unik yang tidak dapat diubah, sehingga menjamin integritas dan keaslian data. Sehingga auditor dapat menggunakan *hash* ini untuk melacak setiap *approvers* yang telah menyetujui realisasi.

Pengujian Sistem

Pengujian sistem adalah proses untuk menguji program atau aplikasi yang telah dikembangkan oleh *developer*. Pengujian ini umumnya dilakukan untuk menemukan masalah seperti *bug* pada *software* untuk memastikan kualitasnya. Pengujian *software* sangat penting dilakukan karena biasanya ada hal-hal yang terlewat dalam pembuatan aplikasi (Cholifah et al., 2018). Pengujian kinerja sebuah program dapat memastikan pengalaman pengguna yang optimal dan keberhasilan fungsionalitasnya (Nurdiana et al., 2024). Pada penelitian ini metode pengujian yang dilakukan adalah *unit testing*, pengujian *gas*, dan *end to end testing*.

a. *Unit Testing*

Unit testing adalah teknik untuk menguji apakah kode program perangkat lunak sudah efektif dan bebas dari kesalahan. Untuk melakukan pengujian ini dibutuhkan *test case*. *Test case* adalah teknik pengujian perangkat lunak yang menggunakan serangkaian skenario eksekusi untuk menentukan apakah modul yang sedang dikembangkan sudah memenuhi spesifikasi atau belum (Hasibuan & Dirgahayu, 2021). Penelitian ini dilakukan pengujian dengan metode *white-box* dan *plugin solidity unit*

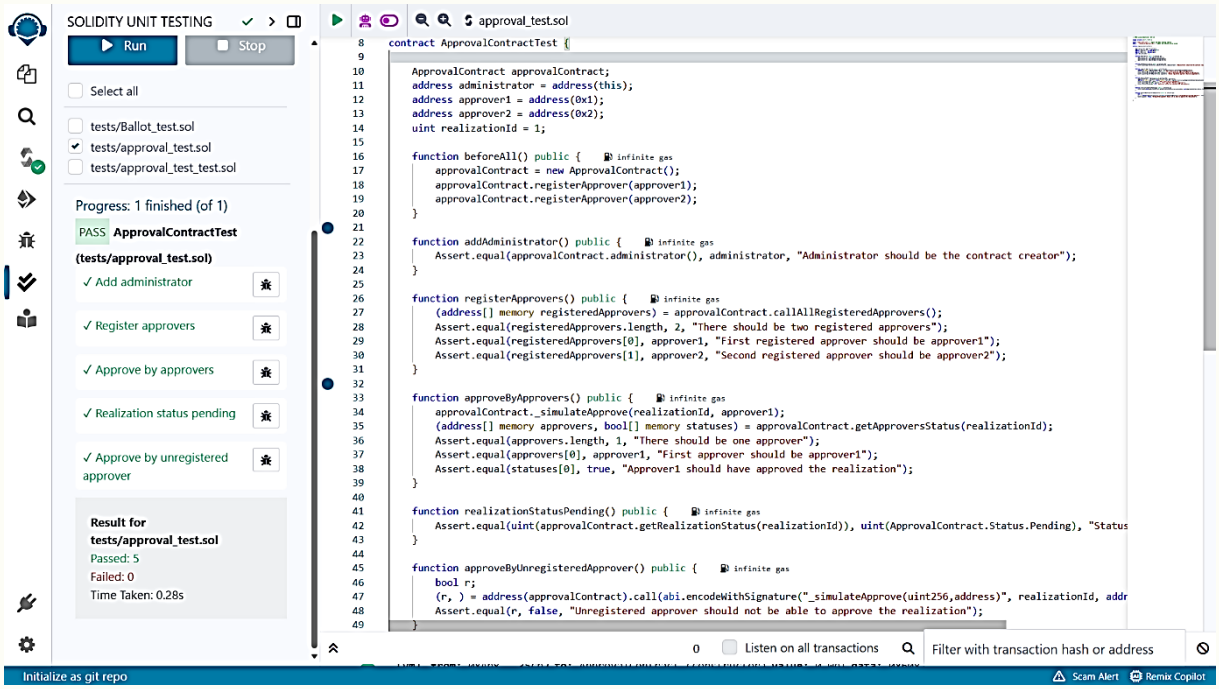
testing dari *remix*. Berikut adalah hasil pengujian yang dapat dilihat pada Tabel 1, seluruh fungsi dapat berjalan dengan baik dan tidak terdapat *bug*.

Tabel 1. Hasil *Unit Testing*

Fungsi	% lulus	Status
addAdministrator()	100%	Pass
registerApprovers()	100%	Pass
realizationStatusPending()	100%	Pass
approveByApprovers()	100%	Pass
approveByUnregiseredApprover()	100%	Pass

Unit test fungsi *addAdministrator* untuk melakukan verifikasi apakah administrator adalah alamat yang membuat *contract* atau bukan. *Unit test* fungsi *registerApprovers* memastikan bahwa pihak-pihak yang memiliki wewenang melakukan approve berhasil didaftarkan. Unit test fungsi *realizationStatusPending* berfungsi untuk memeriksa apakah status awal dari suatu realisasi adalah "*Pending*". *Unit test approveByApprovers* untuk memverifikasi bahwa suatu realisasi berubah status menjadi "*Approved*" setelah semua *approver* menyetujui realisasi tersebut. *Unit test approveByUnregisteredApprover* untuk memastikan bahwa *approver* yang tidak terdaftar tidak dapat memberikan persetujuan kepada realisasi yang ada.

Berikut adalah gambar yang menunjukkan hasil dari *unit testing* yang dijalankan menggunakan *remix*.



Gambar 6. Hasil Unit Testing dengan Remix

b. Pengujian *Gas*

Gas fee adalah biaya yang harus dibayar oleh pengguna saat bertransaksi di dalam jaringan *blockchain* (Fan et al., 2020). Konsep dasarnya adalah membebankan pengguna untuk biaya komputasi (seperti listrik dan CPU) yang diperlukan oleh penambang *(miner)* untuk mengeksekusi transaksi. Penambang adalah individu atau organisasi yang memvalidasi dan menambahkan data baru ke dalam *blockchain* dan diberi kompensasi dengan biaya *gas* tersebut (Pierro & Rocha, 2019). Harga *gas* ini dapat bervariasi tergantung pada jumlah transaksi yang terdapat di jaringan *blockchain*.

Perhitungan pengujian *gas* ini penting bagi pihak perusahaan agar dapat mengkalkulasi biaya yang akan dikeluarkan menjalankan transaksi atau *smart contract*. Hal ini memungkinkan perusahaan untuk mengoptimalkan penggunaan *gas*, menghindari biaya yang tidak perlu, dan mengelola anggaran agar lebih efektif.

Biaya *gas* dibayarkan dalam mata uang *ether* (ETH). Harga *gas* dinyatakan dalam gwei. Berdasarkan *ethereum.org,* setiap gwei setara dengan 0,000000001 ETH (10-9 ETH). Menurut dari *etherscan.io*, rata-rata *price* adalah 20 gwei pada bagian *lower*, sehingga untuk mempermudah perhitungan maka akan menggunakan 20 gwei.

Pengujian *gas* dalam penelitian ini dilakukan pada hari kerja antara pukul 08.00 hingga 16.00, yang merupakan jam kerja kantor. Waktu tersebut dipilih untuk mencerminkan kondisi penggunaan jaringan yang sebenarnya, di mana aktivitas transaksi kemungkinan lebih tinggi dan bervariasi. Dengan melakukan pengujian pada rentang waktu ini, hasil yang diperoleh akan lebih relevan dan akurat dalam menggambarkan kebutuhan *gas* di situasi nyata.

Tabel 2. Hasil Pengujian *Gas*

Time	**Hari**	**Gas yang digunakan (unit)**	**Biaya gas (ETH)**	**Biaya gas (Rupiah)**	**Status transaksi**
14.42 WIB	Rabu, 29 Mei 2024	79.131	0,00158	Rp 97.916	Berhasil
14.49 WIB	Rabu, 29 Mei 2024	87.957	0,00175	Rp 108.451	Berhasil
14.53 WIB	Rabu, 29 Mei 2024	92.679	0,00185	Rp 114.648	Berhasil
15.42 WIB	Rabu, 29 Mei 2024	117.952	0,00235	Rp 145.634	Berhasil
10.26 WIB	Kamis, 30 Mei 2024	117.952	0,00235	Rp 145.634	Berhasil
10.27 WIB	Kamis, 30 Mei 2024	117.952	0,00235	Rp 145.634	Berhasil
14.51WIB	Kamis, 30 Mei 2024	117.952	0,00235	Rp 145.634	Berhasil
13.36 WIB	Jumat, 31 Mei 2024	117.952	0,00235	Rp 145.634	Berhasil
13.37 WIB	Jumat, 31 Mei 2024	97.613	0,00195	Rp 120.845	Berhasil
14.37 WIB	Jumat, 31 Mei 2024	97.613	0,00195	Rp 120.845	Berhasil
Rata-rata		106.375,3	0,00208	Rp 139.129	

Berikut adalah rincian biaya *gas* dalam unit *gas* yang digunakan dalam transaksi *Ethereum* pada sistem ini. Biaya *gas* rata-rata untuk 106.375,3 unit *gas* adalah sekitar Rp 139.129. Seluruh data menunjukkan korelasi antara unit *gas* yang digunakan dan biaya *gas* yang dihasilkan, dengan semua transaksi berstatus berhasil.

c. *End to End Testing*

End-to-end testing adalah metode pengujian yang digunakan untuk memastikan bahwa alur kerja aplikasi berfungsi sesuai dengan desainnya dari awal hingga akhir. Secara umum, dalam pelaksanaan end-to-end testing dengan manual testing, pengujian dilakukan dengan menjalankan perangkat lunak berdasarkan skenario yang telah ditentukan dalam test case. Setelah itu, hasil keluaran dari sistem dibandingkan dengan output yang diharapkan dari setiap test case tersebut (Lian et al., 2020) . Pengujian *end-to-end* dilakukan untuk memastikan bahwa semua fungsi dalam kontrak *Approval Contract* bekerja dengan benar.

Dalam penelitian ini, skenario yang digunakan adalah proses pembelian server IBM *(International business machines corporation)*. Proses dimulai ketika budgeting control staff menginput harga sekitar Rp 150.000.000 untuk pembelian *IBM Power System S822LC Rack Server*. Setelah harga diinput dan realisasi telah berhasil diajukan, kepala departemen melakukan pengecekan dan menyetujui pengajuan tersebut. Proses persetujuan ini berlanjut melalui beberapa tahap, termasuk persetujuan dari berbagai pihak yang terlibat, hingga seluruh pengajuan realisasi disetujui sepenuhnya.

Berikut adalah hasil pengujian yang dicatat dalam Tabel 3:

Tabel 3. Hasil *End-to-End Testing*

Langkah	**Status**
Deployment	Berhasil
Register Approvers	Berhasil
Approve by Budgeting Controller Senior Manager	Berhasil
Approve by Budgering Controller Vice Precident	Berhasil
Approve by Senior Accountant	Berhasil
Approve by Accounting Manager	Berhasil
Approve by Director of Finance	Berhasil

Tabel di atas menunjukkan bahwa semua langkah pengujian menggunakan skenario yang sebenarnya ketika terdapat pengajuan realisasi berhasil dijalankan tanpa kesalahan. Hal ini berarti bahwa *smart contract ApprovalContract* bekerja sesuai dengan yang diharapkan.

Peneltian yang dilakukan oleh Asma Khatoon yang menimplementasikan penerapan teknologi *blockchain* dalam bidang kesehatan dapat membantu meningkatkan auditabilitas dengan cara menyimpan data secara permanen (K., 2020). Penelitian dari Ronald Simanjuntak yang membangun aplikasi OPEX untuk memudahkan memonitoring dan memanajemen biaya operasional perusahaan (Ronald Simanjuntak & Nahdi, 2020). Penelitian oleh Ahmadisheykhsarmast yang menggunakan *blockchain Etherum* terbukti meningkatkan efisiensi dan meningkatkan transparansi dalam setiap transaksi (Ahmadisheykhsarmast & Sonmez, 2020). Dari penelitian-penelitian sebelumnya, peneliti menggabungkannya dan dikembangkan menjadi aplikasi. Berdasarkan penelitian-penelitian sebelumnya, maka peneliti mengembangkan aplikasi yang menggabungkan teknologi *blockchain Ethereum* untuk meningkatkan auditabilitas data perusahaan, monitoring dan manajemen biaya operasional (OPEX), serta efisiensi dan transparansi transaksi.

KESIMPULAN

Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem persetujuan realisasi anggaran berbasis *blockchain* menggunakan metode *waterfall* dan *smart contract*. Setiap tahap pengembangan, mulai dari analisis kebutuhan hingga pengujian, dilakukan secara sistematis, menghasilkan sistem berkualitas tinggi yang memenuhi spesifikasi pengguna. *Smart contract* memastikan transparansi, keamanan, dan efisiensi proses persetujuan, mengurangi kesalahan manusia dan mempercepat transaksi. Pengujian yang dilakukan meliputi *unit testing*, pengujian *gas*, dan *end-to-end testing*. Dalam pengujiannya menunjukkan bahwa seluruh fungsi berjalan baik tanpa *bug*, menunjukkan biaya transaksi rata-rata sebesar 106.375,3 unit gas atau Rp 139.129 untuk setiap transaksi yang berhasil dieksekusi di jaringan *Ethereum*, dengan pengujian dilakukan pada rentang waktu jam kerja kantor untuk mencerminkan kondisi penggunaan jaringan yang sebenarnya. Integrasi *blockchain* memberikan keandalan dan integritas tinggi pada data transaksi, memudahkan proses *auditing*, dan memastikan keaslian data. Hasil penelitian ini menunjukkan bahwa integrasi metode *waterfall* dan *smart contract* dapat menciptakan sistem persetujuan anggaran yang efisien, transparan, dan aman, serta memberikan dasar untuk pengembangan lebih lanjut dalam aplikasi bisnis lainnya.

DAFTAR PUSTAKA

Ahmadisheykhsarmast, S., & Sonmez, R. (2020). A smart contract system for security of payment of construction contracts. *Automation in Construction*, *120*, 103401. https://doi.org/https://doi.org/10.1016/j.autcon.2020.103401

Al-Saqqa, S., Sawalha, S., & Abdelnabi, H. (2020). Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, *14*(11), 246-270. https://doi.org/10.3991/ijim.v14i11.13269

Chaudhuri, A. B. (2020). *Flowchart and algorithm basics: The art of programming*. Mercury Learning and Information.

Cholifah, W., Yulianingsih, Y., & Sagita, S. (2018). Pengujian Black Box Testing pada Aplikasi Action & Strategy Berbasis Android dengan Teknologi Phonegap. *STRING (Satuan Tulisan Riset Dan Inovasi Teknologi)*, *3*(2), 206-210. https://doi.org/10.30998/string.v3i2.3048

Fan, S., Zhang, H., Zeng, Y., & Cai, W. (2020). Hybrid Blockchain-Based Resource Trading System for Federated Learning in Edge Computing. *IEEE Internet of Things Journal*, *XXX*, 2327-4662. https://doi.org/10.1109/JIOT.2020.3028101

Hasibuan, A. N., & Dirgahayu, T. (2021). Pengujian dengan Unit Testing dan Test case pada Proyek Pengembangan Modul Manajemen Pengguna. *AUTOMATA*, *2*(1).

Khatoon, A. (2020). A blockchain-based smart contract system for healthcare management. Electronics, 9(1), 94-116. https://doi.org/10.3390/electronics9010094

Khan, S. N., Loukil, F., Ghedira-Guegan, C., Benkhelifa, E., & Bani-Hani, A. (2021). Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-Peer Networking and Applications*, *14*(5), 2901-2925. https://doi.org/10.1007/s12083-021-01127-0

Kurniawan, T. B., & Syarifuddin. (2020). Perancangan Sistem Aplikasi Pemesanan Makanan Dan Minuman Pada Cafetaria No Caffe Di Tanjung Balai Karimun Menggunakan Bahasa Pemograman PHP dan MYSQL. *Jurnal Tikar*, *1*(2).

Min, J. L., Istiqomah, A., & Rahmani, A. (2020). Evaluasi Penggunaan Manual Dan Automated Software Testing Pada Pelaksanaan End-To-End Testing. *JTT (Jurnal Teknol. Ter*, *6*(1), 18-25.

Luthfiyyah, Z., & Dewayanto, T. (2023). Implikasi Blockchain Pada kecurangan Akuntansi: Telaah Literatur Sistematis (SLR). *Diponegoro Journal of Accounting*, *12*(4), 1-15.

Mailasari, M. (2019). Sistem informasi perpustakaan menggunakan metode waterfall. *Jurnal Sisfokom (Sistem Informasi Dan Komputer)*, *8*(2), 207-214. https://doi.org/10.32736/sisfokom.v8i2.657

Nurdiana, D., Aprijani, D. A., Amastini, F., Maulana, M. R., & Utama, M. R. P. A. (2024). Pengembangan Sistem Informasi Pengelolaan Pembimbing Lapangan Praktik Kerja Lapangan (PKL) Prodi Sistem Informasi Universitas Terbuka. *Decode: Jurnal Pendidikan Teknologi Informasi*, *4*(2), 418-436. https://doi.org/10.51454/decode.v4i2.433

Pierro, G. A., & Rocha, H. (2019). The Influence Factors on Ethereum Transaction Fees. *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, 24-31. https://doi.org/10.1109/WETSEB.2019.00010

Rahardja, U. (2022). Skema Catatan Kesehatan menggunakan Teknologi Blockchain dalam Pendidikan. *Jurnal MENTARI: Manajemen, Pendidikan Dan Teknologi Informasi*, *1*(1), 29-37. https://doi.org/10.33050/mentari.v1i1.134

Simanjuntak, M. A. R., & Nahdi, M. (2020, July). Benefits of the Opex Pro Application in Online Project Monitoring and Evaluation at PT. XYZ. In *IOP Conference Series: Materials Science and*

Engineering (Vol. 852, No. 1, p. 012015). IOP Publishing. https://doi.org/10.1088/1757-899X/852/1/012015

Suherni, P. (2023). Aplikasi Sistem Informasi Transaksi Pelayanan Obat Diapotek Menggunakan Metode Waterfall. *Jurnal SANTI - Sistem Informasi Dan Teknik Informasi*, *1*, 23-31. https://doi.org/10.58794/santi.v1i2.323

Sunarya, P. A. (2022). *Penerapan Sertifikat pada Sistem Keamanan menggunakan Teknologi Blockchain*. *1*(1), 58-67.

Rahmayani, M. T. I., Andriani, F., Utami, D., & Purbolingga, Y. (2024). Penerapan Metode SDLC dalam Rancang Bangun Sistem Informasi Koperasi UED-SP Berbasis Website. *Innovative: Journal Of Social Science Research*, *4*(3), 17325-17343.

Watini, S., Aini, Q., Rahardja, U., Santoso, N. P. L., & Apriliasari, D. (2022). Class DojoLMS in the Interactive Learning of PAUD Educators in the Disruption Era 4.0. *Journal of Innovation in Educational and Cultural Research*, *3*(2), 215-225. https://doi.org/10.46843/jiecr.v3i2.90

Wijaya, W. W. W., & Susanto, E. (2021). New Normal: Pengembangan Sistem Informasi Penjualan Menggunakan Metode SDLC (System Development Life Cycle). *Jurnal Sustainable: Jurnal Hasil Penelitian Dan Industri Terapan*, *10*(1), 1-9. https://doi.org/10.31629/sustainable.v10i1.3190