PENGEMBANGAN APLIKASI TERDESENTRALISASI UNTUK SERTIFIKASI KOMPETENSI BERBASIS BLOCKCHAIN DAN IPFS

Disusun Oleh:

N a m a : Muhammad Risto Abrar

NIM : 20523102

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2025

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN APLIKASI TERDESENTRALISASI UNTUK SERTIFIKASI KOMPETENSI BERBASIS BLOCKCHAIN DAN IPFS

TUGAS AKHIR

Disusun Oleh:

N a m a : Muhammad Risto Abrar
NIM : 20523102

Yogyakarta, 5 November 2025

Pembimbing,

(Dr. Raden Teduh Dirgahayu, S.T, M.Sc)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN APLIKASI TERDESENTRALISASI UNTUK SERTIFIKASI KOMPETENSI BERBASIS BLOCKCHAIN DAN IPFS

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 24 November 2025

Tim Penguji

Dr. Raden Teduh Dirgahayu, S.T., M.Sc. _____

Anggota 1

Dr. Syarif Hidayat, S.Kom., M.I.T. _____

Anggota 2

Arrie Kurniawardhani, S.Si., M.Kom. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Muhammad Risto Abrar

NIM : 20523102

Tugas akhir dengan judul:

PENGEMBANGAN APLIKASI TERDESENTRALISASI UNTUK SERTIFIKASI KOMPETENSI BERBASIS BLOCKCHAIN DAN IPFS

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 13 November 2025

(Muhammad Risto Abrar)

HALAMAN PERSEMBAHAN

Al-hamdu lillahi rabbil 'alamin. Dengan rasa syukur yang tulus kepada Allah SWT. Atas limpahan rahmat, taufik, dan hidayah-Nya sehingga penulis diberi kesehatan, kesempatan, dan kekuatan untuk menyelesaikan tugas akhir ini.

Dengan penuh syukur dan hormat, karya ini penulis persembahkan kepada:

1. Keluarga tercinta

   Terima kasih yang sedalam-dalamnya kepada Ayah dan Ibu serta seluruh keluarga besar atas cinta, doa, dan dukungan tanpa henti. Nasihat, teladan kerja keras, dan kesabaran yang kalian tunjukkan menjadi sumber kekuatan sekaligus arah langkah penulis. Setiap capaian dalam tugas akhir ini tidak mungkin terwujud tanpa bimbingan, pengorbanan, dan kehadiran kalian di setiap proses yang penulis jalani.

2. Rekan-rekan seperjuangan

   Untuk sahabat dan kawan belajar yang senantiasa berbagi waktu, pengetahuan, dan ruang diskusi yang jujur dan kritis. Terima kasih atas semangat kolaborasi, bantuan teknis, dan motivasi yang menguatkan ketika penulis menghadapi kebuntuan. Catatan, masukan, dan kritik kalian telah memperkaya cara pandang penulis dalam menuntaskan penelitian ini.

3. Diri sendiri

   Sebagai pengingat bahwa proses yang konsisten, integritas dalam berkarya, dan kemauan untuk terus belajar adalah jalan yang harus dijaga. Semoga karya ini menjadi pijakan untuk melangkah lebih jauh, tetap rendah hati, dan terus bermanfaat bagi sesama.

Semoga Allah SWT. meridhai dan menjadikan karya ini membawa kebaikan serta kebermanfaatan. Aamiin.

HALAMAN MOTO

"Spend each day trying to be a little wiser than you were when you woke up."

Charlie Munger

"Success is a lousy teacher. It seduces smart people into thinking they can't lose."

Bill Gates

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Puji syukur penulis panjatkan ke hadirat Allah SWT. atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir berjudul "…". Semoga shalawat dan salam selalu tercurah kepada Nabi Muhammad SAW, yang telah membimbing umatnya ke jalan kebenaran dan menjadi rahmat bagi seluruh alam.

Penulisan laporan ini dilakukann untuk memenuhi salah satu syarat guba memperoleh gelar Sarjana Komputer pada Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Selain itu, penulis bertujuan memenuhi kebutuhan akademik dan memperluas khazanah pengetahuan di bidang informatika.

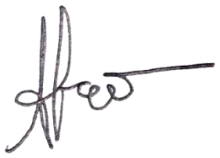
Pada kesempatan ini penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT. atas rahmat, taufik, dan hidayah-Nya sehingga penelitian dan penulisan ini dapat terselesaikan.
2. Keluarga, khususnya Ayah, Ibu, dan seluruh keluarga besar, atas doa, kasih sayang, dan dukungan tanpa henti.
3. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku dosen pembimbing, atas bimbingan, arahan, dan waktu yang dicurahkan sepanjang proses penelitian dan penulisan.
4. Bapak/Ibu Dosen Jurusan Informatika, atas ilmu, teladan, dan dukungan akademik yang sangat berarti.
5. Teman-teman dan sahabat penulis, atas semangat, kerja sama, diskusi, serta bantuan teknis selama penyusunan skripsi.
6. Semua pihak yang tidak dapat disebutkan satu per satu, atas bantuan, dukungan, dan bahkan kebaikan yang turut melancarkan penelitian ini.

Akhir kata, penulis berharap skripsi ini dapat memberikan manfaat bagi pengembangan ilmu pengetahuan dan bagi almamater Universitas Islam Indonesia serta pihak-pihak terkait.

Wassalamu'alaikum Warahmatullahi Wabarakatuh

Yogyakarta, 13 November 2025



(Muhammad Risto Abrar)

SARI

Pengelolaan sertifikat kompetensi pada Lembaga Sertifikasi Profesi (LSP) masih didominasi proses manual yang menimbulkan keterlambatan, proses verifikasi yang tidak efisien, dan kerentanan pemalsuan sertifikat. Penelitian ini bertujuan mengembangkan aplikasi terdesentralisasi (DApp) untuk manajemen sertifikasi yang aman, transparan, dan dapat diverifikasi publik.

Metode yang digunakan mengikuti alur rekayasa perangkat lunak linear: analisis kebutuhan, perancangan arsitektur, implementasi, dan pengujian. Solusi dirancang dengan *smart contract* modular pada lingkungan pengujian Hardhat. Dokumen disimpan secara *off-chain* pada *InterPlanetary File System* (IPFS) dalam keadaan dienkripsi dan diverifikasi melalui *Content Identifier* (CID). Autentikasi menggunakan *wallet address*, tanpa basis data terpusat.

Pengujian *black-box* pada skenario inti verifikasi LSP oleh Badan Nasional Sertifikasi Profesi (BNSP), pendaftaran peserta, penilaian dan penetapan kelulusan oleh LSP, penerbitan sertifikat, serta verifikasi publik berbasis CID menunjukkan seluruh fungsi berjalan sesuai spesifikasi pada lingkungan pengujian Hardhat. Hasil tersebut mengindikasikan bahwa pendekatan terdesentralisasi ini layak untuk meningkatkan integritas data, memudahkan proses verifikasi sertifikat, dan mengurangi ketergantungan pada otoritas pusat.

Kata kunci: Aplikasi Terdesentralisasi, Blockchain, IPFS, Sertifikasi Kompetensi, *Smart Contract*, *Wallet*.

GLOSARIUM

Glosarium memuat daftar kata tertentu yang digunakan dalam laporan dan membutuhkan penjelasan, misalnya kata serapan yang belum lazim digunakan. Urutkan sesuai abjad. Contoh penulisannya seperti di bawah ini:

Compile	proses untuk mengubah berkas kode program dengan berkas lain yang terkait menjadi berkas yang siap untuk dieksekusi oleh sistem operasi secara langsung.
Debug	langkah untuk menelusuri kesalahan kode program.
Waterfall	metode pengembangan perangkat lunak.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Sertifikasi kompetensi merupakan proses sistematis dan objektif untuk menilai serta membuktikan kemampuan individu terhadap standar kompetensi tertentu. Di Indonesia, sertifikasi kompetensi dilakukan antara lain oleh Lembaga Sertifikasi Profesi (LSP) yang berlisensi Badan Nasional Sertifikasi Profesi (BNSP) dengan mengacu pada Standar Kompetensi Kerja Nasional Indonesia (SKKNI) (Sunarya, et al., 2020). Uji kompetensi ini telah diatur dalam Peraturan Pemerintah Republik Indonesia No. 10 Tahun 2018. Dalam regulasi ini, pemerintah mengatur bahwa setiap proses sertifikasi harus objektif, transparan, dan mengacu pada standar yang diakui baik secara nasional maupun internasional, guna memastikan validitas dan keandalan sertifikasi yang dikeluarkan. Sertifikasi kompetensi ini penting dilakukan untuk memberikan legitimasi atas kemampuan profesional individu serta meningkatkan kredibilitas tenaga kerja. Hal ini diperlukan terutama di industri yang sangat bergantung pada kompetensi tenaga kerja, seperti teknologi informasi, manajemen risiko, dan sektor lainnya.

Sertifikasi kompetensi penting untuk meningkatkan kredibilitas tenaga kerja, namun praktik di lapangan masih menghadapi sejumlah kendala, khususnya pada manajemen data dan penjaminan keaslian dokumen. Studi pendahuluan penulis pada salah satu LSP perguruan tinggi (LSP-X) menunjukkan proses pengelolaan sertifikat masih dilakukan secara manual dan berbasis dokumen fisik/digital sederhana, sehingga berkas rentan hilang, proses verifikasi memakan waktu, dan administrasi menjadi tidak efisien. Secara lebih luas, literatur juga menyoroti bahwa sistem sertifikat tradisional memiliki kelemahan pada aspek autentikasi, pelacakan, dan koreksi data (Rahman, 2023). Di saat yang sama, polemik publik terkait tuduhan pemalsuan ijazah di Indonesia menggambarkan sensitifnya isu keaslian dokumen pendidikan serta urgensi mekanisme verifikasi yang andal dan transparan (CNN Indonesia, 2025).

Menjawab persoalan tersebut, penelitian ini mengusulkan pemanfaatan blockchain sebagai buku besar (*ledger*) terdistribusi yang transparan, tahan manipulasi, dan dapat diaudit, sehingga jejak penerbitan maupun perubahan status sertifikat tercatat secara permanen dan dapat diverifikasi publik tanpa otoritas terpusat (Nakamoto, 2008; LaFountain, 2021; Chander,

2023; Saja, et al., 2023). Pada ekosistem Ethereum, aturan layanan dikodekan ke dalam *smart contract* yang dieksekusi secara otomatis ketika syarat terpenuhi, menghilangkan perantara dan meminimalkan risiko *human error* (Kushwaha, et al., 2022). Aplikasi yang memanfaatkan *smart contract* di atas jaringan *peer-to-peer* ini lazim disebut aplikasi terdesentralisasi (DApp) (Chander, 2023). Untuk penanganan berkas berukuran besar, *InterPlanetary File System* (IPFS) menyediakan mekanisme penyimpanan di luar blockchain *(off-chain)* yang tersebar dan berbasis *content addressing* melalui *Content Identifier* (CID), sehingga berkas dapat didistribusikan dan diambil kembali dengan efisien sembari menjaga integritas (Doan, et al, 2022; Pincheira, et al., 2022). Dalam rancangan ini, dokumen sertifikat disimpan dalam keadaan terenkripsi di IPFS, sedangkan CID dan metadata ditempatkan di dalam blockchain (*on-chain*) untuk keperluan pembuktian dan audit. Alur tersebut memungkinkan validasi keaslian serta ketertelusuran sertifikat secara terbuka tanpa bergantung pada satu otoritas/server pusat.

1.2 Rumusan Masalah

Berdasarkan Latar Belakang yang telah diuraikan, maka penulis merumuskan masalah yang akan dibahas sebagai berikut:

1. Bagaimana rancangan dan implementasi aplikasi terdesentralisasi (DApp) berbasis blockchain untuk mendukung proses sertifikasi kompetensi?
2. Bagaimana rancangan mekanissme verifikasi keaslian sertifikat yang dapat diakses oleh publik menggunakan *Content Identifier* (CID) dengan mengurangi ketergantungan pada otoritas pusat serta tanpa membuka data sensitif, sekaligus menjaga integritas dan kerahasiaan dokumen?

1.3 Batasan Masalah

Dalam penelitian ini agar pengembangan DApp sertifikasi kompetensi terselesaikan sesuai tujuan yang diharapkan, maka ditetapkan batasan masalah sebagai berikut:

1. Studi kasus hanya dibatasi pada satu Lembaga Sertifikasi Profesi (LSP) di lingkungan perguruan tinggi.
2. DApp dibangun berbasis Ethereum dan pengujian dilakukan pada jaringan lokal Hardhat. Tidak dilakukan pengujian di testnet maupun mainnet.
3. Fokus pada pendaftaran peserta, input nilai, penetapan lulus/tidak lulus, penerbitan sertifikat, dan verifikasi sertifikat oleh publik.

4. Evaluasi dibatasi hanya pada fungsionalitas, keamanan, dan efisiensi proses.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah merancang dan mengimplementasikan aplikasi terdesentralisasi (DApp) berbasis teknologi blockchain yang digunakan untuk mendukung proses sertifikasi kompetensi mulai dari pendaftaran peserta, penilaian peserta, penetapan kelulusan peserta, hingga penerbitan sertifikat. Selain itu, penelitian ini menghadirkan mekanisme verifikasi keaslian sertifikat yang dapat diakses publik tanpa otoritas pusat, sehingga proses validasi berlangsung lebih cepat dan efisien.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat bagi beberapa pihak berikut:

1. Lembaga Sertifikasi Profesi (LSP) dapat mempercepat alur sertifikasi, menurunkan beban verifikasi manual melalui verifikasi publik, serta meningkatkan integritas.
2. Peserta dapat memperoleh sertifikat digital yang mudah dibuktikan keasliannya, memantau status proses secara transparan, serta menjaga privasi karena dokumen disimpan dalam keadaan terenkripsi.
3. Industri dan perusahaan dapat memverifikasi keaslian sertifikat secara mandiri dan cepat dan memudahkan pemenuhan persyaratan audit berkat pencatatan yang transparan dan dapat diverifikasi secara independen di blockchain.

1.6 Sistematika Laporan

Sistematika laporan ini disusun agar alur pembahasan runtut dari perumusan masalah hingga penarikan kesimpulan, sebagai berikut:

1. BAB 1 Pendahuluan

   Menguraikan konteks penelitian dan alasan pemilihan topik, meliputi Latar Belakang, Rumusan Masalah, Batasan Masalah, Tujuan Penelitian, Manfaat Penelitian, serta Sistematika Laporan.
2. BAB 2 Kajian Pustaka

   Menyajikan landasan teori yang relevan dengan topik, mencakup konsep Sertifikasi Kompetensi, Blockchain, *Smart Contract*, Aplikasi Terdesentralisasi, dan IPFS, diikuti telaah Penelitian Terdahulu beserta ringkasan temuan kunci, kesenjangan riset, dan posisi kontribusi penelitian ini.

3. BAB 3 Metodologi Penelitian

   Menjelaskan pendekatan pengembangan perangkat lunak yang digunakan beserta tahapan kerjanya. Bagian ini mencakup Analisis Kebutuhan (aktor, *use case*, dan tabel rincian *use case*), Perancangan Sistem (arsitektur aplikasi terdesentralisasi, tiga BPMN untuk pendaftaran LSP, proses sertifikasi, dan verifikasi publik, serta diagram kelas *smart contract*), Lingkungan dan Alat Pengembangan yang digunakan, serta Skenario Pengujian yang menjadi acuan evaluasi pada bab berikutnya.

4. BAB 4 Implementasi dan Pembahasan

   Memaparkan realisasi rancangan ke dalam kode dan antarmuka, serta hasil pengujian. Isi bab meliputi pemetaan fungsi dan potongan kode *smart contract*, hasil *black-box testing* sesuai skenario pada BAB 3, ulasan antarmuka pengguna per peran beserta validasi dan konfirmasi transaksi, rekaman *on-chain* dan jejak audit, serta integrasi IPFS untuk penyimpanan terenkripsi berbasis CID.

5. BAB 5 Kesimpulan dan Saran

   Merangkum jawaban atas rumusan masalah berdasarkan hasil implementasi dan pengujian, beserta saran pengembangan dan arah penelitian lanjutan.

BAB II
KAJIAN PUSTAKA

2.1 Landasan Teori

Pada konteks pengembangan aplikasi terdesentralisasi untuk manajemen sertifikasi kompetensi, landasan teori mencakup beberapa pokok bahasan meliputi konsep sertifikasi kompetensi dan tata kelola LSP dan BNSP sebagai domain data dan proses, teknologi blockchain sebagai infrastruktur pencatatan terdistribusi yang tahan perubahan dan dapat diaudit, *smart contract* sebagai logika otomatisasi transaksi dan verifikasi, *InterPlanetary File System* (IPFS) sebagai mekanisme penyimpanan berkas secara terdistribusi, serta *decentralized application* (DApp) sebagai bentuk integrasi antarmuka pengguna dengan layanan *on-chain* dan *off-chain*. Uraian rinci masing-masing teori disajikan pada sub-subbab berikutnya sebagai dasar untuk analisis kebutuhan, perancangan, dan evaluasi sistem.

2.1.1 Sertifikasi Kompetensi

Sertifikasi kompetensi merupakan proses sistematis dan objektif untuk menilai serta membuktikan kemampuan individu terhadap standar kompetensi yang diakui. Di Indonesia, pelaksanaan ssertifikasi dilakukan oleh Lembaga Sertifikasi Profesi (LSP) yang berlisensi Badan Nasional Sertifikasi Profesi (BNSP) melalui uji kompetensi yang mengacu pada Standar Kompetensi Kerja Nasional Indonesia (SKKNI) sesuai ketentuan Peraturan Pemerintah Republik Indonesia Nomor 10 Tahun 2018 tentang BNSP, yang menekankan asas objektivitas, transparansi, akuntabilitas, dan kesesuaian dengan standar dalam setiap tahapan sertifikasi untuk menjamin validitas serta keandalan sertifikat yang diterbitkan (Republik Indonesia, 2018). Dalam ekosistem tersebut, BNSP berperan sebagai otoritas lisensi dan penetapan kebijakan, sedangkan LSP sebagai penyelenggara asesmen kompetensi melalui asesor yang berwenang, dengan peserta ssebagai ssubjek penilaian.

Secara fungsional, sertifikasi kompetensi tidak hanya menjadi pengakuan formal atas kemampuan individu, tetapi juga berfungsi sebagai instrumen penjamin mutu yang menghubungkan capaian pembelajaran atau keahlian dengan kebutuhan dunia kerja. Bagi pemangku kepentingan termasuk institusi pendidikan, LSP, dan industri. Sertifikasi menghadirkan keterukuran kompetensi, memudahkan proses verifikasi oleh pengguna lulusan, serta mendukung pengembangan karier dan kredibilitas tenaga kerja pada berbagai sektor

sseperti teknologi informassi maupun manajemen risiko. Dengan demikian, sertifikasi kompetensi menempati posisi strategis dalam ekosistem peningkatan kualitass sumber daya manusia, sekaligus mendorong keterhubungan antara standar kompetensi dan kebutuhan industri (Sunarya, et al., 2020).

2.1.2 Blockchain

Blockchain pertama kali diperkenalkan melalui *white paper* Bitcoin oleh Satoshi Nakamoto (2008) sebagai *ledger* terdistribusi yang memungkinkan transaksi *peer-to-peer* tanpa perantara terpusat. Dalam praktiknya, blockchain beroperasi sebagai basis data terdistribusi di antara banyak *node* yang saling mereplikasi dan memvalidasi catatan. Data dicatat dalam blok yang dihubungkan secara kriptografis melalui *hash* blok sebelumnya sehingga membentuk rantai yang bersifat *append-only* dan tahan manipulasi. Karakteristik umum yang ditekankan dalam literatur khususnya pada blockchain publik meliputi desentralisasi, transparansi, dan auditabilitas, yang membuat perubahan data sulit dilakukan tanpa terdeteksi oleh jaringan (Nakamoto, 2008; LaFountain, 2021; Sanjay, et al., 2020).

Dari sisi keunggulan, Suryawijaya (2023) menyoroti bahwa blockchain menawarkan keamanan melalui mekanisme kriptografi dan replikasi data, integritas/immutability catatan, serta verifikasi independen oleh pihak terkait tanpa ketergantungan pada satu otoritas. Pada konteks tertentu, hal ini dapat meningkatkan efisiensi proses pemeriksaan bukti. Di sisi lain, terdapat keterbatasan seperti skalabilitas/latensi dan biaya, kompleksitas operasional, serta isu privasi (karena banyak jaringan bersifat terbuka) yang perlu diatasi pada level arsitektur dan aplikasi. Enkripsi konten dapat diterapkan di lapisan aplikasi bila diperlukan, sedangkan ledger menyediakan bukti integritas dan jejak audit yang tidak mudah diubah (Suryawijaya, 2023; LaFountain, 2021).

Blockchain terdiri atas beberapa komponen teknis yang saling terintegrasi untuk menjaga reliabilitas data:

a. Blok: Wadah data yang berisi sekumpulan transaksi, penanda waktu (*timestamp*), dan hash dari blok sebelumnya, sehingga membentuk rantai berurutan secara kronologis.
b. *Hash*: Nilai kriptografi unik yang dihitung dari isi blok dan bertindak sebagai identitas digital untuk memastikan integritas data.
c. *Node*: Komputer atau entitas dalam jaringan yang memelihara salinan lengkap buku besar serta ikut serta dalam memverifikasi transaksi.

d. Protokol konsensus: Aturan untuk menyelaraskan keputusan antar *node* mengenai keabsahan blok baru, misalnya *Proof of Work* (PoW) dan *Proof of Stake* (PoS) (Nguyen, 2016).

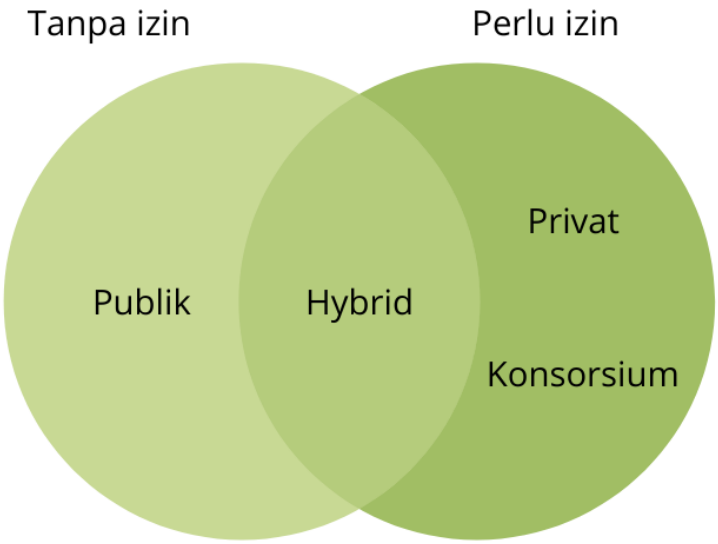
Mekanisme kerja blockchain dimulai saat pengguna mengirimkan transaksi. Transaksi tersebut kemudian diverifikasi oleh sejumlah *node* melalui protokol konsensus. Setelah transaksi dinyatakan sah, transaksi dimasukkan ke dalam blok baru yang dihubungkan menggunakan kriptografis (*hash*) dengan blok sebelumnya. Blok tersebut kemudian di sebarkan ke seluruh jaringan agar setiap *node* memperbarui salinan buku besar terdistribusi secara serentak (Narayanan et al., 2016).

Jenis jaringan blockchain umumnya dibedakan menjadi blockchain publik, privat, dan konsorsium. Blockchain publik bersifat terbuka, siapa pun dapat berpartisipasi sebagai *node* dan memverifikasi transaksi sehingga transparansi dan ketahanan terhadap penyensoran menjadi ciri kuatnya. Blockchain privat membatasi partisipasi pada entitas yang diotorisasi, menekankan kontrol akses dan kepatuhan pada konteks korporasi atau organisasi. Blockchain konsorsium dikelola bersama oleh sejumlah organisasi, menempatkan model kepercayaan di antara para anggota dengan derajat desentralisasi di antara publik dan privat. (Buterin, 2015; Meth, 2019;). Selain tiga kategori tersebut, terdapat pula blockchain *hybrid* yang menggabungkan sifat publik dan privat, yakni sebagian data atau fungsi dibuat terbuka untuk kebutuhan auditabilitas sementara operasi yang sensitif tetap berjalan pada domain permissioned, sehingga memperoleh keseimbangan antara transparansi dan kontrol. (Damre et al., 2022). Ringkasan perbandingan keempat blockchain tersebut dapat dilihat pada Tabel 2.1, sedangkan visualisasi posisi relatif ditunjukkan pada Gambar 2.1

Tabel 2.1 Ringkasan Perbandingan Empat Jenis Blockchain

Jenis	Akses	Pengelolaan	Kelebihan	Kekurangan
Publik	Terbuka untuk umum.	Desentralisasi penuh.	Transparan dan kepercayaan tinggi.	Lambat dan boros energi.
Privat	Tertutup (satu organisasi).	Sentralisasi oleh otoritas pusat.	Cepat dan efisien.	Sentralisasi dan rawan manipulasi.
Konsorsium	Tertutup (antar organisasi).	Kolaborasi multi entitas.	Seimbang antara privasi dengan desentralisasi.	Transparansi terbatas dan koordinasi rumit.

Hybrid	Kombinasi publik dan privat.	Otoritas pusat dengan validasi publik.	Aman, fleksibel, dan terkontrol.	Implementasi kompleks dan perlu tata kelola ketat.



Gambar 2.1 Diagram Venn Jenis Jaringan Blockchain

2.1.3 *Smart Contract*

Sejalan dengan uraian tentang blockchain sebagai *ledger* terdistribusi, *smart contract* merupakan program yang disimpan di jaringan blockchain dan dieksekusi secara otomatis ketika kondisi yang ditentukan terpenuhi. Dengan mekanisme tersebut, pelaksanaan aturan atau kesepakatan tidak lagi bergantung pada otoritas pusat, tetapi dijalankan secara deterministik oleh jaringan itu sendiri. Menurut Pinna et al. (2019), *smart contract* berfungsi sebagai lapisan logika yang mengatur perilaku sistem, di mana setiap instruksi terekam dan direplikasi pada seluruh node sehingga menjamin konsistensi hasil di seluruh jaringan. Dalam konteks Ethereum, kontrak pintar memformalkan kesepakatan atau aturan layanan ke dalam bentuk kode program, dengan status dan hasil eksekusinya tercatat secara permanen di blockchain, menciptakan lingkungan yang bersifat *trustless* dan terverifikasi secara publik (Stark, 2016; Rawat et al., 2019).

Konsep dasar *smart contract* pertama kali dikemukakan oleh Nick Szabo (1997) sebagai serangkaian perjanjian digital yang meniru prinsip hukum konvensional, namun diimplementasikan secara otomatis melalui protokol komputer. Szabo berpendapat bahwa digitalisasi kontrak dapat mengurangi ketergantungan terhadap perantara dan meminimalkan

potensi pelanggaran kesepakatan karena pelaksanaannya dikontrol oleh logika kode, bukan interpretasi manusia. Dua dekade kemudian, Vitalik Buterin (2014) mengembangkan konsep ini dalam platform Ethereum, yang memperluas fungsi smart contract menjadi lebih dinamis dan dapat digunakan untuk membangun aplikasi terdesentralisasi (DApp). Sejak itu, *smart contract* tidak hanya digunakan untuk transaksi finansial, tetapi juga untuk mengelola data, aset digital, serta berbagai proses bisnis yang memerlukan transparansi dan akuntabilitas tinggi.

Secara teknis, *smart contract* memiliki dua komponen utama, yaitu variabel status (*state variables*) dan nilai status (*state value*). Variabel status digunakan untuk menyimpan kondisi sistem, sementara nilai status berisi logika yang menentukan bagaimana status tersebut berubah ketika kontrak dieksekusi. Wu et al. (2022) menjelaskan bahwa eksekusi *smart contract* umumnya berbasis pada logika *if–then* dan *event-driven*, di mana suatu tindakan akan dijalankan ketika kondisi tertentu terpenuhi. Ketika pengguna mengirim transaksi yang memanggil suatu fungsi dalam kontrak, transaksi tersebut diverifikasi oleh *node* jaringan melalui algoritma konsensus. Setelah diverifikasi, hasil eksekusi disimpan secara permanen dalam blok baru yang ditambahkan ke blockchain. Dengan demikian, setiap perubahan keadaan (*state transition*) bersifat transparan, dapat dilacak, dan tidak dapat dimodifikasi secara sepihak.

Selain struktur dan mekanisme eksekusinya, *smart contract* memiliki karakteristik fundamental yang membedakannya dari sistem digital konvensional (Rawat et al., 2019; Nadler Prata et al., 2021):

a. *Immutability* (Tidak Dapat Diubah)

   Setelah kontrak di-*deploy* ke blockchain, kode dan ketentuannya tidak dapat diubah tanpa penerapan ulang versi baru. Hal ini menjamin integritas aturan dan mencegah manipulasi setelah penerapan, namun juga menuntut praktik pengembangan yang hati-hati karena kesalahan kecil dapat bersifat permanen.

b. *Permanence* (Kekekalan Data)

   Semua data dan logika yang terdapat dalam kontrak disimpan secara permanen di blockchain. Kontrak yang telah di-*deploy* tetap tersedia dan dapat diakses untuk audit kapan pun, menjadikannya mekanisme pencatatan yang transparan dan dapat dipercaya.

c. *Automation* (Otomatisasi)

   Smart contract mengeksekusi tindakan secara otomatis tanpa perlu campur tangan manusia ketika kondisi tertentu terpenuhi. Proses ini meningkatkan efisiensi dan

mengurangi risiko kesalahan manusia, terutama dalam sistem yang memerlukan ketepatan aturan dan verifikasi berulang.

d. *Trustless Execution* (Eksekusi Tanpa Kepercayaan)

Pelaksanaan kontrak tidak bergantung pada pihak ketiga. Setiap *node* dalam jaringan menjalankan kode yang sama, sehingga hasilnya dapat dipercaya meskipun pihak-pihak yang terlibat tidak saling mengenal.

e. *Transparency and Auditability* (Transparansi dan Auditabilitas)

Seluruh transaksi dan hasil eksekusi *smart contract* tercatat secara publik di blockchain. Siapa pun dapat meninjau *ledger* untuk memverifikasi tindakan atau peristiwa tertentu, menjadikan sistem lebih terbuka dan dapat diaudit.

f. *Cost Efficiency* (Efisiensi Biaya)

Dengan meniadakan perantara dan mengotomatiskan proses bisnis, *smart contract* mampu memangkas biaya operasional dan administrasi. Ini menjadikannya solusi ideal untuk skenario transaksi digital yang berulang dan berbasis aturan tetap.

Dalam arsitektur blockchain, *smart contract* beroperasi di lapisan aplikasi (*application layer*), sementara validasi transaksi dan penyimpanan data berlangsung di lapisan dasar (*base layer*) (Kushwaha et al., 2022). Interaksi antar lapisan ini dijamin oleh mekanisme konsensus, yang memastikan bahwa setiap eksekusi kontrak memiliki hasil yang sama di seluruh *node* jaringan. Dengan demikian, *smart contract* berperan sebagai jembatan yang menghubungkan antara data di *ledger* dengan logika bisnis yang dapat dieksekusi secara otomatis.

2.1.4 *Interplanetary File System* (IPFS)

InterPlanetary File System (IPFS) merupakan sistem penyimpanan dan distribusi data terdesentralisasi berbasis *peer-to-peer* yang mengidentifikasi data berdasarkan kontennya, bukan berdasarkan lokasi penyimpanan. Pendekatan ini berbeda secara fundamental dari arsitektur web konvensional yang menggunakan HTTP (*Hypertext Transfer Protocol*), di mana sumber daya diakses melalui alamat server tertentu. IPFS menggunakan *Content Identifier* (CID), yaitu sidik jari kriptografis hasil *hashing* terhadap isi berkas. CID berfungsi sebagai identitas unik bagi setiap konten, sehingga file yang identik akan selalu memiliki CID yang sama, sedangkan perubahan sekecil apa pun pada isi file akan menghasilkan CID baru (Trautwein et al., 2022).

Ketika sebuah berkas diunggah ke IPFS, sistem akan membaginya menjadi blok-blok kecil yang kemudian disusun ke dalam struktur *Merkle Directed Acyclic Graph* (Merkle-

DAG). Setiap blok memperoleh CID-nya masing-masing, sementara keseluruhan file memiliki CID induk yang merepresentasikan akar dari struktur DAG tersebut. Dengan pendekatan ini, setiap bagian file dapat diverifikasi secara independen, dan integritas keseluruhan berkas dapat dipastikan melalui proses verifikasi *hash* berantai. Saat pengguna ingin mengambil kembali berkas tersebut, IPFS akan mencari *peer* yang menyimpan blok-blok terkait melalui mekanisme pencarian terdistribusi. Setelah ditemukan, blok-blok tersebut diunduh dari berbagai *node* yang tersedia dan digabungkan kembali menjadi satu file utuh di sisi klien.

Mekanisme ini membuat IPFS jauh lebih tahan terhadap kegagalan jaringan dibanding sistem penyimpanan terpusat. Tidak ada satu server tunggal yang menjadi sumber kebenaran data, sehingga jika satu atau beberapa *node* tidak tersedia, file tetap dapat diakses dari *node* lain yang menyimpan replika. Selain itu, karena CID merupakan *hash* dari konten, pengguna dapat memverifikasi keaslian data secara mandiri hanya dengan mencocokkan nilai *hash*-nya, tanpa memerlukan otoritas pusat atau sertifikat digital tambahan.

Menurut Trautwein et al. (2022), IPFS memiliki beberapa karakteristik utama yang menjadikannya sangat relevan untuk mendukung sistem berbasis blockchain, khususnya dalam penyimpanan data bersifat sensitif dan berukuran besar:

a. *Content-Based Addressing*

   IPFS tidak menggunakan alamat lokasi (*location-based addressing*), melainkan pengalamatan berbasis konten. Setiap file dan potongan datanya diidentifikasi dengan CID, yang dihasilkan dari fungsi *hash* kriptografis terhadap isi data tersebut. Dengan cara ini, data yang disimpan akan selalu dapat diverifikasi integritasnya karena perubahan apa pun pada konten akan mengubah CID.

b. Desentralisasi Penuh

   IPFS beroperasi di atas jaringan *peer-to-peer* yang sepenuhnya terdesentralisasi. Tidak ada entitas tunggal yang mengontrol jaringan, dan setiap *node* berperan sebagai penyimpan maupun penyedia data. Desentralisasi ini meningkatkan ketersediaan (*availability*) dan mengurangi risiko *single point of failure* yang umum terjadi pada sistem berbasis server pusat.

c. *Immutability* dan Verifikasi Mandiri

   Karena CID didasarkan pada *hash* konten, data dalam IPFS bersifat *immutable* (tidak dapat diubah) dan *self-verifiable*. Pengguna dapat memverifikasi keaslian data hanya dengan membandingkan CID-nya, tanpa perlu mempercayai penyedia data atau otoritas

eksternal. Ini menjadikan IPFS sangat sesuai untuk skenario yang menuntut integritas tinggi seperti arsip dokumen, rekam transaksi, atau sertifikat digital.

d. Partisipasi Terbuka (*Open Participation*)

IPFS bersifat *open-source* dan inklusif. Siapa pun dapat menjalankan *node* IPFS tanpa izin khusus, berpartisipasi dalam replikasi data, serta berkontribusi terhadap pengembangan protokol. Karakteristik ini menjadikan IPFS sebagai teknologi yang terbuka, kolaboratif, dan terus berkembang melalui dukungan komunitas global.

e. Ketahanan terhadap *Churn* dan Kegagalan (*Fault Tolerance*)

IPFS dirancang untuk beroperasi di lingkungan yang dinamis, di mana *node* dapat bergabung atau keluar kapan saja (*node churn*). Dengan mereplikasi data ke beberapa *node* terdekat, IPFS tetap dapat menyediakan layanan meskipun sebagian *node* offline. Kemampuan ini menjadikan IPFS solusi penyimpanan yang tahan terhadap gangguan dan kehilangan data.

Dari karakteristik tersebut, dapat disimpulkan bahwa IPFS menawarkan keunggulan dalam hal keamanan, ketersediaan, dan integritas data, terutama untuk skenario penyimpanan berbasis blockchain. Karena blockchain memiliki keterbatasan dalam menyimpan data berukuran besar akibat biaya gas dan efisiensi penyimpanan, IPFS digunakan sebagai lapisan penyimpanan *off-chain* yang melengkapi fungsi blockchain sebagai lapisan pencatatan *on-chain*. Integrasi antara kedua sistem ini memungkinkan pencatatan *hash* atau CID dari suatu berkas ke dalam blockchain sebagai bukti keaslian (*proof of authenticity*), sementara file aslinya disimpan secara terdistribusi di jaringan IPFS.

2.1.5 Aplikasi Terdesntralisasi

Setelah munculnya *smart contract* sebagai logika otomatis di blockchain, berkembanglah konsep aplikasi terdesentralisasi atau *Decentralized Application* (DApp) sebagai bentuk implementasi lanjutan dari teknologi tersebut. DApp merupakan aplikasi yang berjalan di atas jaringan blockchain dan memanfaatkan *smart contract* sebagai komponen inti untuk mengeksekusi aturan bisnis tanpa otoritas pusat. Dengan demikian, DApp berperan sebagai lapisan aplikasi (*application layer*) yang memungkinkan pengguna berinteraksi langsung dengan blockchain melalui antarmuka pengguna yang ramah (*frontend*) tanpa bergantung pada entitas perantara (FinCEN, 2019; Ethereum, n.d.).

Secara arsitektural, DApp menggabungkan tiga komponen utama, yaitu:

a. *Smart contract* yang berfungsi sebagai logika bisnis dan penyimpanan *state* di blockchain,

b. *Frontend* yang berperan sebagai antarmuka pengguna berbasis web atau aplikasi, dan
c. *Wallet* yang menjadi penghubung antara pengguna dengan blockchain.

Wallet mengelola kunci privat dan alamat publik yang digunakan untuk menandatangani transaksi, sehingga otentikasi pengguna bersifat kriptografis dan independen dari sistem login terpusat (Voshmgir, 2019). Mekanisme ini memungkinkan pengguna untuk mempertahankan kendali penuh atas identitas digital dan aset yang mereka miliki di dalam jaringan.

Menurut dokumentasi resmi Ethereum (n.d.), *smart contract* yang di-*deploy* pada jaringan bersifat terbuka dan transparan seperti open API, sehingga DApp dapat memanggil fungsi atau kontrak milik pihak lain dengan mudah. Konsep ini disebut *reusability*, di mana berbagai aplikasi dapat saling berinteraksi secara modular di atas jaringan yang sama tanpa memerlukan izin tambahan. *Reusability* inilah yang menjadikan ekosistem DApp berkembang pesat, karena pengembang dapat membangun aplikasi baru dengan memanfaatkan kontrak atau layanan yang sudah ada.

Dari sisi karakteristik, DApp memiliki sejumlah ciri utama yang membedakannya dari aplikasi konvensional:

a. Terdesentralisasi

Seluruh logika bisnis dan eksekusi aturan berjalan di jaringan blockchain yang terdiri dari banyak *node*, bukan pada server pusat. Setiap *node* menyimpan salinan kontrak dan data, sehingga sistem lebih tahan terhadap kegagalan (*fault-tolerant*) dan tidak bergantung pada satu penyedia layanan.

b. Deterministik

DApp menjamin bahwa untuk setiap masukan yang sama, hasil eksekusi akan selalu sama di seluruh *node* jaringan. Hal ini memastikan konsistensi perilaku aplikasi di berbagai lingkungan, karena semua *node* menjalankan kode yang identik.

c. Transparan dan Terverifikasi

Semua transaksi dan interaksi yang dilakukan melalui DApp tercatat secara publik pada blockchain. Setiap tindakan pengguna dapat diaudit secara terbuka, meningkatkan akuntabilitas dan mengurangi risiko manipulasi data.

d. Turing Complete

DApp mendukung logika komputasi kompleks melalui kemampuan *Turing completeness* dari *smart contract* (misalnya pada Ethereum Virtual Machine). Dengan

demikian, pengembang dapat membangun logika dinamis seperti perhitungan kondisi, manajemen data, dan pengambilan keputusan otomatis.

e. Lingkungan Eksekusi Terisolasi

DApp dijalankan di lingkungan virtual seperti Ethereum Virtual Machine (EVM) yang memisahkan eksekusi tiap kontrak. Isolasi ini memastikan bahwa kegagalan atau bug pada satu kontrak tidak memengaruhi seluruh jaringan, menjaga stabilitas dan keamanan ekosistem blockchain.

Selain karakteristik di atas, DApp juga mendukung prinsip keterbukaan kode sumber (*open source*) dan interoperabilitas, di mana pengembang dapat memeriksa, memodifikasi, atau menggunakan ulang kode kontrak yang telah dipublikasikan. Model ini menciptakan ekosistem yang kolaboratif dan mendorong inovasi karena setiap kontrak baru dapat memperluas fungsionalitas yang telah ada tanpa membangun ulang seluruh sistem.

Secara teknis, eksekusi DApp tidak bergantung pada server aplikasi tradisional. Proses penegakan aturan dan penyimpanan *state* yang relevan dilakukan di jaringan *peer-to-peer*, sementara antarmuka pengguna dapat di-*host* secara terpisah menggunakan layanan web biasa atau bahkan didistribusikan melalui sistem seperti IPFS. Pola ini memungkinkan pemisahan tanggung jawab antara lapisan tampilan dan lapisan logika bisnis. Dalam praktiknya, antarmuka DApp beroperasi layaknya aplikasi web modern yang berinteraksi dengan blockchain melalui *wallet* pengguna untuk membaca dan menulis data ke *smart contract* sesuai kebutuhan (Voshmgir, 2019).

DApp juga memperkenalkan paradigma baru dalam autentikasi dan kepemilikan identitas. Identitas pengguna tidak lagi dikelola oleh server pusat, melainkan berbasis kriptografi kunci publik. Dengan menandatangani transaksi menggunakan *private key* yang disimpan di *wallet*, pengguna dapat membuktikan kepemilikan akun tanpa mengungkapkan data pribadi. Pendekatan ini memperkuat keamanan sistem sekaligus menjaga privasi pengguna, karena tidak ada penyimpanan kredensial di sisi server yang rentan terhadap kebocoran.

2.2 Penelitian Terdahulu

Dalam penyusunan skripsi ini, peneliti menelaah berbagai karya ilmiah yang relevan sebagai landasan konseptual dan bahan pembanding.

Saleh et al. (2020) mengusulkan kerangka verifikasi sertifikat pendidikan berbasis blockchain untuk menjawab maraknya pemalsuan dan keterbatasan transparansi proses

autentikasi konvensional. Mereka memformulasikan *workflow* verifikasi dan menguji implementasi menggunakan Hyperledger Fabric sebagai infrastruktur, sehingga memperjelas bagaimana desentralisasi dapat menjadi basis "*single source of truth*" bagi pemangku kepentingan (penerbit, pemegang, pihak ketiga). Studi ini menegaskan manfaat berupa penguatan keandalan dan kepercayaan antar pihak melalui jejak audit yang tidak mudah diubah.

Rahardja et al. (2021) merancang skema autentikasi sertifikat yang "*immutable*" dan dapat diverifikasi secara mandiri (*trust-minimized*). Pendekatannya memanfaatkan kriptografi kunci publik dan hashing SHA-256; *hash*/QR disematkan pada sertifikat lalu "di-*anchor*" ke blockchain publik Vexanium, dengan proses verifikasi melalui situs resmi dan tautan ke block explorer untuk pembuktian silang. Hasilnya menunjukkan alur validasi yang jelas (menu Verify) dan karakter sertifikat yang *immutable-authenticated-ubiquitous*, sehingga mengurangi ketergantungan pada otoritas pusat.

Tarigan (2022) membangun sistem penerbitan SKPI sebagai Non-Fungible Token (NFT) berstandar ERC-721. Dengan metodologi SDLC, sistem memadukan *smart contract* dan IPFS. Metadata disimpan di IPFS, sementara kontrol akses pada token mencegah transfer tidak sah, sehingga penerbitan dan kepemilikan dapat dibuktikan secara kriptografis. Seluruh fungsi berjalan sesuai spesifikasi dan memperkuat integritas sertifikat lulusan.

Swastika et al. (2022) berfokus pada penyimpanan sertifikat digital yang aman untuk *platform* belajar daring. Mereka menggarisbawahi risiko pemalsuan serta kelemahan sistem terpusat, lalu merancang dan menguji sistem berbasis Ethereum (Geth + *smart contract*). Pengujian menunjukkan sistem memproses ±200 transaksi per 8 detik dan estimasi kebutuhan 22,6 GB untuk 10 juta blok memberi indikasi praktis soal performa dan skalabilitas dalam skenario *e-learning*.

Faaroek et al. (2022) merancang situs akademik dengan penyimpanan sertifikat digital berbasis blockchain. Fokusnya pada penguatan keamanan, integritas, dan kepercayaan terhadap sertifikat yang diterbitkan institusi, dengan tahapan pengembangan dari analisis hingga pengujian. Hasil akhirnya menegaskan efektivitas blockchain untuk menerbitkan dan menyimpan sertifikat dalam *platform* daring dengan tingkat keamanan, keandalan, dan skalabilitas yang memadai.

Pu dan Lam (2023) menyajikan analisis multi kasus tentang manfaat blockchain untuk sertifikat digital lintas konteks bisnis. Mereka menyoroti peningkatan kualitas sistem, kemudahan verifikasi, dan dukungan terhadap pertumbuhan bisnis, sembari mengingatkan

tantangan nyata menghubungkan catatan *on-chain* dengan objek fisik serta tata kelola pemangku kepentingan. Studi ini berguna sebagai peta manfaat dan risiko untuk adopsi berskala organisasi.

Nguyen et al. (2020) memaparkan VECefblock, prototipe autentikasi sertifikat berbasis blockchain untuk menanggapi maraknya pemalsuan di Vietnam. Mereka menelusuri tren penerapan, memetakan arsitektur, dan menunjukkan efektivitas VECefblock dalam meningkatkan keandalan kredensial. Pendekatan ini diposisikan sebagai model yang berpotensi diadaptasi secara lintas negara.

Aini et al. (2023) menggarap sisi yang kerap terlewat: manajemen verifikasi dan pencabutan sertifikat (revocation) dalam PKI. Dengan desain sistem berbasis blockchain dan optimasi bloom filter, mereka mengevaluasi kinerja lewat waktu eksekusi dan konsumsi data, serta menunjukkan keunggulan dibanding CRL/OCSP—mengarah pada verifikasi status yang lebih cepat, aman, dan bebas titik gagal tunggal, relevan untuk skala jaringan yang kian padat.

Tabel 2.2 Penelitian Terdahulu

No	Referensi	Tujuan	Metode	Hasil
1	Saleh et al. (2020)	Mengidentifikasi celah solusi verifikasi yang ada dan mengusulkan kerangka verifikasi sertifikat berbasis blockchain.	Tinjauan solusi eksisting dan tema keamanan; usulan framework berbasis Hyperledger Fabric (permissioned).	Menghasilkan framework yang menekankan authentication, authorization, confidentiality, privacy, ownership sebagai kebutuhan inti verifikasi di HLF.
2	Rahardja et al. (2021)	Merumuskan protokol autentikasi sertifikat digital yang immutable dan ubiquitous berbasis blockchain untuk meningkatkan keamanan e-certificate.	Metode kualitatif, perancangan protokol, dan penggunaan SHA-256 sebagai mekanisme *hash*.	Menyimpulkan bahwa blockchain meningkatkan keamanan data e-certificate, memungkinkan administrasi terdistribusi, dan menurunkan biaya operasional; eksekusi *hash* SHA-256 mendukung keandalan autentikasi.

3	Tarigan (2022)	Menerbitkan SKPI sebagai NFT untuk menjamin keaslian dan kepemilikan.	Perancangan dan implementasi standar ERC-721 di ekosistem Web3; uji coba pada test-net.	Sistem berhasil dibangun, diimplementasikan, dan diuji pada jaringan blockchain uji; SKPI direpresentasikan sebagai NFT ERC-721.
4	Swastika et al. (2022)	Merancang website akademik untuk penerbitan dan validasi sertifikat digital berbasis blockchain.	Implementasi Ethereum (Geth) dan *smart contract*; uji reliabilitas dan skalabilitas.	Sistem memproses 200 transaksi ≈ 8 detik dan estimasi 10 juta blok ≈ 22,6 GB; penerbitan dan validasi sertifikat berjalan baik
5	Faaroek et al. (2022)	Menerapkan blockchain untuk penyimpanan sertifikat digital pada website akademik dan menguji efektivitas/keamanannya.	embangunan jaringan Ethereum lokal (Geth), *smart contract* (Solidity/Truffle), integrasi MetaMask; unit testing dan uji fungsi.	Jaringan lokal berfungsi untuk platform kelas daring; dinyatakan memiliki tingkat keamanan, reliabilitas, dan skalabilitas yang baik.
6	Pu dan Lam (2023)	Menganalisis manfaat blockchain untuk sertifikat digital secara holistik.	Multiple case study; pengembangan model analisis manfaat lintas dimensi (teknis–individu–organisasi–sosial).	Mengidentifikasi manfaat umum: penurunan biaya verifikasi, peningkatan pengambilan keputusan, daya tarik pengguna, dan pertumbuhan bisnis.
7	Nguyen et al. (2020)	Membangun sistem autentikasi sertifikat VECefblock untuk menanggulangi pemalsuan ijazah di Vietnam.	Perancangan arsitektur, proses bisnis dan struktur data; implementasi pada Hyperledger Fabric yang dideploy di Amazon EC2; evaluasi performa.	Sistem terbukti beroperasi baik pada lingkungan praktis; uji performa menunjukkan kelayakan penerapan VECefblock.

8	Aini et al. (2023)	Merancang manajemen status dan pencabutan sertifikat berbasis blockchain menggunakan Bloom filter.	Desain struktur informasi pencabutan (RSI/LRSI), distribusi Bloom filter per distribution point; implementasi pada Namecoin; evaluasi vs CRL/OCSP.	Terbukti melampaui skema CRL/OCSP pada testbed nyata, mengurangi waktu penyampaian informasi pencabutan, dan tetap kompatibel dengan standar web saat ini.

Berdasarkan ringkasan pada Tabel 2.2, tampak bahwa karya-karya sebelumnya telah memberikan landasan kuat mulai dari kerangka verifikasi, protokol autentikasi yang tidak dapat diubah dan dapat diverifikasi mandiri, implementasi penerbitan sertifikat/SKPI sebagai NFT, penyimpanan serta validasi sertifikat berbasis blockchain di lingkungan akademik beserta metrik kinerjanya, hingga telaah manfaat lintas kasus dan manajemen status maupun pencabutan sertifikat. Namun, sebagian studi berfokus pada arsitektur verifikasi tanpa integrasi penyimpanan dokumen terenkripsi *off-chain* yang terstandar, sementara yang lain menonjolkan penerbitan atau NFT tetapi belum menyediakan verifikasi publik berbasis penunjuk konten (seperti CID). Di sisi lain, aspek peran kelembagaan yang spesifik untuk LSP dan BNSP, pengelolaan kunci enkripsi, serta evaluasi fungsional dalam skenario operasional LSP belum ditangani secara terpadu. Oleh karena itu, penelitian ini memposisikan kontribusinya pada perancangan dan implementasi DApp berbasis *smart contract* yang menggabungkan penyimpanan sertifikat terenkripsi di IPFS dengan verifikasi publik melalui CID, terarah pada alur kerja LSP mulai dari pendaftaran, penilaian, penerbitan, hingga verifikasi, dan divalidasi melalui pengujian *end-to-end*, yakni pengujian yang mencakup seluruh rantai proses dari interaksi antarmuka pengguna, validasi isian, enkripsi sisi klien, pengunggahan berkas ke Pinata untuk memperoleh CID, pemanggilan fungsi kontrak dan konfirmasi *wallet*, pemeriksaan tanda terima serta *event* di blockchain, sampai pembacaan kembali status dan keberhasilan verifikasi publik berbasis CID pada antarmuka.

BAB III
METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan rekayasa perangkat lunak (*software engineering approach*) yang berorientasi pada pengembangan sistem terdesentralisasi berbasis blockchain dan *InterPlanetary File System* (IPFS). Pendekatan ini dipilih karena sesuai dengan tujuan penelitian, yaitu menghasilkan prototipe *decentralized application* (DApp) yang mampu mengotomatisasi proses sertifikasi kompetensi dengan prinsip keamanan, transparansi, dan verifiabilitas publik. Melalui pendekatan ini, penelitian tidak hanya berfokus pada hasil konseptual, tetapi juga pada realisasi sistem yang dapat diuji secara fungsional sesuai kebutuhan Lembaga Sertifikasi Profesi (LSP) dan standar Badan Nasional Sertifikasi Profesi (BNSP).

Model pengembangan yang digunakan adalah model pengembangan linear, yang merepresentasikan proses pengembangan sistem secara berurutan melalui empat tahap utama, yaitu analisis kebutuhan, perancangan, implementasi, dan pengujian. Setiap tahap memiliki keluaran yang menjadi dasar bagi tahap berikutnya, sehingga proses berjalan sistematis dan mudah dilacak. Pemilihan model ini didasari oleh karakteristik penelitian yang ruang lingkup dan kebutuhannya telah didefinisikan secara jelas sejak awal.

Menurut Murdiani dan Sobirin (2022), model linear cocok diterapkan pada proyek dengan kebutuhan yang stabil dan dokumentasi yang ketat, karena setiap tahap harus diselesaikan secara penuh sebelum melangkah ke tahap berikutnya. Pendapat ini diperkuat oleh Thesing et al. (2021), yang menyatakan bahwa model linear efektif digunakan dalam konteks akademik dan penelitian prototipe karena mendorong kedisiplinan dokumentasi dan memungkinkan evaluasi bertahap terhadap kemajuan sistem.

Dalam konteks penelitian ini, model linear dipilih karena mendukung pengembangan DApp dengan struktur logika dan dependensi antar komponen yang telah ditentukan, seperti *smart contract*, integrasi IPFS, dan antarmuka. Pendekatan linear membantu menghindari perubahan yang tidak perlu selama proses pengembangan, memastikan setiap komponen diuji secara mandiri, dan memudahkan penelusuran hubungan antara hasil pengujian dengan kebutuhan awal.

Adapun tahapan-tahapan dalam model pengembangan linear pada penelitian ini dijelaskan sebagai berikut:

a. Analisis Kebutuhan

Tahap analisis kebutuhan bertujuan untuk mengidentifikasi dan mendefinisikan kebutuhan fungsional dan non-fungsional dari sistem sertifikasi kompetensi yang akan dikembangkan. Analisis dilakukan dengan meninjau proses bisnis LSP UII, regulasi BNSP, serta praktik umum pada sistem sertifikasi konvensional.

b. Perancangan

Tahap ini bertujuan untuk merancang arsitektur sistem dan struktur komponen yang akan diimplementasikan.

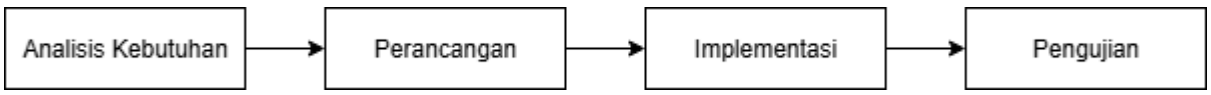
c. Implementasi

Tahap implementasi merupakan proses realisasi dari desain sistem menjadi bentuk kode program yang dapat dijalankan.

d. Pengujian

Tahap pengujian bertujuan untuk memastikan bahwa sistem berfungsi sesuai dengan kebutuhan yang telah didefinisikan. Metode pengujian yang digunakan adalah Black Box Testing, di mana fokusnya pada hasil keluaran dari setiap fungsi tanpa menguji kode internal.

Alur tahapan model pengembangan linear pada penelitian ini ditunjukkan pada Gambar 3.1.



Gambar 3.1 Diagram Metode Linear

Melalui penerapan model pengembangan linear ini, penelitian dapat dilaksanakan secara sistematis dan terukur. Setiap tahap memberikan keluaran yang jelas mulai dari definisi kebutuhan hingga hasil pengujian sehingga proses pengembangan sistem menjadi terarah dan dapat dipertanggungjawabkan. Model linear juga membantu memastikan keterkaitan logis antara kebutuhan, perancangan, implementasi, dan pengujian, yang pada akhirnya menghasilkan prototipe DApp sertifikasi kompetensi berbasis blockchain dan IPFS yang sesuai dengan tujuan penelitian.

3.1 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk memahami alur proses sertifikasi kompetensi yang berjalan pada Lembaga Sertifikasi Profesi (LSP) di salah satu perguruan tinggi serta mengidentifikasi kebutuhan sistem yang dapat diselesaikan melalui penerapan teknologi blockchain dan *InterPlanetary File System* (IPFS). Tahap ini bertujuan memperoleh gambaran menyeluruh mengenai peran aktor, data, dan aliran proses yang akan diotomatisasi melalui aplikasi terdesentralisasi, sehingga hasilnya menjadi dasar perancangan *smart contract*, model data, dan antarmuka.

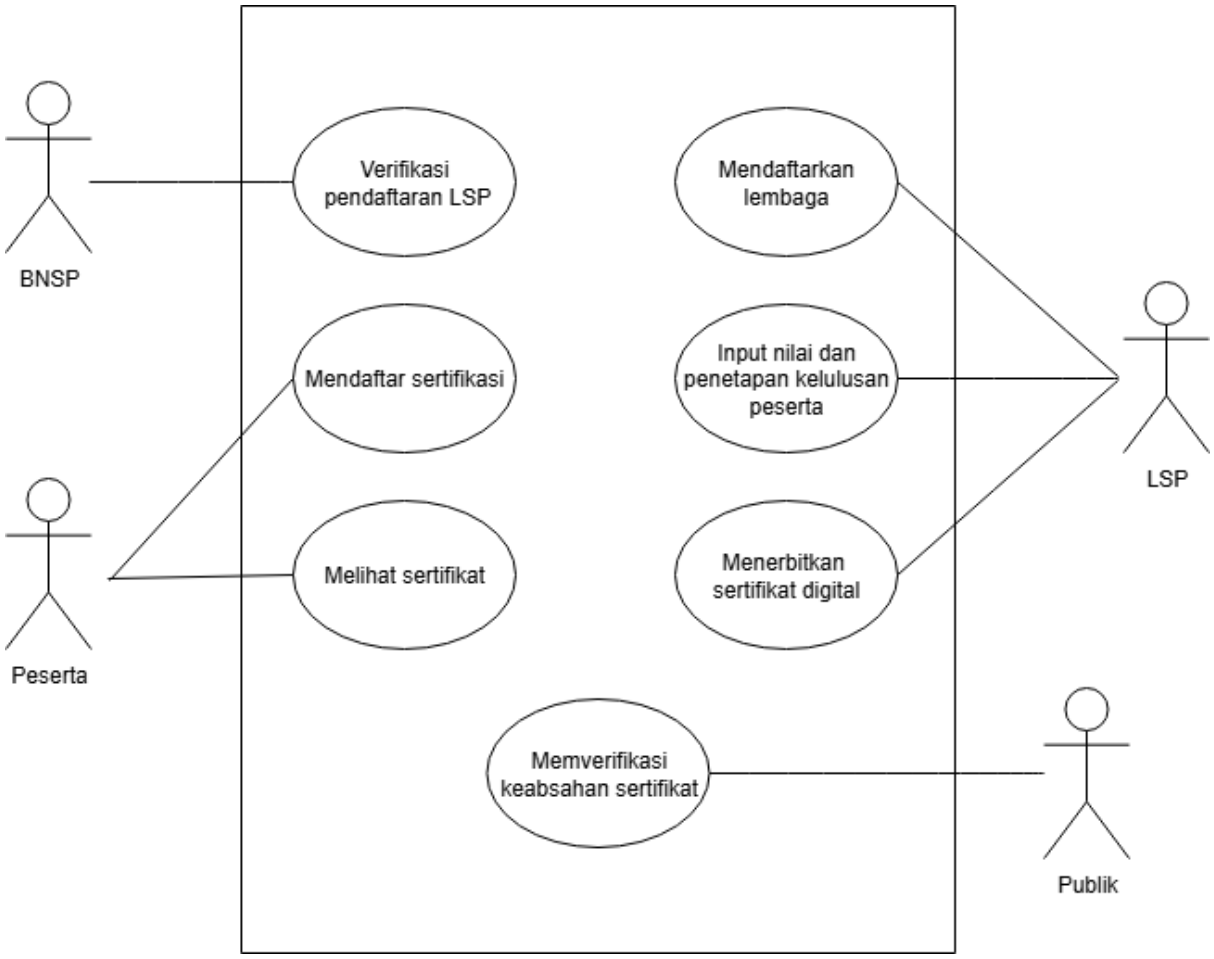
Proses analisis dilakukan melalui tiga pendekatan utama. Pertama, studi proses bisnis untuk memetakan tahapan pendaftaran peserta, pelaksanaan asesmen, penetapan hasil, dan penerbitan sertifikat, termasuk mekanisme pengawasan oleh BNSP. Kedua, studi regulatif terhadap rujukan formal seperti Peraturan Pemerintah Nomor 10 Tahun 2018 tentang BNSP, pedoman BNSP, dan Standar Kompetensi Kerja Nasional Indonesia (SKKNI) guna menegaskan entitas, aturan, dan bukti yang harus dicatat. Ketiga, studi literatur dan konsultasi teknis mengenai integrasi *smart contract*, penyimpanan terenkripsi di IPFS, serta konsep *trustless verification* untuk memastikan rancangan selaras dengan praktik terbaik teknologi terdesentralisasi.

Hasil analisis menunjukkan bahwa beberapa aktivitas administratif pada proses sertifikasi di LSP masih semi-manual, khususnya pada pengumpulan dan validasi berkas, sehingga pelacakan status antar pihak belum transparan secara *real-time*. Sertifikat yang diterbitkan dalam bentuk digital (PDF) juga belum memiliki mekanisme autentikasi kriptografis yang dapat diverifikasi secara independen.

Sistem melibatkan empat aktor utama, yaitu BNSP, LSP, Peserta, dan Publik yang berinteraksi melalui DApp. Aktor BNSP, LSP, dan Peserta melakukan autentikasi menggunakan wallet sebelum mengakses fungsi aplikasi. Publik tidak memerlukan autentikasi karena hanya melakukan verifikasi keabsahan sertifikat berbasis CID. Hubungan aktor dan fungsi sistem digambarkan pada Gambar 3.2 dengan tujuh *use case* utama sebagai berikut:

a. Verifikasi pendaftaran LSP (oleh BNSP).
b. Mendaftarkan lembaga (oleh LSP).
c. Input nilai dan penetapan kelulusan peserta (oleh LSP).
d. Menerbitkan sertifikat digital (oleh LSP)
e. Mendaftar sertifikasi (oleh Peserta).
f. Melihat Setifikat (oleh Peserta).

g. Memverifikasi keabsahan sertifikat (oleh Publik).



Gambar 3.2 Diagram *Use Case*

Setelah itu, uraian singkat *use case* adalah sebagai berikut. Terdapat tujuh *use case* yang dijalankan oleh keempat aktor, yaitu: BNSP memverifikasi pendaftaran LSP sebagai langkah pemberian lisensi; LSP mendaftarkan lembaga agar terdaftar pada aplikasi; LSP melakukan input nilai dan penetapan kelulusan peserta berdasarkan hasil asesmen; LSP menerbitkan sertifikat digital bagi peserta yang lulus; Peserta mendaftar sertifikasi melalui formulir pada DApp; Peserta melihat sertifikat yang telah diterbitkan; dan Publik memverifikasi keabsahan sertifikat dengan memasukkan CID yang diperoleh dari sertifikat digital yang dibagikan oleh pemilik sertifikat, Pada aplikasi, CID ditampilkan pada *dashboard* peserta yang dapat disalin dan dibagikan sehingga status validitas ditampilkan. Tabel 3.1 hingga Tabel 3.4 selanjutnya merinci untuk masing-masing *use case* sesuai peran aktor.

Tabel 3.1 *Use Case* oleh BNSP

Nama UC	Keterangan
Verifikasi pendaftaran LSP	BNSP memeriksa pengajuan pendaftaran lembaga oleh LSP, mencocokkan kelengkapan perizinan dan data legal, lalu menetapkan status terverifikasi atau ditolak. Hasil verifikasi tersimpan pada aplikasi terdesentralisasi dan menjadi prasyarat LSP untuk mengoperasikan fungsi lain di aplikasi terdesentralisasi.

Tabel 3.2 *Use Case* oleh LSP

Nama UC	Keterangan
Mendaftarkan lembaga	LSP menghubungkan *wallet* lalu mengajukan pendaftaran lembaga melalui DApp dengan mengisi profil lembaga pada formulir. Pengajuan dikirim ke BNSP untuk diverifikasi dan status pengajuan dapat dipantau oleh LSP.
Input nilai dan penetapan kelulusan peserta	LSP memasukkan hasil asesmen peserta, menghitung pencapaian terhadap unit kompetensi, kemudian menetapkan status lulus atau tidak lulus. Keputusan kelulusan menjadi dasar penerbitan sertifikat.
Menerbitkan sertifikat digital	LSP membuat sertifikat digital bagi peserta yang lulus dengan mengunggah dokumen sertifikat melalui DApp.

Tabel 3.3 *Use Case* oleh Peserta

Nama UC	Keterangan
Mendaftar sertifikasi	Peserta menghubungkan *wallet* ke DApp, mengisi formulir pendaftaran sertifikasi, dan mengirimkan data yang diperlukan untuk proses asesmen. Peserta dapat memantau status pendaftaran dan kelulusan.
Melihat sertifikat	Setelah dinyatakan lulus dan sertifikat diterbitkan, peserta dapat melihat dan mengunduh sertifikat.

Tabel 3.4 *Use Case* oleh Publik

Nama UC	Keterangan
Memverifikasi keabsahan sertifikat	Publik memasukkan CID ke DApp, kemudian aplikasi terdesentralisasi akan memeriksa kesesuaian CID pada IPFS. Aplikasi terdesentralisasi menampilkan status valid atau tidak valid sehingga keaslian sertifikat dapat dipastikan tanpa perlu autentikasi.

3.2 Perancangan

Berdasarkan hasil analisis kebutuhan, tahap perancangan difokuskan untuk menyusun rancangan sistem terdesentralisasi yang mampu mengotomatisasi tahapan sertifikasi, meningkatkan transparansi antar-aktor, serta menjaga keamanan dan integritas dokumen melalui pemanfaatan blockchain dan IPFS. Perancangan ini menempatkan *smart contract*

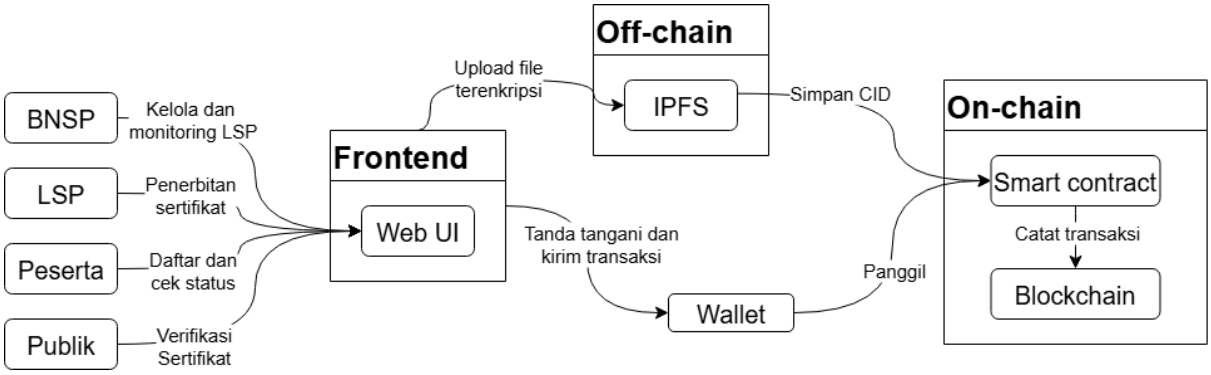
sebagai inti penegakan aturan dan memisahkan penyimpanan berkas berukuran besar ke IPFS agar efisien, sambil tetap menyediakan jejak audit *on-chain* yang dapat diverifikasi publik.

Rancangan disusun secara berurutan agar mudah ditelusuri dan diuji pada model pengembangan linear. Bagian ini menyajikan gambaran arsitektur DApp (komponen frontend, wallet, *smart contract*, dan batas *on*-chain/off-*chain*), diikuti model proses menggunakan *Business Process Model and Notation* (BPMN) untuk skenario inti (pendaftaran LSP, proses sertifikasi, penerbitan sertifikat, dan verifikasi sertifikat), serta model desain detail berupa diagram kelas *smart contract* yang merepresentasikan pemisahan tanggung jawab modul. Urutan tersebut memberikan pandangan menyeluruh mulai dari komponen, aliran kerja, hingga struktur logika yang akan diwujudkan pada tahap implementasi.

Selain itu, perancangan juga merumuskan struktur data yang dicatat *on-chain* (metadata minimal, status, dan referensi CID) serta yang disimpan *off-chain* (berkas terenkripsi), berikut kebijakan keamanan meliputi autentikasi berbasis *wallet* dan enkripsi di sisi klien. Hasil perancangan ini menjadi acuan utama ketika mengembangkan kode, menyiapkan skenario uji *black-box*, dan mengevaluasi kesesuaian sistem terhadap tujuan fungsional yaitu pendaftaran peserta, verifikasi lembaga, penetapan kelulusan, penerbitan sertifikat, dan verifikasi publik berbasis CID.

3.2.1 Arsitektur Aplikasi Terdesentralisasi

Tahap perancangan arsitektur bertujuan memetakan komponen inti serta hubungan antarbagiannya pada sistem terdesentralisasi untuk sertifikasi kompetensi. Arsitektur dirancang agar tiga pilar utama saling terintegrasi: *smart contract* sebagai pengendali logika dan pencatat status di blockchain, IPFS sebagai media penyimpanan dokumen terenkripsi secara terdistribusi, serta aplikasi terdesentralisasi (DApp) sebagai antarmuka interaksi pengguna. Rancangan ini disusun berdasarkan hasil analisis kebutuhan, yakni mengotomatisasi proses pendaftaran dan verifikasi lembaga, asesmen peserta, penerbitan sertifikat, serta verifikasi publik yang transparan dan dapat diaudit.



Gambar 3.3 Diagram Arsitektur Sistem

Hubungan antarkomponen ditunjukkan pada Gambar 3.3. Sistem melibatkan empat komponen utama yang saling terhubung melalui jaringan blockchain dan protokol IPFS. Peran masing-masing komponen diringkas pada Tabel 3.5.

Tabel 3.5 Deskripsi Umum Arsitektur

No	Komponen	Deskripsi Fungsi
1.	*frontend*	Antarmuka web yang digunakan oleh BNSP, LSP, Peserta, dan Publik. Berkomunikasi dengan *smart contract* melalui *library* Ethers.js. Seluruh interaksi pendaftaran peserta, verifikasi lembaga, input hasil asesmen, penerbitan dan penelusuran sertifikat, serta verifikasi berbasis CID dilakukan melalui DApp.
2.	*on-chain*	Terdiri atas *smart contract* dan blockchain. *Smart contract* mengeksekusi aturan (otorisasi peran, perubahan status, penerbitan sertifikat) dan mencatat *event* serta status secara permanen. Blockchain menjamin integritas, keterlacakan, dan ketahanan terhadap manipulasi.
3.	*off-chain*	Penyimpanan dokumen berukuran besar secara terdistribusi. Berkas (mis. sertifikat) dienkripsi di sisi klien sebelum diunggah ke IPFS. Hasil unggahan berupa CID unik disimpan pada *smart contract* sebagai rujukan kriptografis terhadap dokumen.
4.	*wallet*	Mekanisme autentikasi dan otorisasi transaksi untuk BNSP, LSP, dan Peserta (Publik tidak membutuhkan autentikasi

		untuk verifikasi). Wallet seperti MetaMask mengelola kunci privat dan menandatangani transaksi saat fungsi smart contract dipanggil.

Berdasarkan tabel di atas, seluruh komponen saling berinteraksi secara terintegrasi. Lapisan frontend bertugas sebagai penghubung antara pengguna dengan sistem, sedangkan smart contract di lapisan onchain menangani seluruh proses bisnis dan pencatatan transaksi. Lapisan offchain berperan sebagai media penyimpanan dokumen terenkripsi, sementara wallet digunakan sebagai mekanisme otentikasi dan penandatanganan transaksi secara terdesentralisasi. Dengan demikian, setiap proses yang terjadi di sistem dapat diverifikasi dan ditelusuri tanpa bergantung pada server pusat.

Secara umum, alur interaksi dalam sistem dapat dijelaskan sebagai berikut:

1. Peserta mendaftar sertifikasi, LSP mengajukan pendaftaran lembaga ke BNSP, BNSP melakukan verifikasi pendaftaran LSP melalui DApp.
2. DApp menyiapkan dan mengirim transaksi ke *smart contract* (melalui *wallet*) untuk mencatat atau mengubah status *on-chain*.
3. Jika diperlukan unggahan dokumen (mis. sertifikat), berkas dienkripsi di sisi klien, kemudian diunggah ke IPFS melalui DApp.
4. CID dari IPFS dikembalikan ke aplikasi dan disimpan di blockchain sebagai referensi yang dapat diverifikasi publik.
5. Publik melakukan verifikasi keabsahan sertifikat dengan memasukkan CID pada DApp, sistem mencocokkan CID dengan catatan *on-chain* tanpa membuka isi dokumen.
6. Seluruh transaksi ditandatangani melalui *wallet*, sehingga identitas dan otorisasi pengguna tervalidasi secara kriptografis.

Untuk memudahkan pemahaman, arsitektur dapat dipandang sebagai tiga lapisan:

1. Lapisan Presentasi (*Frontend Layer*)

   Menyediakan antarmuka pengguna berbasis web (DApp). Lapisan ini menjadi titik interaksi antara pengguna dan sistem, serta berfungsi mengelola input, menampilkan status, dan memfasilitasi melihat dan mengunduh dokumen.
2. Lapisan Logika Bisnis (*Smart Contract Layer*)

Menangani seluruh logika proses bisnis sesuai aturan proses sertifikasi, termasuk pendaftaran, pengajuan, verifikasi, dan penerbitan dokumen. Setiap tindakan direkam sebagai transaksi dalam blockchain.

3. Lapisan Data (*Data Layer*)

   Menyimpan metadata dan status secara *on-chain* (alamat *wallet*, peran, status peserta, CID, *timestamp*) serta menyimpan dokumen terenkripsi secara *off-chain* di IPFS.

Integrasi antara blockchain dan IPFS bersifat komplementer: blockchain menyediakan jejak audit yang tidak dapat diubah dan mekanisme verifikasi publik, sedangkan IPFS menyediakan penyimpanan efisien untuk dokumen terenkripsi dengan CID sebagai bukti kriptografis. Rancangan arsitektur ini menjadi dasar penurunan struktur data dan rancangan *smart contract* pada bagian berikutnya, serta memandu penyusunan skenario uji *black-box* terhadap fungsi utama sistem.

3.2.2 *Business Process Model and Notation* (BPMN)

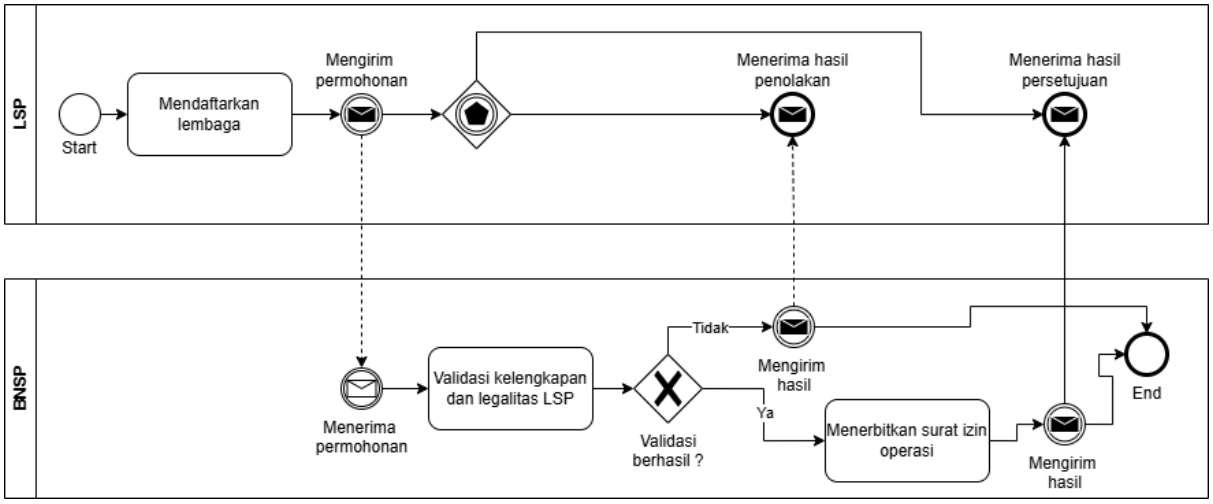
Bagian ini menampilkan tiga proses utama dalam bentuk BPMN, yaitu proses pendaftaran LSP, proses sertifikasi, proses verifikasi sertifikat. Pada Gambar 3.4 menampilkan alur pendaftaran LSP dalam dua *lane*, yaitu LSP sebagai pemohon dan BNSP sebagai otoritas. Tujuan proses ini adalah menetapkan status lembaga pada aplikasi terdesentralisasi, apakah terverifikasi atau ditolak, serta menerbitkan surat izin operasi bagi lembaga yang memenuhi syarat.

Alur dimulai ketika LSP melakukan aktivitas mendaftarkan lembaga dengan melengkapi profil, berkas legalitas, dan alamat *wallet* penanggung jawab. Setelah data dinyatakan siap, LSP mengirim permohonan. Permohonan diterima oleh BNSP yang kemudian menjalankan validasi kelengkapan dan legalitas LSP meliputi pemeriksaan kebenaran identitas, kelengkapan dokumen, masa berlaku perizinan, serta konsistensi data.

Selanjutnya terjadi pengambilan keputusan pada gateway validasi berhasil. Apabila validasi tidak berhasil, BNSP mengirim hasil penolakan kepada LSP berisi alasan penolakan dan rekomendasi perbaikan, lalu proses berakhir pada sisi LSP dengan status ditolak. Sebaliknya, apabila validasi berhasil, BNSP menerbitkan surat izin operasi, kemudian mengirim hasil persetujuan. LSP menerima hasil persetujuan dan status lembaga di sistem berubah menjadi terverifikasi.

Kondisi awal adalah LSP belum terdaftar dan belum memiliki izin operasi. Kondisi akhir adalah salah satu dari dua keadaan berikut: status LSP tercatat pada sistem sebagai

terverifikasi beserta nomor surat izin, atau ditolak dengan alasan penolakan yang terdokumentasi. Permohonan hanya dapat dikirim jika seluruh atribut wajib terisi dan dokumen pendukung terunggah. Keputusan BNSP berdasarkan kesesuaian dokumen dan legalitas yang berlaku. Setiap keputusan harus menghasilkan pemberitahuan resmi kepada LSP dan terekam pada jejak audit sistem. Dalam implementasi DApp, metadata registrasi dan status hasil keputusan dicatat pada *smart contract*, sedangkan dokumen pendukung dapat disimpan terenkripsi di IPFS dengan CID yang dicantumkan pada metadata untuk keperluan audit.



Gambar 3.4 BPMN Permohonan LSP ke BNSP

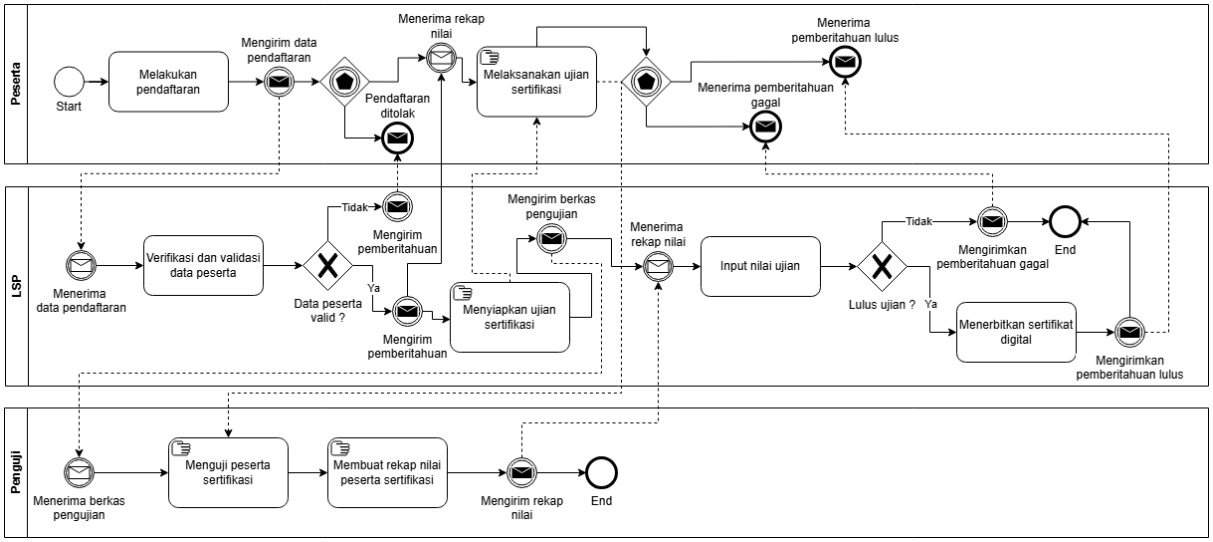
Pada Gambar 3.5 menampilkan BPMN sertifikasi peserta dengan tiga *lane*, yaitu Peserta, LSP, dan Penguji. Tujuan proses ini adalah memvalidasi data peserta, menyelenggarakan asesmen, menetapkan kelulusan, serta menerbitkan sertifikat digital bagi peserta yang memenuhi syarat.

Alur dimulai pada Peserta dengan aktivitas melakukan pendaftaran. Peserta melengkapi formulir dan mengirim data pendaftaran. LSP menerima berkas melalui *event* menerima data pendaftaran, kemudian menjalankan verifikasi dan validasi data peserta yang mencakup pemeriksaan identitas, kelengkapan dokumen, dan persyaratan administratif. Pada *gateway* bila data tidak valid, LSP mengirim pemberitahuan penolakan dan proses berakhir pada sisi Peserta setelah menerima notifikasi pendaftaran ditolak. Bila data valid, LSP mengirim pemberitahuan persetujuan dan menyiapkan ujian sertifikasi.

Penguji tidak berinteraksi langsung dengan aplikasi. Penguji hanya menguji peserta sertifikasi sesuai jadwal dan prosedur, lalu menyusun rekap nilai menggunakan mekanisme yang dikendalikan di luar aplikasi terdesentralisasi. Rekap nilai diserahkan kepada LSP melalui

kanal administrasi yang ditetapkan. LSP kemudian menerima rekap nilai dan melakukan input nilai ujian ke dalam aplikasi terdesentralisasi. Pada *gateway* jika hasil tidak lulus maka LSP mengirimkan pemberitahuan gagal kepada peserta dan proses ditutup. Jika hasil lulus maka LSP menerbitkan sertifikat digital dan mengirimkan pemberitahuan lulus. Peserta menerima rekap nilai serta pemberitahuan lulus atau gagal sesuai keputusan.

Kondisi awal adalah LSP telah terverifikasi, pendaftaran dibuka, dan Peserta memiliki akun *wallet* yang aktif. Kondisi akhir berupa salah satu dari dua keadaan, yaitu Peserta lulus dan sertifikat digital terbit, atau Peserta tidak lulus dengan pemberitahuan resmi yang terdokumentasi. Dalam implementasi, seluruh aksi terhadap aplikasi terdesentralisasi dilakukan oleh LSP. Rekap nilai dari Penguji menjadi dasar penetapan kelulusan yang dicatat pada *smart contract*. Dokumen sertifikat terenkripsi diunggah ke IPFS, sedangkan CID beserta metadata disimpan *on-chain* untuk keperluan verifikasi publik.

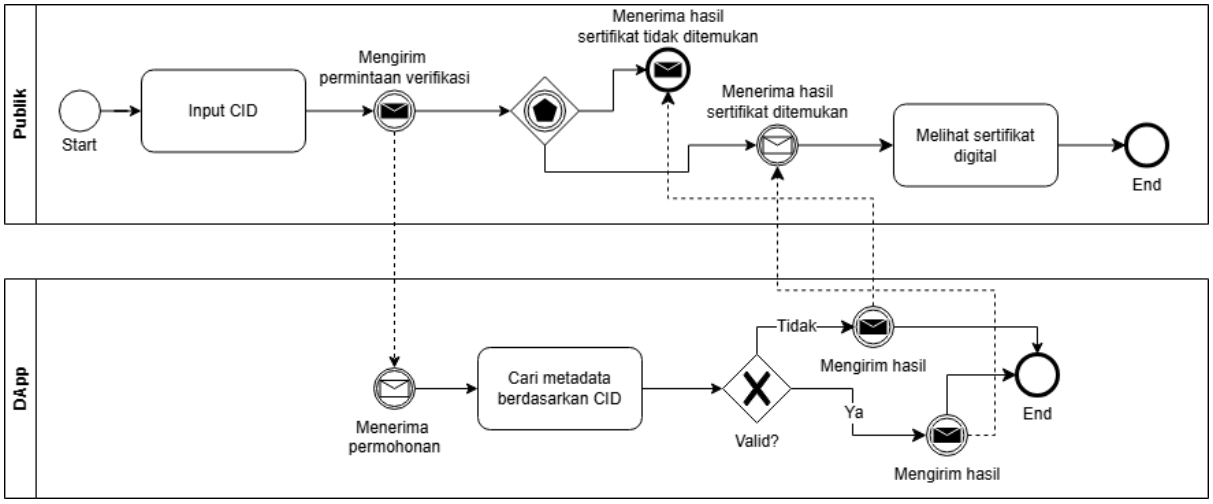


Gambar 3.5 BPMN Proses Sertifikasi

Pada Gambar 3.6 menampilkan BPMN verifikasi publik dengan dua *lane*, yaitu Publik sebagai pemverifikasi dan Sistem yang mewakili DApp, *smart contract*, serta IPFS. Tujuan proses ini adalah memastikan keaslian sertifikat berbasis konten tanpa otoritas pusat dengan cara mencocokkan *Content Identifier* (CID) yang dimasukkan pengguna dengan CID dan metadata yang tercatat pada kontrak. Proses verifikasi tidak memerlukan login dan tidak membuka isi dokumen karena berkas sertifikat disimpan terenkripsi di IPFS.

Alur dimulai ketika Publik membuka halaman verifikasi lalu memasukkan CID. Sistem menerima permintaan, melakukan validasi format CID, kemudian melakukan pencarian

pada *smart contract* berdasarkan CID tersebut. Jika entri tidak ditemukan, Sistem menampilkan status tidak valid dan proses berakhir. Jika entri ditemukan, Sistem memeriksa ketersediaan objek di IPFS dengan CID yang sama, bila objek dapat diakses, Sistem menampilkan status valid beserta metadata minimum yang boleh diekspos kepada publik, misalnya nama LSP penerbit, tanggal terbit, dan status sertifikat. Jika objek tidak tersedia atau pemeriksaan akses gagal, Sistem menampilkan status tidak valid/tidak ditemukan dengan pesan penyebab yang jelas.



Gambar 3.6 BPMN Verifikasi Sertifikat oleh Publik

3.2.3 Diagram Kelas *Smart Contract*

Rancangan mengikuti prinsip modular dan *separation of concerns* dimana setiap domain tanggung jawab ditempatkan pada kontrak terpisah, sementara data persisten dipusatkan pada satu kontrak penyimpanan bersama. Gambar 3.7 memperlihatkan hubungan antarmodul: satu kontrak penyimpanan (SertifikasiStorage), empat kontrak fungsional (RegistrasiLSP, PesertaManager, SertifikasiManager, dan VerifikasiPublik) yang mewarisi penyimpanan tersebut, serta sebuah kontrak penghubung (MainContract) yang mengomposisi dan menjadi gerbang utama dari antarmuka pengguna.

Peran masing-masing kontrak:

- SertifikasiStorage (lapisan data bersama)

  Menyediakan struktur dan *state* yang dipakai bersama oleh seluruh modul: *struct* untuk LSP, peserta, dan sertifikat; *mapping* indeks; *enum* status (mis. Terverifikasi, Aktif/Tidak Aktif, Lulus/Tidak Lulus); serta *event* audit (pendaftaran, verifikasi, penilaian, penerbitan, dan referensi CID IPFS). Dengan memusatkan *state* pada satu

kontrak, seluruh modul berbagi sumber kebenaran yang sama, mencegah duplikasi dan ketidakkonsistenan *storage*, serta memudahkan penelusuran *event*.

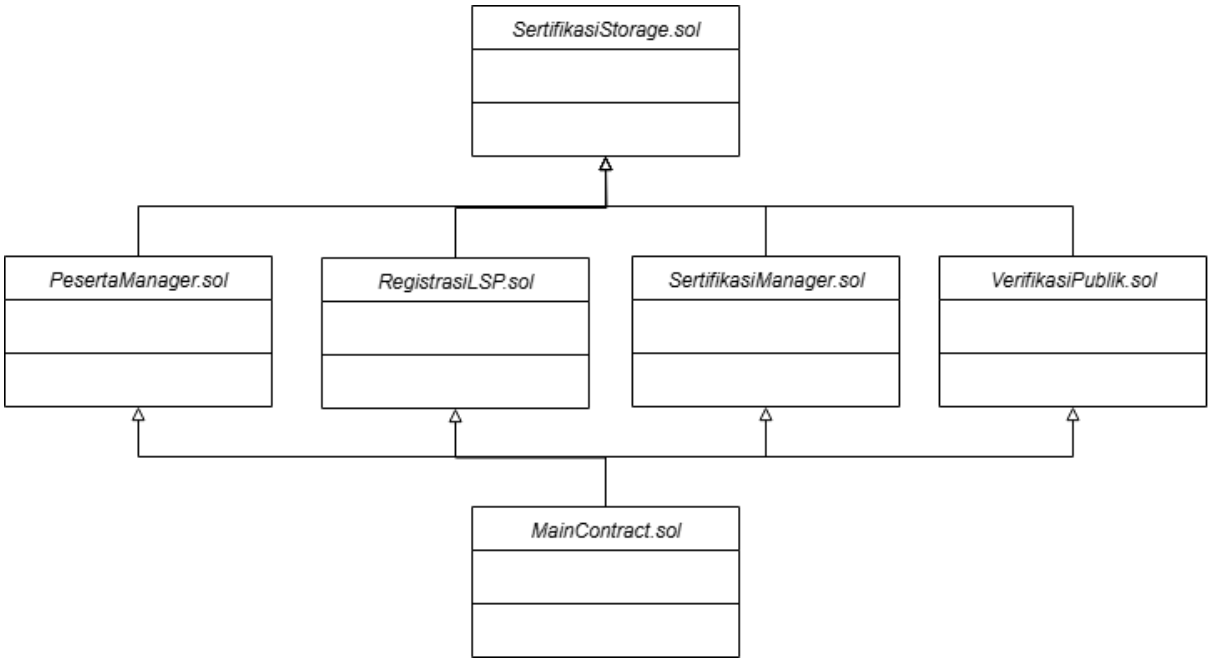
- RegistrasiLSP (domain perizinan LSP)
  Menangani pendaftaran awal LSP (menyimpan CID akta notaris terenkripsi) dan proses verifikasi/penolakan oleh BNSP. Fungsi tulis dibatasi dengan *modifier* peran (contoh: onlyBNSP untuk verifikasi), memeriksa keunikan alamat, dan menerbitkan *event* LSPRegistered / LSPVerified.
- PesertaManager (domain identitas peserta)
  Mengelola pendaftaran peserta (metadata ringkas dan CID berkas terenkripsi di IPFS), serta pembaruan terbatas sesuai aturan. Setiap transaksi memicu *event* untuk jejak audit dan validasi *precondition* (contoh: peserta belum terdaftar pada *wallet* yang sama).
- SertifikatManager (domain asesmen dan penerbitan sertifikat)
  Mengakomodasi alur inti LSP: pengajuan skema, input nilai, penetapan kelulusan, dan penerbitan sertifikat (menyimpan CID sertifikat terenkripsi). Fungsi yang mengubah *state* dibatasi untuk LSP yang telah terverifikasi, *event* seperti ScoreSubmited, ResultSet, dan CertificateIssued dicatat untuk audit.
- VerifikasiPublik (domain cek dan baca sertifikat)
  Menyediakan operasi hanya baca untuk memeriksa validitas sertifikat dan mengambil metadata minimum yang boleh diekspos (mis. nama LSP, skema, tanggal terbit, dan status). Modul ini tidak memiliki fungsi tulis sehingga aman diakses publik.
- MainContract (gerbang utama dan pengendali akses)
  Menjadi titik masuk tunggal bagi DApp. Kontrak ini menyimpan referensi ke modul fungsional dan meneruskan panggilan ke kontrak yang relevan. Sentralisasi gerbang panggilan menyederhanakan penerapan kontrol peran, kebijakan p*ause/emergency stop* (jika digunakan), dan validasi lintas-modul, sekaligus mengurangi *attack surface* karena hanya satu ABI yang terekspos ke antarmuka.

Pemisahan domain ke dalam modul yang mewarisi SertifikasiStorage memberikan beberapa keuntungan:

1. Keterjagaan konsistensi data: satu *storage* bersama mencegah duplikasi *slot* dan konflik tata letak.
2. Kemudahan pengujian: tiap modul dapat diuji secara unit tanpa membawa seluruh logika.

3. Keterbacaan dan pemeliharaan: perubahan di satu domain tidak memengaruhi domain lain.
4. Pengurangan permukaan serangan: hanya MainContract yang terekspos sebagai *entry point*, kontrak baca (*read-only*) tidak memiliki fungsi tulis, dan fungsi sensitif dibatasi dengan *modifier* peran (mis. onlyBNSP, onlyVerifiedLSP).

Dengan demikian, diagram pada Gambar 3.6 tidak hanya menunjukkan susunan modul, tetapi juga menegaskan strategi pemisahan tanggung jawab, konsolidasi *state*, dan pola penghubung yang dipilih untuk mencapai kejelasan arsitektur, jejak audit yang kuat, serta risiko keamanan yang lebih rendah.



Gambar 3.7 Diagram Kelas *Smart Contract*

3.2.4 Struktur Data

Perancangan struktur data ditujukan untuk merepresentasikan entitas-entitas utama dalam proses sertifikasi kompetensi dan memastikan pengelolaannya efisien pada dua ranah penyimpanan: *on-chain* (melalui *smart contract* di blockchain) dan *off-chain* (melalui IPFS). Data yang dicatat di blockchain bersifat *immutable* dan berfungsi sebagai catatan status/metadata serta rujukan dokumen, sedangkan berkas berukuran besar atau sensitif disimpan terenkripsi di IPFS.

Struktur data sistem dibagi menjadi dua kategori utama:

a. Data *on-chain*: mencatat transaksi, status proses (registrasi LSP, pendaftaran & penilaian peserta, penerbitan/penarikan sertifikat), metadata minimal, dan CID dokumen.
b. Data *off-chain*: menyimpan berkas berukuran besar atau sensitif (mis. sertifikat digital dan metadata detail) dalam bentuk file terenkripsi di IPFS.

Keterkaitan antara keduanya dijembatani melalui CID IPFS yang disimpan di blockchain sebagai referensi unik terhadap data *off-chain*. Data *on-chain* direpresentasikan dalam beberapa *struct* dan *enum* pada *smart contract* untuk mendukung proses bisnis sistem. Struktur utama yang digunakan dipetakan ke dalam Tabel 3.6. Struktur tersebut memastikan bahwa setiap transaksi dan perubahan status permohonan dapat dilacak secara historis melalui *event log* di blockchain.

Tabel 3.6 Struktur Data *On-chain*

Nama *Struct/Enum*	Deskripsi
enum SkemaSertifikasi	Enumerasi jenis skema sertifikasi dipakai sebagai acuan penetapan jenis sertifikasi pada entitas Sertifikasi serta validasi input.
struct Peserta	Merepresentasikan akun peserta pada sistem. Menyimpan CID metadata peserta (IPFS), status pendaftaran dan keaktifan, referensi kontrak sertifikasi yang sedang aktif, daftar riwayat sertifikasi yang pernah diikuti, serta timestamp pendaftaran.
struct Sertifikasi	Merepresentasikan satu proses sertifikasi untuk seorang peserta pada suatu skema. Memuat alamat *wallet* peserta, jenis skema (SkemaSertifikasi), CID berkas sertifikat terenkripsi di IPFS, status kelulusan dan keaktifan, waktu pengajuan–selesai–kedaluwarsa, alamat *wallet* LSP penilai, serta alasan ketidaklulusan bila ada.
struct Nilai	Menyimpan komponen penilaian peserta (tulis, praktik, wawancara) dan penanda sudah/tidaknya entri nilai. Sebagai dasar penetapan kelulusan pada entitas Sertifikasi.

Berbeda dengan data *on-chain*, data *off-chain* pada sistem ini mencakup berkas berukuran besar atau bersifat privat, seperti file sertifikat digital (PDF), metadata peserta dan LSP, serta rincian hasil asesmen. Seluruh berkas sensitif dienkripsi di sisi klien sebelum

diunggah ke IPFS, sedangkan data terstruktur disimpan sebagai berkas JSON agar mudah diintegrasikan dengan *smart contract* dan antarmuka DApp.

Penyimpanan dokumen dan metadata rinci dilakukan *off-chain* karena jika dicatat langsung *on-chain*, ukuran data akan meningkatkan biaya transaksi (gas) secara signifikan. Di sisi lain, sifat blockchain yang transparan tidak ideal untuk informasi pribadi. Karenanya, IPFS dipilih untuk tetap menjaga prinsip desentralisasi dan keterlacakan tanpa mengorbankan privasi dan efisiensi penyimpanan.

Hubungan antara data *on-chain* dan *off-chain* bersifat referensial: blockchain hanya menyimpan pointer berupa CID IPFS beserta metadata minimal (misalnya jenis dokumen dan status sertifikasi), sementara isi berkas dan detail evaluasi tetap berada di IPFS. Dengan mekanisme ini, sistem menyeimbangkan efisiensi dan keamanan sekaligus memastikan integritas melalui verifikasi CID yang unik dan tidak dapat dipalsukan. Contoh struktur JSON untuk berkas *off-chain* akan diuraikan pada Tabel 3.7.

Tabel 3.7 Struktur JSON Data Peserta

Bagian	Deskripsi
Nama Lengkap	Nama resmi peserta sesuai identitas kependudukan.
NIK	Nomor Induk Kependudukan peserta sebagai identifikasi unik.
Tempat Lahir	Kota/kabupaten tempat kelahiran peserta.
Tanggal Lahir	Tanggal lahir peserta
Jenis Kelamin	Jenis kelamin peserta sebagaimana tercantum pada identitas.
Alamat KTP	Alamat domisili peserta sesuai KTP.
Email Peserta	Alamat surat elektronik peserta untuk korespondensi resmi.
Nomor HP	Nomor telepon peserta yang aktif untuk kontak.
Id Sosmed	Identitas akun media sosial peserta sebagai kontak alternatif.

Penataan data tersebut menjadi landasan bagi perancangan *smart contract* pada bagian berikutnya, sehingga proses registrasi LSP, pendaftaran dan penilaian peserta, serta penerbitan dan verifikasi sertifikat dapat dikelola secara otomatis dan terukur.

3.2.5 Mekanisme Keamanan dan Enkripsi

Aspek keamanan merupakan komponen krusial mengingat data sertifikasi memuat informasi sensitif dan harus dijaga kerahasiaannya. Sistem menerapkan gabungan mekanisme

berbasis blockchain dan kriptografi untuk memastikan integritas, autentikasi, dan privasi data di seluruh alur pendaftaran, penilaian, penerbitan, hingga verifikasi sertifikat.

Pada lapisan *on-chain*, keamanan ditopang oleh sifat bawaan blockchain: *immutability* (catatan tidak dapat diubah setelah tercatat) dan *transparency* (rekam transaksi terverifikasi publik). Setiap tindakan penting seperti verifikasi LSP, pencatatan hasil asesmen, penerbitan atau pencabutan sertifikat memicu *event log* yang tersimpan permanen sebagai jejak audit. Kontrol akses diimplementasikan di *smart contract* dengan pendekatan *Role-Based Access Control* (RBAC) melalui modifier seperti onlyBNSP, onlyVerifiedLSP, atau pembatasan lain yang relevan, sehingga hanya alamat *wallet* berwenang yang dapat memanggil fungsi tertentu.

Pada lapisan *off-chain*, seluruh berkas berukuran besar atau privat terutama sertifikat digital (PDF) dan metadata terperinci dienkripsi di sisi klien sebelum diunggah ke IPFS. Enkripsi menggunakan algoritma *Advanced Encryption Standard* (AES-256-CBC) dengan *Initial Vector* (IV) acak per berkas. Kunci enkripsi tidak disimpan di IPFS maupun *on-chain* dan hanya digunakan kembali saat proses dekripsi melalui aplikasi. Setelah berkas terenkripsi diunggah, sistem hanya menyimpan *Content Identifier* (CID) di blockchain sebagai referensi. Dengan demikian, integritas dijaga (perubahan sekecil apa pun pada konten mengubah CID), sementara kerahasiaan tetap terlindungi walaupun IPFS bersifat publik.

Setiap aktivitas kritis menghasilkan *event log* di blockchain (mis. LSPVerified, AssessmentRecorded, CertificateIssued, CertificateRevoked) sehingga memenuhi prinsip *accountability* dan *non-repudiation* aksi ditautkan ke alamat *wallet* yang menandatangani transaksi.

Tabel 3.8 berikut merangkum terkait mekanisme keamanan yang digunakan dalam pengembangan aplikasi terdesentralisasi.

Tabel 3.8 Ringkasan Mekanisme Keamanan

Lapisan	Mekanisme	Pendekatan	Tujuan
On-chain	*Role-Based Access Control* (RBAC).	Solidity *Smart Contract*	Membatasi hak akses fungsi sesuai otoritas.
On-chain	*Immutable event log*	Blockchain	Menjamin integritas proses dan keterlacakan.
Off-chain	Enkripsi dokumen dan data JSON	AES-256-CBC (client-side), IV acak	Melindungi privasi isi dokumen dan data sensitif.

Off-chain	Penyimpanan terdesentralisasi	IPFS dengan rujukan CID tersimpan *on-chain*	Menghindari ketergantungan server pusat dan memungkinkan verifikasi integritas melalui CID.

Dengan kombinasi antara mekanisme kontrol akses di *smart contract*, pencatatan transaksi yang transparan di blockchain, serta enkripsi dokumen menggunakan AES-256-CBC sebelum penyimpanan di IPFS, sistem ini mampu menjaga keamanan, integritas, dan privasi data sertifikasi secara menyeluruh. Pendekatan ini memastikan bahwa setiap proses tidak hanya efisien dan transparan, tetapi juga memenuhi standar keamanan informasi dalam konteks sistem terdesentralisasi.

3.3 Lingkungan dan Alat Pengembangan

Untuk memastikan konsistensi implementasi rancangan arsitektur, alur BPMN, dan struktur kontrak pada Subbab 3.2, penelitian ini menetapkan lingkungan dan alat pengembangan yang terdokumentasi. Bagian ini menguraikan perangkat yang digunakan beserta perannya dalam pengembangan, *deployment*, dan pengujian. Ringkasan disajikan pada Tabel 3.5, diikuti penjelasan singkat mengenai cara komponen saling terhubung.

Tabel 3.9 Perangkat dan Teknologi Utama

No	Komponen	Deskripsi
1	Hardhat	Jaringan Ethereum lokal, kompilasi, *deployment*, dan pengujian *smart contract*.
2	Solidity	Bahasa pemrograman untuk *smart contract*.
3	React.js	Antarmuka web untuk BNSP, LSP, Peserta, dan Publik.
4	Ethers.js	Jembatan DApp ke *node* dan *smart contract*.
5	Metamask	*Wallet* pengguna dan penandatanganan transaksi.
6	IPFS (Pinata)	Penyimpanan *off-chain* dan pengelolaan CID.

Mengacu pada Tabel 3.9, seluruh komponen bekerja terintegrasi dalam satu rantai kerja yang konsisten dari sisi pengembangan hingga pengujian. Hardhat menyediakan jaringan Ethereum lokal untuk *compilation*, *deployment*, dan simulasi transaksi. Solidity digunakan untuk menulis *smart contract* yang menjadi inti logika sistem. Di sisi aplikasi, React.js berperan sebagai antarmuka untuk aktor (BNSP, LSP, Peserta, dan Publik), sedangkan Ethers.js menjadi jembatan komunikasi antara *frontend* dan *node* blockchain. MetaMask menangani manajemen identitas digital serta penandatanganan transaksi, sehingga setiap

eksekusi fungsi kontrak tercatat dan dapat ditelusuri. Untuk penyimpanan dokumen, IPFS (Pinata) dimanfaatkan sebagai media *off-chain* yang menerbitkan CID sebagai penunjuk unik objek.

Pada sisi *runtime* dan pengelolaan paket, Node.js (npm) digunakan untuk menjalankan skrip pengembangan (mis. build, dev server, test) dan mengatur dependensi proyek. Konfigurasi lingkungan diseragamkan melalui berkas variabel lingkungan (.env) yang memuat parameter penting seperti RPC_URL jaringan, CHAIN_ID, alamat kontrak hasil *deployment*, serta kredensial Pinata (API/JWT). Pendekatan ini memastikan *endpoint*, akun, dan alamat kontrak dapat diganti tanpa mengubah kode sumber, sekaligus mengurangi risiko kebocoran rahasia karena berkas .env tidak disertakan dalam sistem kendali.

Untuk menjaga kualitas dan konsistensi perilaku kontrak, rangkaian pengujian otomatis dijalankan di atas Hardhat dengan *test runner* bawaan (Mocha) dan Chai.js sebagai *assertion library*. Pengujian dilakukan pada fungsi-fungsi utama misalnya validasi peran, pencatatan nilai, penerbitan/pencabutan sertifikat, serta emisi *event* untuk memastikan hasil sesuai rancangan. Selama uji integrasi, Ethers.js dipakai untuk membaca *event log* dan memverifikasi *state* akhir, termasuk kecocokan CID yang tertulis pada *on-chain*.

Seluruh berkas sensitif (terutama sertifikat digital dan metadata rinci peserta/LSP) dienkripsi di sisi klien menggunakan AES-256-CBC sebelum diunggah ke IPFS melalui Pinata API. Praktik ini memastikan tidak ada data mentah yang meninggalkan perangkat pengguna tanpa perlindungan. Aplikasi hanya menyimpan CID di *smart contract*, sehingga integritas konten dapat diverifikasi publik (perubahan sekecil apa pun akan menghasilkan CID baru), sementara kerahasiaan tetap terjaga karena kunci enkripsi tidak disimpan di IPFS maupun *on-chain*.

Untuk menjamin *reproducibility*, konfigurasi jaringan lokal Hardhat, alamat kontrak hasil *deployment*, serta *endpoint* Pinata didokumentasikan sebagai variabel lingkungan dan dicantumkan pada dokumen .env. Antarmuka memuat ABI dan alamat kontrak dari variabel ini saat *runtime*, sehingga pengujian dapat diulang tanpa perubahan kode. Pemisahan tanggung jawab juga dijaga: blockchain memegang pembuktian dan jejak audit transaksi, sedangkan IPFS menyimpan isi dokumen terenkripsi, *frontend* menghubungkan keduanya melalui Ethers.js dan MetaMask.

Dengan susunan lingkungan seperti ini, implementasi dapat mengikuti rancangan arsitektur secara konsisten: kontrak dikompilasi dan di-*deploy* pada jaringan lokal, antarmuka React berkomunikasi dengan *node* melalui Ethers.js untuk setiap aksi (pendaftaran lembaga,

pencatatan nilai, penerbitan sertifikat), *event* dipantau untuk memastikan status transaksi, dan dokumen terenkripsi dikelola melalui IPFS (Pinata) dengan CID sebagai rujukan *on-chain*. Pendekatan ini menjaga keselarasan antara rancangan, *tooling*, dan hasil uji, sekaligus memudahkan migrasi ke jaringan uji/produksi di tahap berikutnya.

3.4 Skenario Pengujian

Pengujian disusun secara terstruktur untuk mengevaluasi prototipe DApp sertifikasi kompetensi sebelum digunakan lebih lanjut. Tujuannya adalah mendeteksi dan memperbaiki potensi masalah sejak dini agar fungsi inti berjalan benar pada berbagai kondisi. Pengujian fungsional menggunakan pendekatan *black-box*, berfokus pada kesesuaian *output* terhadap *input* tanpa menelaah kode sumber. Evaluasi dijalankan pada lingkungan pengembangan yang dijelaskan pada Subbab 3.3. Bukti uji didokumentasikan dalam bentuk tangkapan layar, *transaction receipt*, dan *event log*.

Ruang lingkup pengujian meliputi dua sisi: *on-chain* (*smart contract*) untuk memverifikasi kontrol akses berbasis peran, konsistensi *state*, emisi *event*, dan validasi masukan, serta *frontend* (DApp) untuk menilai alur antarmuka, interaksi *wallet*, enkripsi sisi klien, integrasi IPFS, dan verifikasi berbasis CID. Selain skenario positif (sesuai), disiapkan pula skenario negatif guna memastikan *guard clauses* dan pembatasan otorisasi berfungsi dengan benar. Rincian rencana uji diringkas pada Tabel 3.10 dan Tabel 3.11.

Tabel 3.10 Rencana Pengujian *Smart Contract*

No	Fungsi	Deskripsi	Keluaran
1.	tambahLSP()	Mendaftarkan LSP dengan metadata CID	LSP terdaftar, *event* LSPDitambahkan terbit
2.	verifikasiLSP()	Mengubah status LSP menjadi terverifikasi	Status LSP = terverifikasi; event verifikasi terbit
3.	daftarPeserta()	Pendaftaran peserta beserta CID metadata	Peserta aktif, *event* PesertaTerdaftar, *state* konsisten
4.	ajukanSertifikasi()	Peserta mengajukan sertifikasi (pilih skema)	Sertifikasi aktif, *event* SertifikasiDiajukan, *state* peserta diperbarui

5.	inputNilai()	LSP memasukkan nilai komponen asesmen	Nilai.sudahInput == true, *event* NilaiDiinput
6.	updateKelulusan()	LSP menetapkan lulus/tidak	lulus/aktif/sertifikatCID/tanggalSelesai terbarui, *event* KelulusanDiupdate
7.	terbitkanSertifikat()	LSP mengupload sertifikat peserta	*Event* CertificateIssued, CID tersimpan, *state* peserta konsisten
8.	verifikasiPublikByCID()	Verifikasi valid (CID ada)	valid = true + metadata minimum
9.	verifikasiPublikByCID()	Verifikasi invalid (CID tidak ada)	valid = false / "tidak ditemukan"

Setiap pengujian pada *smart contract* dijalankan dengan skenario masukan dan keluaran yang telah ditentukan. Contoh, pada verifikasiLSP() sistem diuji untuk memastikan perubahan status hanya dapat dilakukan oleh BNSP, jika dipanggil oleh akun lain, transaksi harus ditolak dengan pesan kesalahan (revert). Pada updateKelulusan(), hanya LSP yang terverifikasi yang berwenang menetapkan hasil asesmen dan mencatat sertifikatCID, setelah sukses, *state* (lulus/aktif/tanggal) dan *event* terkait harus konsisten. Uji serupa diterapkan pada ajukanSertifikasi() (mencegah pengajuan ganda saat masih aktif) serta verifikasiPublikByCID() untuk membedakan kasus valid dan tidak ditemukan secara tegas.

Pengujian *frontend* menggunakan *black-box testing* dengan fokus pada alur interaksi pengguna dan kesinambungan antara UI, Ethers.js, dan *smart contract*. Skenario yang diuji mencakup login MetaMask, pengisian formulir, enkripsi sisi klien dan unggah *ciphertext* ke IPFS (Pinata), pelacakan status berbasis *event on-chain*, serta unduh berkas. Hasil yang diharapkan: seluruh langkah berjalan mulus, status antarmuka konsisten dengan *state on-chain*, dan privasi dokumen tetap terjaga.

Tabel 3.11 Rencana Pengujian *Black-box*

No	Fitur	Deskripsi	Keluaran
1.	Login MetaMask	Autentikasi pengguna dengan *wallet*	*Wallet* terhubung, alamat terdeteksi, jaringan sesuai
2.	Registrasi LSP	LSP mengajukan pendaftaran dan unggah dokumen pendukung	Notifikasi sukses, *event* LSP terbitm, status menunggu verifikasi

3.	Verifikasi LSP (BNSP)	Aksi verifikasi pada UI peran BNSP	Status LSP terverifikasi, sinkron dengan *state on-chain*
4.	Pendaftaran Peserta	Isi formulir pendaftaran	*Event* peserta, *dashboard* peserta aktif
5.	Input Nilai	LSP mengisi nilai komponen	Nilai terbaca ulang dengan benar, status asesmen konsisten
6.	Penetapan Kelulusan	LSP set lulus/tidak dan (jika lulus) unggah sertifikat	*Event* kelulusan/terbit, *state* dan UI sinkron
7.	Verifikasi Publik (CID valid)	Pengguna memasukkan CID tercatat	Status valid dan metadata minimum tampil
8.	Verifikasi Publik (CID tidak valid)	Pengguna memasukkan CID acak/tidak ada	Status tidak valid/tidak ditemukan
9.	Pelacakan Status	Peserta memeriksa status setelah tiap aksi	Status sesuai pembaruan *event on-chain*
10.	Unduh sertifikat	Peserta mengunduh sertifikat	File hasil dekripsi valid dan dapat dibuka

Kriteria keberhasilan pengujian ditetapkan berdasarkan kesesuaian keluaran sistem dengan kondisi yang diharapkan. Suatu fungsi dinyatakan berhasil apabila:

1. Sistem menghasilkan keluaran yang konsisten dengan skenario dan data uji.
2. Transaksi blockchain tereksekusi tanpa kesalahan (*revert*) pada uji positif, serta menolak dengan *revert* yang tepat pada uji negatif, dan
3. *Event log* tercatat lengkap dan selaras dengan transaksi maupun *state* yang diuji.

Seluruh hasil pengujian didokumentasikan untuk memastikan setiap modul telah memenuhi fungsinya sebelum integrasi menyeluruh. Pendekatan ini menjamin ketiga lapisan *on-chain*, *off-chain*, dan *frontend* berjalan konsisten, terhubung, dan stabil.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Gambaran Umum Sistem

Bagian ini menjelaskan secara umum sistem yang telah dikembangkan berdasarkan rancangan pada Bab 3. Sistem merupakan aplikasi terdesentralisasi (DApp) untuk mendukung proses sertifikasi kompetensi dengan mengintegrasikan blockchain, *smart contract*, dan *InterPlanetary File System* (IPFS). Tujuan utama pengembangan adalah menghadirkan mekanisme penerbitan dan verifikasi sertifikat yang transparan, terukur, dan tahan manipulasi, sekaligus meminimalkan ketergantungan pada basis data terpusat. Pendekatan *hybrid on-chain/off-chain* digunakan: aturan dan jejak audit dicatat di blockchain, sementara dokumen berukuran besar disimpan terenkripsi di IPFS.

Sistem ini menyeimbangkan otomatisasi melalui *smart contract* dengan tahap verifikasi substantif oleh pihak berwenang. *Smart contract* menegakkan aturan secara deterministik misalnya otorisasi peran, penetapan kelulusan, dan penerbitan sertifikat. Aspek penilaian kompetensi tetap berada pada kewenangan lembaga penilai sesuai tata kelola yang berlaku. Dengan demikian, sistem menjaga akuntabilitas tanpa mengorbankan kepatuhan proses.

Aplikasi Terdesentralisasi ini melibatkan empat aktor utama, yaitu:

1. BNSP, berperan melakukan verifikasi pendaftaran LSP, serta otorisasi/penarikan kewenangan lembaga pada jaringan.
2. LSP, bertugas mendaftarkan lembaga, mengelola skema sertifikasi, mencatat hasil asesmen, menetapkan kelulusan peserta, dan menerbitkan sertifikat digital bagi peserta yang lulus.
3. Peserta, melakukan pendaftaran, mengikuti asesmen, serta mengakses sertifikat yang telah diterbitkan untuk alamat *wallet*-nya.
4. Publik, melakukan verifikasi keabsahan sertifikat secara mandiri menggunakan *Content Identifier* (CID) tanpa perlu autentikasi maupun membuka isi dokumen.

Secara teknis, sistem terdiri atas dua komponen utama, yaitu:

1. Komponen *on-chain*, berupa serangkaian *smart contract* modular pada jaringan Ethereum. Kontrak menyimpan metadata minimal (alamat pihak, skema sertifikasi, status, timestamp, dan CID), menerapkan kontrol akses berbasis peran (RBAC), serta memancarkan *event log* sebagai jejak audit permanen. Modul yang digunakan

mencakup penyimpanan terpusat *state* (*storage*), pengelolaan pendaftaran LSP, administrasi peserta, manajemen proses sertifikasi (nilai/kelulusan/penerbitan), verifikasi publik, dan satu kontrak sebagai gerbang (*main*) untuk pengintegrasian antarmuka pemanggilan.

2. Komponen *off-chain*, meliputi antarmuka web (*frontend*), enkripsi sisi klien (AES-256-CBC dengan IV acak per berkas), serta integrasi IPFS untuk penyimpanan sertifikat digital terenkripsi dan metadata terstruktur (JSON) bila diperlukan. Blockchain hanya menyimpan CID sebagai penunjuk unik ke objek di IPFS, sehingga integritas dapat diverifikasi tanpa mengekspos isi dokumen.

Integrasi kedua komponen memungkinkan pemrosesan *end-to-end*: LSP mendaftar dan diverifikasi oleh BNSP, peserta melakukan pendaftaran, LSP mencatat nilai dan menetapkan kelulusan, sertifikat dienkripsi di sisi klien, diunggah ke IPFS untuk memperoleh CID, lalu CID dan metadata minimal dicatat pada *on-chain*. Publik dapat melakukan verifikasi berbasis CID dengan mencocokkan rujukan *on-chain* tanpa membuka dokumen, sehingga privasi tetap terlindungi. Setiap perubahan status memicu *event log* di blockchain, memungkinkan seluruh pihak menelusuri riwayat dengan tingkat transparansi dan ketertelusuran yang tinggi.

Dengan demikian, sistem yang dikembangkan tidak hanya berfungsi sebagai prototipe teknis, tetapi juga sebagai model konseptual penerapan blockchain dalam layanan sertifikasi profesi: jejak audit tidak dapat diubah, verifikasi publik mandiri berbasis CID, dan perlindungan privasi melalui enkripsi *off-chain*. Pendekatan ini mendukung transformasi digital proses sertifikasi menuju tata kelola data yang lebih terbuka, akuntabel, dan terpercaya.

4.2 Kode dan Fungsi *Smart Contract*

Fokus bagian ini adalah memetakan fungsi inti *smart contract* yang mewujudkan alur kerja DApp, dimulai dari SertifikasiStorage sebagai lapisan data bersama. Empat modul RegistrasiLSP, PesertaManager, SertifikasiManager, dan VerifikasiPublik mewarisi SertifikasiStorage dan menggunakan *state* serta *event* yang sama, sedangkan MainContract sebagai gerbang terpadu yang menerapkan kontrol akses dan meneruskan panggilan ke modul terkait. Untuk setiap modul disajikan tujuan, prasyarat akses, perubahan *state*, serta *event* sebagai jejak audit, disertai cuplikan kode yang representatif agar keterlacakan antara rancangan dan implementasi terlihat jelas.

4.2.1 Struktur Data *On-Chain*

Implementasi struktur data *on-chain* pada *smart contract* ini ditulis menggunakan Solidity 0.8.28 dan dijalankan pada Hardhat sebagai lingkungan pengembangan yang menyediakan kompilasi, pengujian, serta simulasi jaringan EVM. Arsitektur *smart contract* disusun secara modular agar setiap tanggung jawab dapat dikembangkan dan diuji terpisah, namun tetap saling terhubung saat dijalankan sebagai satu kesatuan. Pada bagian ini dibahas representasi data melalui *enum*, *struct*, dan *mapping* yang menjadi fondasi alur sertifikasi.

Rancangan data *on-chain* mengikuti tiga prinsip: minimisasi data di blockchain dengan hanya menyimpan metadata penting dan penunjuk, auditabilitas melalui status eksplisit dan *event* terstruktur, pemisahan kepentingan antara identitas/status proses (*on-chain*) dan konten sensitif (*off-chain*). Dengan demikian, verifikasi publik dapat dilakukan tanpa mengekspos data pribadi.

a. *Enum* Skema Sertifikasi

Sistem menggunakan satu *enumeration* untuk mendefinisikan himpunan tetap skema sertifikasi yang diakui. Pendekatan ini membatasi input hanya pada nilai yang sah, menjaga konsistensi istilah lintas modul, serta lebih hemat penyimpanan dibandingkan string bebas. Implementasi daftar nilai tetap tersebut dideklarasikan seperti pada Gambar 4.1. Dengan *enum*, perilaku kontrak menjadi deterministik karena setiap alur asesmen selalu merujuk pada salah satu label yang telah diatur sejak kompilasi.

Nilai-nilai pada *enum* merepresentasikan jalur asesmen faktual:

Okupasi_PJOI_Pengendalian_Pencemaran_Udara, Okupasi_PJ_Pengendalian_Pencemaran_Udara, Okupasi_PJO_Pengolahan_Air_Limbah, dan Okupasi_PJ_Pengendalian_Pencemaran_Air. Label ini menjadi bagian dari identitas sertifikasi sejak pengajuan oleh peserta, digunakan kembali saat LSP melakukan penilaian dan penetapan hasil, hingga tahap penerbitan sertifikat. Karena skema adalah atribut utama entitas sertifikasi, nilai *enum* juga ikut tercantum pada muatan *event* sehingga proses penelusuran audit *on-chain* lebih terarah.

Dari sisi teknis, *enum* dikodekan sebagai bilangan bulat kecil sehingga komparasi dan penyimpanan lebih efisien di EVM. Ikatan tipe yang ketat membantu pencegahan input di luar rentang yang diizinkan, menyederhanakan validasi di fungsi-fungsi inti, dan memudahkan binding di antarmuka pengguna. Pada *frontend*, *dropdown* pemilihan skema dapat memetakan label tampilan ke indeks *enum* yang setara di kontrak melalui Ethers.js. Konsistensi inilah yang

membuat filter *log* berbasis skema, rekonstruksi alur asesmen, serta agregasi statistik per skema bisa dilakukan tanpa ambiguitas.

```
enum SkemaSertifikasi {
        Okupasi_PJOI_Pengendalian_Pencemaran_Udara,
        Okupasi_PJ_Pengendalian_Pencemaran_Udara,
        Okupasi_PJO_Pengolahan_Air_Limbah,
        Okupasi_PJ_Pengendalian_Pencemaran_Air
    }
```

Gambar 4.1 Kode *Enum* Skema Sertifikasi

Selain SkemaSertifikasi, sistem juga menggunakan *enumeration* StatusLSP untuk merepresentasikan daur hidup lembaga di dalam kontrak. Implementasi StatusLSP ditunjukkan pada Gambar 4.2. Nilai Menunggu digunakan sebagai keadaan awal setelah pendaftaran melalui fungsi registrasi. Nilai Aktif menandai bahwa lembaga telah diverifikasi oleh otoritas yang berwenang. Nilai Ditolak menegaskan hasil evaluasi yang tidak memenuhi syarat.

Status ini dipakai sebagai prasyarat pada fungsi yang membutuhkan otorisasi LSP. Hanya LSP berstatus Aktif yang dapat melakukan tindakan seperti menginput nilai, menetapkan kelulusan, atau menerbitkan sertifikat. Dengan demikian, kontrak dapat menolak lebih awal setiap pemanggilan dari LSP yang masih Menunggu atau berstatus Ditolak. Pola ini menyederhanakan validasi, menurunkan risiko eksekusi tidak sah, dan menjaga *traceability* karena perubahan status selalu diikuti oleh penerbitan *event* domain seperti LSPDitambahkan, LSPDiverifikasi, atau event penolakan yang relevan.

Dari sisi antarmuka, label StatusLSP dipetakan langsung ke tampilan agar perbedaan peran dan kewenangan terlihat jelas bagi pengguna. Alur ini juga memudahkan audit. Pemeriksa dapat menelusuri kronologi perubahan dari Menunggu menjadi Aktif beserta waktu dan akun pemverifikasi, karena seluruh aksi tercatat sebagai *event on-chain* yang terikat pada identitas lembaga.

```
enum StatusLSP {
        Menunggu,
        Aktif,
        Ditolak
}
```

Gambar 4.2 Kode *Enum* Status LSP

b. *Struct* sebagai Model Data *On-Chain*

Model data *on-chain* direpresentasikan melalui tiga *struct* inti yang saling melengkapi, yaitu Peserta, Sertifikasi, dan Nilai. Ketiganya memetakan entitas dan status proses secara ringkas namun auditabel, serta menjadi dasar bagi fungsi-fungsi kontrak untuk memvalidasi prasyarat, memperbarui status, dan menerbitkan *event*.

1. LSP

   LSP merepresentasikan lembaga penerbit pada sistem. Atribut metadataCID menjadi penunjuk ke profil lembaga yang disimpan terenkripsi di IPFS, sehingga detail sensitif tetap disimpan secara *off-chain* namun dapat dirujuk secara pasti. status bertipe StatusLSP {Menunggu, Aktif, Ditolak} berfungsi sebagai penanda otorisasi operasional hanya LSP berstatus Aktif yang berwenang mengakses fungsi kritikal seperti input nilai, penetapan kelulusan, dan penerbitan sertifikat.

   suratIzinCID menyimpan rujukan dokumen izin dari otoritas (hasil unggah terenkripsi) sebagai bukti legalitas yang mudah diaudit, sementara alasanTolak menampung keterangan ketika pengajuan LSP belum memenuhi syarat. Dengan rancangan ini, kebijakan kontrol akses dapat mengacu pada *state* yang jelas, dan setiap perubahan status meninggalkan jejak referensial yang dapat ditelusuri melalui *event* dan CID tanpa menyingkap isi dokumen seperti terlihat pada Gambar 4.3.

```
struct LSP {
        string metadataCID;
        StatusLSP status;
        string suratIzinCID;
        string alasanTolak;
}
```

Gambar 4.3 Kode *Struct* LSP

2. Peserta

   Peserta mencatat identitas operasional peserta tanpa menaruh data pribadi secara terbuka. Atribut metadataCID menunjuk berkas profil terenkripsi di IPFS sehingga detail sensitif tetap berada *off-chain*, sedangkan *state* penting disimpan di kontrak untuk verifikasi publik. Bendera terdaftar dan aktif membedakan pendaftaran awal dari kondisi siap mengikuti asesmen.

Atribut sertifikasiAktif menandai proses asesmen yang sedang berjalan untuk mencegah lebih dari satu sertifikasi aktif pada saat yang sama. Riwayat pada sertifikasiDiikuti menyimpan daftar penunjuk ke proses sebelumnya agar rekam jejak asesmen mudah ditelusuri. tanggalDaftar berfungsi sebagai *timestamp* audit sehingga perubahan status dapat dibaca secara kronologis. Dengan pemisahan ini, kontrak tetap hemat penyimpanan namun tetap menyediakan *proof* yang cukup untuk keperluan audit dan pelacakan seperti terlihat pada Gambar 4.4.

```
struct Peserta {
        string metadataCID;
        bool terdaftar;
        bool aktif;
        address sertifikasiAktif;
        address[] sertifikasiDiikuti;
        uint256 tanggalDaftar;
}
```

Gambar 4.4 Kode *Struct* Peserta

3. Sertifikasi

   Sertifikasi mewakili satu siklus asesmen dari pengajuan hingga keputusan akhir. Atribut peserta mengikat proses ke alamat *wallet* pemohon, sementara skema merekam jalur asesmen melalui nilai *enum* yang telah ditetapkan. sertifikatCID diisi setelah peserta dinyatakan lulus dan dokumen terenkripsi diunggah, sehingga publik dapat memverifikasi keberadaan sertifikat tanpa membuka isinya.

   Penanda lulus dan aktif mengendalikan alur bisnis: sertifikasi aktif menunjukkan proses berjalan, sedangkan kelulusan menentukan kelayakan penerbitan sertifikat. Tiga penanda waktu tanggalPengajuan, tanggalSelesai, dan tanggalExpiry memungkinkan pemeriksaan masa berlaku secara deterministik. Atribut lspPenilai mencatat pihak penanggung jawab asesmen, dan alasanGagal menyimpan keterangan ketika hasilnya tidak lulus. Desain ini membuat kontrak mampu menjawab pertanyaan praktis seperti apakah sertifikat masih berlaku, siapa penilainya, dan kapan proses diselesaikan, sekaligus menyediakan muatan minimum bagi *event* domain yang terkait seperti terlihat pada Gambar 4.5.

```
struct Sertifikasi {
        address peserta;
        SkemaSertifikasi skema;
        string sertifikatCID;
        bool lulus;
        bool aktif;
        uint256 tanggalPengajuan;
        uint256 tanggalSelesai;
        uint256 tanggalExpiry;
        address lspPenilai;
        string alasanGagal;
}
```

Gambar 4.5 Kode *Struct* Sertifikasi

4. Nilai

   Nilai menyimpan komponen asesmen tulis, praktek, dan wawancara dalam tipe uint8 agar penggunaan gas efisien. Penanda status sudahInput menjadi gerbang validasi sebelum penetapan kelulusan, kontrak menolak penetapan hasil jika komponen nilai belum lengkap. Pola ini menempatkan prasyarat alur secara eksplisit pada data sehingga validasi sederhana, mencegah kelalaian input, dan menjaga konsistensi antar tahapan seperti terlihat pada Gambar 4.6.

```
struct Nilai {
        uint8 tulis;
        uint8 praktek;
        uint8 wawancara;
        bool sudahInput;
}
```

Gambar 4.6 Kode *Struct* Niai

Secara keseluruhan, keempat *struct* tersebut menyediakan representasi yang padat untuk kebutuhan verifikasi, pelacakan status, dan audit. Data disusun sedemikian rupa agar efisien untuk penyimpanan *on-chain*, tetap selaras dengan prinsip privasi melalui *content addressing* ke IPFS, serta mudah ditautkan dengan *event* domain untuk jejak bukti yang dapat diverifikasi.

c. *Mapping* sebagai Indeks dan *Source of Truth*

Berbeda dari basis data relasional, *mapping* di Solidity bekerja seperti *hash* table sehingga pencarian berdasarkan kunci unik dapat dilakukan tanpa iterasi yang mahal. Deklarasi *mapping* utama pada kontrak ditunjukkan pada Gambar 4.7 dan berperan sebagai sumber kebenaran untuk pembacaan state di seluruh modul.

1. Registri LSP

   isLSP : mapping(address => bool) menyimpan penanda status lembaga pada tingkat alamat *wallet*. Nilai *true* menandai lembaga yang telah melalui proses verifikasi, sedangkan *false* menandai belum atau tidak sah. Registri ini menjadi gerbang otorisasi untuk tindakan yang bersifat menulis, misalnya input nilai, penetapan kelulusan, dan penerbitan sertifikat.

2. Indeks Peserta

   pesertaList : mapping(address => Peserta) mengikat satu alamat *wallet* ke satu entitas Peserta. Pasangan ini menjaga invarian satu alamat untuk satu identitas peserta, sehingga status aktif, sertifikasi yang sedang berjalan, serta riwayat dapat diambil secara langsung. Penanda cepat isPesertaTerdaftar : mapping(address => bool) memudahkan pemeriksaan awal sebelum fungsi inti dijalankan, sedangkan pesertaCount : uint dipakai sebagai penghitung total peserta terdaftar untuk kebutuhan pelaporan dan audit.

3. Indeks Sertifikasi

   Struktur sertifikasiList : mapping(address => Sertifikasi) menautkan alamat proses sertifikasi ke *state* lengkapnya. Dengan kunci berupa alamat proses, kontrak dapat membaca dan memperbarui status asesmen, masa berlaku, serta atribut terkait peserta tanpa harus melakukan pemindaian menyeluruh.

4. Indeks Nilai

   nilaiPeserta : mapping(address => Nilai) memisahkan data komponen asesmen dari *state* sertifikasi. Pemisahan ini membuat pembaruan nilai lebih ringan dan menyediakan jalur validasi yang jelas. Kontrak mengecek penanda status sudahInput di entitas Nilai sebelum mengizinkan penetapan kelulusan.

5. Indeks CID untuk Verifikasi Publik

   sertifikasiByCID : mapping(string => address) menjaga relasi *one-to-one* antara *Content Identifier* dari berkas sertifikat terenkripsi dan alamat proses sertifikasi yang

sah. Melalui indeks ini, halaman verifikasi publik cukup memasukkan CID untuk mendapatkan status keabsahan serta metadata minimum tanpa membuka isi dokumen.

6. Penghitung *Nonce*

   mapping(address => uint256) internal nonces menyediakan penghitung per alamat untuk kebutuhan penomoran unik. Mekanisme ini bermanfaat sebagai perlindungan terhadap penggunaan ulang data yang sama dan membantu menjaga keterurutan tindakan saat beberapa operasi dilakukan beruntun oleh alamat yang sama.

```
mapping(address => bool) public isLSP;
mapping(address => Peserta) public pesertaList;
mapping(address => Sertifikasi) public sertifikasiList;
mapping(address => bool) public isPesertaTerdaftar;
mapping(address => uint256) internal nonces;
uint public pesertaCount;
mapping(address => Nilai) public nilaiPeserta;
mapping(string => address) public sertifikasiByCID;
```

Gambar 4.7 Kode Struktur *Mapping*

Dengan susunan *mapping* tersebut, kontrak memperoleh jalur baca-tulis yang langsung ke entitas kunci, menjaga biaya eksekusi tetap efisien, dan memastikan setiap modul mengacu pada sumber data yang konsisten. Kombinasi registri peran, indeks entitas, serta pemetaan CID memungkinkan auditabilitas dan verifikasi publik dilakukan tanpa ketergantungan pada penyimpanan terpusat.

d. *Modifier*: Kontrol Akses dan Penjagaan Alur

Seperti terlihat pada Gambar 4.8, kontrak menerapkan serangkaian modifier untuk membatasi eksekusi fungsi berdasarkan peran, validitas argumen, serta kondisi awal yang wajib dipenuhi. Pendekatan ini menegakkan prinsip *least privilege*, menjaga integritas alur, dan memastikan setiap perubahan *state* hanya dilakukan oleh pihak yang tepat pada waktu yang tepat.

1. Modifier berbasis peran

   onlyBNSP, onlyLSP, dan onlyPeserta menjadi pagar utama otorisasi. onlyBNSP membatasi fungsi pengesahan lembaga pada akun otoritatif. onlyLSP memastikan aksi seperti input nilai, penetapan kelulusan, dan penerbitan sertifikat hanya dapat dipanggil oleh LSP yang sah. onlyPeserta memverifikasi bahwa pemanggil adalah peserta

terdaftar dan aktif sebelum mengakses pendaftaran lanjutan atau pengajuan sertifikasi. Pesan kesalahan yang tegas memudahkan penelusuran kegagalan saat pengujian maupun audit.

2. Validasi argumen dasar

   validAddress(address addr) menolak alamat nol untuk mencegah penunjuk tidak sah, sedangkan notEmpty(string memory str) menolak masukan string kosong. Keduanya menjaga konsistensi data sejak awal, misalnya saat mencatat alamat lembaga, alamat proses sertifikasi, atau saat menyimpan referensi penting seperti CID.

3. Penolakan terarah dan sifat *fail-fast*

   Ketika prasyarat pada modifier tidak terpenuhi, transaksi di-*revert* sehingga *state* tetap dan tidak ada *event* yang diterbitkan. Pola *fail-fast* ini mengurangi pemborosan gas untuk eksekusi yang sejak awal tidak sah. Jika seluruh modifier terpenuhi, barulah eksekusi dilanjutkan ke logika inti yang umumnya diakhiri dengan penerbitan *event* agar jejak audit dapat ditelusuri secara kronologis di blockchain.

4. Integrasi dengan prasyarat alur proses

   Prasyarat domain seperti kewajiban nilai lengkap sebelum kelulusan atau keaktifan sertifikasi sebelum penerbitan sertifikat dapat diterapkan pada modifier khusus atau sebagai pemeriksaan awal di fungsi inti. Kombinasi ini menjaga urutan kerja tetap konsisten, mencegah perubahan lintas entitas yang tidak sah, serta menyediakan umpan balik yang eksplisit kepada pemanggil ketika prasyarat alur belum terpenuhi.

```
modifier onlyBNSP() {
        require(msg.sender == bnsp, "Hanya BNSP");
        _;
}
modifier onlyLSP() {
        require(isLSP[msg.sender], "Hanya LSP");
        _;
}
modifier onlyPeserta() {
        require(pesertaList[msg.sender].terdaftar &&
pesertaList[msg.sender].aktif, "Hanya peserta aktif");
        _;
}
modifier validAddress(address addr) {
        require(addr != address(0), "Alamat tidak valid");
        _;
```

```
}
modifier notEmpty(string memory str) {
        require(bytes(str).length != 0, "String tidak boleh kosong");
        _;
}
```

Gambar 4.8 Kode *Modifier*

Dengan penerapan kontrol akses tersebut, kontrak memastikan bahwa perubahan data dan status hanya dilakukan oleh pihak yang tepat pada waktu yang tepat, sesuai hierarki dan alur yang telah ditetapkan. Hasilnya adalah proses yang aman, konsisten, dan terbuka untuk diaudit melalui catatan permanen di blockchain.

e. *Event Logging* dan Audit *On-Chain*

Seperti terlihat pada Gambar 4.9, kontrak mendefinisikan sejumlah *event* domain untuk merekam aktivitas penting secara permanen di blockchain. Tujuannya adalah menyediakan jejak audit yang dapat diverifikasi publik tanpa mengungkap data sensitif, sekaligus menjadi sinyal sinkronisasi antarmuka setelah transaksi terkonfirmasi.

1. Pendaftaran dan profil peserta

   PesertaTerdaftar(address peserta, string metadataCID, uint256 tanggal) menandai aktivasi identitas operasional peserta. Parameter peserta memudahkan penyaringan *log* per alamat, metadataCID merujuk ke profil terenkripsi di IPFS, dan tanggal menjadi penanda waktu.

   MetadataDiupdate(address peserta, string newCID) mencatat perubahan rujukan profil ketika peserta memperbarui data di sisi klien. Dengan struktur muatan yang minimal, privasi tetap terjaga karena hanya pointer yang disiarkan.
2. Inisiasi siklus sertifikasi

   SertifikasiDiajukan(address peserta, address sertifikasiID, SkemaSertifikasi skema, uint256 tanggal) merekam awal proses asesmen. Kombinasi sertifikasiID dan skema memudahkan pelacakan lintas modul dan penataan riwayat pada antarmuka. Auditor dapat menelusuri “siapa, skema apa, dan kapan” tanpa membaca isi dokumen.
3. Penetapan hasil dan penerbitan dokumen

   KelulusanDiupdate(address lsp, address sertifikasiID, string sertifikatCID, uint256 tanggal) mengumumkan keputusan lulus atau tidak lulus beserta rujukan berkas

sertifikat terenkripsi. Nilai sertifikatCID memungkinkan verifikasi publik berbasis CID, sementara lsp memperjelas akuntabilitas penilai.

4. Pembatalan proses oleh penanggung jawab asesmen

   SertifikasiBatal(address lsp, address sertifikasiID, string alasan, uint256 tanggal) mendokumentasikan penghentian terarah berikut alasan yang relevan. Catatan ini penting untuk audit keputusan tidak lulus yang tidak berujung pada penerbitan sertifikat.

5. Registri dan administrasi LSP

   Pada konteks kelembagaan, rangkaian *event* dirancang untuk menangkap seluruh siklus kewenangan secara utuh mulai dari pendaftaran, pengesahan, kemungkinan penolakan, pembaruan profil, hingga pencabutan. LSPDidaftarkan diterbitkan saat calon LSP mengajukan diri beserta perujuk profil melalui metadataCID. Ketika otoritas menetapkan legalitas operasional, LSPDiverifikasi diterbitkan dengan memuat suratIzinCID sebagai penunjuk dokumen izin yang terenkripsi. Jika pengajuan belum memenuhi ketentuan, LSPDitolak merekam keputusan tersebut berikut alasan ringkas agar jejak keputusan mudah diaudit. Perubahan informasi lembaga selanjutnya tercatat melalui LSPMetadataUpdated, sedangkan penghentian atau penonaktifan kewenangan dicatat oleh LSPDihapus.

   Setiap *event* membawa muatan minimum yang relevan. Alamat lembaga yang ditandai sebagai *indexed* untuk pemfilteran efisien, perujuk CID ke berkas terenkripsi, dan penanda waktu. Susunan parameter ini memungkinkan rekonstruksi kronologi dan pelacakan status tanpa membuka isi dokumen sensitif. Di sisi antarmuka, aliran *event* tersebut digunakan sebagai sinyal sinkronisasi: setelah transaksi terkonfirmasi, status pendaftaran, verifikasi, atau pencabutan segera tercermin pada tampilan, sementara catatan di blockchain menyediakan dasar audit yang transparan dan konsisten.

6. Muatan minimum dan perlindungan privasi

   Seluruh *event* membawa parameter esensial seperti alamat pihak, penunjuk proses (sertifikasiID), CID, dan timestamp. Desain ini cukup untuk merekonstruksi alur, namun tidak memaparkan konten dokumen ataupun atribut pribadi yang sensitif.

7. Keterkaitan dengan *state* dan indeks

   Log event dikorelasikan dengan *mapping* yang telah dijelaskan sebelumnya, misalnya sertifikasiByCID untuk verifikasi publik dan pesertaList untuk pelacakan riwayat.

Dengan parameter bertipe *indexed* pada alamat, penelusuran *log* menjadi efisien karena dapat difilter langsung per peserta atau per LSP.

8. Perilaku pada kegagalan

   Apabila prasyarat *modifier* atau validasi domain tidak terpenuhi, transaksi di-*revert* sehingga tidak ada *event* yang dipancarkan dan *state* tetap. Dengan demikian, jejak audit bersih dari *noise* dan hanya memuat tindakan yang sah.

```
event PesertaTerdaftar(
     address indexed peserta,
     string metadataCID,
     uint256 tanggal
);

event MetadataDiupdate(
     address indexed peserta,
     string newCID
);


event SertifikasiDiajukan(
     address indexed peserta,
     address sertifikasiID,
     SkemaSertifikasi skema,
     uint256 tanggal
);


event KelulusanDiupdate(
     address indexed lsp,
     address sertifikasiID,
     string sertifikatCID,
     uint256 tanggal
);


event SertifikasiBatal(
     address indexed lsp,
     address sertifikasiID,
     string alasan,
     uint256 tanggal
);
```

```
event LSPDitambahkan(
     address indexed lsp,
     string metadataCID
);


event LSPDihapus(
     address indexed lsp
);

event LSPMetadataUpdated(
     address indexed lsp,
     string newMetadataCID
);

event LSPDidaftarkan(
     address indexed lsp,
     string metadataCID
);

event LSPDiverifikasi(
     address indexed lsp,
     string suratIzinCID
);

event LSPDitolak(
     address indexed lsp,
     string alasan
);
```

Gambar 4.9 Kode Deklarasi *Event*

Dengan perancangan *event* yang terikat erat dengan *modifier* dan *mapping*, kontrak menjaga setiap perubahan tercatat secara akuntabel, dapat ditelusuri, dan konsisten dengan keadaan *on-chain*. Muatan parameter yang ringkas seperti alamat pelaku, penunjuk proses, CID, dan *timestamp* cukup untuk verifikasi publik dan rekonstruksi kronologi tanpa membuka data sensitif. Dampaknya, antarmuka dapat tersinkronisasi segera setelah transaksi terkonfirmasi, sementara proses tetap transparan dan siap diaudit melalui catatan permanen di blockchain.

f. Aturan Konsistensi (*Invariants*)

Agar alur berjalan benar dan dapat diaudit, kontrak menegakkan sejumlah invarian:

1. Hanya satu sertifikasi aktif per peserta pada satu waktu.
2. Kelulusan hanya boleh ditetapkan jika nilai lengkap.
3. CID sertifikat dicatat setelah peserta dinyatakan lulus.
4. Akses berbasis peran (pendaftaran/penilaian/kelulusan oleh LSP terverifikasi; verifikasi LSP oleh otoritas; pendaftaran peserta untuk alamat yang belum terdaftar).
5. Setiap perubahan penting menerbitkan *event* dengan parameter minimal (alamat *wallet* pihak, ID sertifikasi, CID, *timestamp*) agar jejak audit tidak ambigu.

g. Implikasi terhadap Efisiensi, Privasi, dan Audit

Dengan menyimpan metadata minimum dan penunjuk di *on-chain*, sementara dokumen disimpan terenkripsi di IPFS, sistem mencapai efisiensi gas sekaligus perlindungan privasi. Penggunaan *enum* dan *mapping* membuat validasi deterministik dan cepat, sementara pemisahan *struct* serta indeks riwayat menjaga keteraturan operasi baca/tulis. Pada akhirnya, model data ini memungkinkan publik memverifikasi status sertifikasi secara mandiri melalui CID, dan auditor menelusuri setiap langkah proses melalui *event log* semuanya tanpa perlu membuka data sensitif.

4.2.2 Modul RegistrasiLSP

Modul ini mengelola siklus kelembagaan sejak pendaftaran oleh calon LSP hingga pengesahan oleh BNSP. Alur utamanya diwujudkan melalui dua fungsi inti: daftarLSP untuk inisiasi pendaftaran menggunakan wallet lembaga, dan verifikasiLSP untuk keputusan pengesahan oleh BNSP. Rincian implementasi fungsi dan guard yang menyertainya disajikan berurutan pada Gambar 4.10 sampai dengan Gambar 4.14.

Pada daftarLSP, kontrak memastikan pemanggil belum terdaftar, argumen metadataCID tidak kosong, serta *wallet* BNSP tidak diperkenankan mendaftar sebagai LSP. Jika prasyarat terpenuhi, entri LSP berstatus awal Menunggu dibentuk, metadataCID disimpan sebagai perujuk profil terenkripsi di IPFS, penanda terdaftar diaktifkan, lalu *event* LSPDidaftarkan diterbitkan sebagai jejak audit. Rangkaian langkah ini ditunjukkan pada Gambar 4.10.

```
function daftarLSP(string calldata metadataCID) external notEmpty(metadataCID) {
    require(lspList[msg.sender].status == StatusLSP(uint8(0)),
            "Sudah pernah daftar atau status bukan Menunggu");
    require(bytes(lspList[msg.sender].metadataCID).length == 0,
            "Sudah pernah daftar");
    require(msg.sender != bnsp, "BNSP tidak bisa daftar sebagai LSP");

    lspList[msg.sender] = LSP({
        metadataCID: metadataCID,
        status: StatusLSP.Menunggu,
        suratIzinCID: "",
        alasanTolak: ""
    });
    isLSPTerdaftar[msg.sender] = true;

    emit LSPDidaftarkan(msg.sender, metadataCID);
}
```

Gambar 4.10 Kode Fungsi daftarLSP

Fungsi verifikasiLSP membatasi eksekusi hanya bagi BNSP. Kontrak memvalidasi alamat target, memastikan suratIzinCID terisi, dan memeriksa bahwa status masih Menunggu. Jika valid, status diperbarui menjadi Aktif, suratIzinCID disimpan, alasanTolak dikosongkan, hak operasional diaktifkan melalui pemutakhiran indeks peran, dan *event* LSPDiverifikasi diterbitkan. Urutan validasi hingga pembaruan status tersebut ditunjukkan pada Gambar 4.11.

```
function verifikasiLSP(address lspAddress, string calldata suratIzinCID)
    external
    onlyBNSP
    validAddress(lspAddress)
    notEmpty(suratIzinCID)
{
    LSP storage lsp = lspList[lspAddress];
    require(lsp.status == StatusLSP.Menunggu, "LSP tidak dalam status menunggu");

    lsp.status = StatusLSP.Aktif;
    lsp.suratIzinCID = suratIzinCID;
    lsp.alasanTolak = "";
    isLSP[lspAddress] = true;

    emit LSPDiverifikasi(lspAddress, suratIzinCID);
}
```

Gambar 4.11 Kode Fungsi verifikasiLSP

Pembatasan peran ditegakkan konsisten melalui *modifier* onlyBNSP pada lapisan bersama, dibantu utilitas validAddress dan notEmpty untuk menolak masukan tidak sah sejak awal. Penerapan kontrol akses tersebut dirangkum pada Gambar 4.12.

```
modifier onlyBNSP() {
    require(msg.sender == bnsp, "Hanya BNSP");
    _;
}
```

Gambar 4.12 Kode *Modifier* onlyBNSP

Selain jalur persetujuan, modul menyediakan keputusan administratif berupa penolakan pendaftaran. Fungsi tolakLSP(lspAddress, alasan) mensyaratkan alamat valid, alasan tidak kosong, serta status Menunggu, kemudian memperbarui status menjadi Ditolak, mencatat alasanTolak, mengosongkan suratIzinCID, dan menerbitkan event LSPDitolak. Logika penolakan ini ditunjukkan pada Gambar 4.13.

```
function tolakLSP(address lspAddress, string calldata alasan) external onlyBNSP
validAddress(lspAddress) notEmpty(alasan) {
    LSP storage lsp = lspList[lspAddress];
    require(lsp.status == StatusLSP.Menunggu, "LSP tidak dalam status menunggu");
```

```
    lsp.status = StatusLSP.Ditolak;
    lsp.alasanTolak = alasan;
    lsp.suratIzinCID = "";
    emit LSPDitolak(lspAddress, alasan);
}
```

Gambar 4.13 Kode Fungsi tolakLSP

Pembaruan informasi kelembagaan difasilitasi oleh updateMetadataLSP(newMetadataCID) yang dipanggil langsung oleh akun LSP. Kontrak menolak masukan kosong dan memastikan lembaga sudah terdaftar, ketika valid, metadataCID diperbarui tanpa mengubah status operasional lalu *event* LSPMetadataUpdated diterbitkan. Mekanisme pembaruan profil tersebut ditampilkan pada Gambar 4.14.

```
function updateMetadataLSP(string calldata newMetadataCID) external
notEmpty(newMetadataCID) {
    require(lspList[msg.sender].status != StatusLSP(uint8(0)), "Belum pernah
daftar");
    lspList[msg.sender].metadataCID = newMetadataCID;
    emit LSPMetadataUpdated(msg.sender, newMetadataCID);
}
```

Gambar 4.14 Kode Fungsi updateMetadataLSP

Konsistensi data dijaga melalui invarian yang jelas: satu alamat *wallet* hanya merepresentasikan satu LSP, status awal pendaftaran adalah Menunggu, verifikasi hanya sah dari Menunggu ke Aktif, dan *wallet* BNSP tidak bisa mendaftar sebagai LSP. Pelanggaran prasyarat menyebabkan transaksi di-*revert* dengan pesan tegas, *state* tetap, dan tidak ada *event* yang diterbitkan sehingga *log* bersih dari entri tidak sah.

Rujukan dokumen non-publik ditangani melalui metadataCID dan suratIzinCID yang menunjuk ke objek terenkripsi di IPFS. Pendekatan ini menjaga privasi di *off-chain*, sambil menyediakan bukti referensial di dalam rantai. Kombinasi *event* seperti LSPDidaftarkan, LSPDiverifikasi, LSPDitolak, dan LSPMetadataUpdated memudahkan antarmuka menyelaraskan status setelah konfirmasi transaksi serta mendukung audit yang transparan.

Dengan alur pendaftaran yang terdefinisi, kontrol akses yang ketat, dan *log event* yang komprehensif, modul Registrasi LSP memastikan hanya lembaga berizin yang dapat beroperasi dalam sistem. Proses menjadi aman, konsisten, dan siap ditelusuri dari pendaftaran hingga pengesahan maupun perubahan profil berikutnya.

4.2.3 Modul PesertaManager

Modul ini mengelola identitas operasional peserta berbasis *Content Identifier* (CID) metadata yang disimpan terenkripsi di IPFS. Identitas peserta diikat pada alamat *wallet* sehingga satu alamat merepresentasikan satu peserta. Pencatatan di *on-chain* memuat status, relasi dengan sertifikasi, serta jejak audit melalui *event* agar proses mudah ditelusuri dan tetap menjaga kerahasiaan data di luar rantai.

Fungsi daftarPeserta(metadataCID) membentuk profil dasar peserta. Kontrak terlebih dahulu memastikan peserta belum terdaftar dan argumen metadataCID tidak kosong. Ketika valid, struktur Peserta diisi dengan penanda terdaftar=true, aktif=true, sertifikasiAktif=address(0), daftar sertifikasiDiikuti kosong, serta tanggalDaftar sesuai waktu transaksi. Penanda indeks isPesertaTerdaftar[msg.sender] diaktifkan, penghitung pesertaCount bertambah, dan *event* PesertaTerdaftar diterbitkan sebagai jejak audit. Alur ini ditampilkan pada Gambar 4.15.

```
function daftarPeserta(string calldata metadataCID) external notEmpty(metadataCID)
{
    require(!pesertaList[msg.sender].terdaftar, "Sudah terdaftar");
    pesertaList[msg.sender] = Peserta({
        metadataCID: metadataCID,
        terdaftar: true,
        aktif: true,
        sertifikasiAktif: address(0),
        sertifikasiDiikuti: new address,
        tanggalDaftar: block.timestamp
    });
    isPesertaTerdaftar[msg.sender] = true;
    pesertaCount++;
    emit PesertaTerdaftar(msg.sender, metadataCID, block.timestamp);
}
```

Gambar 4.15 Kode Fungsi daftarPeserta

Fungsi updateMetadata(newCID) melayani pembaruan rujukan profil terenkripsi di IPFS oleh peserta aktif. Akses dibatasi *modifier* onlyPeserta, diikuti validasi newCID tidak kosong. Setelah sah, kontrak memperbarui metadataCID tanpa menyentuh atribut lain seperti status keaktifan, relasi sertifikasi, atau waktu pendaftaran, lalu menerbitkan *event*

MetadataDiupdate agar antarmuka dapat menyelaraskan tampilan profil secara reaktif. Mekanisme ini bisa dilihat pada Gambar 4.16.

```
function updateMetadata(string calldata newCID) external onlyPeserta
notEmpty(newCID) {
    pesertaList[msg.sender].metadataCID = newCID;
    emit MetadataDiupdate(msg.sender, newCID);
}
```

Gambar 4.16 Kode Fungsi updateMetadata

Akses riwayat disediakan melalui fungsi view lihatRiwayatSertifikasi(peserta) yang mengembalikan daftar penunjuk sertifikasi yang pernah diikuti. Karena hanya membaca *state*, pemanggilan ini bebas biaya gas di sisi pengguna dan akan mengembalikan data kosong jika peserta belum memiliki riwayat. Pendekatan referensial ini menjaga jejak tanpa duplikasi data karena detail sertifikasi tetap dirujuk, bukan disalin. Implementasinya diperlihatkan pada Gambar 4.17.

```
function lihatRiwayatSertifikasi(address peserta) external view returns (address[]
memory) {
    return pesertaList[peserta].sertifikasiDiikuti;
}
```

Gambar 4.17 Kode Fungsi lihatRiwayatSertifikasi

Keseluruhan rancangan menjaga pemisahan tanggung jawab: informasi rinci berada di *off-chain* melalui CID, sementara status minimum, relasi, dan *event* dicatat di *on-chain* untuk auditabilitas. Dengan kontrol akses berbasis onlyPeserta dan penerbitan *event* yang konsisten, pengelolaan identitas berlangsung aman, pembaruan profil tercermin cepat pada antarmuka, dan riwayat sertifikasi dapat ditelusuri tanpa membuka data sensitif.

4.2.4 Modul SertifikasiManager

Modul ini mengelola alur sertifikasi dari awal sampai akhir mulai dari pengajuan oleh peserta, pengisian nilai oleh LSP, penetapan hasil (lulus atau gagal), hingga penerbitan sertifikat dan penautan CID untuk verifikasi publik. Eksekusi fungsi dibatasi oleh modifier kontrol akses (mis. onlyPeserta, onlyLSP), disertai validasi parameter dan penegakan invarian domain khususnya satu sertifikasi aktif per peserta agar setiap perubahan *state* tetap sah. Setiap

aksi krusial memunculkan *event* sehingga jejak audit *on-chain* terjaga secara utuh, kronologis, dan konsisten.

Pengajuan dimulai melalui ajukanSertifikasi. Pada tahap ini kontrak memastikan peserta tidak memiliki proses aktif, membentuk sertifikasiID sebagai alamat *pseudo* yang unik, mencatat entitas Sertifikasi, memperbarui relasi pada profil peserta, lalu memancarkan *event* domain untuk audit. Rangkaian pembentukan identitas, penataan *state*, dan emisi *event* tersebut ditunjukkan pada Gambar 4.18.

```
function ajukanSertifikasi(SkemaSertifikasi skema) external onlyPeserta {
    require(pesertaList[msg.sender].sertifikasiAktif == address(0), "Masih ada
sertifikasi aktif");
    _sertifikasiIdCounter++;
    uint256 counterId = _sertifikasiIdCounter;
    address sertifikasiID = address(uint160(uint256(keccak256(
        abi.encode("SERTIFIKASI_", counterId, msg.sender, block.timestamp,
nonces[msg.sender]++)
    ))));

    sertifikasiList[sertifikasiID] = Sertifikasi({
        peserta: msg.sender,
        skema: skema,
        sertifikatCID: "",
        lulus: false,
        aktif: true,
        tanggalPengajuan: block.timestamp,
        tanggalSelesai: 0,
        tanggalExpiry: block.timestamp + (3 * 365 days),
        lspPenilai: address(0),
        alasanGagal: ""
    });

    pesertaList[msg.sender].sertifikasiAktif = sertifikasiID;
    pesertaList[msg.sender].sertifikasiDiikuti.push(sertifikasiID);
    emit SertifikasiDiajukan(msg.sender, sertifikasiID, skema, block.timestamp);
}
```

Gambar 4.18 Kode Fungsi ajukanSertifikasi

Pada sertifikasiID digenerasikan secara deterministik dari gabungan prefiks, penghitung internal, alamat *wallet* pemohon, *timestamp*, dan *nonce*. Pola ini memudahkan pemetaan dan pencarian pada struktur data karena identitas sertifikasi langsung berwujud

alamat *wallet* yang dapat dijadikan kunci pada berbagai *mapping* tanpa penyimpanan pengindeks tambahan.

Rekap nilai asesmen dicatat melalui inputNilaiPeserta. Fungsi ini membatasi eksekusi pada peran LSP, menolak pengisian ulang lewat *flag* sudahInput, dan memvalidasi rentang skor 0–100 untuk tiap komponen. Nilai yang sah disimpan di nilaiPeserta[sertifikasiID] dan diselaraskan dengan *state* proses melalui pembaruan atribut lulus pada entitas Sertifikasi. Implementasi komponen yang terjaga tersebut disajikan pada Gambar 4.19.

```
function inputNilaiPeserta(
    address sertifikasiID,
    uint8 tulis,
    uint8 praktek,
    uint8 wawancara,
         bool lulus
    ) external onlyLSP validAddress(sertifikasiID) {
         require(!nilaiPeserta[sertifikasiID].sudahInput, "Nilai sudah diinput");
         require(tulis <= 100 && praktek <= 100 && wawancara <= 100, "Nilai maksimal
100");
         nilaiPeserta[sertifikasiID] = Nilai({
             tulis: tulis,
             praktek: praktek,
             wawancara: wawancara,
             sudahInput: true
         });
         Sertifikasi storage s = sertifikasiList[sertifikasiID];
         s.lulus = lulus;
    }
```

Gambar 4.19 Kode Fungsi inputNilaiPeserta

Penetapan kelulusan dan penerbitan sertifikat dilakukan oleh LSP melalui updateKelulusan. Kontrak memverifikasi entitas serta status aktif sebelum menandai lulus = true, mengarsipkan sertifikatCID yang bersumber dari dokumen terenkripsi di IPFS, menutup proses, mengisi penanda waktu selesai beserta identitas penilai, serta membangun indeks sertifikasiByCID untuk mendukung verifikasi publik berbasis CID. Relasi sertifikasiAktif pada profil peserta turut dibersihkan agar peserta dapat menginisiasi proses berikutnya. Rangkaian tersebut diperlihatkan pada Gambar 4.20.

```
function updateKelulusan(address sertifikasiID, string calldata sertifikatCID)
    external
    onlyLSP
    validAddress(sertifikasiID)
    notEmpty(sertifikatCID)
{
    Sertifikasi storage s = sertifikasiList[sertifikasiID];
    require(s.aktif, "Sertifikasi tidak aktif");
    require(s.peserta != address(0), "Sertifikasi tidak ditemukan");

    s.lulus = true;
    s.sertifikatCID = sertifikatCID;
    s.aktif = false;
    s.tanggalSelesai = block.timestamp;
    s.lspPenilai = msg.sender;

    sertifikasiByCID[sertifikatCID] = sertifikasiID;

    pesertaList[s.peserta].sertifikasiAktif = address(0);

    emit KelulusanDiupdate(msg.sender, sertifikasiID, sertifikatCID,
block.timestamp);
}
```

Gambar 4.20 Kode Fungsi updateKelulusan

Jalur kegagalan ditangani melalui updateKegagalan. Ketika LSP memutuskan peserta tidak lulus, kontrak menutup proses, menyimpan alasan kegagalan, mengisi waktu selesai, mengikat identitas penilai, dan membersihkan relasi aktif pada profil peserta. Perubahan *state* ini disertai *event* SertifikasiBatal sehingga kronologi keputusan dapat ditelusuri tanpa membuka isi dokumen. Alur tersebut bisa dilihat pada Gambar 4.21.

```
function updateKegagalan(address sertifikasiID, string calldata alasan)
    external
    onlyLSP
    validAddress(sertifikasiID)
    notEmpty(alasan)
{
    Sertifikasi storage s = sertifikasiList[sertifikasiID];
    require(s.aktif, "Sertifikasi tidak aktif");
    require(s.peserta != address(0), "Sertifikasi tidak ditemukan");
```

```
    s.lulus = false;
    s.aktif = false;
    s.tanggalSelesai = block.timestamp;
    s.lspPenilai = msg.sender;
    s.alasanGagal = alasan;

    pesertaList[s.peserta].sertifikasiAktif = address(0);
    emit SertifikasiBatal(msg.sender, sertifikasiID, alasan, block.timestamp);
}
```

Gambar 4.21 Kode Fungsi updateKegagalan

Untuk kebutuhan tampilan dan audit, detail lengkap satu sertifikasi disediakan oleh getSertifikasiDetail. Validasi awal memastikan *record* tersedia, setelah itu kontrak mengembalikan sembilan atribut utama, termasuk identitas peserta, skema, *flag* kelulusan dan aktif, tiga penanda waktu, penilai, serta alasan gagal. Susunan keluaran yang utuh ini mengurangi jumlah panggilan terpisah di antarmuka. Jalur tersebut ditampilkan pada Gambar 4.22.

```
function getSertifikasiDetail(address sertifikasiID) external view returns (
         address peserta,
         SkemaSertifikasi skema,
         string memory sertifikatCID,
         bool lulus,
         bool aktif,
         uint256 tanggalPengajuan,
         uint256 tanggalSelesai,
         address lspPenilai,
         string memory alasanGagal
     ) {
         require(sertifikasiList[sertifikasiID].peserta != address(0),
"Sertifikasi tidak ditemukan");
         Sertifikasi memory s = sertifikasiList[sertifikasiID];
         return (
             s.peserta,
             s.skema,
             s.sertifikatCID,
             s.lulus,
             s.aktif,
             s.tanggalPengajuan,
             s.tanggalSelesai,
```

```
            s.lspPenilai,
            s.alasanGagal
        );
}
```

Gambar 4.22 Kode Fungsi getSertifikasiDetail

Ringkasan nilai dapat diakses melalui getNilaiPeserta yang mengembalikan skor dari nilai ujian tulis, praktik, wawancara, serta *flag* sudahInput. Keluaran *flag* memberi sinyal kontrol bagi antarmuka untuk menonaktifkan aksi yang mensyaratkan rekap lengkap atau menampilkan peringatan bila data belum tersedia. Implementasinya terlihat pada Gambar 4.23.

```
function getNilaiPeserta(address sertifikasiID) external view returns (uint8,
uint8, uint8, bool) {
        Nilai memory n = nilaiPeserta[sertifikasiID];
        return (n.tulis, n.praktek, n.wawancara, n.sudahInput);
}
```

Gambar 4.23 Kode Fungsi getNilaiPeserta

Secara keseluruhan, modul ini menjaga integritas proses melalui invarian yang jelas: satu sertifikasi aktif per peserta, pengisian nilai satu kali, keputusan hanya oleh LSP, dan setiap perubahan penting selalu memunculkan *event*. Indeks sertifikasiByCID menghubungkan *ciphertext* di IPFS dengan catatan *on-chain* sehingga verifikasi publik cukup dilakukan dengan memasukkan CID tanpa membuka dokumen. Dengan alur penulisan *state* yang terjaga, alur pembacaan yang efisien, serta pencatatan *event* yang konsisten, siklus sertifikasi dapat ditelusuri dari awal hingga akhir secara transparan dan akuntabel.

4.2.5 Modul VerifikasiPublik

Modul ini menyediakan jalur verifikasi yang dapat diakses oleh siapa pun untuk memastikan keabsahan sertifikat tanpa perlu otorisasi khusus dan tanpa membuka isi dokumen. Seluruh fungsi bersifat *read-only* pada kontrak, sehingga tidak mengubah *state* dan aman digunakan oleh publik maupun antarmuka. Inti desainnya bertumpu pada pemetaan sertifikasiByCID sebagai indeks yang menghubungkan CID di IPFS dengan identitas sertifikasi pada *ledger*, sehingga proses pengecekan dapat berlangsung cepat dan transparan.

Verifikasi berbasis CID dilakukan melalui verifikasiKelulusanByCID. Fungsi ini terlebih dahulu memastikan parameter CID tidak kosong, kemudian menelusuri alamat

sertifikasi melalui indeks sertifikasiByCID. Jika tidak ditemukan, fungsi segera mengembalikan tidak valid. Jika ditemukan, kontrak memuat ringkasan data sertifikasi dan memutuskan validitas berdasarkan kondisi lulus == true dan aktif == false. Pola ini memastikan bahwa hanya sertifikasi yang sudah difinalisasi sebagai lulus dan tidak lagi berada pada fase proses yang dianggap valid. Mekanisme tersebut ditampilkan pada Gambar 4.24.

```
function verifikasiKelulusanByCID(string calldata sertifikatCID)
    external
    view
    returns (bool valid, address sertifikasiID)
{
    require(bytes(sertifikatCID).length > 0, "Masukkan CID");
    sertifikasiID = sertifikasiByCID[sertifikatCID];
    if (sertifikasiID == address(0)) {
        return (false, address(0));
    }
    Sertifikasi memory s = sertifikasiList[sertifikasiID];
    valid = s.lulus && !s.aktif;
    return (valid, sertifikasiID);
}
```

Gambar 4.24 Kode Fungsi verifikasiKelulusanByCID

Selain berbasis CID, sistem juga mendukung verifikasi langsung menggunakan identitas sertifikasi. Fungsi verifikasiKelulusan(sertifikasiID) mengembalikan nilai benar jika sertifikasi yang dimaksud berstatus lulus dan tidak aktif (telah selesai), serta mengembalikan salah apabila sertifikasi tidak ditemukan atau belum memenuhi kriteria valid. Pendekatan ini bermanfaat untuk penggunaan internal antarmuka ketika alamat sertifikasi telah diketahui sebelumnya, sebagaimana ditunjukkan pada Gambar 4.25.

```
function verifikasiKelulusan(address sertifikasiID) external view returns (bool
valid) {
        if (sertifikasiList[sertifikasiID].peserta == address(0)) {
            return false;
        }

        Sertifikasi memory s = sertifikasiList[sertifikasiID];
        return s.lulus && !s.aktif;
}
```

Gambar 4.25 Kode Fungsi verifikasiKelulusan

Untuk mendapatkan rujukan menuju berkas sertifikat terenkripsi yang disimpan di IPFS, kontrak menyediakan getSertifikatCID(sertifikasiID). Fungsi ini memverifikasi keberadaan sertifikasi terlebih dahulu, lalu mengembalikan CID yang tersimpan pada struktur data *on-chain*. Antarmuka memanfaatkan nilai ini untuk menautkan verifikasi visual di sisi pengguna tanpa perlu menyentuh data sensitif. Alur pengambilan rujukan tersebut dapat dilihat pada Gambar 4.26.

```
function getSertifikatCID(address sertifikasiID) external view returns (string
memory cid) {
        require(sertifikasiList[sertifikasiID].peserta != address(0),
"Sertifikasi tidak ditemukan");
        return sertifikasiList[sertifikasiID].sertifikatCID;
}
```

Gambar 4.26 Kode Fungsi getSertifikatCID

Arah sebaliknya, yaitu penelusuran dari CID ke identitas sertifikasi, disediakan oleh cariSertifikasiByCID(sertifikatCID). Dengan memanfaatkan *mapping* sebagai indeks primer, fungsi ini mengembalikan alamat sertifikasi dalam satu langkah pencarian sehingga efisien untuk *lookup* berulang. Validasi parameter mencegah masukan kosong yang tidak bermakna. Implementasinya ditunjukkan pada Gambar 4.27.

```
function cariSertifikasiByCID(string calldata sertifikatCID) external view returns
(address sertifikasiID) {
        require(bytes(sertifikatCID).length > 0, "CID tidak boleh kosong");

        return sertifikasiByCID[sertifikatCID];
}
```

Gambar 4.27 Kode Fungsi cariSertifikasiByCID

Terakhir, modul ini juga membantu kebutuhan audit ringan terhadap profil peserta melalui getPesertaMetadata(peserta). Fungsi ini mengembalikan CID metadata peserta yang telah terenkripsi dan disimpan di IPFS, setelah memastikan akun tersebut memang terdaftar. Dengan demikian, antarmuka dapat menyajikan ringkasan identitas operasional peserta secara konsisten tanpa mengekspos atribut pribadi di *on-chain*. Rangkaian logika tersebut ditampilkan pada Gambar 4.28.

```
function getPesertaMetadata(address peserta) external view returns (string memory
cid) {
        require(pesertaList[peserta].terdaftar, "Peserta tidak terdaftar");
        return pesertaList[peserta].metadataCID;
}
```

Gambar 4.28 Kode Fungsi getPesertaMetadata

Secara keseluruhan, kombinasi fungsi-fungsi di atas menghasilkan jalur verifikasi yang ringkas, hemat biaya, dan tahan sensor: publik dapat memeriksa keabsahan sertifikat menggunakan CID, sistem dapat menelusuri ulang identitas sertifikasi secara deterministik, dan antarmuka memperoleh pointer yang diperlukan untuk menghadirkan informasi yang relevan tanpa melanggar privasi. Dengan *read path* yang jelas, index yang efisien, dan *event* yang konsisten pada modul-modul sebelumnya, proses autentikasi sertifikat menjadi transparan serta mudah diaudit dari hulu ke hilir.

4.2.6 MainContract dan Mekanisme Umum

Seluruh modul digabungkan dalam MainContract sehingga aplikasi berinteraksi melalui satu pintu. Pada saat *deploy*, pengirim transaksi ditetapkan sebagai BNSP, yang selanjutnya menjadi otoritas pengesahan LSP. Dengan komposisi ini, *modifier* akses, *event*,

dan struktur data bersama berada pada satu hirarki turunan yang konsisten seperti pada Gambar 4.29.

```
contract MainContract is
    PesertaManager,
    SertifikasiManager,
    RegistrasiLSP,
    VerifikasiPublik
{
    constructor() {
        bnsp = msg.sender; // menetapkan otoritas BNSP saat deploy
    }
}
```

Gambar 4.29 Kode Program MainContract

Rangkaian fungsi pada lima bagian di atas membentuk jalur lengkap dari pendaftaran LSP, pendaftaran peserta, pengajuan dan penyelesaian sertifikasi, hingga verifikasi publik berbasis CID.

4.3 Hasil Pengujian Sistem

4.3.1 Pengujian *Smart Contract*

Pengujian *smart contract* bertujuan memastikan alur sertifikasi berjalan sesuai rancangan, kontrol akses berbasis peran konsisten, prasyarat proses ditegakkan, serta setiap perubahan penting menerbitkan *event* sebagai jejak audit di blockchain. Lingkungan pengujian menggunakan Solidity 0.8.28 pada jaringan lokal Hardhat, dengan Chai.js untuk *assertion*. Pendekatannya mengombinasikan unit testing untuk memverifikasi fungsi inti secara terpisah dan integration testing untuk menilai interaksi antarmodul sesuai skenario pada Bab 3.

Sebelum setiap skenario, lingkungan diinisialisasi ulang agar kondisi awal bersih dan terkontrol. Kontrak di-*deploy*, akun peran disiapkan untuk BNSP, LSP, dan Peserta, lalu data minimal seperti registrasi LSP atau pendaftaran Peserta diisi sesuai kebutuhan. Hasil eksekusi diuji melalui pembacaan *state*, pengecekan pesan *revert* untuk kasus penolakan yang diharapkan, dan verifikasi parameter *event*.

Ringkasan skenario dan hasil uji disajikan pada Tabel 4.1. Tabel ini memetakan jalur utama yang telah ditetapkan pada Bab 3, meliputi registrasi dan verifikasi LSP, pendaftaran Peserta, pengajuan sertifikasi, pengisian nilai, penetapan kelulusan, penerbitan sertifikat, serta verifikasi publik berbasis CID.

Tabel 4.1 Ringkasan Skenario Uji *Smart Contract*

No	Fokus Uji	Fungsi	Kondisi Awal	Ekspektasi	Status
1.	Registrasi LSP	tambahLSP()	Akun pemanggil berwenang, metadata valid	LSP terdaftar, *event* LSPDitambahkan terbit, CID tercatat	Lulus
2.	Verifikasi LSP	verifikasiLSP()	LSP sudah terdaftar, pemanggil BNSP	Status LSP menjadi terverifikasi, *event* verifikasi terbit	Lulus
3.	Pendaftaran Peserta	daftarPeserta()	Alamat belum terdaftar, data valid	Peserta aktif, *event* PesertaTerdaftar, *state* konsisten	Lulus
4.	Pengajuan Sertifikasi	ajukanSertifikasi()	Peserta aktif, belum ada sertifikasi aktif	Sertifikasi aktif, *event* SertifikasiDiajukan, penunjuk sertifikasi aktif terset	Lulus
5.	Input Nilai	inputNilai()	LSP terverifikasi, sertifikasi aktif	Nilai.sudahInput == true, *event* NilaiDiinput terbit	Lulus
6.	Penetapan Kelulusan	updateKelulusan()	Nilai lengkap, LSP penilai sah	Status lulus/tidak lulus terset, tanggal selesai tercatat, *event* KelulusanDiupdate	Lulus
7.	Penerbitan Sertifikat	terbitkanSertifikat()	Peserta lulus, CID valid, belum pernah terbit	CID tersimpan, *event* CertificateIssued, *state* peserta konsisten	Lulus
8.	Verifikasi Publik (valid)	verifikasiPublikByCID()	CID terdaftar	Keluaran valid = true dan metadata minimum tampil	Lulus

9.	Verifikasi Publik (tidak valid)	verifikasiPublikByCID()	CID acak/tidak ada	Keluaran valid = false / “tidak ditemukan”	Lulus

Penjelasan hasil uji berikut mengikat langsung ke Tabel 4.1. Baris 1 dan 2 memastikan gerbang otorisasi LSP berjalan benar: pendaftaran berhasil untuk akun yang berwenang dan pengesahan hanya dapat dilakukan oleh BNSP. Baris 3 dan 4 memvalidasi identitas Peserta dan pengajuan sertifikasi, termasuk pencegahan lebih dari satu sertifikasi aktif pada saat yang sama. Baris 5 dan 7 menilai fase asesmen yang menuntut nilai lengkap sebelum kelulusan serta pencatatan CID sertifikat, dengan *event* domain sebagai bukti audit. Baris 8 dan 9 memverifikasi akses publik berbasis CID untuk menentukan keabsahan tanpa membuka dokumen.

Rekap eksekusi otomatis seluruh skenario ditunjukkan pada Gambar 4.30. *Log* Hardhat memperlihatkan enam kelompok uji yang seluruhnya lulus: Deploy Kontrak, Siklus LSP, Manajemen Peserta, Proses Sertifikasi, Verifikasi Publik, dan Kasus Keamanan. Kelompok “Siklus LSP” menguatkan baris 1 dan 2 pada Tabel 4.1, “Manajemen Peserta” menguatkan baris 3, “Proses Sertifikasi” menguatkan baris 4 sampai 7, sedangkan “Verifikasi Publik” menguatkan baris 8 dan 9. Kelompok “Kasus Keamanan” menegaskan *guard* generik seperti validasi alamat, validasi *string*, penolakan tindakan oleh pihak tidak berwenang, jaminan keunikan ID sertifikasi, dan konsistensi penyimpanan riwayat.

```
murito@LAPTOP-NGMIUV9S:~/skillchain$ npx hardhat test


  Pengujian DApp Sertifikasi
    1. Deploy Kontrak
      ✓ mengatur alamat BNSP dengan benar
      ✓ mulai dengan jumlah peserta nol

    2. Siklus Hidup LSP
      ✓ mengizinkan wallet mendaftar sebagai LSP
      ✓ menolak BNSP yang mencoba mendaftar sebagai LSP
      ✓ menolak pendaftaran LSP yang duplikat
      ✓ mengizinkan BNSP memverifikasi LSP yang terdaftar
      ✓ mengizinkan BNSP menolak pendaftaran LSP
      ✓ mengizinkan LSP memperbarui metadata-nya
```

```
      ✓ mengizinkan BNSP menambahkan LSP ke daftar tunggu

    3. Manajemen Peserta
      ✓ mengizinkan pendaftaran peserta baru
      ✓ menolak pendaftaran peserta yang duplikat
      ✓ menolak metadata CID kosong
      ✓ mengizinkan peserta memperbarui metadata

    4. Proses Sertifikasi
      ✓ mengizinkan peserta mengajukan sertifikasi
      ✓ menolak lebih dari satu sertifikasi aktif per peserta
      ✓ mengizinkan LSP terverifikasi menyetujui sertifikasi
      ✓ mengizinkan LSP terverifikasi menolak sertifikasi
      ✓ mengizinkan peserta membatalkan sertifikasi
      ✓ mencegah non-LSP mengubah status sertifikasi
      ✓ mengizinkan LSP menginput nilai peserta satu kali saja

    5. Verifikasi Publik
      ✓ memverifikasi sertifikasi yang valid
      ✓ mengembalikan false untuk ID sertifikasi yang tidak valid
      ✓ bisa mengambil CID sertifikat
      ✓ bisa mengambil detail verifikasi lengkap
      ✓ bisa mengambil status peserta
      ✓ bisa mengambil nama skema sertifikasi
      ✓ memverifikasi sertifikasi berdasarkan CID sertifikat yang valid
      ✓ mengembalikan tidak valid untuk CID sertifikat yang tidak terdaftar

    6. Kasus Keamanan
      ✓ menerapkan validasi alamat yang valid
      ✓ menerapkan validasi string tidak boleh kosong
      ✓ mencegah verifikasi LSP oleh pihak tidak berwenang
      ✓ menghasilkan ID sertifikasi yang unik
      ✓ menyimpan riwayat sertifikasi dengan benar

  33 passing (1s)
```

Gambar 4.30 Hasil Pengujian *Smart Contract*

Secara keseluruhan, hasil pada Tabel 4.1 dan Gambar 4.30 menunjukkan implementasi memenuhi jalur fungsional yang dirancang, kontrol akses berbasis peran berjalan ketat, invarian proses terjaga, dan seluruh perubahan penting dapat ditelusuri melalui *state* serta *event*. Temuan ini menjadi dasar bahwa *smart contract* siap mendukung pengujian antarmuka pada subbab berikutnya.

Selain menampilkan hasil akhir seluruh skenario pengujian, beberapa cuplikan kode pengujian berikut disertakan untuk memberikan gambaran konkret mengenai cara pengujian dilakukan pada lingkungan Hardhat menggunakan framework Chai.js. Setiap cuplikan dirancang untuk mewakili jenis pengujian yang berbeda, mulai dari pengujian fungsi utama, pengendalian hak akses, validasi logika status, hingga pencatatan *event on-chain* sebagai mekanisme audit.

Contoh pada Gambar 4.31 memperlihatkan peserta aktif memanggil ajukanSertifikasi() lalu sistem menghasilkan ID sertifikasi yang sah, menetapkan penunjuk sertifikasi aktif, menambah total sertifikasi, dan menerbitkan *event* SertifikasiDiajukan. Rangkaian asersi memastikan ID terbentuk benar, *state* konsisten, serta parameter *event* sesuai skema yang dipilih. Cuplikan ini berkaitan langsung dengan Tabel 4.1 baris 4.

```
it("mengizinkan peserta mengajukan sertifikasi", async function () {
        const sertifikasiID = await submitCertification(
            peserta1,
            SkemaSertifikasi.Okupasi_PJOI_Pengendalian_Pencemaran_Udara
        );
        expect(sertifikasiID).to.be.properAddress;
        expect(await mainContract.getTotalSertifikasi()).to.equal(1n);
});
```

Gambar 4.31 Kode Pengujian Sertifikasi Berhasil

Berikutnya, Gambar 4.32 menunjukkan LSP terverifikasi memanggil updateKelulusan() sehingga *state* berubah menjadi lulus dan tidak aktif, sertifikatCID tersimpan, serta *event* KelulusanDiupdate terbit dengan argumen yang tepat. Asersi juga memverifikasi bahwa perubahan status dan pencatatan CID selaras dengan rancangan audit *on-chain*. Cuplikan ini berhubungan dengan Tabel 4.1 baris 6–7.

```
it("mengizinkan LSP terverifikasi menyetujui sertifikasi", async function () {
        const sertifikasiID = await submitCertification(peserta1);

        const tx = await
mainContract.connect(lsp1).updateKelulusan(sertifikasiID, "QmSertifikatCID");
        await expect(tx)
            .to.emit(mainContract, "KelulusanDiupdate")
            .withArgs(lsp1.address, sertifikasiID, "QmSertifikatCID", anyValue);

        const data = await mainContract.getSertifikasi(sertifikasiID);
        expect(data.lulus).to.be.true;
        expect(data.aktif).to.be.false;
});
```

Gambar 4.32 Kode Pengujian Emisi *Event* Saat Kelulusan

Untuk menjaga konsistensi alur, Gambar 4.33 menguji invarian “satu sertifikasi aktif per peserta”. Setelah pengajuan pertama berhasil, upaya pengajuan kedua harus ditolak dengan pesan kesalahan yang tepat. Hasil ini menegaskan bahwa kontrak menegakkan prasyarat proses sebelum menerima transaksi baru. Cuplikan ini terhubung dengan Tabel 4.1 baris 4.

```
it("menolak lebih dari satu sertifikasi aktif per peserta", async function () {
        await submitCertification(peserta1);
        await expect(submitCertification(peserta1,
SkemaSertifikasi.Okupasi_PJ_Pengendalian_Pencemaran_Udara)).to.be.revertedWith(
            "Masih ada sertifikasi aktif"
        );
});
```

Gambar 4.33 Kode Uji Satu Sertifikasi Aktif per Peserta

Terakhir, Gambar 4.34 menguji penegakan peran pada fungsi verifikasi lembaga. Upaya verifikasi oleh akun non-BNSP dipastikan gagal (*revert*) dengan pesan “Hanya BNSP”, sehingga kebijakan otorisasi di tingkat kontrak terbukti bekerja konsisten. Cuplikan ini selaras dengan Tabel 4.1 baris 2.

```
it("mencegah verifikasi LSP oleh pihak tidak berwenang", async function () {
        await mainContract.connect(lsp2).daftarLSP("QmLSP2MetadataCID");
        await expect(
            mainContract.connect(peserta1).verifikasiLSP(lsp2.address, "QmSurat")
        ).to.be.revertedWith("Hanya BNSP");
});
```

Gambar 4.34 Kode Uji Kontrol Akses: Only BNSP

Hasil pengujian setelah rangkaian cuplikan kode pada Gambar 4.31 hingga Gambar 4.34 memperlihatkan bahwa kontrak tidak hanya berfungsi secara logis, tetapi juga konsisten menegakkan batas otorisasi. Merujuk Tabel 4.1, seluruh jalur utama berjalan sesuai rancangan dan setiap perubahan *state* terikat pada prasyarat yang tepat. Upaya di luar kewenangan ditolak dengan pesan kesalahan yang jelas, sedangkan eksekusi yang sah selalu meninggalkan jejak melalui *event on-chain*.

Kontrol akses berbasis peran terbukti efektif. Modifier onlyBNSP, onlyLSP, dan onlyPeserta bekerja konsisten menolak pemanggilan fungsi oleh akun yang tidak berwenang, sebagaimana ditunjukkan pada skenario verifikasi lembaga dan penetapan hasil oleh LSP. Hasil ini selaras dengan baris-baris verifikasi pada Tabel 4.1, serta diperkuat oleh pengujian kontrol akses pada Gambar 4.34.

Konsistensi alur juga terjaga. Pengajuan hanya dapat dilakukan oleh peserta aktif, keputusan kelulusan baru valid setelah nilai tercatat, dan invarian “satu sertifikasi aktif per peserta” ditegakkan, sebagaimana terlihat pada Gambar 4.33. Kontrak mencegah lompatan status dan memastikan setiap tahap dicapai melalui urutan yang benar, sehingga tidak muncul keadaan ambigu.

Dari sisi keterlacakan, *event* domain seperti PesertaTerdaftar, SertifikasiDiajukan, KelulusanDiupdate, dan SertifikasiBatal tercatat dengan parameter yang akurat. Pencatatan sertifikatCID dan indeks sertifikasiByCID memungkinkan verifikasi publik tanpa membuka dokumen, sekaligus menjaga keterhubungan antara data *on-chain* dan objek terenkripsi di IPFS. Dependensi antarmuka terhadap *event* untuk menyelaraskan tampilan status juga terkonfirmasi pada skenario uji yang dirujuk oleh Tabel 4.1.

Secara keseluruhan, pengujian membuktikan bahwa *smart contract* memenuhi spesifikasi fungsional, aman terhadap pelanggaran peran, dan transparan untuk diaudit. Kombinasi kontrol akses, validasi alur status, dan pencatatan *event* memastikan proses dari

pendaftaran LSP, pendaftaran peserta, asesmen, hingga verifikasi publik berbasis CID berjalan sesuai tujuan penelitian dan siap ditelusuri melalui jejak transaksi di blockchain.

4.3.2 Pengujian *Black-Box*

Pengujian *black-box* menilai kesesuaian fungsional dari sudut pandang pengguna, berfokus pada hubungan *input* dan *output* tanpa meninjau implementasi kode. Antarmuka berbasis React.js terhubung ke *node* lokal Hardhat melalui Ethers.js, autentikasi dilakukan dengan MetaMask, dan penyimpanan berkas memanfaatkan IPFS (Pinata) sesuai rancangan pada Bab 3. Skenario dieksekusi dengan peran BNSP, LSP, Peserta, dan Publik menurut otoritasnya masing-masing.

Setiap skenario menelusuri satu alur fungsional utuh. Pada Login MetaMask, sistem harus mengenali alamat *wallet* dan jaringan tanpa perlu pemuatan ulang halaman. Pada Registrasi LSP dan Verifikasi LSP, UI diharapkan menampilkan perubahan status yang sejalan dengan *state on-chain*. Pada Pendaftaran Peserta, *dashboard* aktif setelah *event* Peserta terdaftar terbit. Fase Input Nilai dan Penetapan Kelulusan menuntut konsistensi antara data yang dibaca ulang dari *smart contract* dan tampilan antarmuka. Verifikasi Publik memeriksa CID sah dan tidak sah untuk memastikan jalur validasi berjalan tegas. Unduh sertifikat menguji proses dekripsi klien agar berkas hasil terbit dapat dibuka dengan benar.

Hasil pengamatan terperinci dan keterkaitan dengan ekspektasi disajikan pada Tabel 4.2. Secara keseluruhan, Tabel 4.2 menunjukkan bahwa seluruh skenario uji memenuhi ekspektasi fungsional.

Tabel 4.2 Hasil Pengujian *Black-Box*

No	Fitur	Deskripsi	Keluaran	Hasil Aktual	Status
1.	Login MetaMask	Autentikasi pengguna dengan *wallet*	*Wallet* terhubung, alamat terdeteksi, jaringan sesuai	Alamat *wallet* tampil di UI, status koneksi Terhubung	Sesuai
2.	Registrasi LSP	LSP mengajukan pendaftaran dan unggah dokumen pendukung	*event* LSP terdaftar terbit, status menunggu verifikasi	Pendaftaran tercatat, *event* sesuai, status Menunggu Verifikasi	Sesuai
3.	Verifikasi LSP (BNSP)	Verifikasi pada LSP pada	Status LSP terverifikasi,	Status berubah menjadi	Sesuai

		dashboard BNSP	sinkron dengan *state on-chain*	Terverifikasi, UI dan *on-chain* konsisten	
4.	Pendaftaran Peserta	Isi formulir pendaftaran	*Event* peserta terdaftar terbit, *dashboard* peserta aktif	Peserta aktif, *event* sesuai, navigasi ke *dashboard* peserta berhasil	Sesuai
5.	Input Nilai (LSP)	LSP mengisi nilai komponen asesmen	Nilai terbaca ulang dengan benar, status asesmen konsisten	Nilai tersimpan, baca ulang sesuai, tidak ada inkonsistensi data	Sesuai
6.	Penetapan Kelulusan	LSP set lulus atau gagal, dan jika lulus unggah sertifikat	Event kelulusan terbit, *state* dan UI sinkron	Status dan *timestamp* tampil, *event* tercatat, UI sesuai	Sesuai
7.	Verifikasi Publik (CID valid)	Pengguna memasukkan CID benar	Status valid dan metadata minimum tampil	Status valid ditampilkan, metadata minimum sesuai	Sesuai
8.	Verifikasi Publik (CID tidak valid)	Pengguna memasukkan CID acak atau salah	Status tidak ditemukan	UI menampilkan tidak ditemukan dengan tegas	Sesuai
9.	Pelacakan Status	Peserta memeriksa status tiap aksi setelah pendaftaran	Status sesuai pembaruan *event on-chain*	Status pada UI tersinkron setelah konfirmasi transaksi	Sesuai
10.	Unduh sertifikat	Peserta mengunduh sertifikat	Berkas hasil dekripsi valid dan dapat dibuka	Dekripsi AES-256-CBC berhasil, berkas dapat dibuka	Sesuai

Autentikasi melalui MetaMask berjalan stabil. Alamat *wallet* terdeteksi konsisten tanpa perlu memuat ulang halaman, jaringan yang aktif dikenali dengan benar, dan status koneksi di antarmuka segera berpindah menjadi “Terhubung”. Kondisi ini memastikan proses login tidak mengganggu alur uji sehingga fokus dapat berlanjut ke skenario fungsional berikutnya.

Pada alur kelembagaan, registrasi dan verifikasi LSP menunjukkan perilaku yang sesuai rancangan. Pengajuan pendaftaran menimbulkan perubahan status menjadi “Menunggu” dan menerbitkan jejak audit melalui *event* seperti LSPDidaftarkan atau LSPDitambahkan sesuai jalur administratif yang dipilih. Tindakan pengesahan oleh BNSP memicu *event*

LSPDiverifikasi, sedangkan keputusan penolakan menghasilkan LSPDitolak dengan alasan yang tercatat. Tampilan status di UI selaras dengan pembacaan *state* dari kontrak, sehingga pengguna lembaga memperoleh umpan balik yang jelas setelah transaksi dikonfirmasi.

Pada sisi peserta, pendaftaran memunculkan *dashboard* aktif dan *event* PesertaTerdaftar, diikuti pengajuan asesmen yang menghasilkan *event* SertifikasiDiajukan beserta penunjuk sertifikasi yang unik. Pembatasan satu sertifikasi aktif per peserta tertegap, sehingga tidak ada duplikasi proses berjalan bersamaan. Pengisian nilai oleh LSP dapat dibaca ulang dari kontrak dengan hasil yang konsisten di antarmuka, lalu penetapan hasil memperbarui status kelulusan, mematikan sertifikasi yang sudah selesai, mencatat waktu penyelesaian, dan menyimpan sertifikatCID. Perubahan tersebut tercermin di UI segera setelah transaksi terkonfirmasi, dan terekam melalui *event* KelulusanDiupdate atau SertifikasiBatal untuk kasus tidak lulus.

Fitur verifikasi publik berbasis CID berfungsi sesuai tujuan. Untuk CID yang tercatat, antarmuka menampilkan status valid dan metadata minimum yang diizinkan tanpa membuka isi dokumen. Untuk CID acak atau tidak terdaftar, antarmuka menampilkan “tidak ditemukan” secara tegas. Proses unduh sertifikat yang terenkripsi berhasil didekripsi di sisi klien menggunakan AES-256-CBC, sehingga kerahasiaan tetap terjaga sementara integritas berkas dijamin oleh sifat *content addressing*: setiap perubahan konten akan menghasilkan CID yang berbeda.

Aspek transparansi dan akuntabilitas terjaga melalui *event* domain yang konsisten, seperti LSPDidaftarkan, LSPDiverifikasi, LSPDitolak, LSPDitambahkan, PesertaTerdaftar, MetadataDiupdate, SertifikasiDiajukan, KelulusanDiupdate, dan SertifikasiBatal. *Event-event* ini dapat ditelusuri kembali melalui *log* transaksi untuk menjawab pertanyaan “siapa melakukan apa dan kapan”, tanpa akses ke data sensitif. Di saat yang sama, kontrol akses berbasis *modifier* onlyBNSP, onlyLSP, dan onlyPeserta menolak eksekusi dari akun yang tidak berwenang, dibuktikan oleh pesan *revert* yang tepat seperti “Hanya BNSP”, “Hanya LSP”, “Hanya peserta aktif”, “Alamat tidak valid”, “String tidak boleh kosong”, atau “Masih ada sertifikasi aktif”.

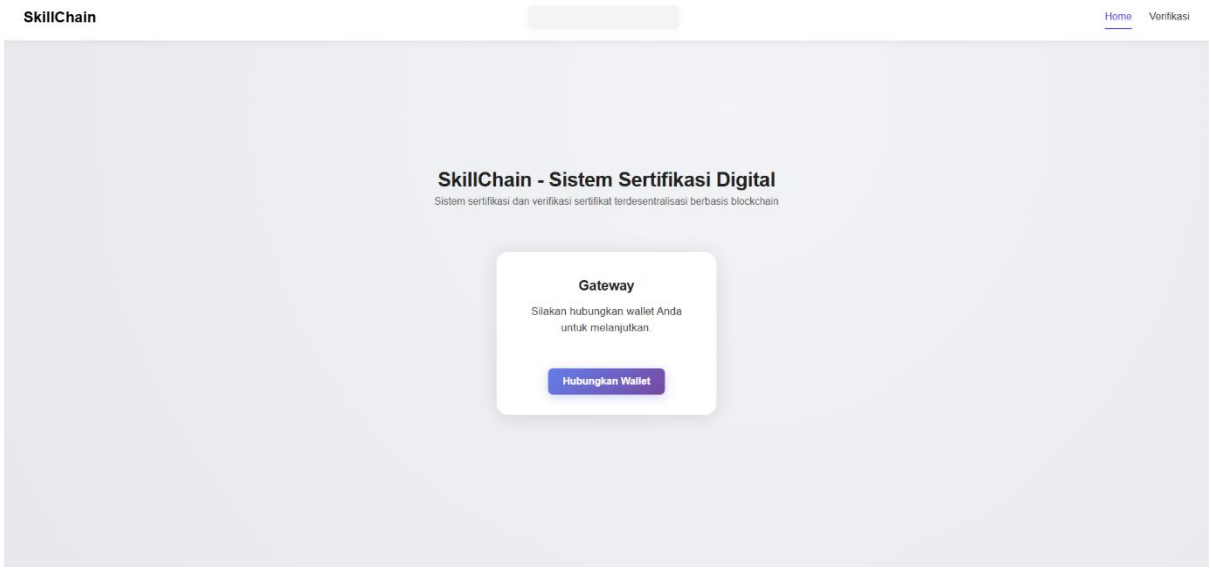
Secara keseluruhan, hasil pengujian *black-box* memperlihatkan bahwa integrasi antara UI, *smart contract*, dan IPFS berjalan selaras. Perubahan *state* di blockchain segera tercermin pada antarmuka, jejak audit *on-chain* lengkap dan dapat diverifikasi publik, serta jalur privasi tetap terjaga melalui enkripsi sisi klien. Seluruh poin pada Tabel 4.2 terpenuhi, sehingga dari

sudut pandang pengguna sistem telah berfungsi sesuai rancangan: aman, konsisten, dan transparan untuk mendukung proses sertifikasi kompetensi *end-to-end*.

4.4 Ulasan Antarmuka Pengguna

Ulasan antarmuka pengguna difokuskan pada halaman-halaman utama DApp menurut peran: BNSP, LSP, Peserta, dan Publik. Ulasan ini memetakan alur interaksi pada antarmuka pengguna dengan fungsi *smart contract* pada Subbab 4.1, termasuk validasi, pembatasan akses berbasis peran, dan artefak audit seperti *event* serta CID. Ketika peran dari *wallet* tidak sesuai, pengguna dialihkan ke halaman yang tepat dan tidak ada transaksi yang dikirim.

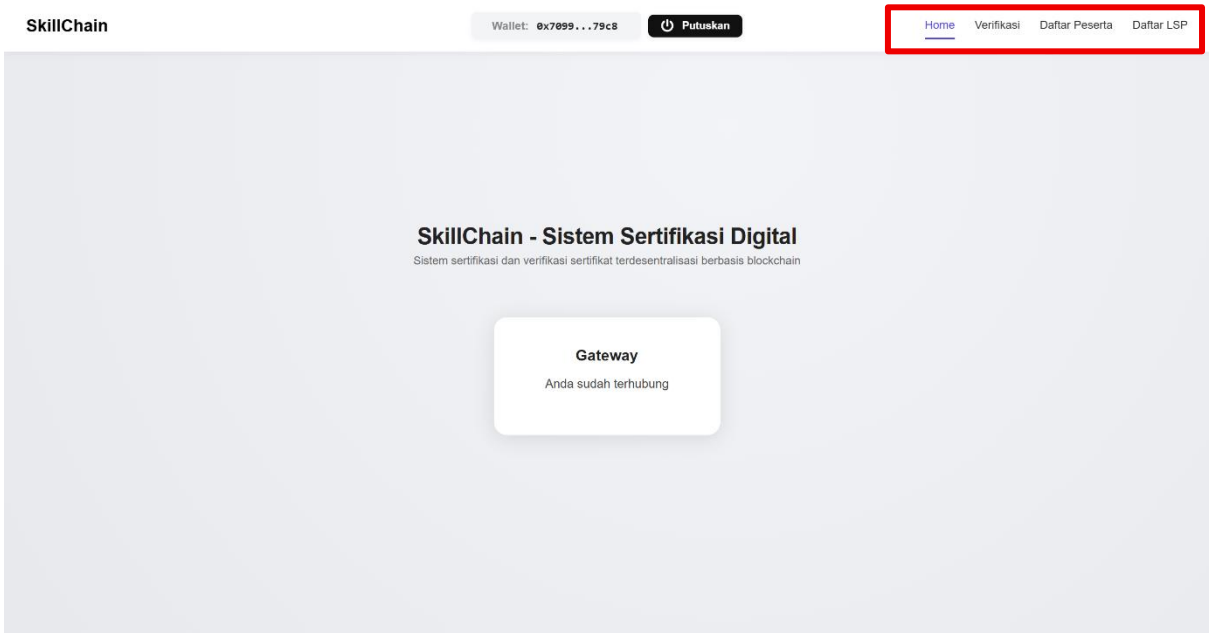
Sebagai gerbang awal, halaman Home menampilkan identitas aplikasi, menu (Home dan Verifikasi), serta Gateway untuk menghubungkan *wallet*. Sampai koneksi berhasil, fitur peran dibatasi. Setelah terhubung, aplikasi menentukan peran dan mengarahkan pengguna ke *dashboard* yang tepat, sementara menu Verifikasi tetap terbuka untuk publik, sebagaimana ditunjukkan pada Gambar 4.35.



Gambar 4.35 Halaman *Home*

Pada kondisi sudah terhubung, halaman Home berfungsi sebagai ringkasan status koneksi, bar navigasi menampilkan cuplikan alamat *wallet* beserta tombol putuskan, sedangkan kartu Gateway berubah menjadi “Anda sudah terhubung”. Jika *wallet* belum memiliki peran apa pun, sistem menampilkan tautan Daftar Peserta dan Daftar LSP sebagai pintu masuk registrasi. Setelah salah satu peran terdaftar, tautan registrasi tersebut disembunyikan dan diganti dengan menu *dashboard* sesuai peran. Pergantian akun di MetaMask memicu

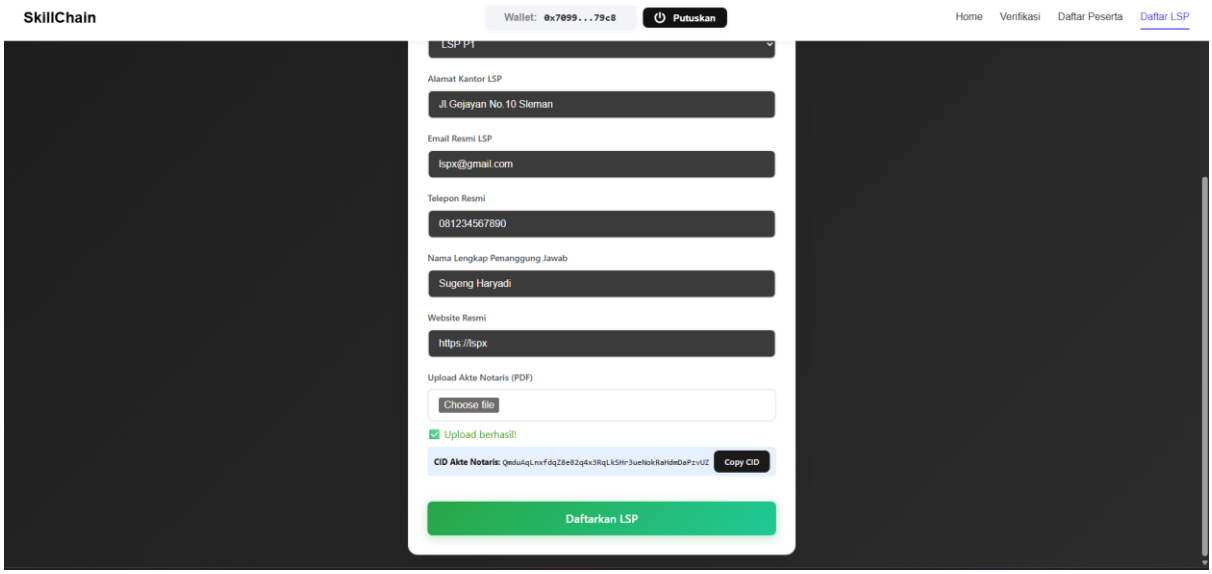
penyegaran peran dan penyesuaian menu secara otomatis. Perlu dicatat, proses penyambungan tidak mengeksekusi transaksi *on-chain* dan konfirmasi MetaMask baru muncul ketika pengguna melakukan aksi yang mengubah *state*, sebagaimana ditunjukkan pada Gambar 4.36.



Gambar 4.36 Halaman Home Ketika Sudah Terhubung dengan *Wallet*

4.4.1 Halaman Pendaftaran LSP

Halaman ini digunakan calon LSP untuk mendaftarkan lembaga sebelum memperoleh akses ke *Dashboard* LSP. Pengguna mengisi metadata lembaga seperti pada Gambar 4.37.



Gambar 4.37 Halaman Pendaftaran LSP

Setelah seluruh kolom terisi, aplikasi melakukan validasi sisi klien (semua kolom terisi, format email dan URL, serta tipe berkas PDF). Jika ada isian yang belum benar, antarmuka menampilkan pesan korektif dan tidak mengirim transaksi. Bila valid, berkas Akte Notaris dienkripsi terlebih dahulu kemudian diunggah ke IPFS (Pinata) sehingga diperoleh CID_akte .

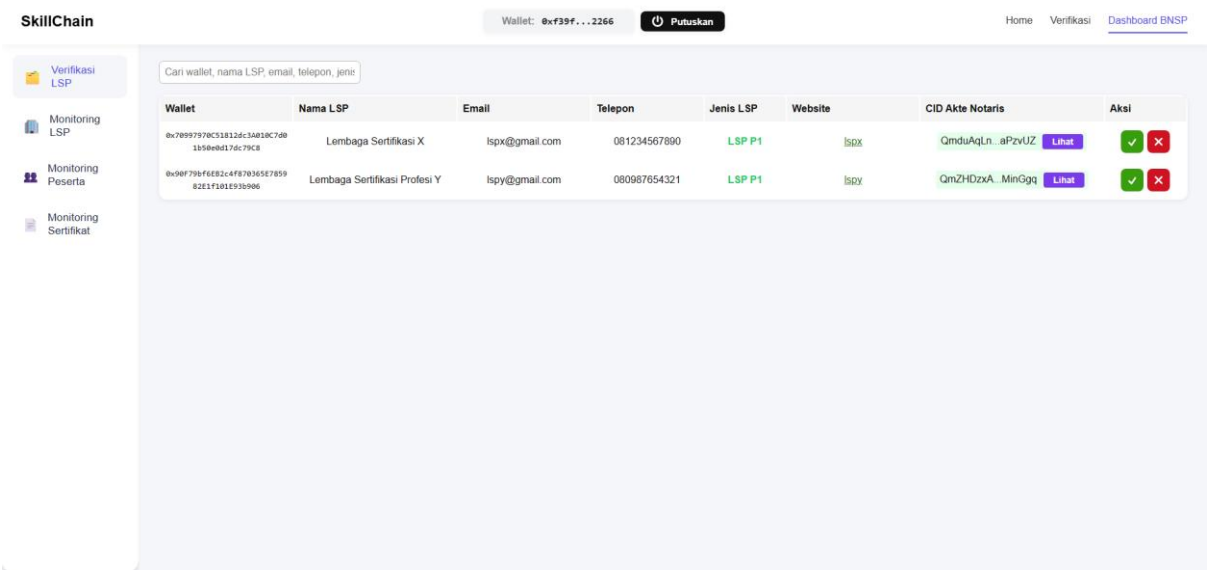
Berikutnya aplikasi menyusun dokumen metadata dalam format JSON yang memuat informasi identitas lembaga yang sudah diisikan pada formulir pendaftaran. Dokumen JSON tersebut dienkripsi di sisi klien, lalu hasil enkripsinya diunggah ke Pinata untuk memperoleh CID_metadata . Tahap final adalah pemanggilan fungsi kontrak daftarLSP(CID_metadata) .

Sesudah transaksi tercatat, antarmuka menampilkan status "Menunggu verifikasi BNSP", menyimpan jejak CID yang diajukan, dan menutup seluruh fitur LSP sampai verifikasi selesai. Bila BNSP menyetujui melalui *dashboard* BNSP, status berubah menjadi "Terverifikasi" dan menu menuju *Dashboard* LSP otomatis aktif. Kegagalan unggah IPFS atau penandatanganan transaksi ditangani dengan notifikasi yang jelas tanpa mengubah *state* apa pun.

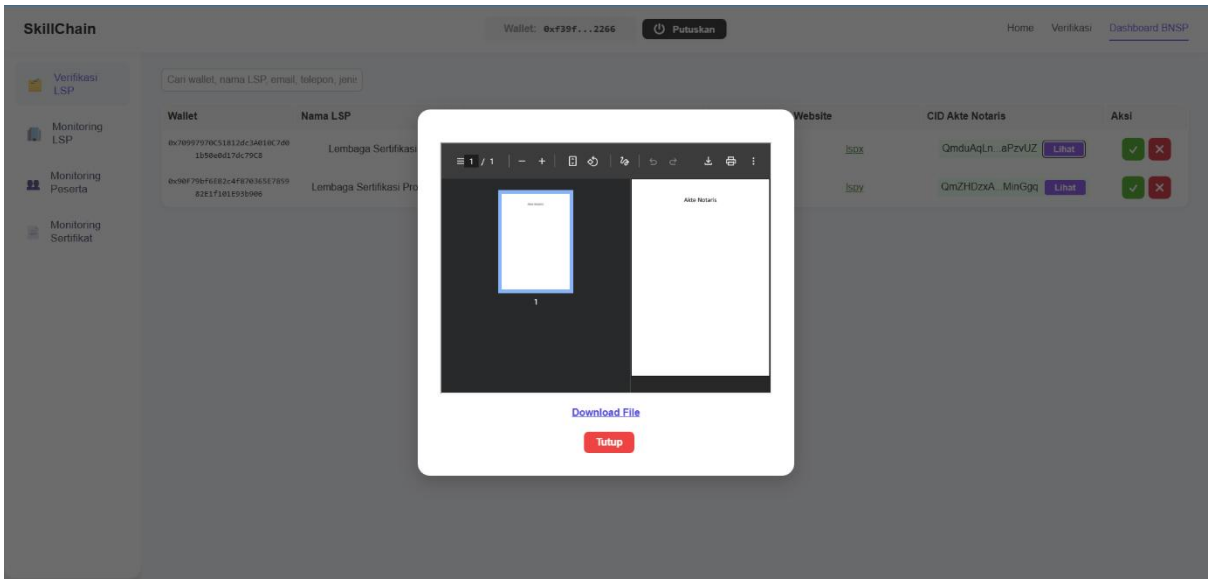
4.4.2 *Dashboard* BNSP

Dashboard BNSP berfungsi sebagai pusat kendali otorisasi LSP dan pemantauan aktivitas di seluruh platform. Akses ke halaman ini dibatasi hanya untuk *wallet* yang memiliki peran BNSP di kontrak. Jika peran tidak sesuai, antarmuka mengarahkan pengguna ke halaman yang tepat dan tidak ada transaksi yang dikirim. Navigasi sisi kiri memuat empat menu: Verifikasi LSP, Monitoring LSP, Monitoring Peserta, dan Monitoring Sertifikat.

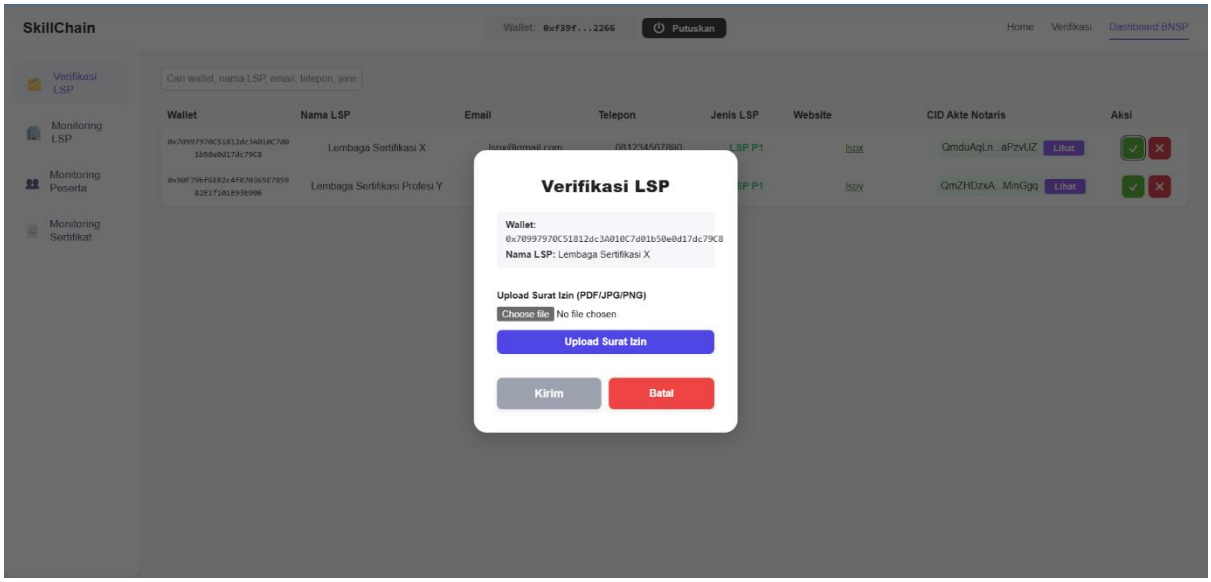
Verifikasi LSP menampilkan daftar pengajuan lembaga yang belum diputuskan dalam bentuk tabel seperti pada Gambar 4.38. Tombol Lihat pada kolom CID membuka pratinjau berkas yang sudah diupload oleh LSP ketika melakukan pendaftaran yang diambil dari IPFS berdasarkan CID, lengkap dengan opsi Download seperti terlihat pada Gambar 4.39. Untuk memutuskan, pengguna menekan ikon centang untuk verifikasi atau ikon silang untuk penolakan. Saat memilih verifikasi, muncul tampilan untuk mengunggah surat izin, tombol Upload Surat Izin untuk mengirim berkas terenkripsi ke Pinata hingga memperoleh CID surat izin, lalu tombol Kirim untuk menyimpan keputusan ke kontrak beserta CID tersebut seperti pada Gambar 4.40. Jika verifikasi berhasil, status lembaga akan berpindah ke daftar aktif pada menu Monitoring LSP. Apabila ditolak, keputusan tercatat dan lembaga tidak memperoleh akses operasional.



Gambar 4.38 Halaman Verifikikasi LSP

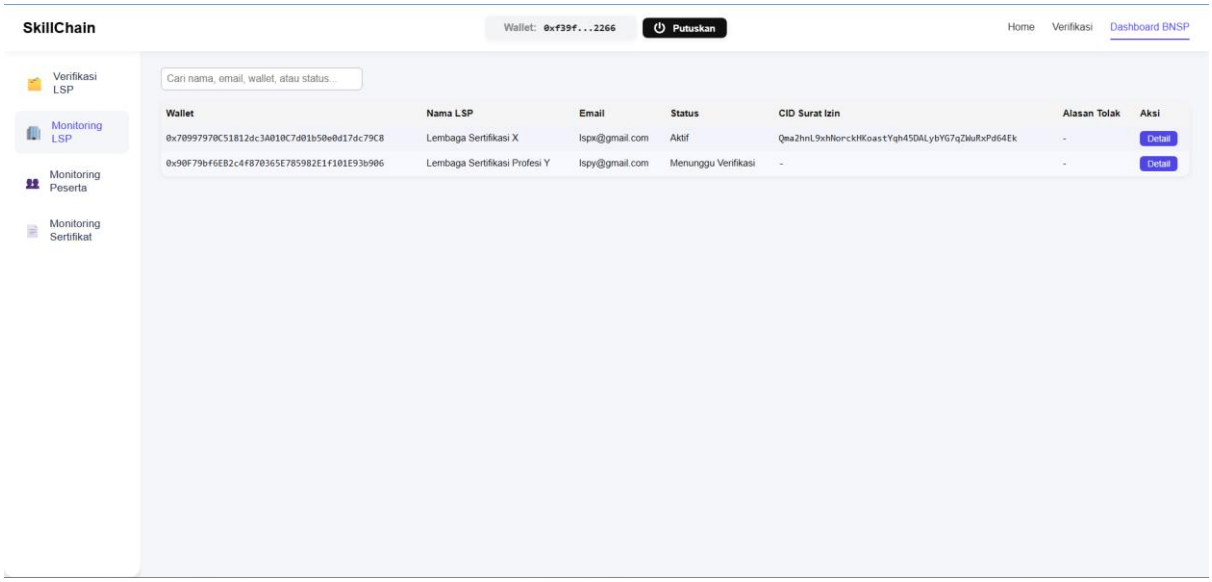


Gambar 4.39 Tampilan Proses Melihat Berkas LSP



Gambar 4.40 Tampilan Proses Unggah Surat Izin LSP

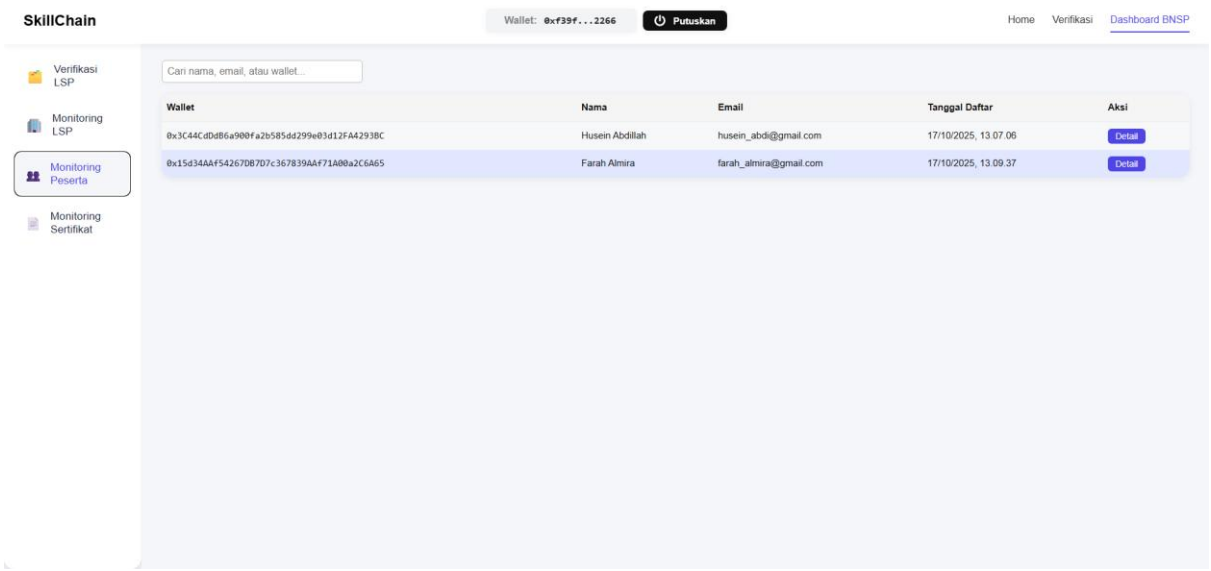
Monitoring LSP menampilkan daftar seluruh LSP beserta statusnya (Aktif atau Menunggu Verifikasi). Kotak pencarian di bagian atas memudahkan penyaringan berdasarkan nama, *wallet*, atau status. Tautan pada CID memungkinkan inspeksi cepat dokumen izin di IPFS guna keperluan audit terlihat seperti pada Gambar 4.41.



Gambar 4.41 Halaman Monitoring LSP

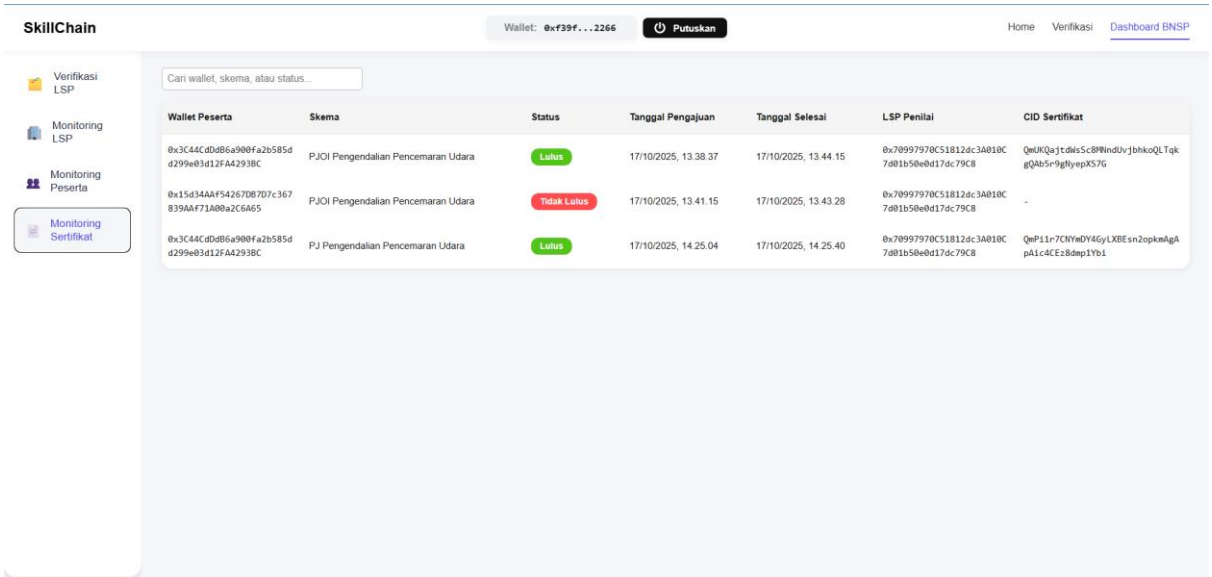
Monitoring Peserta menampilkan seluruh peserta terdaftar di sistem dalam bentuk tabel. Tombol Detail membuka ringkasan identitas peserta seperti pada Gambar 4.42. Fasilitas

pencarian dapat memfilter berdasarkan nama, email, atau *wallet* untuk kebutuhan pengawasan.



Gambar 4.42 Halaman Monitoring Peserta

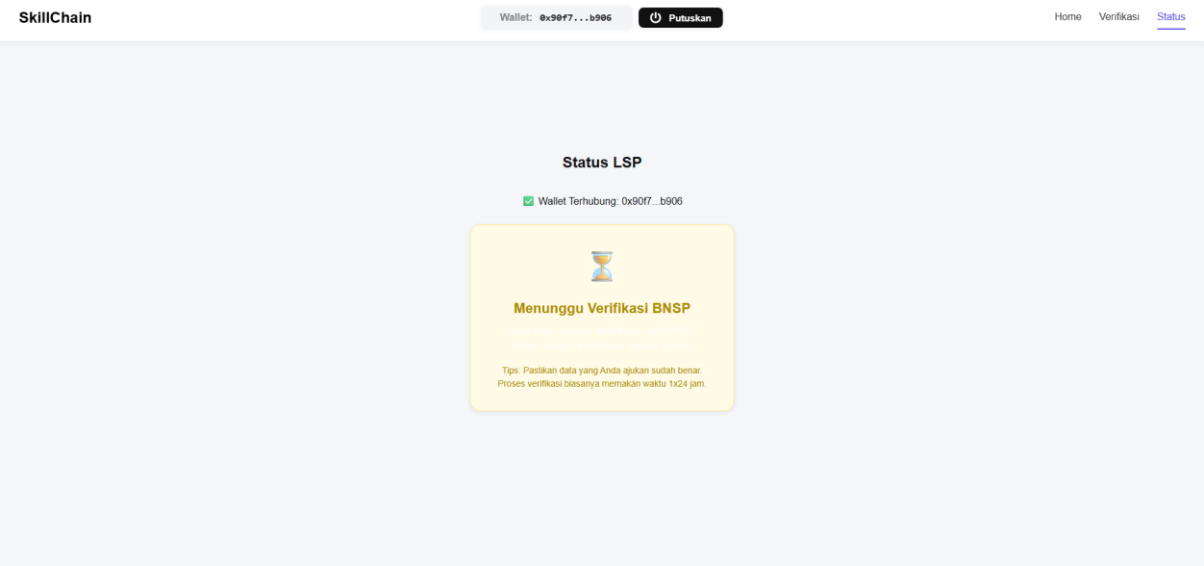
Monitoring Sertifikat menampilkan daftar sertifikasi yang sudah diambil oleh setiap peserta. Untuk peserta yang sudah dinyatakan lulus, pada kolom CID akan menampilkan CID sertifikat peserta di IPFS yang dapat ditelusuri oleh publik seperti pada Gambar 4.43. Untuk peserta yang dinyatakan tidak lulus, pada kolom CID akan kosong. Terdapat fitur pencarian sehingga BNSP dapat melakukan audit cepat.



Gambar 4.43 Halaman Monitoring Sertifikat

4.4.3 *Dashboard* LSP

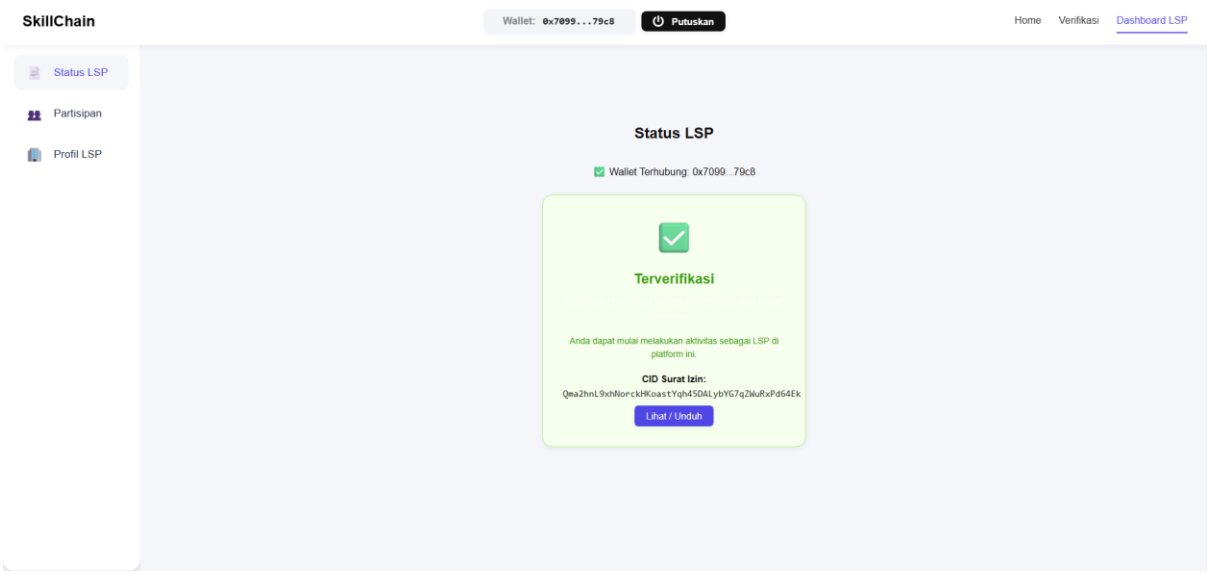
Akses ke *dashboard* LSP hanya bisa diakses bagi LSP yang sudah terverifikasi oleh BNSP. Jika belum, pengguna otomatis diarahkan ke halaman Status LSP yang menampilkan kartu informasi status sedang menunggu konfirmasi, seluruh fitur operasional LSP dinonaktifkan sampai proses verifikasi selesai seperti terlihat pada Gambar 4.44.



Gambar 4.44 Halaman Status LSP Ketika Menunggu Proses Verifikasi

h. Status LSP

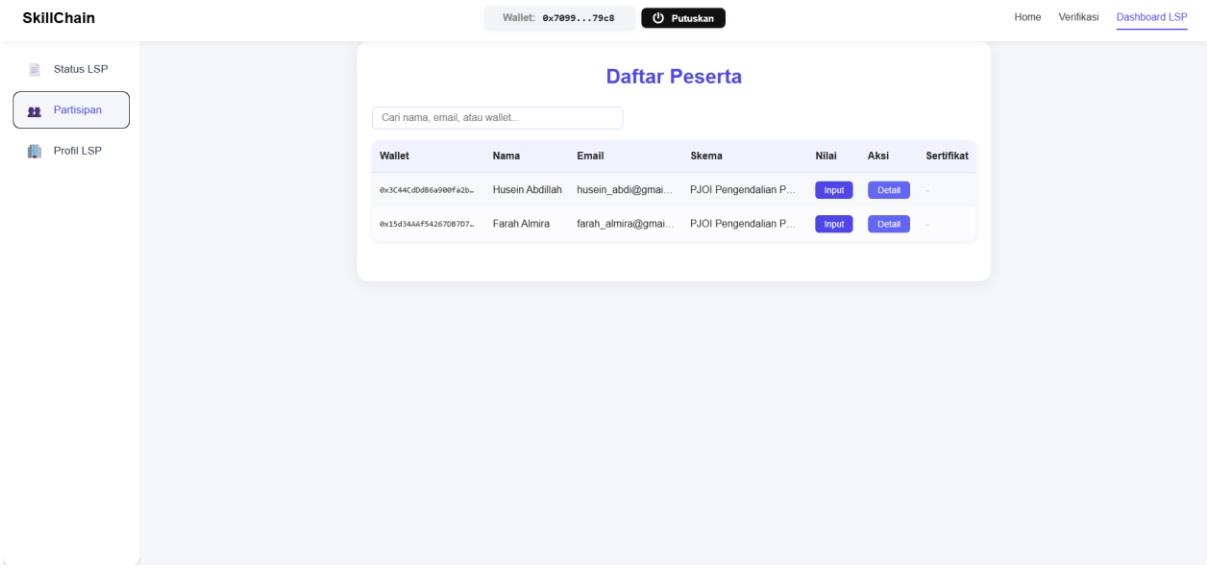
Bagian ini menampilkan ringkasan *wallet* yang terhubung, status verifikasi, serta CID surat izin seperti pada Gambar 4.45. Tombol Lihat/Unduh membuka dokumen surat izin yang disimpan terenkripsi di IPFS melalui Pinata.



Gambar 4.45 Halaman Status LSP Ketika Sudah Terverifikasi

i. Partisipan

Halaman ini menyajikan daftar peserta yang sedang mengikuti sertifikasi dalam sebuah tabel dengan kolom: Wallet, Nama, Email, Skema, Nilai, Aksi, dan Sertifikat. Di atas tabel tersedia kotak pencarian untuk memfilter berdasarkan nama, email, atau alamat *wallet* terlihat seperti pada Gambar 4.46.

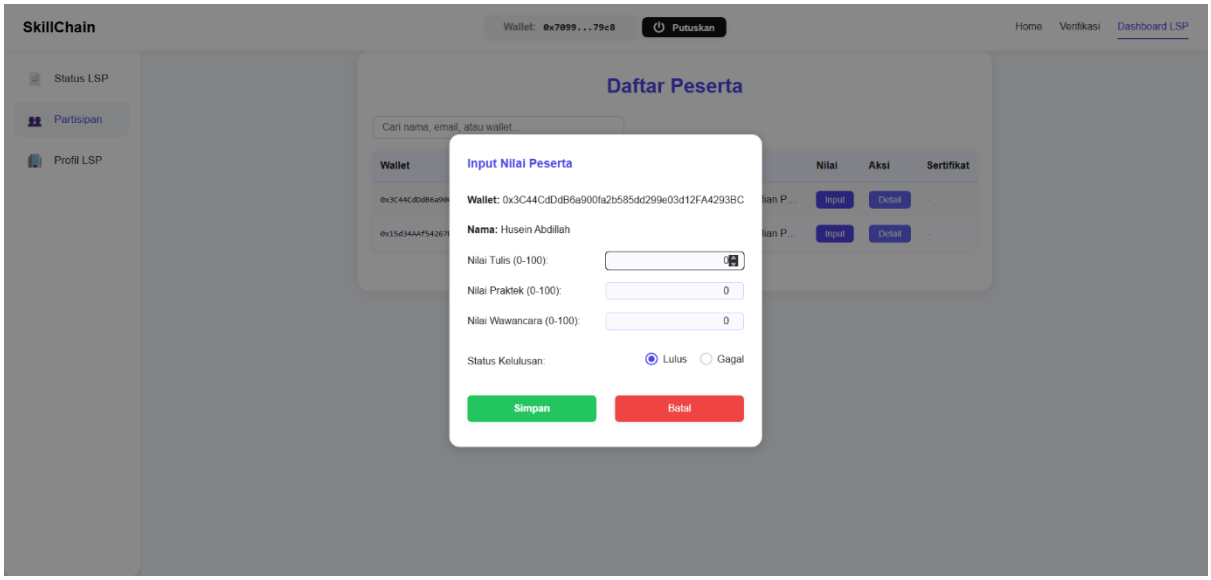


Gambar 4.46 Halaman Partisipan

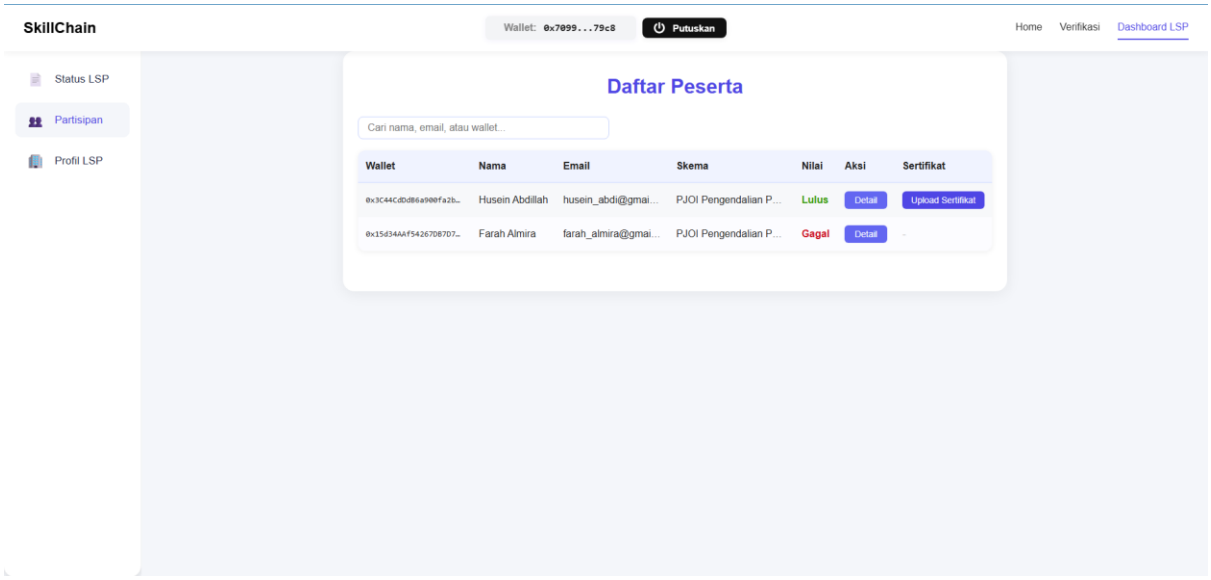
Tombol Input membuka formulir penilaian berisi tiga kolom angka 0–100 (Nilai Tulis, Nilai Praktik, Nilai Wawancara) dan pilihan Status Kelulusan (Lulus/Gagal). Tombol Simpan memicu transaksi penetapan nilai dan status pada kontrak. Tombol Batal menutup formulir

tanpa mengubah *state*. Validasi di sisi antarmuka memastikan rentang nilai benar dan status dipilih. Jika tidak, sistem menampilkan pesan korektif dan tidak mengirim transaksi.

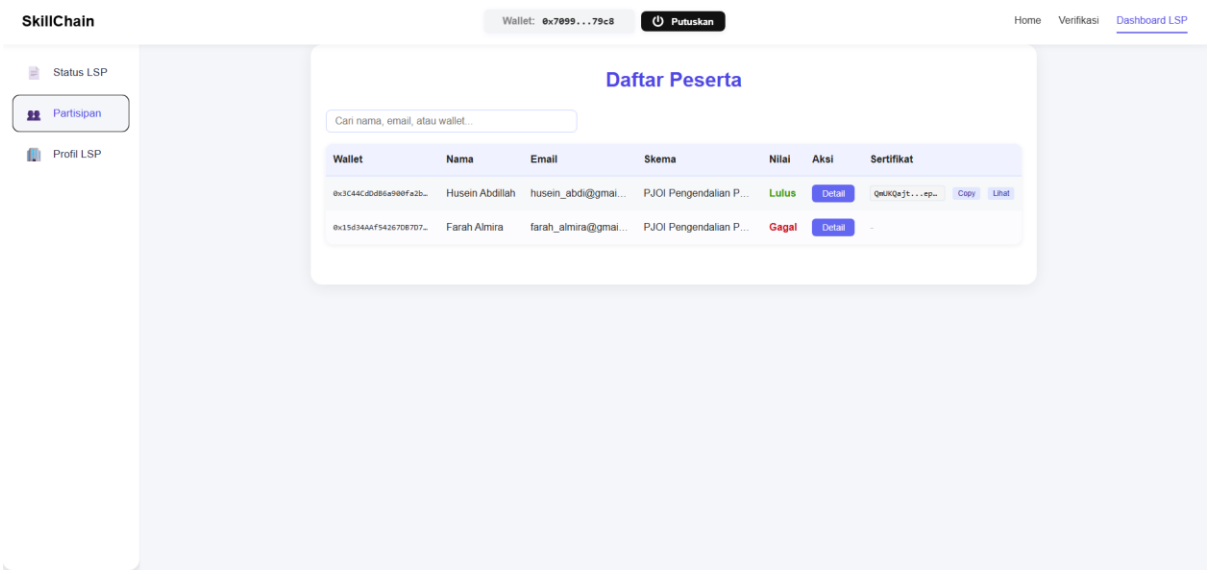
Setelah status disimpan Lulus, pada baris peserta muncul tombol Upload Sertifikat seperti pada Gambar 4.47. LSP memilih berkas sertifikat PDF, aplikasi akan mengenkripsi berkas sertifikat kemudian mengunggah ke Pinata untuk memperoleh CID seperti pada Gambar 4.48, lalu memanggil fungsi penerbitan pada kontrak. Bila berhasil, kolom Sertifikat menampilkan CID beserta tombol Copy dan Lihat untuk memudahkan audit seperti pada Gambar 4.49.



Gambar 4.47 Tampilan Proses Penilaian oleh LSP

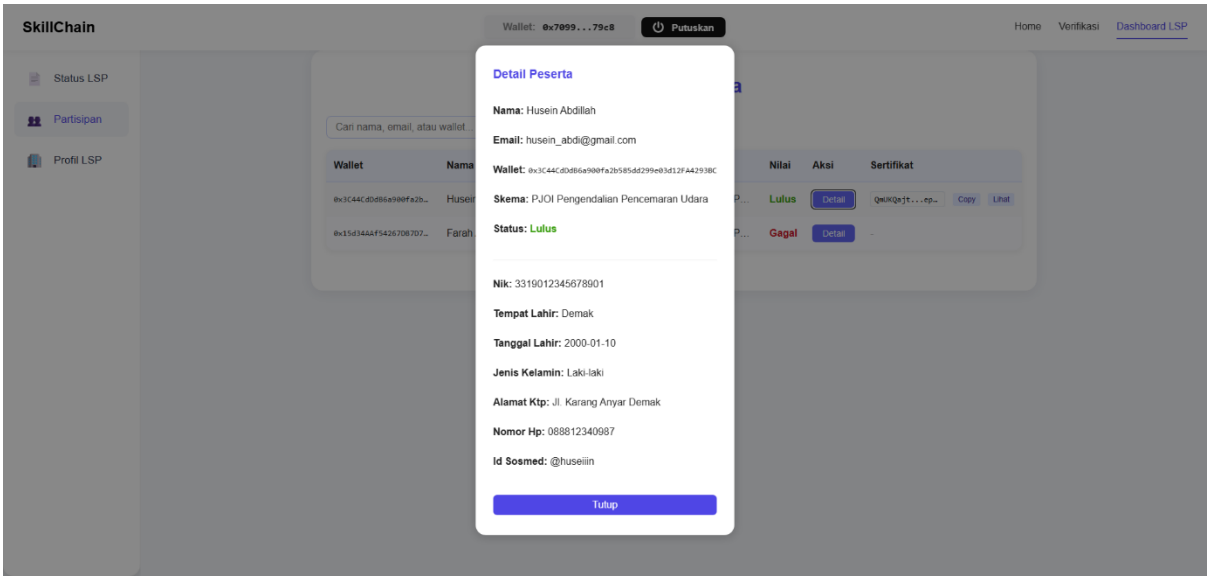


Gambar 4.48 Tampilan Proses Ketika Sudah Penilaian



Gambar 4.49 Tampilan Proses Ketika Sudah Selesai Unggah Sertifikat

Tombol Detail menampilkan profil peserta dan ringkasan asesmen untuk membantu verifikasi seperti pada Gambar 4.50.

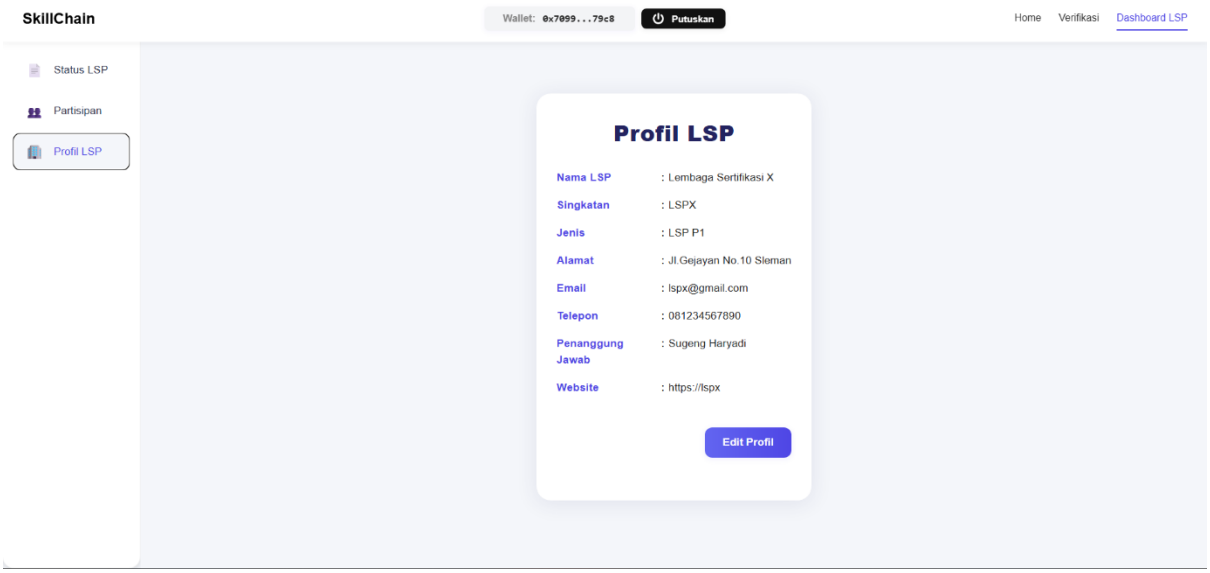


Gambar 4.50 Tampilan Proses Melihat Data Peserta oleh LSP

j. Profil LSP

Menampilkan metadata lembaga seperti pada Gambar 4.51. Tombol Edit Profil membuka formulir pembaruan. Data hasil pembaruan dikemas ke JSON, dienkripsi di sisi klien, diunggah ke Pinata untuk menghasilkan CID baru, lalu CID tersebut disimpan ke kontrak

melalui transaksi pembaruan. Perubahan metadata tidak memengaruhi status verifikasi, namun seluruh pembaruan tercatat melalui *event* sehingga mudah diaudit.



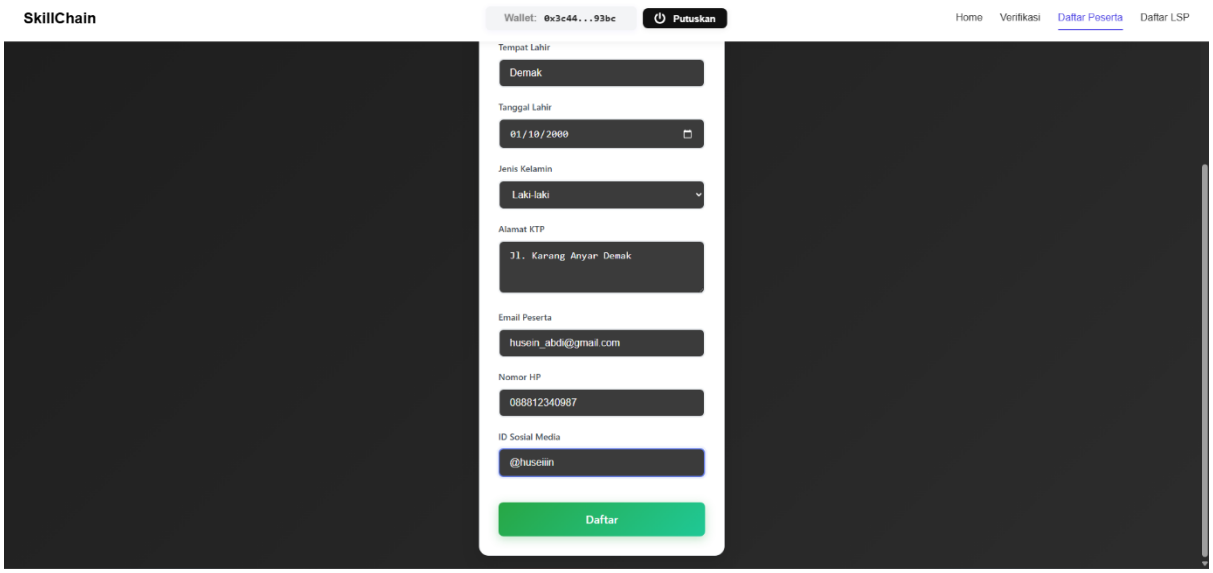
Gambar 4.51 Halaman Profil LSP

4.4.4 Halaman Pendaftaran Peserta

Halaman ini adalah prasyarat sebelum *Dashboard* Peserta dapat diakses. Calon peserta mengisi metadata pribadi sesuai pada Gambar 4.52. Setelah seluruh kolom terisi, aplikasi melakukan validasi di sisi klien, seperti kolom tidak boleh kosong, format email dan *username* media sosial diperiksa, tanggal lahir dipilih dari komponen kalender, dan nomor HP dicek agar hanya berisi sepuluh sampai tiga belas digit angka. Jika ada isian tidak valid, sistem menampilkan pesan korektif dan tidak mengirim transaksi.

Begitu validasi lolos, aplikasi menyusun dokumen metadata dalam format JSON yang memuat isian sesuai formulir, lalu hasil enkripsinya diunggah ke IPFS (Pinata) untuk memperoleh CID_metadata. Tahap berikutnya, aplikasi memanggil fungsi kontrak daftarPeserta(CID_metadata) . Jika transaksi berhasil, antarmuka menampilkan notifikasi keberhasilan, status keanggotaan berubah menjadi Aktif, tanggal pendaftaran tercatat, dan pengguna dialihkan ke *Dashboard* Peserta.

Mekanisme pengaman diberlakukan di dua lapis. Di antarmuka, tombol Daftar hanya aktif bila validasi terpenuhi. Di *smart contract*, terdapat pengecekan pendaftaran ganda sehingga *wallet* yang sudah terdaftar akan ditolak bila mencoba mendaftar ulang. Dengan pola ini, CID_metadata tersimpan *on-chain* sebagai rujukan, isi dokumen tetap privat karena terenkripsi, dan seluruh tindakan memiliki jejak audit melalui *event* transaksi.



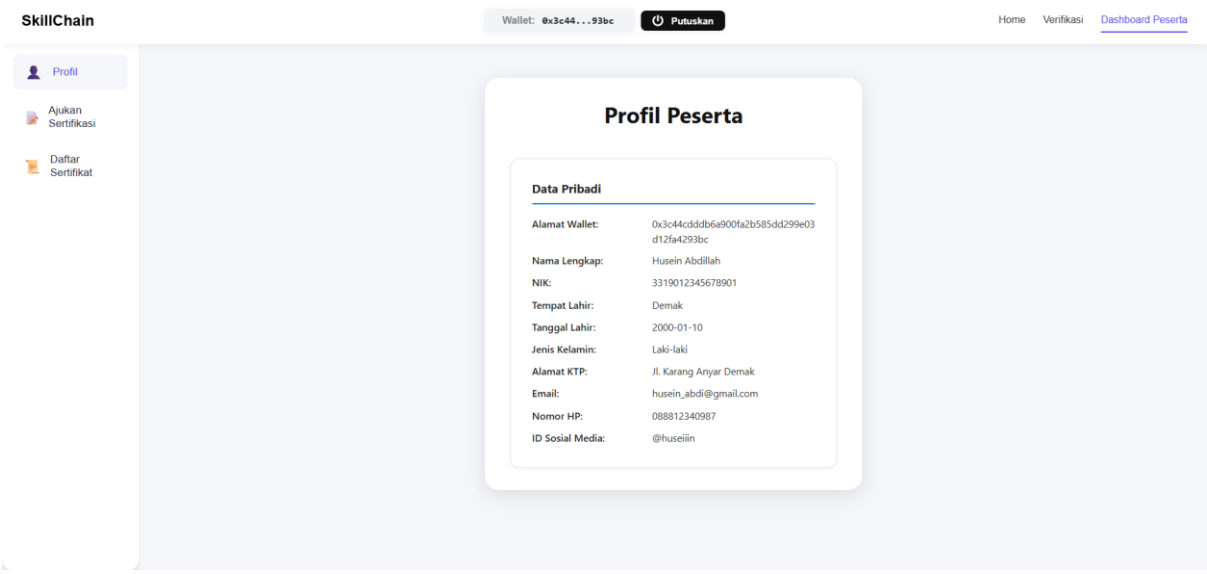
Gambar 4.52 Halaman Pendaftaran Peserta

4.4.5 *Dashboard* Peserta

Dashboard Peserta hanya dapat diakses setelah *wallet* melakukan pendaftaran sebagai peserta. Selama belum terdaftar, antarmuka mengarahkan pengguna ke halaman pendaftaran dan seluruh menu khusus peserta disembunyikan sehingga tidak ada transaksi yang dikirim selain permintaan pendaftaran.

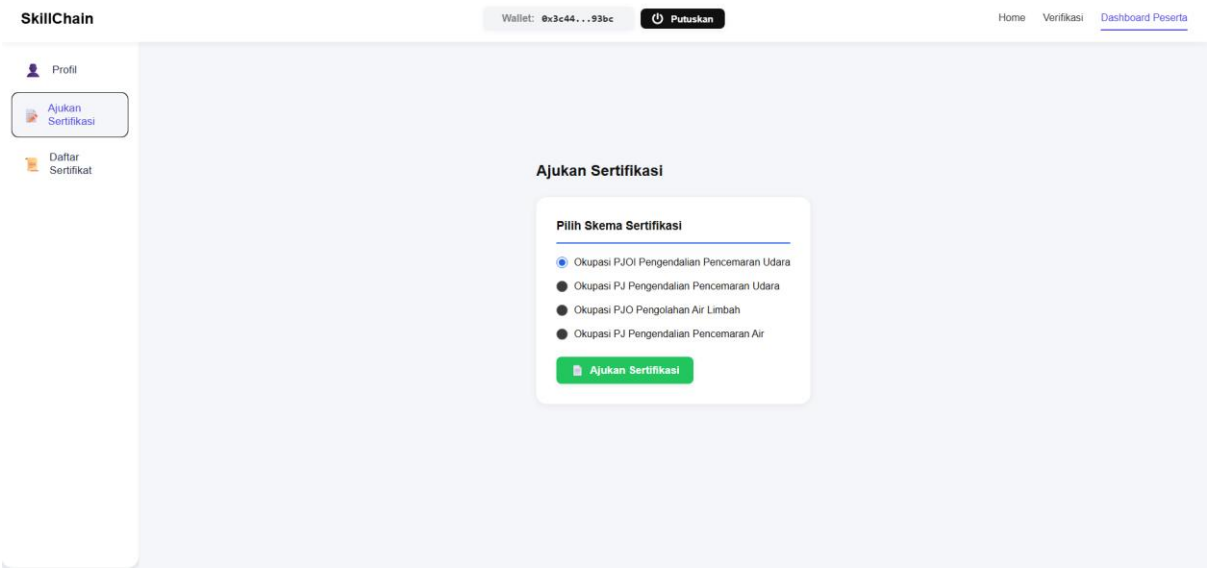
Di sisi kiri tersedia tiga menu: Profil, Ajukan Sertifikasi, dan Daftar Sertifikat. Ketiganya saling terhubung dengan logika pada *smart contract* sehingga setiap aksi yang mengubah keadaan selalu memicu tanda tangan menggunakan MetaMask dan menghasilkan catatan *on-chain* serta *event* untuk keperluan audit.

Profil Peserta menampilkan ringkasan identitas yang dikumpulkan saat pendaftaran, antara lain alamat *wallet*, nama lengkap, NIK, tempat dan tanggal lahir, jenis kelamin, alamat KTP, email, nomor HP, dan ID media sosial. Data ditampilkan hanya baca agar konsisten dengan metadata yang diajukan saat pendaftaran. Tampilan ini memudahkan peserta memastikan informasi yang diajukan sudah benar sebelum mengajukan sertifikasi seperti pada Gambar 4.53.



Gambar 4.53 Halaman Profil Peserta

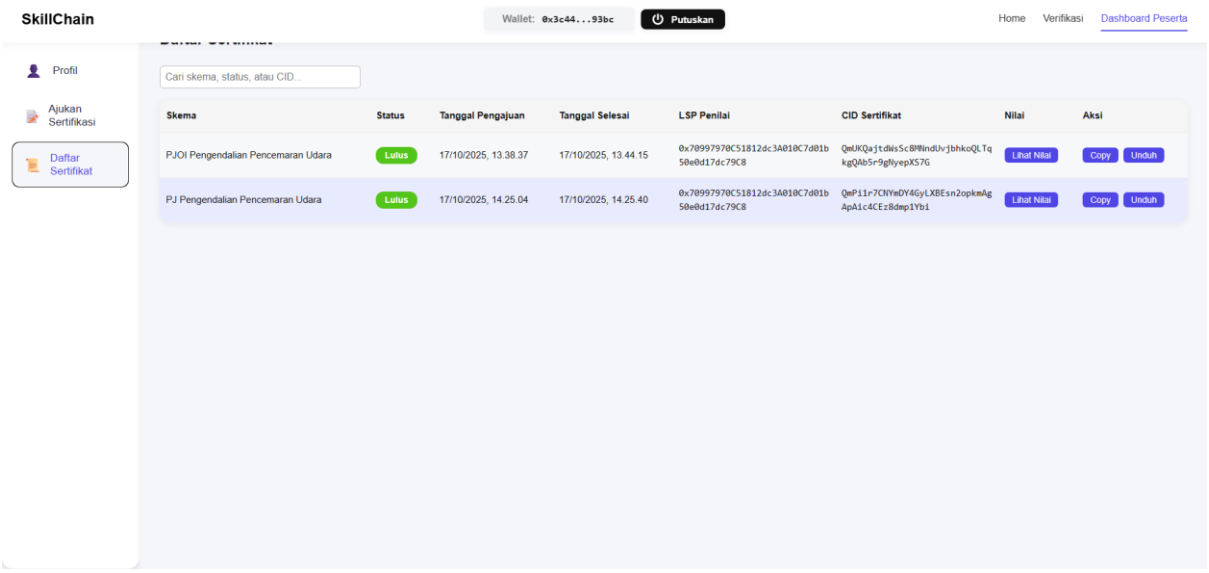
Ajukan Sertifikasi digunakan peserta untuk memilih satu skema dari daftar yang tersedia lalu menekan tombol Ajukan Sertifikasi seperti pada Gambar 4.54. Aplikasi melakukan validasi lokal, seperti melarang pengajuan baru ketika masih terdapat satu sertifikasi aktif. Setelah valid, antarmuka meminta tanda tangan transaksi menggunakan MetaMask dan memanggil fungsi pengajuan pada kontrak.



Gambar 4.54 Halaman Ajukan Sertifikasi

Daftar Sertifikat menampilkan seluruh sertifikasi yang sudah pernah diikuti oleh peserta dalam bentuk tabel seperti yang terlihat pada Gambar 4.55. Pada halaman ini terdapat tiga aksi:

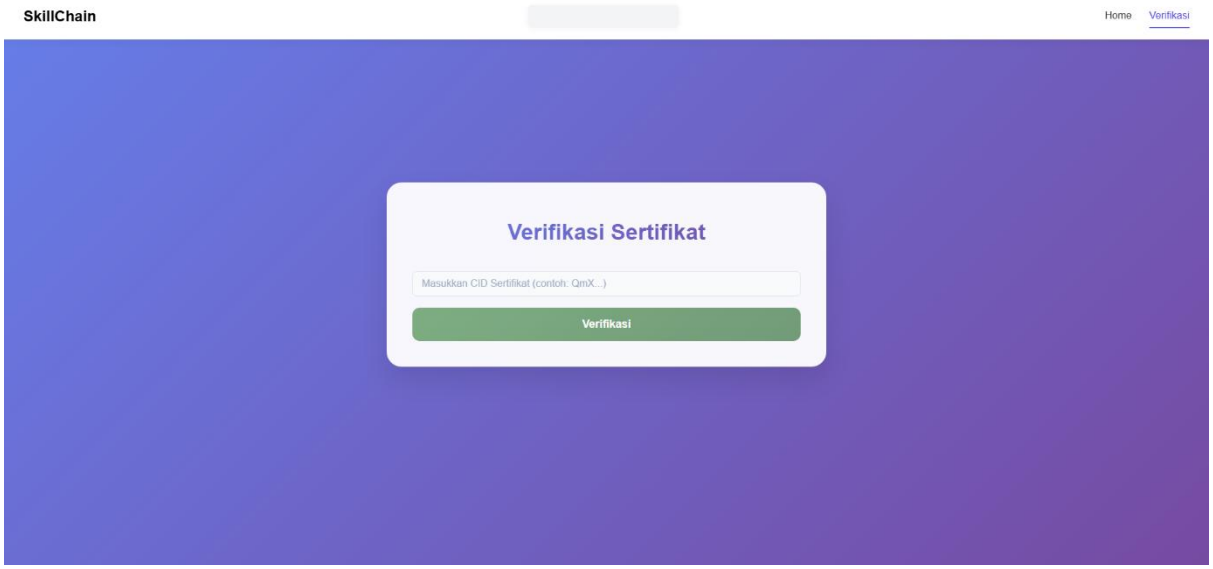
- Lihat Nilai untuk membuka tampilan yang menampilkan nilai yang diperoleh saat ujian.
- Copy untuk menyalin CID sertifikat agar mudah dibagikan untuk verifikasi.
- Unduh untuk mengambil berkas sertifikat langsung dari IPFS berdasarkan CID yang tercatat. Jika status masih kosong atau tidak lulus maka CID akan kosong, tombol unduh dinonaktifkan untuk mencegah pengambilan berkas yang tidak tersedia.



Gambar 4.55 Halaman Daftar Sertifikat

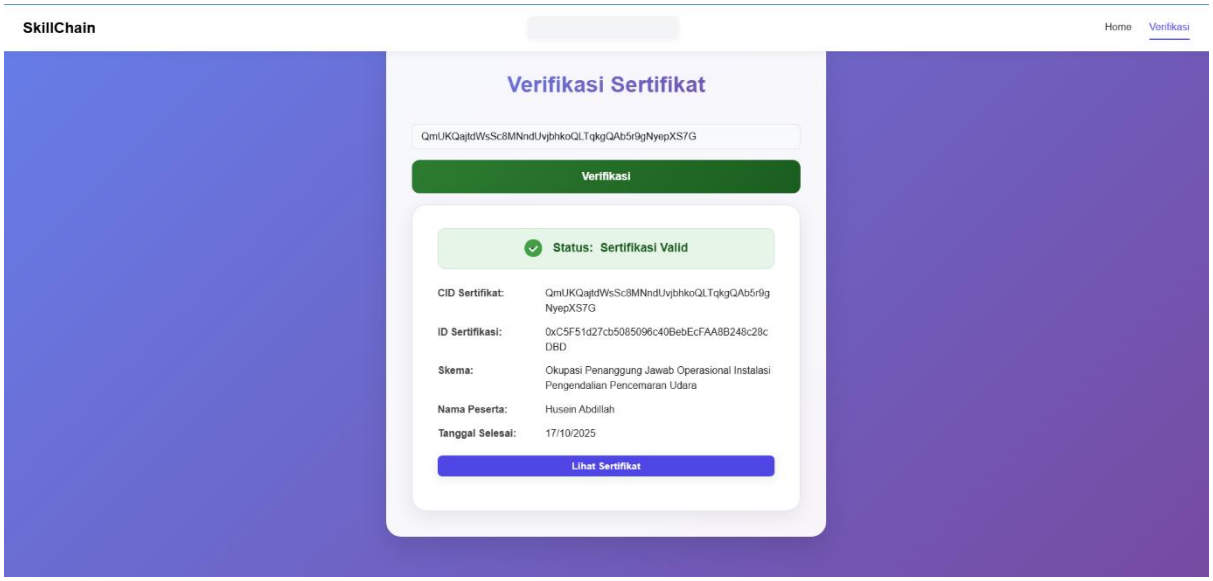
4.4.6 Halaman Verifikasi Sertifikat

Halaman Verifikasi ditujukan untuk pengguna umum yang ingin memastikan keaslian sertifikat tanpa perlu masuk sebagai BNSP, LSP, atau Peserta. Antarmuka menampilkan satu bidang isian untuk CID sertifikat dan sebuah tombol Verifikasi seperti pada Gambar 4.56. Setelah CID dimasukkan, aplikasi melakukan pemanggilan baca ke *smart contract* untuk memeriksa apakah CID tersebut terdaftar, masih berlaku, dan terhubung ke satu entri sertifikasi yang sah. Proses ini tidak membutuhkan *wallet* MetaMask dan tidak menimbulkan transaksi, sehingga aman bagi publik dalam arti tidak mengekspos data sensitif, tidak meminta kredensial pengguna, serta tidak memiliki konsekuensi finansial.



Gambar 4.56 Halaman Verifikasi Sertifikat

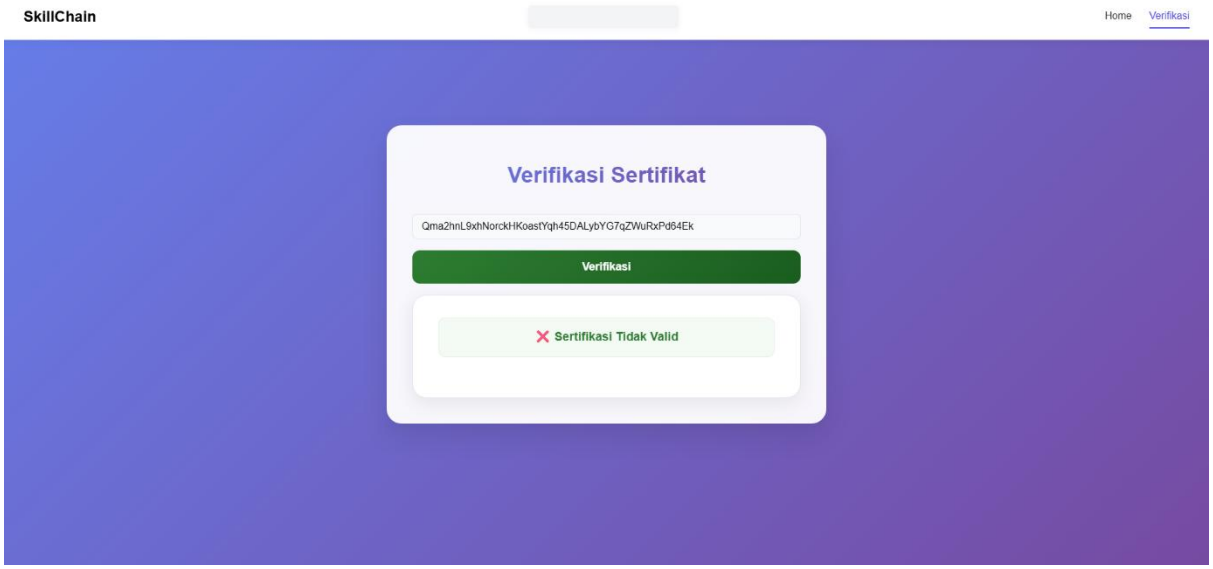
Jika pemeriksaan berhasil, halaman menampilkan ringkasan hasil verifikasi seperti pada Gambar 4.57. yang meliputi: CID Sertifikat, ID Sertifikasi, Skema, Nama Peserta, serta Tanggal Selesai. Status ditampilkan secara jelas, misalnya “Sertifikasi Valid”. Tersedia tombol Lihat Sertifikat yang mengarahkan ke objek pada IPFS sesuai CID untuk tujuan audit institusional.



Gambar 4.57 Tampilan Ketika CID Benar

Apabila bidang CID kosong atau formatnya tidak sesuai, antarmuka menampilkan pesan korektif dan tidak mengirim permintaan. Jika CID tidak ditemukan pada kontrak, sistem menampilkan status “Sertifikasi Tidak Valid” tanpa mengungkapkan data sensitif apa pun

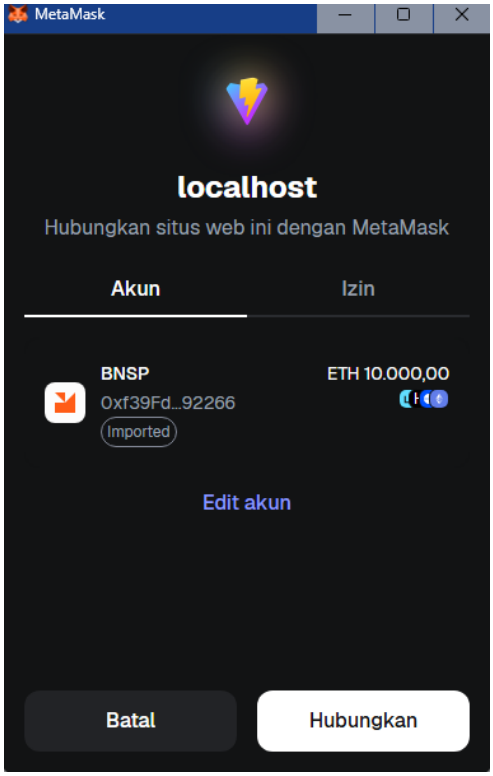
seperti pada Gambar 4.58. Dengan pola ini, verifikasi keaslian berlangsung terbuka dan dapat diulang kapan saja, namun isi dokumen tetap terlindungi.



Gambar 4.58 Tampilan Ketika CID Tidak Benar

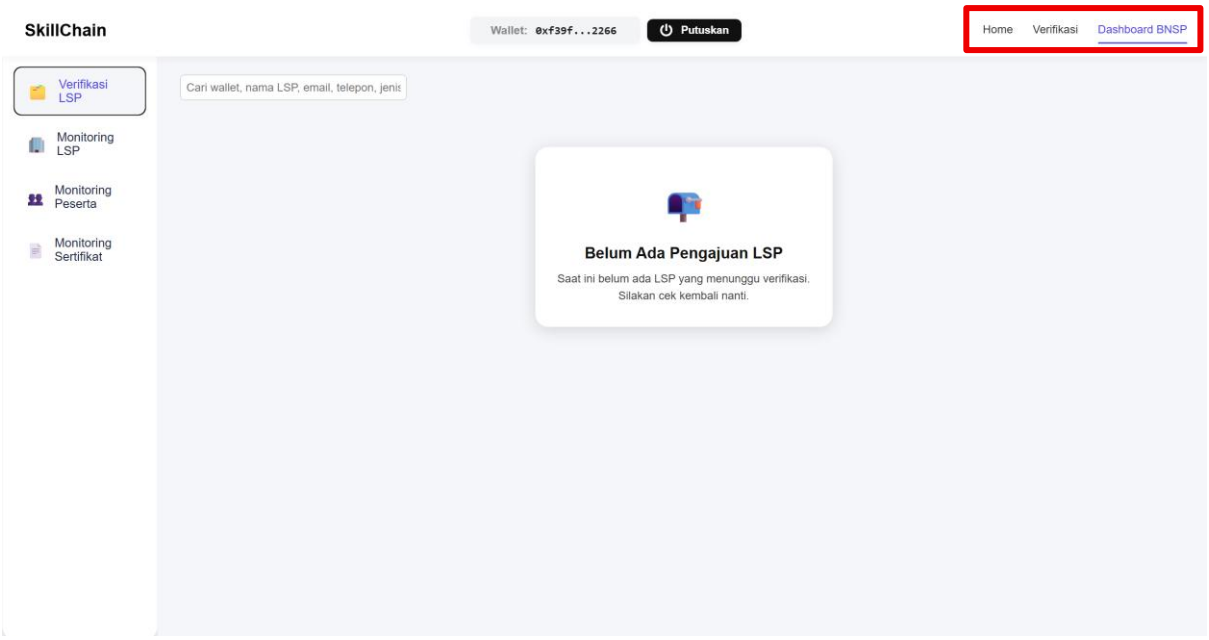
4.4.7 Pola Interaksi, Validasi, dan Konfirmasi Transaksi

Proses pada DApp selalu diawali dengan penyambungan *wallet*, kecuali untuk fitur verifikasi sertifikat tidak perlu terhubung dengan *wallet*. Saat tombol Hubungkan Wallet ditekan, MetaMask menampilkan dialog untuk memilih akun dan menyetujui koneksi situs pada jaringan lokal Hardhat seperti pada Gambar 4.59. Setelah koneksi disetujui, status akun berubah menjadi terhubung.



Gambar 4.59 Dialog MetaMask Ketika Menghubungkan *Wallet*

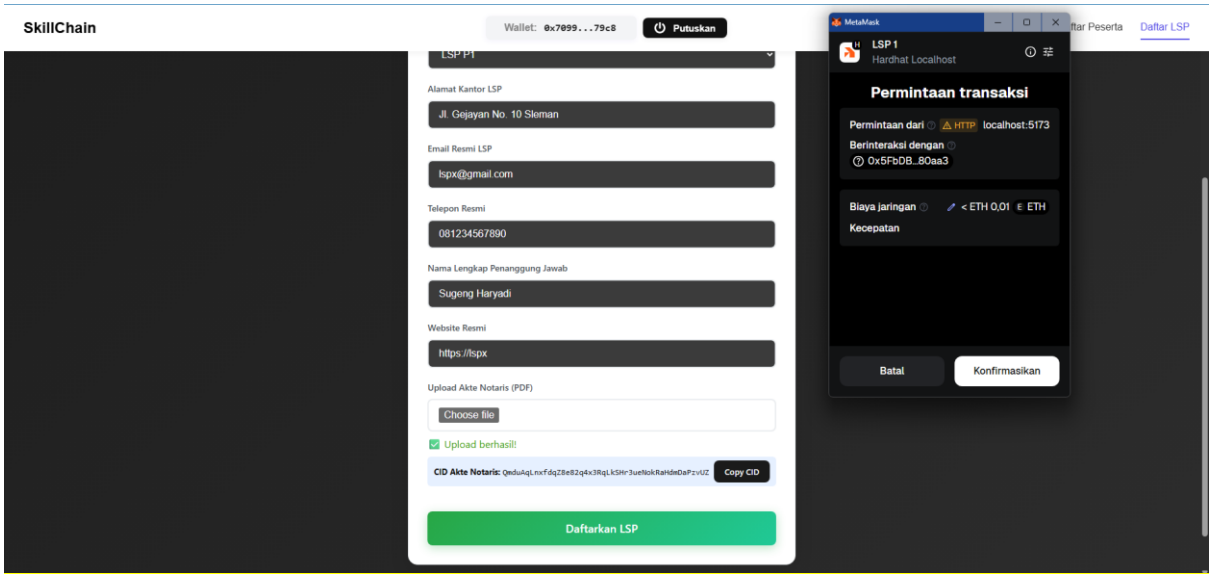
Sesudah tersambung, aplikasi membaca peran dari *smart contract* melalui Ethers.js lalu menata navigasi sesuai hasilnya. Apabila *wallet* belum memiliki peran, halaman Home menampilkan pesan “Anda sudah terhubung” beserta tautan ke halaman Daftar LSP dan Daftar Peserta sebagai pintu masuk untuk memperoleh peran. Bila *wallet* sudah berperan sebagai BNSP, LSP, atau Peserta, pengguna diarahkan ke *dashboard* yang sesuai, sedangkan menu lain yang tidak relevan disembunyikan atau menolak akses. Perilaku ini memastikan pengguna hanya melihat aksi yang sah sesuai perannya seperti pada Gambar 4.60.



Gambar 4.60 Kondisi Menu Setelah Koneksi *Wallet* yang Memiliki Peran BNSP

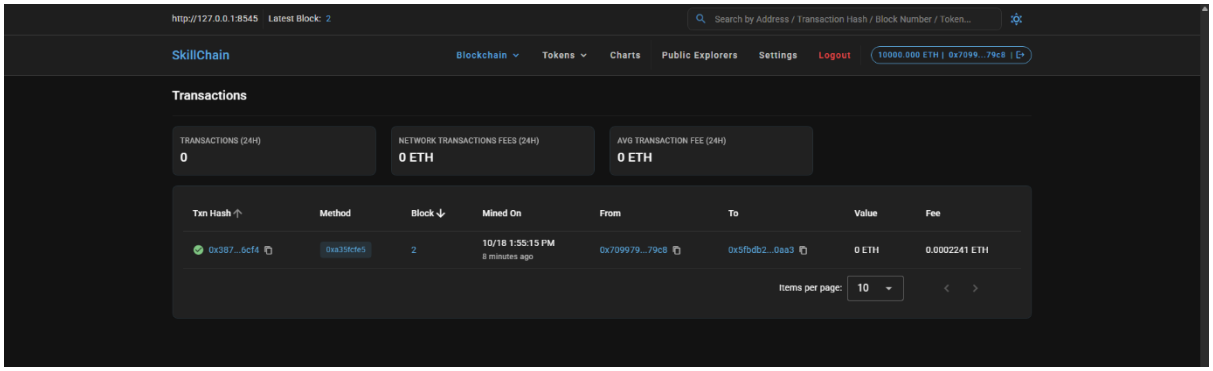
Sebelum transaksi dikirim, setiap formulir melakukan validasi sisi klien. Contohnya, pendaftaran LSP memeriksa kolom wajib seperti nama, singkatan, jenis LSP, alamat, email, telepon, penanggung jawab, serta keterbatasan tipe berkas akta notaris yang harus berformat PDF. Pada pendaftaran peserta, validasi meliputi NIK numerik, tanggal lahir yang valid, serta format email. Setelah semua isian benar, metadata dirakit ke berkas JSON, dienkripsi di sisi klien, diunggah ke Pinata untuk memperoleh CID, lalu tombol kirim diaktifkan. Gagal validasi menampilkan pesan korektif dan transaksi tidak akan membuka dialog MetaMask.

Ketika data lolos validasi dan CID berhasil didapat, MetaMask menampilkan dialog konfirmasi transaksi berisi alamat kontrak tujuan, nilai gas, serta data panggilan fungsi. Pengguna dapat meninjau ringkasan tindakan yang akan dilakukan, misalnya memanggil daftarLSP(CID) atau terbitkanSertifikat(peserta, cid) seperti pada Gambar 4.61, lalu memilih Konfirmasi untuk menandatangani atau Tolak untuk membatalkan. Jika ditolak, aplikasi menampilkan notifikasi penolakan tanpa ada perubahan pada blockchain.



Gambar 4.61 Dialog Transaksi Pendaftaran LSP

Setelah LSP menekan “Daftarkan LSP” dan tanda tangan transaksi pada MetaMask berhasil, hasil pencatatan *on-chain* dapat dilihat pada penjelajah blok lokal (Hardhat/Ethernal) sebagaimana ditunjukkan pada Gambar 4.62. Baris transaksi berikon centang hijau menandakan status berhasil, kolom Txn Hash (mis. 0x387…6cf4) dapat diklik untuk membuka detail tanda terima, kolom Method menampilkan selektor fungsi yang dipanggil (hasil kompilasi dari daftarLSP(CID)), kolom Block menunjukkan nomor blok tempat transaksi ditambang (mis. blok 2) beserta waktu penambangan, kolom From berisi alamat *wallet* LSP pengaju, sedangkan To adalah alamat kontrak utama yang menerima panggilan, kolom Value bernilai 0 ETH karena tidak ada transfer ether, dan Fee menampilkan biaya gas yang terpakai (mis. 0.0002241 ETH). Tampilan ini mengonfirmasi bahwa permohonan pendaftaran LSP telah tercatat di blockchain dan dapat diaudit kembali melalui *hash* transaksi.



Gambar 4.62 Bukti *on-chain* Transaksi Pendaftaran LSP

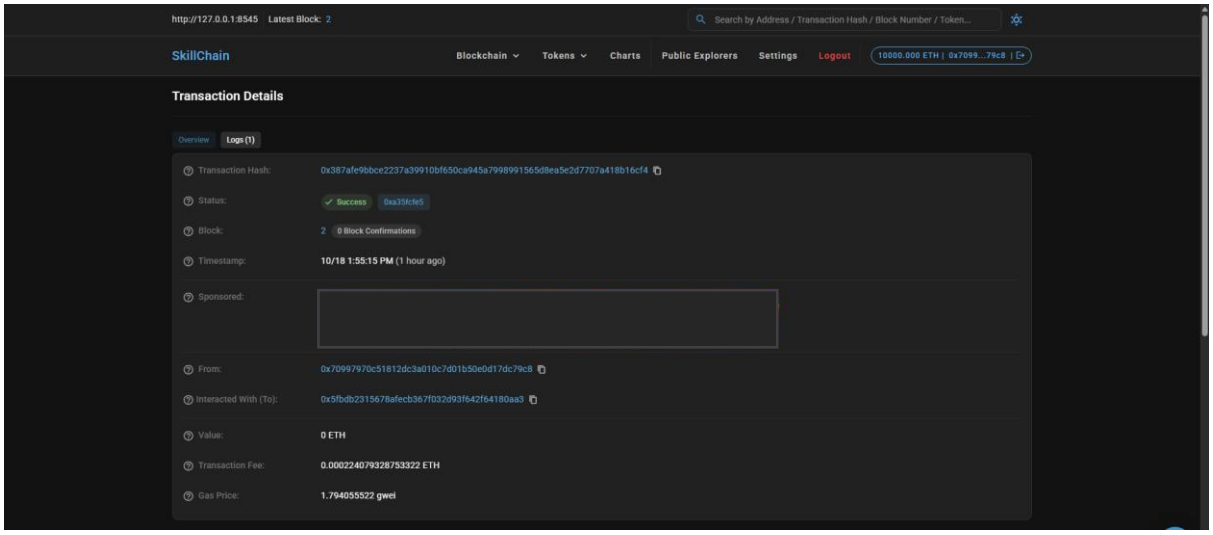
Setiap tindakan yang dikonfirmasi melalui MetaMask menghasilkan transaksi di jaringan pengujian dan memicu *event* pada *smart contract*. Bukti pencatatan *on-chain* beserta rincian transaksinya ditampilkan pada sub-subbab 4.3.8.

4.4.8 Rekaman *On-chain* dan Jejak Audit

Setiap aksi yang mengubah *state* pada DApp, seperti pendaftaran LSP, penilaian peserta, dan penerbitan sertifikat, dieksekusi sebagai transaksi di jaringan pengujian Hardhat. Transaksi tersebut menghasilkan tanda terima dan *event* pada *smart contract* yang berfungsi sebagai jejak audit. Bukti pencatatan dapat diperiksa melalui penjelajah blok lokal (Ethernal). Bagian ini menampilkan cuplikan hasil *on-chain* dan menjelaskan elemen yang relevan untuk penelusuran kembali.

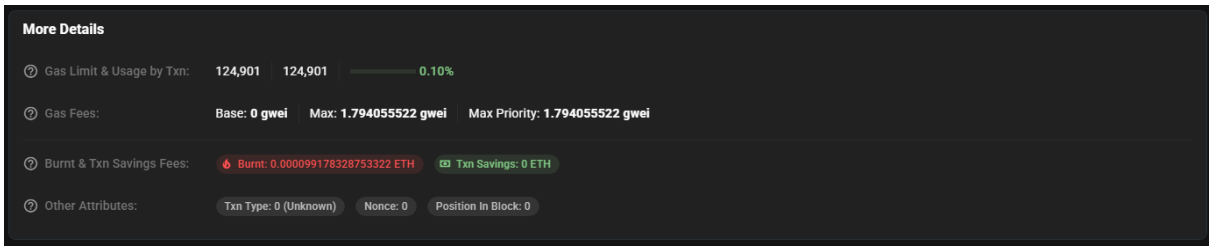
Setelah LSP menekan tombol Daftarkan LSP dan menandatangani transaksi di MetaMask, ringkasan transaksi muncul pada daftar transaksi Ethernal sebagaimana terlihat pada Gambar 4.63. Baris transaksi yang berikon centang hijau menandakan status sukses. Kolom Txn Hash dapat diklik untuk membuka detail transaksi, Method menampilkan selektor fungsi yang dipanggil, Block menunjukkan nomor blok serta waktu penambangan, From berisi alamat *wallet* pengaju, To adalah alamat kontrak yang dipanggil, Value bernilai 0 ETH karena tidak ada transfer ether, sedangkan Fee memperlihatkan biaya gas yang terpakai. Tampilan ini memastikan bahwa permohonan pendaftaran LSP benar-benar tercatat di blockchain.

Dengan membuka *hash* transaksi pada daftar tersebut, Ethernal menampilkan Transaction Details sebagaimana pada Gambar 4.63. Bagian ini memuat status transaksi, nomor blok, waktu penambangan, gas limit, gas used, dan gas price. Informasi ini adalah artefak audit utama karena memverifikasi kapan transaksi terjadi, pada blok ke berapa, serta berapa sumber daya gas yang dikonsumsi.



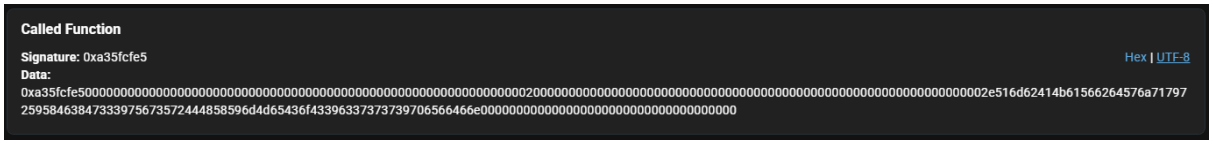
Gambar 4.63 Detail Transaksi Pendaftaran LSP

Panel *More Details* memperlihatkan Gas Limit dan *Usage*, komponen Gas *Fees* (*base, max fee, max priority*), nilai ETH yang terbakar pada baris Burnt, serta atribut lain seperti Nonce dan Position in Block. Data ini menggambarkan biaya dan efisiensi eksekusi transaksi pendaftaran LSP, sebagaimana terlihat pada Gambar 4.64.



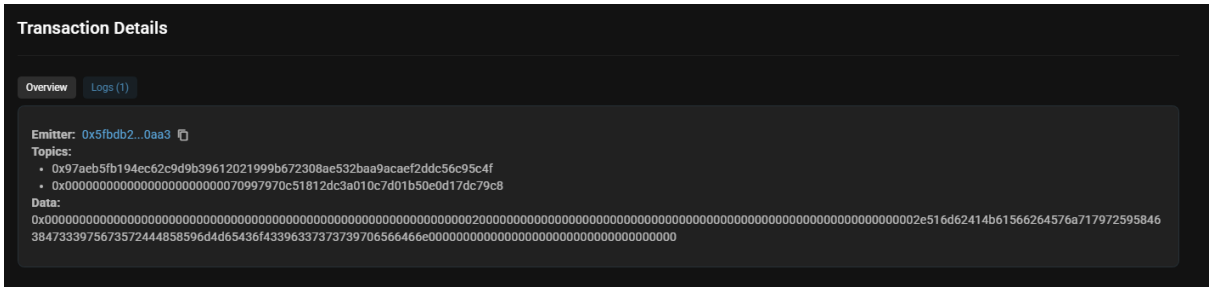
Gambar 4.64 Rincian Konsumsi Gas Transaksi Pendaftaran LSP pada Ethernal

Panel *Called Function* menampilkan selector 0xa35fcfe5 yang merupakan 4 byte awal *hash* tanda tangan fungsi pendaftaran pada kontrak (misalnya daftarLSP(...)). Kolom Data menunjukkan isi permintaan ke kontrak termasuk CID dari LSP yang sudah dikemas dalam format standar (ABI). Rincian ini membuktikan bahwa kontrak yang dieksekusi sesuai dengan rancangan dan menerima parameter yang benar, seperti terlihat pada Gambar 4.65.



Gambar 4.65 Tab *Called Function* Transaksi Pendaftaran LSP pada Ethernal

Panel Logs merekam *event* yang diterbitkan (emit) oleh *smart contract* saat transaksi diproses. Seperti terlihat pada Gambar 4.66, baris Emitter menunjukkan alamat *smart contract* yang menerbitkan *event*. Topics[0] berisi *event signature* hasil keccak256 terhadap definisi *event*, sedangkan Topics[1] menyimpan parameter yang ditandai *indexed* (alamat *wallet* pengaju). Bagian Data berisi parameter *non-indexed* yang dikodekan menurut ABI. Pada kasus pendaftaran LSP, bagian ini memuat nilai seperti kode jenis LSP dan CID dokumen di IPFS. Pola heksadesimal pada Data memuat representasi ASCII CID (misalnya awalan "Qm" muncul sebagai 51 6d), sehingga dapat ditelusuri balik keterkaitan transaksi dengan berkas di IPFS.



Gambar 4.66 *Logs* Pendaftaran LSP pada Ethernal

Sementara itu, verifikasi publik terhadap CID sertifikat tidak menghasilkan transaksi karena hanya memanggil fungsi baca. Oleh sebab itu tidak ada entri baru pada Ethernal untuk proses verifikasi. Absennya transaksi justru menjadi indikator bahwa operasi memang bersifat *read-only* dan tidak mengubah *state* kontrak.

4.4.9 Keterkaitan Antarmuka dengan Fungsi Kontrak

Keterkaitan antarmuka dan kontrak ditunjukkan oleh pemetaan langsung antara aksi pada antarmuka dan fungsi pada *smart contract*, yang dibatasi oleh peran pengguna. Contohnya: pada *Dashboard* BNSP, tombol Verifikasi LSP memanggil verifikasiLSP(lspAddress, suratIzinCID) (atau penolakan melalui fungsi penolakan), pada *Dashboard LSP*, formulir Input Nilai memanggil fungsi pencatatan nilai, tombol Tetapkan Kelulusan memanggil fungsi penetapan hasil asesmen, dan tombol Terbitkan Sertifikat memanggil fungsi penyimpanan CID sertifikat, pada *Dashboard Peserta*, tombol Daftar

Peserta memanggil daftarPeserta(cidMetadata) dan Ajukan Sertifikasi memanggil fungsi pengajuan skema, sedangkan pada Halaman Verifikasi Publik, isian CID memanggil fungsi baca untuk mengecek validitas tanpa transaksi. Aksi tulis memerlukan tanda tangan MetaMask dan menghasilkan *event* sebagai jejak audit, sementara aksi baca tidak mengubah *state* dan tidak menimbulkan biaya gas. Selain itu, antarmuka menyembunyikan atau menonaktifkan aksi yang tidak sesuai peran sehingga fungsi yang dipanggil selalu sejalan dengan otorisasi pengguna.

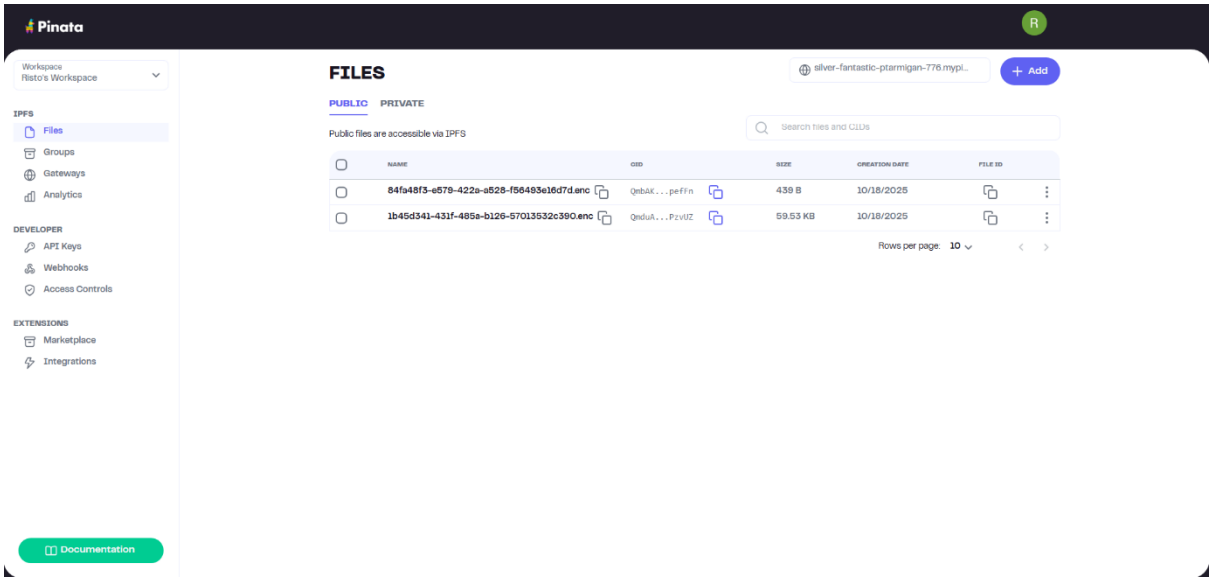
4.5 Bukti Penyimpanan *Off-chain* (Pinata)

Semua berkas yang diunggah pada DApp disimpan secara *off-chain* di IPFS melalui Pinata. Setiap berkas memperoleh *Content Identifier* (CID) yang kemudian dicatat pada *smart contract*, sehingga perubahan status di blockchain terhubung langsung dengan objek dokumen yang berada di luar rantai.

Pada skenario pendaftaran LSP, aplikasi:

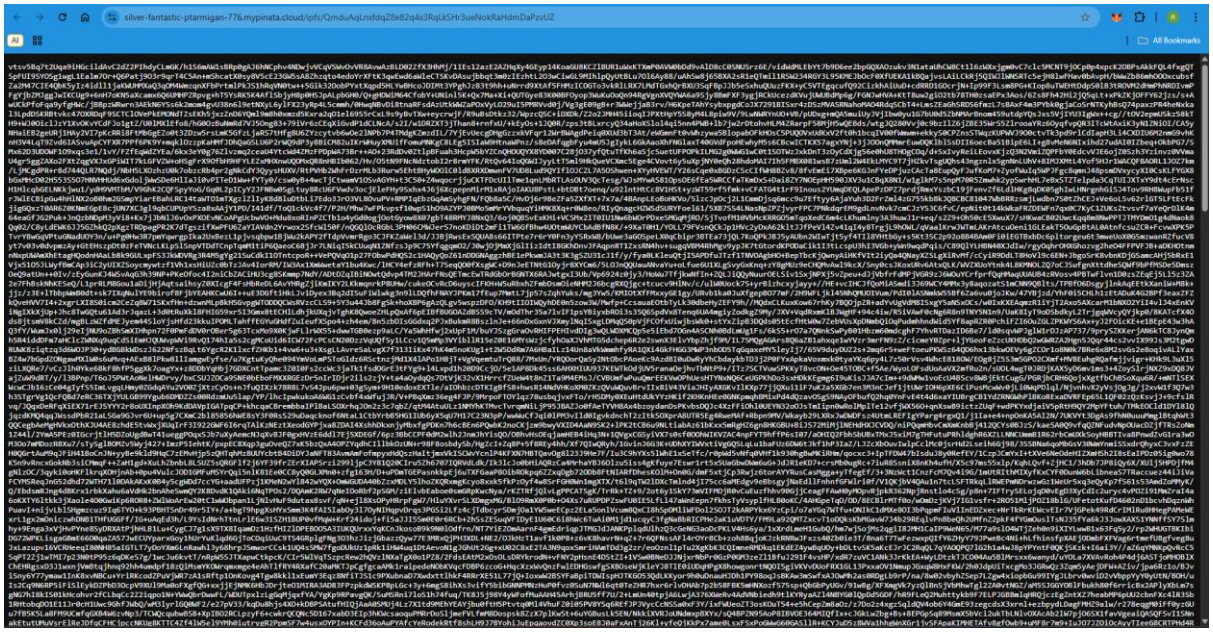
1. Mengemas data lembaga menjadi berkas JSON.
2. Melakukan enkripsi terhadap berkas JSON dan akta notaris hingga menghasilkan berkas dengan format .enc
3. Mengunggah hasil enkripsi ke Pinata untuk memperoleh CID, dan
4. Menyimpan CID tersebut ke kontrak saat transaksi pendaftaran dipanggil.

Gambar 4.67 Menampilkan halaman *Files* pada Pinata setelah pendaftaran LSP. Tampak dua berkas terenkripsi dengan format .enc yang mewakili akta notaris dan metadata LSP. Kolom CID pada tampilan ini harus sama dengan CID yang tercatat *on-chain* pada transaksi pendaftaran, sehingga tautan dari kontrak benar-benar menunjuk ke objek *off-chain* yang dimaksud.



Gambar 4.67 Daftar Berkas Terenkripsi Hasil Pendaftaran LSP di Pinata

Saat salah satu berkas diakses langsung melalui *gateway* Pinata tanpa proses dekripsi, yang muncul hanyalah *cipher-text* sebagaimana terlihat pada Gambar 4.68. Hal ini menegaskan bahwa enkripsi dilakukan di sisi klien sebelum penggunggahan. Untuk membaca isi dokumen, aplikasi memerlukan kunci dekripsi yang tidak disimpan di IPFS atau Pinata, sehingga kerahasiaan tetap terjaga walaupun CID bersifat publik.



Gambar 4.68 Tampilan *cipher-text* Berkas Akta Notaris

CID pada Gambar 4.68 berkorespondensi dengan CID pada tanda terima transaksi di penjelajah blok yang disajikan di Subbab 4.3.8. Kecocokan ini memberikan jejak audit *end-to-*

end. Tindakan di UI, persetujuan MetaMask, transaksi tercatat di blockchain, serta objek dokumen terenkripsi yang dapat ditelusuri kembali melalui CID yang sama di Pinata seperti pada Gambar 4.34. Kombinasi keduanya memastikan integritas, keaslian, dan nirsangkal setiap perubahan status sertifikasi.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

DApp sertifikasi kompetensi berhasil dirancang dan diimplementasikan di atas jaringan Ethereum dengan arsitektur modular yang mencakup SertifikasiStorage sebagai lapisan data bersama, modul RegistrasiLSP, PesertaManager, SertifikasiManager, dan VerifikasiPublik, serta MainContract sebagai gerbang utama. Alur operasional yang ditetapkan pada perancangan terealisasi pada aplikasi, meliputi pendaftaran LSP, pendaftaran peserta, pengajuan sertifikasi, input nilai dan penetapan hasil oleh LSP, penerbitan sertifikat, hingga akses riwayat pada *dashboard* sesuai peran. Pengujian *black-box* pada skenario utama menunjukkan seluruh fungsi berjalan sesuai spesifikasi dan perubahan *state* dapat dilihat melalui *event* serta pembacaan nilai *on-chain*, sehingga DApp terbukti mendukung proses sertifikasi kompetensi dari hulu hingga hilir.

Pendekatan modular tidak hanya memudahkan pengembangan, tetapi juga memperjelas penegakan aturan proses pada setiap batas modul. Invarian kunci, seperti larangan satu peserta memiliki dua sertifikasi aktif pada skema yang sama atau berbeda, syarat kelulusan setelah seluruh komponen nilai tercatat, serta penerbitan sertifikat yang hanya bisa ketika status lulus terpenuhi, ditegakkan melalui validasi di *smart contract*, *modifier* berbasis peran, dan *event* yang konsisten.

Dengan mekanisme tersebut, perilaku sistem menjadi deterministik. Untuk masukan yang sama dan urutan tindakan yang sah, keluaran selalu identik serta terekam sebagai jejak audit *on-chain*. Struktur ini memudahkan keterlacakan dari hasil perancangan ke implementasi, karena setiap transisi status memiliki padanan langsung pada fungsi *smart contract* dan *event* yang diterbitkan.

Mekanisme verifikasi keaslian sertifikat yang dapat diakses publik berbasis *Content Identifier* (CID) telah dirancang dan diuji. Publik memasukkan CID pada halaman Verifikasi, sistem melakukan pemanggilan baca pada *smart contract* untuk memastikan adanya pemetaan CID ke satu entitas sertifikasi yang sah, serta menampilkan metadata minimum yang diperbolehkan seperti penerbit dan tanggal terbit. Verifikasi ini tidak membutuhkan otoritas pusat maupun koneksi *wallet*, tidak memicu transaksi, dan tidak membuka isi dokumen karena berkas sertifikat disimpan secara terenkripsi di IPFS melalui Pinata. Dengan *content*

addressing, integritas berkas terjaga karena perubahan pada konten akan menghasilkan CID yang berbeda, sehingga pemalsuan mudah terdeteksi.

Skema verifikasi tersebut merepresentasikan prinsip *privacy-by-design*. Pihak publik memperoleh kepastian keaslian melalui kecocokan CID dan metadata *on-chain* tanpa melihat isi berkas. Kerahasiaan terjaga karena yang disimpan di IPFS adalah *ciphertext* hasil enkripsi di sisi klien. Pada kasus CID tidak terdaftar, sistem mengembalikan status tidak valid atau tidak ditemukan secara tegas sehingga tidak menimbulkan ambiguitas.

Proses verifikasi juga bersifat tanpa biaya gas karena hanya melibatkan pemanggilan baca. Selain itu, *reproducibility* terjaga. Setiap pihak dapat mengambil objek di IPFS berdasarkan CID, menghitung ulang *hash* konten secara lokal, lalu mencocokkannya dengan catatan *on-chain*. Ketersediaan jangka panjang bergantung pada keberadaan *node* di jaringan yang menyimpan objek terkait. Aspek ini tidak menjadi fokus pengujian pada penelitian dan dicatat sebagai batasan operasional untuk pekerjaan lanjutan.

Temuan pendukung dan batasan:

1. Keterlacakan *on-chain* memadai karena setiap tindakan penting memancarkan *event* dan memiliki tanda terima transaksi yang dapat diaudit.
2. Kontrol akses berbasis peran berjalan pada level antarmuka dan kontrak, sehingga tindakan tulis hanya dapat dilakukan oleh pihak yang berwenang.
3. Pemisahan data *on-chain* dan *off-chain* menjaga efisiensi dan kerahasiaan, dengan CID sebagai penghubung yang terverifikasi.
4. Implementasi masih bersifat prototipe pada jaringan lokal Hardhat, belum mencakup pengukuran performa skala besar maupun integrasi kelembagaan penuh, sehingga aspek tersebut menjadi ruang pengembangan berikutnya.

Secara keseluruhan, hasil perancangan, implementasi, dan pengujian menunjukkan bahwa DApp yang dibangun telah mencapai tujuan fungsional. Alur operasional berjalan *end-to-end*, aturan proses ditegakkan di tingkat *smart contract*, dan verifikasi keaslian sertifikat dapat dilakukan publik tanpa mengorbankan privasi.

5.2 Saran

1. Pengujian pada jaringan publik atau testnet.

   Pengujian di Sepolia atau Holesky dijalankan untuk menilai biaya gas yang sebenarnya, latensi transaksi, serta kompatibilitas dengan penjelajah blok dan *wallet*. Hasilnya menjadi dasar estimasi biaya operasional dan kelayakan penerapan.

2. Mekanisme pencabutan dan siklus hidup sertifikat.

   Fitur pencabutan atau penangguhan, masa berlaku, serta versi skema disediakan agar status terbaru tercermin pada halaman verifikasi. Kebijakan ini mendukung audit, mengurangi risiko penyalahgunaan sertifikat, dan memenuhi kebutuhan tata kelola.
3. Indeksasi dan pencarian terindeks.

   Penerapan indexer seperti The Graph dimanfaatkan untuk query cepat berdasarkan skema, status, LSP, atau CID tanpa pemindaian blok manual. Pendekatan ini menurunkan beban RPC dan meningkatkan respons antarmuka.
4. Perlindungan berkas dan manajemen kunci.

   Enkripsi sisi klien digabungkan dengan manajemen kunci yang terstruktur (misalnya *envelope key* atau *key-encryption key*), token akses yang terikat pada CID, serta audit akses IPFS. Tujuannya menjaga kerahasiaan dokumen dan menyediakan jejak pemakaian yang jelas.
5. Akses dan keandalan *wallet*.

   Account Abstraction (ERC-4337), *session keys*, dan *social recovery* dievaluasi untuk menurunkan friksi pengguna non-teknis, memungkinkan biaya gas terbatas pada aksi tertentu, serta menyediakan pemulihan akun yang lebih aman.
6. Observabilitas operasional.

   Monitoring transaksi, peringatan kesalahan, mekanisme *retry*, dan *dashboard* audit disediakan bagi BNSP dan LSP. Langkah ini mempercepat penanganan insiden dan meningkatkan akuntabilitas proses.
7. Penguatan keamanan kontrak.

   Penelaahan keamanan menggunakan alat seperti Slither, Echidna, dan fuzz testing Foundry dilengkapi dengan cakupan uji yang tinggi. Pola peningkatan versi berbasis *proxy* dipertimbangkan untuk kebutuhan pembaruan di lingkungan produksi.
8. Kepatuhan dan tata kelola.

   Proses diselaraskan dengan peraturan BNSP, kebijakan perlindungan data, serta prosedur operasional LSP. Dokumen SOP dan kebijakan pengelolaan kunci disiapkan sebagai prasyarat penerapan resmi.
9. Pengalaman pengguna dan aksesibilitas.

   Panduan bertahap, *empty state* yang informatif, dukungan perangkat bergerak (*mobile*), opsi aksesibilitas, serta pusat bantuan ditambahkan agar sistem mudah diadopsi oleh pengguna dengan latar belakang non-teknis.

10. Integrasi eksternal.

    Notifikasi berbasis webhook (misalnya email atau WhatsApp), *single sign-on* untuk panel LSP, dan ekspor data dalam format CSV atau JSON disediakan guna meningkatkan interoperabilitas dan efisiensi operasional.

11. Evaluasi kuantitatif.

    Pengukuran waktu proses, *throughput* transaksi, dan biaya per sertifikat dilaksanakan untuk menyediakan dasar pengambilan keputusan terkait optimasi dan skala penerapan.

Dengan pematangan pada indeksasi dan pencarian, penguatan keamanan berkas dan manajemen kunci, serta pengujian pada jaringan publik atau testnet, DApp sertifikasi kompetensi ini memiliki peluang untuk bertransisi dari prototipe ke penerapan terbatas bersama mitra LSP dan BNSP. Integrasi mekanisme pencabutan sertifikat, observabilitas yang memadai, serta penguatan kontrak akan memperkuat tata kelola dan kepatuhan, sementara peningkatan pengalaman pengguna melalui pendekatan account abstraction dapat menurunkan hambatan adopsi. Kombinasi pencatatan on-chain dan penyimpanan *off-chain* berbasis CID tetap menjadi fondasi untuk menjaga integritas dan efisiensi sistem.

DAFTAR PUSTAKA

Aini, Q., Rahardja, U., Tangkaw, M. R., Santoso, N. P. L., & Khoirunisa, A. (2021). Embedding aBlockchain Technology Pattern Into the QR Code for an Authentication Certificate. *Journal of Applied Research and Technology*, *19*(4), 308–321

Buterin, V. (2015). *On Public and Private Blockchains | Ethereum Foundation Blog*. Ethereum Foundation Blog. https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains?

Chander, Er. N. (2023). Decentralized Healthcare System using Blockchain. *INTERANTIONAL JOURNAL of SCIENTIFIC RESEARCH in ENGINEERING and MANAGEMENT*, *07*(11), 1–11. https://doi.org/10.55041/ijsrem27214

CNN. (2025, April 12). *Jokowi Buka Suara Tudingan Ijazah Palsu: Siapa Menuduh Harus Buktikan*. Nasional; cnnindonesia.com. https://www.cnnindonesia.com/nasional/20250412103312-12-1218052/jokowi-buka-suara-tudingan-ijazah-palsu-siapa-menuduh-harus-buktikan

Damre, J., Kharche, A., Jungade, S., Sanap, V., Sudesh, P., Bachwani, A., & Babasaheb Ambedkar. (2022). BLOCKCHAIN: TYPES AND BENEFITS. *International Journal of Current Science*, *12*(1), 2250–1770. https://rjpn.org/ijcspub/papers/IJCSP22A1066.pdf

Di Angelo, M., Durieux, T., Ferreira, J., & Wien, T. (2023). *SmartBugs 2.0: An Execution Framework for Weakness Detection in Ethereum Smart Contracts Gernot Salzer*

Doan, T. V., Bajpai, V., & Psaras, Y. (2022). Towards Decentralised Cloud Storage with IPFS: Opportunities, Challenges, and Future Directions. *IEEE Internet Computing*, *26*(6), 7–15

Indonesia, R. (2018). *Peraturan Pemerintah Republik Indonesia Nomor 10 Tahun 2018 tentang Badan Nasional Sertifikasi Profesi*

Introduction to dapps. (2025, July 28). Ethereum.org. https://ethereum.org/en/developers/docs/dapps/?

Kushwaha, S. S., Joshi, S., Singh, D., Kaur, M., & Lee, H.-N. (2022). Ethereum Smart Contract Analysis Tools: A Systematic Review. *IEEE Access*, *10*, 1–1. https://doi.org/10.1109/access.2022.3169902

LaFountain, C. (2021). Blockchain, Cryptocurrencies, and Non-Fungible Tokens: What Libraries Need to Know. *Computers in Libraries*, *41*(4).

Lamichhane, S., & Herbke, P. (2024). *Verifiable Decentralized IPFS Cluster: Unlocking Trustworthy Data Permanency for Off-Chain Storage*

Meth, M. (2019). Blockchain in Libraries. *Library Technology Reports*, *55*(8). https://doi.org/10.5860/ltr.55n8

Murdiani, D., & Sobirin, M. (2022). PERBANDINGAN METODOLOGI WATERFALL DAN RAD (RAPID APPLICATION DEVELOPMENT) DALAM PENGEMBANGAN SISTEM INFORMASI. Jurnal Informatika Teknologi Dan Sains, 4(4), 302–306. https://doi.org/10.51401/jinteks.v4i4.2008

Nadler Prata, D., de Araújo, H. X., & Santos, C. (2021). A Literature Review about Smart Contracts Technology. International Journal of Advanced Engineering Research and Science, 8(2), 001–004. https://doi.org/10.22161/ijaers.82.1

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*.

Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). Bitcoin and Cryptocurrency Technologies. Princeton University Press.

Nguyen, Q.K. (2016). Blockchain: A financial technology for future sustainable development. Proceedings of the 3rd International Conference on Green Technology and Sustainable Development.

Pincheira, M., Donini, E., Vecchio, M., & Kanhere, S. (2022). A Decentralized Architecture for Trusted Dataset Sharing Using Smart Contracts and Distributed Storage. *Sensors*, *22*(23), 9118. https://doi.org/10.3390/s22239118

Pinna, A., Ibba, S., Baralla, G., Tonelli, R., & Marchesi, M. (2019). A Massive Analysis of Ethereum Smart Contracts Empirical Study and Code Metrics. *IEEE Access*, *7*, 78194–78213. https://doi.org/10.1109/access.2019.2921936

Rahman, Md. M., Tonmoy, M. T. K., Shihab, S. R., & Farhana, R. (2023). Blockchain-based certificate authentication system with enabling correction. *ArXiv Preprint*

Rawat, R., Chougule, R., Singh, S., Dixit, S., & Kadam, A. (2019). Smart Contracts using Blockchain. *International Research Journal of Engineering and Technology*, *6*(6).

Saja, K., & Stecyk, A. (2023). *Blockchain-Based Certification: Enhancing Transparency and Trust in Higher Education*

Saleh, O. S., Ghazali, O., & Rana, M. E. (2020). BLOCKCHAIN BASED FRAMEWORK FOR EDUCATIONAL CERTIFICATES VERIFICATION. *Journal of Critical Reviews*, *7*(3), 79–84

Sanjay, & Nabihasan. (2020). Blockchain technology and its application in libraries. *LIBRARY HERALD*, *58*(4), 118–125. https://doi.org/10.5958/0976-2469.2020.00030.10

Stark, J. (2016, June 4). *Making Sense of Blockchain Smart Contracts*. CoinDesk. https://www.coindesk.com/markets/2016/06/04/making-sense-of-blockchain-smart-contracts?

Sunarya, Po, A., Lutfiani, N., Dinda, Pratiwi, S., Kunci, K., & Sertifikasi. (2020). *Analisis Sistem Sertifikasi Profesi Untuk Pengembangan Kompetensi Mahasiswa*.

Suryawijaya, T. W. E. (2023). Memperkuat Keamanan Data melalui Teknologi Blockchain: Mengeksplorasi Implementasi Sukses dalam Transformasi Digital di Indonesia. *JSKP: Jurnal Studi Kebijakan Publik*, *2*(1), 55–68. https://doi.org/10.21787/jskp.2.2023.55-68

Szabo, N. (1997). Formalizing and Securing Relationships on Public Networks. First Monday, 2(9). https://doi.org/10.5210/fm.v2i9.548

Tarigan, A. (2022). Rancang bangun sistem penerbitan Surat Keterangan Pendamping Ijazah (SKPI) sebagai Non-Fungible Token (NFT) berbasis blockchain. *Jurnal Ilmiah Informatika Komputer*, *27*(3), 248–3257\

Thesing, T., Feldmann, C., & Burchardt, M. (2021). Agile versus Waterfall Project management: Decision Model for Selecting the Appropriate Approach to a Project. Procedia Computer Science, 181(1), 746–756. https://doi.org/10.1016/j.procs.2021.01.227

Trautwein, D., Raman, A., Tyson, G., Castro, I., Scott, W., Schubotz, M., Gipp, B., & Psaras, Y. (2022a). Design and evaluation of IPFS. *Proceedings of the ACM SIGCOMM 2022 Conference*. https://doi.org/10.1145/3544216.3544232

Trautwein, D., Raman, A., Tyson, G., Castro, I., Scott, W., Schubotz, M., Gipp, B., & Psaras, Y. (2022b). Design and Evaluation of IPFS: A Storage Layer for the Decentralized Web KEYWORDS Interplanetary file system, content addressing, decentralized web, libp2p, content addressable storage ACM Reference Format. *In Proceedings of the ACM SIGCOMM 2022 Conference*. https://doi.org/10.1145/3544216.3544232

Voshmgir, S. (2019). *Token Economy: How Blockchains and Smart Contracts Revolutionize the Economy* (berilustrasi). BlockchainHub. (Original work published 2019)

Wu, C., Xiong, J., Xiong, H., Zhao, Y., & Yi, W. (2022). A Review on Recent Progress of Smart Contract in Blockchain. IEEE Access, 10, 50839–50863. https://doi.org/10.1109/ACCESS.2022.3174052

LAMPIRAN