BAB II
LANDASAN TEORI

2.1 Teori-Teori Terkait Bahasan Penelitian (*Relevan Theories*)

Peneliti menyajikan teori-teori fundamental yang berkaitan erat dengan bahasan penelitian sebagai landasan. Hal ini bertujuan untuk memastikan bahwa setiap analisis yang dilakukan peneliti memiliki pijakan teoretis yang kuat, relevan, dan teruji secara akademis.

2.1.1 Rancang Bangun

Rancang bangun adalah suatu proses pembangunan sebuah sistem untuk menciptakan sistem baru maupun dapat mengganti atau memperbaiki sebuah sistem yang telah lama ada baik secara keseluruhan maupun hanya sebagian (Pratama et al., 2024).

2.1.2 Web

World Wide Web atau yang secara umum dikenal sebagai web merupakan sebuah sistem informasi berbasis internet yang memungkinkan dokumen dan sumber daya lainnya diidentifikasi oleh *Uniform Resource Locator* (URL) (Sibero, 2013). Web berfungsi sebagai media pertukaran data yang menampilkan teks, gambar, serta elemen multimedia melalui protokol komunikasi HTTP atau HTTPS (Rahmawati & Sari, 2024). Sebagai sebuah ekosistem digital, web memungkinkan terjadinya interaksi dua arah antara pengguna melalui peramban (*browser*) dan *server*.

2.1.3 Aplikasi

Aplikasi merupakan perangkat lunak (*software*) atau program komputer yang dirancang dan dikembangkan untuk menjalankan perintah tertentu guna membantu pengguna dalam menyelesaikan berbagai tugas secara spesifik sesuai dengan kebutuhan. Aplikasi dapat diklasifikasikan menjadi tiga jenis utama, yaitu aplikasi *desktop*, aplikasi web, dan aplikasi *mobile* (Hawari Nasution et al., 2023).

2.1.4 Bahasa Pemrogramman

Bahasa pemrograman adalah sarana komunikasi antara manusia dan komputer yang memiliki aturan sintaksis tertentu untuk mengeksekusi instruksi (Dipraja, 2014). Dalam konteks pengembangan aplikasi *mobile*, pemilihan bahasa pemrograman sangat menentukan performa aplikasi. Bahasa pemrograman modern dirancang untuk mendukung implementasi yang berkinerja tinggi serta memiliki kemampuan integrasi yang baik dengan berbagai sistem operasi (Rahmawati & Sari, 2024).

Adapun dalam penelitian ini Bahasa Pemrogramman yang digunakan adalah:

2.1.4.1 Javascript

JavaScript merupakan bahasa pemrograman tingkat tinggi yang bersifat dinamis dan mendukung berbagai paradigma pemrograman. JavaScript memungkinkan pengembang untuk menciptakan konten yang diperbarui secara dinamis, mengontrol elemen multimedia, serta mengimplementasikan animasi pada halaman web (MDN).

Sebagai bahasa pemrograman berbasis skrip, JavaScript berfungsi untuk menyempurnakan tampilan serta fungsionalitas sistem pada aplikasi berbasis web. Karakteristik utama dari JavaScript meliputi: (1) kategori bahasa pemrograman tingkat tinggi yang memiliki abstraksi kuat; (2) operasional yang bersifat *client-side*; (3) berorientasi pada objek; serta (4) bersifat *loosely typed*, di mana penentuan tipe data variabel dilakukan secara fleksibel saat *runtime* (Azizi et al., 2024).

2.1.4.2 Go

Berdasarkan dokumentasi resmi go.dev, Go merupakan bahasa pemrograman *open-source* yang dikembangkan oleh Google. Go dilengkapi dengan dukungan *built-in concurrency*

dan *standard library* yang kokoh untuk mempermudah pengembangan perangkat lunak yang kompleks. Selain keunggulan teknisnya, Go didukung oleh ekosistem yang luas juga ketersediaan tools pengembangan yang komprehensif.

Bahasa pemrograman Go dikembangkan dengan tujuan utama untuk menciptakan bahasa yang unggul dalam aspek kecepatan, keandalan, skalabilitas, dan kesederhanaan. Sebagai bahasa pemrograman bertipe statis, Go mampu menghasilkan kode biner yang dikompilasi secara efisien untuk berjalan langsung di atas mesin target. Meskipun desainnya terinspirasi dari efisiensi bahasa pemrograman C, Go telah dimodernisasi untuk menjawab tantangan komputasi abad ke-21. Dalam implementasinya, Go sangat fleksibel dan dapat digunakan secara luas untuk membangun aplikasi sistem, pengembangan situs web, serta berbagai perangkat lunak berskala besar lainnya. (Azizi et al., 2024).

2.1.4.3 Solidity

Berdasarkan dokumentasi resmi (soliditylang.org), Solidity didefinisikan sebagai bahasa pemrograman tingkat tinggi yang berorientasi pada objek untuk mengimplementasikan *smart contracts*. Bahasa ini dirancang secara spesifik untuk dijalankan di atas *Ethereum Virtual Machine* (EVM) serta berbagai jaringan *blockchain* lain yang kompatibel.

Sebagai bahasa utama dalam ekosistem Ethereum, Solidity berfungsi untuk menyusun logika bisnis yang kompleks ke dalam bentuk *smart contract*. Karakteristik utama dari bahasa ini adalah kemampuannya untuk menangani transaksi di dalam jaringan terdesentralisasi dengan tingkat keamanan yang tinggi. (Alim et al., 2025).

2.1.5 *Framework*

Framework atau kerangka kerja merupakan sekumpulan pustaka (*libraries*) dan aturan yang dirancang untuk mempermudah pengembang dalam membangun aplikasi secara terstruktur (Gunawan et al., 2024). Penggunaan *framework* sangat krusial dalam pengembangan perangkat lunak modern karena mampu meningkatkan efisiensi waktu serta menjamin konsistensi kode di seluruh bagian aplikasi. Dengan adanya *framework*, pengembang tidak perlu membangun fungsi dasar dari nol, melainkan dapat lebih fokus pada pengembangan logika bisnis utama aplikasi.

Selain itu, *framework* memiliki multifungsi yang dapat dimanfaatkan untuk mempercepat proses pembuatan aplikasi, memudahkan pembacaan kode, hingga membantu seluruh siklus pengembangan mulai dari perencanaan, pembuatan, pengujian, hingga pemeliharaan (Haniefardy et al., 2019).

Adapun dalam penelitian ini *framework* yang digunakan adalah:

2.1.5.1 TailwindCSS

Berdasarkan dokumentasi resminya (tailwindcss.com), Tailwind CSS didefinisikan sebagai sebuah kerangka kerja (*framework*) CSS yang mengusung konsep *utility-first*. Berbeda dengan *framework* CSS konvensional yang menyediakan komponen siap pakai, Tailwind CSS bekerja dengan cara menyediakan kumpulan *utility classes* tingkat rendah (*low-level*) yang memungkinkan pengembang untuk membangun desain khusus secara langsung di dalam struktur HTML tanpa harus menulis baris kode CSS eksternal.

Tailwind CSS merupakan *framework* yang sangat efektif digunakan untuk membangun *User Interface* dengan proses desain yang jauh lebih cepat (Azizi et al., 2024).

2.1.5.2 NextJS

Next.js merupakan *framework* React tingkat lanjut yang dirancang untuk membangun aplikasi web berperforma tinggi dengan fitur *full-stack*, yang mencakup pengelolaan *front-end* sekaligus solusi *back-end* untuk perutean dan manajemen basis data.

Next JS menggunakan metode *Static Site Generation* (SSG) unggul sebagai teknik tercepat dengan efisiensi hingga 34,6% dibandingkan *Server-Side Rendering* (SSR) pada halaman tertentu. Meskipun demikian, Next.js memberikan fleksibilitas bagi pengembang untuk menggabungkan metode SSR, SSG, dan *Client-Side Rendering* (CSR) dalam satu aplikasi guna menyeimbangkan kebutuhan antara kecepatan akses, performa server, dan keakuratan data secara *real-time* (Hanafi et al., 2024).

2.1.5.3 Go-Gin

Berdasarkan dokumentasi resmi (gin-gonic.com), Gin (atau sering disebut Gin Gonic) adalah sebuah HTTP *web framework* berperforma tinggi yang ditulis menggunakan bahasa pemrograman Go. *Framework* ini dirancang secara khusus untuk memfasilitasi pengembangan aplikasi web dan *microservice* melalui fitur-fitur unggulan seperti sistem routing yang sangat cepat, manajemen *middleware* yang fleksibel, serta kemampuan *rendering* JSON yang efisien.

Sebagai *framework* yang dikembangkan oleh komunitas Go, Gin menawarkan keunggulan dalam menyederhanakan serta mempercepat siklus pengembangan aplikasi perangkat lunak. Salah satu aspek fundamental dari Gin adalah fleksibilitas kodenya yang memungkinkan komponen tertentu untuk diperluas dan digunakan kembali. Hal ini secara signifikan

meningkatkan efisiensi proses *coding* karena pengembang tidak perlu membangun fungsionalitas dasar dari awal, sehingga konsistensi dan skalabilitas sistem dapat tetap terjaga dengan baik. (Avatara & Tan, 2024)

2.1.5.4 Flutter

Menurut dokumentasi resmi dari flutter.dev, Flutter merupakan *UI toolkit open source* yang dikembangkan oleh Google untuk membangun aplikasi yang dikompilasi secara native untuk perangkat seluler, web, dan desktop. Sejalan dengan hal tersebut, dalam penelitian yang dilakukan oleh Rahmawati dan Sari (2024) mengenai pengembangan aplikasi perizinan tugas keluar, dijelaskan bahwa Flutter adalah *framework* yang memfasilitasi pengembang dalam menciptakan antarmuka pengguna yang fleksibel, responsif, juga dapat dijalankan pada platform Android maupun iOS melalui satu basis kode (*single codebase*).

Keunggulan utama Flutter terletak pada arsitekturnya yang membedakannya dengan *framework* lain, yaitu proses kompilasi kode langsung ke *native* (seperti Android NDK, LLVM, dan AOT *compiled*) tanpa melibatkan *interpreter*. Hal ini memungkinkan aplikasi memiliki performa yang lebih cepat dan efisien. Seluruh komponen dalam Flutter diimplementasikan melalui *widget*, baik berupa komponen visual maupun sebagai wadah penampung bagi *widget* lainnya (Rahmawati & Sari, 2024).

Dalam pengembangan aplikasinya, Flutter menggunakan bahasa pemrograman Dart yang juga dikembangkan oleh Google untuk menggantikan JavaScript. Bahasa Pemrograman Dart dipilih karena kemudahannya dalam pengembangan aplikasi modern, dukungan implementasi berkinerja tinggi, serta

kemampuannya untuk dijalankan sebelum dikompilasi. Integrasi antara arsitektur widget dan penggunaan bahasa Dart ini menjadikan Flutter sebagai pilihan utama dalam pengembangan aplikasi mobile modern yang membutuhkan pengalaman pengguna yang konsisten di berbagai platform (Rahmawati & Sari, 2024).

2.1.6 *Blockchain Technology*

Teknologi *blockchain* pertama kali diperkenalkan oleh Nakamoto (2008) melalui konsep yang ia sebut sebagai '*chain of blocks*'. Dalam jurnal fundamentalnya, Nakamoto mendefinisikan infrastruktur ini sebagai sebuah server stempel waktu (*time-stamp server*) yang bekerja secara terdesentralisasi. Secara teknis, struktur data ini terbentuk dari rangkaian blok yang saling terhubung melalui *hash*.

Cara kerja *blockchain* sendiri yaitu dengan server stempel waktu yang bekerja mengambil *hash* dari sebuah blok berisi data untuk diberi stempel waktu dan menyebarkan *hash* tersebut secara luas. Setiap stempel waktu menyertakan *hash* dari stempel waktu sebelumnya, sehingga membentuk sebuah rantai di mana setiap stempel waktu tambahan akan memperkuat stempel waktu sebelumnya (Nakamoto, n.d.)

Selain aspek struktur data, aspek keamanan dalam *blockchain* sangat bergantung pada mekanisme konsensus antarjaringan. Nakamoto menekankan bahwa sebuah catatan dalam *blockchain* tidak dapat diubah selama mayoritas jaringan bersifat jujur. Mekanisme operasional jaringan ini dijelaskan sebagai berikut:

> "Langkah-langkah untuk menjalankan jaringan adalah sebagai berikut: ... Unit jaringan (*nodes*) menerima blok tersebut hanya jika semua transaksi di dalamnya valid dan belum digunakan. Unit jaringan menyatakan penerimaan mereka terhadap blok tersebut dengan bekerja

membuat blok berikutnya dalam rantai, menggunakan *hash* dari blok yang diterima sebagai *hash* sebelumnya." (Nakamoto, n.d.).

Dengan demikian, *blockchain* menurut rujukan aslinya bukan sekadar media penyimpanan, melainkan sebuah protokol kepercayaan di mana setiap unit jaringan (*nodes*) melakukan validasi mandiri terhadap setiap data sebelum menyatakannya sah dan menyambungkannya ke dalam rantai utama.

2.1.7 *Smart Contract*

Smart contract adalah program komputer yang tersimpan di dalam jaringan *blockchain* yang berfungsi untuk menjalankan "perjanjian" secara otomatis tanpa perlu pihak ketiga (Ganapathy et al., 2022). Sistem ini bekerja berdasarkan logika pemrograman yang akan mengeksekusi perintah setelah syarat-syarat yang ditentukan terpenuhi. Berbeda dengan kontrak kertas yang membutuhkan keterlibatan pihak ketiga seperti bank atau notaris, *smart contracts* beroperasi secara mandiri tanpa campur tangan manusia. Dengan pemanfaatan teknologi *distributed ledger*, setiap transaksi bersifat transparan, *immutable* (tidak dapat diubah), dan terproteksi dari risiko manipulasi data.

2.1.8 Kriptografi

Secara etimologi, kriptografi berasal dari bahasa Yunani, yaitu *kryptos* (tersembunyi atau rahasia) dan *graphein* (menulis atau tulisan). Secara harfiah, kriptografi merupakan seni atau teknik menulis secara tersembunyi untuk menyampaikan pesan yang bersifat rahasia (Amalya et al., 2023).

Dalam konteks modern, kriptografi didefinisikan sebagai ilmu yang mempelajari teknik-teknik matematika yang berkaitan dengan aspek keamanan informasi, seperti menjaga kerahasiaan data, keabsahan data, serta autentikasi data. Kriptografi mengandalkan dua proses utama, yaitu:

- Enkripsi (*Encryption*)**:** Proses menyandikan atau mengubah pesan asli yang dapat dibaca (*plaintext*) menjadi pesan acak yang tidak dapat dimengerti (*ciphertext*).
- Deskripsi (*Decryption*)**:** Proses kebalikan dari enkripsi, yaitu mengembalikan isi pesan acak (*ciphertext*) menjadi bentuk informasi semula (*plaintext*).

2.1.9 Algoritma

Algoritma merupakan serangkaian instruksi atau langkah-langkah logis, terstruktur, dan sistematis yang dirancang untuk memecahkan suatu masalah atau mencapai hasil tertentu (Selviana Anggreani & Yahfizham Yahfizham, 2024). Istilah ini awalnya berakar dari nama ilmuwan muslim Abu Ja'far Muhammad Ibnu Musa Al-Khwarizmi dan berkembang dalam dunia komputer sebagai metode baku untuk menyelesaikan persoalan. Sebuah algoritma yang baik tidak hanya harus akurat dalam menghasilkan keluaran yang benar dan mendekati nilai sebenarnya, tetapi juga harus efisien. Efisiensi ini diukur berdasarkan optimasi waktu eksekusi yang cepat serta penggunaan memori komputer yang seminimal mungkin, karena logika yang tepat akan menjadi sia-sia jika prosesnya memakan waktu terlalu lama atau menghabiskan terlalu banyak sumber daya komputer.

Dalam konteks komputasi, terdapat hubungan yang sangat erat antara algoritma dan pemrograman. Algoritma bertindak sebagai cetak biru atau rancangan konseptual yang berisi urutan penyelesaian masalah, sedangkan program adalah implementasi dari algoritma tersebut menggunakan bahasa pemrograman tertentu agar dapat dieksekusi oleh komputer. Dengan kata lain, algoritma pemrograman merupakan jembatan penerjemah logika manusia ke dalam bahasa komputer guna menjalankan fungsi spesifik secara padu. Meskipun setiap orang dapat menyusun algoritma yang berbeda untuk menyelesaikan satu masalah yang sama,

fokus utamanya tetap pada pencarian langkah yang paling logis, efektif, dan efisien (Selviana Anggreani & Yahfizham Yahfizham, 2024).

2.1.10 Elliptic Curve Digital Signature Algorithm

Elliptic Curve Digital Signature Algorithm (ECDSA) merupakan sebuah algoritma kriptografi kunci asimetris yang berfungsi sebagai analogi kurva eliptik dari *Digital Signature Algorithm* (DSA) untuk menghasilkan tanda tangan digital (Johnson et al., n.d.). Secara teknis, ECDSA didefinisikan sebagai skema tanda tangan digital yang beroperasi pada grup titik-titik dalam kurva eliptik di atas bidang berhingga (*finite field*). Mekanisme keamanan utama dari algoritma ini bersandar pada tingkat kesulitan matematis dari *Elliptic Curve Discrete Logarithm Problem* (ECDLP), di mana hingga saat ini belum ditemukan algoritma waktu polinomial yang mampu memecahkan masalah logaritma diskrit tersebut pada kurva yang dipilih dengan benar (Johnson et al., n.d.).

Implementasi ECDSA dalam sistem autentikasi produk menawarkan keunggulan signifikan dibandingkan algoritma kunci publik konvensional lainnya, seperti RSA. Keunggulan utamanya terletak pada efisiensi ukuran kunci; ECDSA mampu menyediakan tingkat keamanan yang setara dengan kunci yang jauh lebih kecil, sehingga proses komputasi menjadi lebih cepat dan penggunaan memori lebih hemat (Johnson et al., n.d.). Karakteristik ini menjadikan ECDSA sangat ideal untuk diterapkan pada perangkat dengan sumber daya terbatas, seperti pemindaian berbasis aplikasi *mobile* untuk memverifikasi keaslian produk melalui *Quick Response* (QR) *Code*. Dalam prosesnya, produsen menggunakan kunci privat untuk menandatangani data produk guna menjamin integritas dan autentikasi, sementara konsumen menggunakan kunci publik untuk melakukan verifikasi guna memastikan bahwa data tersebut benar-benar berasal dari otoritas yang sah dan belum mengalami manipulasi.

2.1.11 Quick Response Code

Quick Response Code (QR Code) adalah sebuah simbologi matriks dua dimensi yang terdiri dari kumpulan modul berbentuk persegi yang disusun dalam pola persegi keseluruhan (ISO/IEC 18004, 2024). Berdasarkan standar ISO/IEC 18004, QR Code telah mengalami evolusi teknis yang terdiri dari empat model utama, yaitu:

1. QR Code Model 1: Spesifikasi asli yang direkomendasikan untuk sistem tertutup.
2. QR Code Model 2**:** Bentuk yang disempurnakan dengan tambahan *alignment patterns* untuk mendukung navigasi pada simbol yang lebih besar.
3. QR Code (Standar Utama): Pengembangan dari Model 2 yang mendukung fitur pencerminan (*mirror image*) dan pembalikan reflektansi (simbol terang di atas latar gelap). Model ini merupakan standar yang direkomendasikan untuk aplikasi sistem terbuka.
4. *Micro* QR Code: Varian dengan jumlah modul *overhead* yang dikurangi untuk penggunaan pada area yang sangat terbatas.

Secara struktural, QR Code dicirikan oleh *finder pattern* unik pada tiga sudut simbol yang berfungsi untuk mengoptimalkan kecepatan pemindaian dalam berbagai posisi dan kemiringan. Dalam implementasi autentikasi, fitur *error correction* pada QR Code menjamin ketahanan data tanda tangan digital (ECDSA) terhadap kerusakan fisik, sehingga menjadikannya jembatan yang andal menuju sistem keamanan berbasis *blockchain*. Integrasi ini menciptakan mekanisme verifikasi yang sangat efisien; alih-alih melalui proses birokrasi yang memakan waktu, pengguna dapat memindai QR Code untuk berinteraksi langsung dengan *smart contract* guna mendapatkan umpan balik verifikasi secara instan (*real-time verification feedback*) (Noorhizam et al., 2023).

2.1.12 Tools Perangkat Lunak

2.1.12.1 Visual Studio Code

Visual Studio Code adalah *code editor* yang dikembangkan oleh Microsoft untuk Windows, Linux dan mac OS. Visual Studio Code memiliki fitur untuk *debugging, git control*, penyorotan sintaksis, penyelesaian kode cerdas, *snippet* dan *refactoring code* (Azizi et al., 2024). Code editor ini peneliti gunakan karena memiliki ekosistem plugin yang sangat luas, bersifat ringan (*lightweight*), serta mendukung berbagai bahasa pemrograman secara efisien melalui fitur *IntelliSense* yang mempercepat proses penulisan kode.

2.1.12.2 PgAdmin 4

Berdasarkan web resminya, pgAdmin 4 adalah software *Open Source* terkemuka untuk Postgres. pgAdmin 4 dirancang khusus untuk memenuhi kebutuhan pengguna Postgres, baik pemula maupun pengguna berpengalaman, dengan menyediakan antarmuka grafis yang kuat. Kehadiran versi terbaru ini semakin menyederhanakan proses pembuatan, pemeliharaan, serta penggunaan berbagai objek database melalui fitur-fitur yang lebih modern dan fleksibel.

Didalam jurnal dan penelitian lain, PgAdmin 4 dijelaskan sebagai perangkat lunak atau *tool* bawaan dari PostgreSQL yang dirancang dengan antarmuka visual (GUI) untuk mempermudah pengelolaan *database*. Kehadiran pgAdmin 4 menjadi solusi bagi pengguna agar dapat membuat dan memanipulasi *database* hanya melalui menu-menu yang tersedia tanpa harus menghafal atau mengetikkan perintah SQL secara manual dalam mode teks. Hal ini menjadikan proses administrasi *database* jauh lebih efisien, praktis, dan ramah bagi

pengguna yang baru mempelajari sistem *database* (Munawaroh, 2005).

2.1.12.3 Postman

Postman merupakan platform kolaborasi yang dirancang khusus untuk mempermudah siklus pengembangan *Application Programming Interface* (API). Perangkat lunak yang digagas oleh Abhinav Asthana ini telah menjadi standar industri dalam melakukan pengujian, pengembangan, hingga dokumentasi API secara efisien. (Azizi et al., 2024).

Penggunaan Postman dalam penelitian ini didasarkan pada kemampuannya untuk melakukan pengujian fungsionalitas *backend* tanpa harus menunggu implementasi antarmuka pengguna (*user interface*) selesai. Cara kerja Postman adalah dengan mengirimkan permintaan (*request*) ke *server*.

2.1.13 Tools Pemodelan Sistem

2.1.13.1 Flowchart

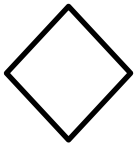
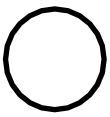
Flowchart digunakan sebagai sebuah strategi visual yang sistematis dan efektif untuk menggambarkan urutan logika, langkah-langkah, atau prosedur penyelesaian masalah dalam suatu program atau proses operasional. *Flowchart* menggunakan simbol-simbol grafis untuk merinci setiap aspek proses secara mendalam, sehingga memudahkan pemahaman terhadap instruksi yang kompleks, identifikasi masalah potensial, serta pengambilan keputusan yang terinformasi (Kus Indrani Listyoningrum et al., 2023).

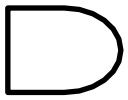
Pada jurnal lain, *flowchart* didefinisikan sebagai diagram dengan simbol-simbol grafis yang menyatakan aliran proses untuk memecahkan masalah ke dalam bagian-bagian yang lebih kecil. Fungsi utamanya adalah memberikan gambaran proses produksi agar mudah dipahami berdasarkan urutan langkah dari

satu tahap ke tahap berikutnya. Agar efektif, pembuatan *flowchart* harus mengikuti kaidah tertentu, seperti penggambaran dari atas ke bawah atau kiri ke kanan, memiliki kejelasan titik awal dan akhir, serta menggunakan deskripsi kata kerja pada setiap tahapannya dengan simbol-simbol yang baku (Malabay, 2016).

Berdasarkan Jurnal Penelitian (Malabay, 2016) Berikut adalah daftar simbol beserta fungsinya:

Tabel 2.1 Simbol-simbol Flowchart

Gambar	Nama
	*Terminator* (awal dan akhir)
	*Input* Data dan *Output* informasi
	Proses
	*Predefine* Proses
	*Dokumen*
	Keputusan
	Penghubung

	*Delay*

2.1.13.2 Rich Picture

Metode *Rich Pictures* (RP) merupakan teknik penelitian kualitatif inovatif yang menggunakan media gambar atau karton untuk menangkap persepsi, pengalaman, dan perasaan individu maupun kelompok terhadap suatu situasi yang kompleks atau "berantakan". Berakar dari *Soft Systems Methodology* yang dikembangkan oleh Checkland pada tahun 1970-an, metode ini berfungsi sebagai alat penggali pengetahuan kontemporer yang mampu menyampaikan informasi yang sulit diungkapkan melalui kata-kata. Dengan mengandalkan visual, RP memungkinkan untuk melampaui hambatan bahasa, budaya, dan narasi yang sudah ada, sehingga pembaca dapat mengakses pemikiran bawah sadar dan melihat pola atau hubungan antar sistem yang sebelumnya tidak disadari (Gisby et al., 2023).

2.1.13.3 *Unified Modelling Language* (UML)

Unified Modelling Language (UML) digunakan untuk memvisualisasikan proses bisnis, rancangan basis data, hingga komponen pemrograman yang dibutuhkan (Siska Narulita et al., 2024).

UML terdiri dari beberapa diagram yang menggambarkan permasalahan dan solusinya (Siska Narulita et al., 2024). Berikut adalah diagram-diagram yang peneliti gunakan di penelitian ini.

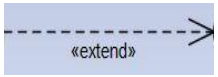
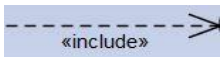
1. *Use Case Diagram*

Use Case Diagram merupakan visualisasi fungsionalitas sistem yang terdiri dari komponen utama berupa *actor*, *use case*, dan relasi antar komponen tersebut. Diagram ini sangat penting bagi analis dalam menyusun kebutuhan pengembangan sistem. Dengan menggunakan berbagai simbol atau notasi khusus, diagram ini membantu menjelaskan perancangan fitur-fitur sistem kepada pengguna sehingga seluruh kebutuhan fungsionalitas yang akan dibangun dapat terpetakan dengan jelas (Siska Narulita et al., 2024).

Tabel 2.2 ini merupakan simbol-simbol yang digunakan dalam *use case diagram* diantaranya:

Tabel 2.2 Simbol Use Case Diagram (Shalahuddin, 2018)

Simbol	Deskripsi
Use Case 	Fungsionalitas sistem digambarkan sebagai sekumpulan unit yang saling berinteraksi melalui pertukaran pesan, baik antar unit maupun dengan aktor. Biasanya, fungsi tersebut direpresentasikan dalam bentuk frasa kata kerja sebagai nama *use case*.
Aktor 	Entitas seperti pengguna, proses, atau sistem lain yang berada di luar sistem yang dikembangkan, namun tetap berinteraksi dengan sistem informasi tersebut.
Asosiasi 	Bentuk komunikasi antara aktor dengan *use case* yang menggambarkan adanya interaksi

	atau partisipasi aktor terhadap fungsi-fungsi dalam sistem.
Ekstensi 	Suatu hubungan antara *use case* utama dan *use case* tambahan, di mana keberadaan *use case* tambahan tidak memengaruhi kemampuan *use case* utama untuk tetap berjalan secara independen.
Generalisai 	Suatu hubungan generalisasi antara dua *use case* yang menunjukkan bahwa satu *use case* memiliki cakupan fungsi yang lebih luas, sementara *use case* lainnya merupakan bentuk spesialisasi dari fungsi tersebut.
Include 	Suatu hubungan antara *use case* yang menunjukkan adanya ketergantungan, di mana pelaksanaan suatu *use case* mensyaratkan keterlibatan *use case* lain agar fungsi sistem dapat berjalan dengan baik.

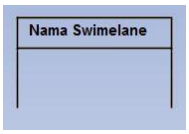
2. *Activity Diagram*

Activity Diagram digunakan untuk merepresentasikan aliran proses atau rangkaian aktivitas di dalam sistem, mulai dari titik awal, titik pengambilan keputusan, hingga proses tersebut berakhir. Berbeda dengan *use case* yang menunjukkan cara aktor memakai sistem, *activity diagram* lebih fokus pada tahapan teknis atau langkah-langkah yang terjadi di dalam sistem, termasuk visualisasi proses paralel. Secara struktural, diagram ini dirancang berdasarkan satu atau

lebih *use case* yang telah didefinisikan sebelumnya (Siska Narulita et al., 2024).

Tabel 2.3 ini merupakan simbol-simbol yang digunakan dalam *activity diagram* diantaranya:

Tabel 2.3 Simbol Activity Diagram (Shalahuddin, 2018)

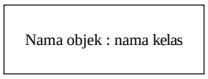
Simbol	Deskripsi
Status awal 	Status yang menunjukkan awal dari suatu aktivitas sistem, yang menjadi titik mulai dalam *activity diagram*.
Aktivitas 	Aktivitas sistem yang menggambarkan suatu proses, yang biasanya dituliskan dengan menggunakan kata kerja sebagai penjelasannya.
Percabangan 	Percabangan pada aktivitas yang menunjukkan adanya lebih dari satu kemungkinan alur yang dapat dipilih dalam sistem.
Penggabungan / *join* 	Penggabungan aktivitas yang menunjukkan bahwa beberapa alur proses bertemu dan dilanjutkan menjadi satu aktivitas.
Status akhir 	Status yang menandai berakhirnya suatu aktivitas sistem sebagai titik akhir dalam *activity diagram*.
Swinelane 	Memisahkan pihak atau unit organisasi yang memiliki tanggung jawab terhadap masing-masing aktivitas dalam proses.

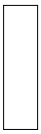
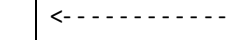
3. *Sequence Diagram*

Sequence Diagram menggambarkan interaksi dinamis antar-objek di dalam sistem melalui pesan yang dikirimkan antar objek dari waktu ke waktu. Diagram ini memvisualisasikan seluruh objek yang berkaitan dalam sebuah *use case* dan menunjukkan rangkaian pesan serta kolaborasi yang terjadi selama sistem dieksekusi. Melalui diagram ini, pengembang dapat melihat urutan interaksi yang terjadi secara detail untuk memastikan komunikasi antar komponen sistem berjalan sesuai rancangan.

Tabel 2.4 ini merupakan simbol-simbol yang digunakan dalam *sequence diagram* diantaranya:

Tabel 2.4 Simbol Sequence Diagram (Shalahuddin, 2018)

Simbol	Deskripsi
Aktor 	Orang, proses, atau sistem lain di luar sistem yang tetap terlibat dalam interaksi dengan sistem informasi.
Garis hidup (*lifetime*) 	Menunjukkan perjalanan atau perubahan keadaan suatu objek sepanjang siklus hidupnya.
Objek 	Menjelaskan objek yang terlibat dalam interaksi dengan cara saling mengirim dan menerima pesan.

Waktu aktif 	Menjelaskan objek yang aktif dalam sistem dan terlibat dalam proses pengiriman serta penerimaan pesan.
Pesan tipe *create* 	Menjelaskan bahwa sebuah objek menghasilkan objek lain, yang ditandai dengan arah panah menuju objek yang dibuat.
Return Message 	Menjelaskan bahwa suatu objek atau metode mengembalikan hasil kepada objek lain, yang ditandai dengan arah panah menuju objek yang menerima kembalian.

2.1.14 Pengujian Sistem

2.1.14.1 Black Box Testing

Black box testing juga dikenal sebagai *functional testing* atau *behavioral testing*, yaitu metode pengujian perangkat lunak yang menitikberatkan pada fungsi sistem tanpa melihat struktur internal atau kode program. Dalam metode ini, sistem dianggap sebagai "kotak hitam", sehingga penguji hanya memperhatikan input yang diberikan dan output yang dihasilkan tanpa mengetahui proses di dalamnya.

Black box testing merupakan pengujian yang berfokus pada spesifikasi fungsional perangkat lunak, di mana penguji menyusun berbagai kondisi input untuk memastikan sistem berjalan sesuai dengan yang diharapkan. menyatakan bahwa pengujian ini menggunakan data uji yang bersumber dari spesifikasi fungsional tanpa memperhatikan struktur internal program (Maulida et al., 2025).

Black box testing dikenal sebagai pengujian perilaku, di mana penguji tidak mengetahui struktur internal maupun logika dari perangkat lunak yang diuji. Pengujian ini didasarkan pada spesifikasi kebutuhan tanpa perlu menganalisis kode program, serta dilakukan dari sudut pandang pengguna akhir.

Terdapat beberapa teknik dalam *black box testing*, seperti partisi, analisis nilai batas, grafik sebab-akibat, pengujian *orthogonal array*, transisi *state*, dan *fuzzing*. Metode ini memiliki kelebihan yaitu dapat membantu menemukan bagian yang belum sesuai dengan spesifikasi kebutuhan. Namun, kekurangannya adalah pengujian tidak dapat dilakukan secara menyeluruh karena keterbatasan pengetahuan penguji terhadap sistem yang diuji (Praniffa et al., 2023).

Berdasarkan beberapa definisi yang telah diuraikan, *Black box testing* merupakan metode pengujian perangkat lunak yang berfokus pada fungsi sistem berdasarkan spesifikasi kebutuhan tanpa melihat kode atau struktur internal. Pengujian dilakukan dengan menguji *input* dan *output* dari sudut pandang pengguna, serta memiliki kelebihan dalam menemukan ketidaksesuaian fungsi, namun terbatas karena penguji tidak mengetahui bagian dalam sistem.

2.1.14.2 White Box Testing

White box testing merupakan metode pengujian struktural di mana perangkat lunak bersifat transparan bagi penguji, sehingga pengujian dilakukan dari sudut pandang pengembang dengan memahami struktur internal dan kode sumber untuk menemukan kesalahan logika. Pengujian ini dapat diterapkan pada berbagai tahapan pengembangan perangkat lunak, karena penguji dapat mengidentifikasi bagian kode yang tidak berfungsi dengan baik serta menemukan kesalahan atau asumsi yang keliru dalam program (Praniffa et al., 2023).

Menurut (Ghibran & Khamaeni, 2023) “Pengujian *white box* adalah pengujian perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi masukan dan keluaran yang sesuai dengan spesifikasi kebutuhan”.

Dapat disimpulkan bahwa *White box testing* memerlukan akses langsung ke kode sumber dan juga dikenal sebagai *clear box*, *glass box*, atau *open box*. Selain itu, metode ini memiliki kelebihan seperti mampu mendeteksi kode tersembunyi yang tidak diperlukan, melakukan pengujian secara menyeluruh terhadap struktur dan logika program, membantu dalam optimasi kode, serta dapat dilakukan meskipun antarmuka (GUI) masih dalam tahap pengembangan.

2.1.14.3 UAT

UAT *(User Acceptance Testing*) yaitu pengujian aplikasi terhadap pengguna yang dilakukan sehubungan dengan kebutuhan pengguna terakhir atau *end user* (Wulandari et al., 2023). UAT merupakan tahap pengujian yang melibatkan pengguna akhir untuk menilai apakah sistem yang dikembangkan sudah memenuhi harapan dan kebutuhan mereka sebelum resmi dioperasikan (Siti Muthiah Nuralifah et al., 2025).

2.2 Penelitian Sebelumnya *(Previous Work)*

Peninjauan terhadap penelitian terdahulu yang relevan merupakan langkah krusial dalam menyusun kerangka pemikiran penelitian ini. Melalui kajian terhadap berbagai studi sebelumnya, diharapkan dapat ditemukan alternatif solusi, baik melalui penggunaan kembali, modifikasi, maupun pengembangan model baru yang lebih efektif.

Selain itu, penelitian terdahulu berfungsi sebagai komparasi untuk memetakan perbedaan serta kebaruan antara studi yang sudah ada dengan penelitian yang sedang dilakukan. Hal ini bertujuan untuk memastikan bahwa solusi yang ditawarkan oleh peneliti mampu menjawab celah permasalahan yang belum terselesaikan pada penelitian sebelumnya.

Tabel 2.5 Penelitian Sebelumnya

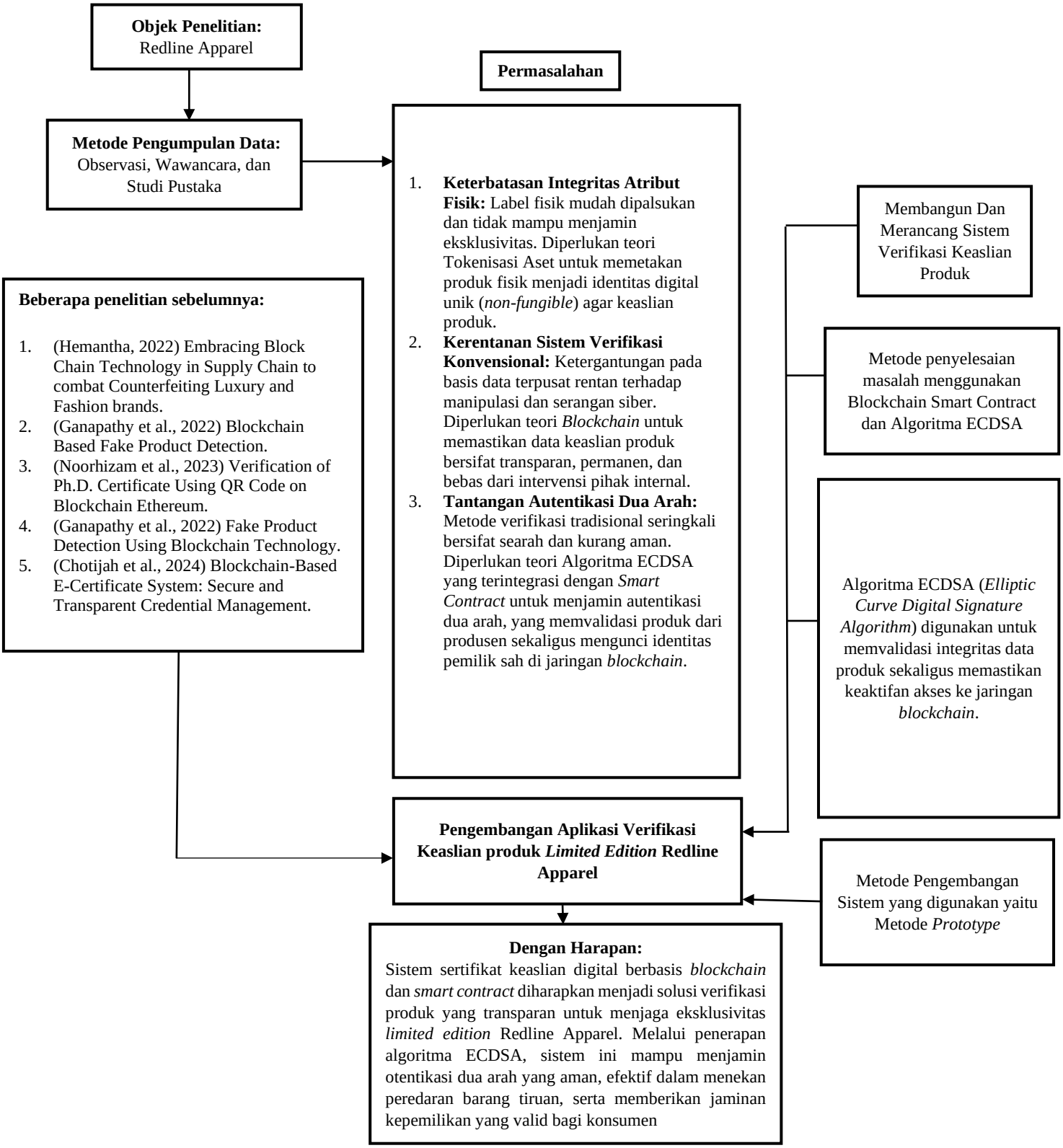
No	Judul & Penulis	Tujuan Penelitian	Hasil Penelitian	Persamaan & Perbedaan dengan Penelitian Saat Ini
1	*Embracing Block Chain Technology in Supply Chain to combat Counterfeiting Luxury and Fashion brands*, (Hemantha, 2022)	Mempresentasikan implementasi *blockchain* pada industri barang mewah untuk melindungi identitas merek.	*Blockchain* memberikan manfaat signifikan pada transparansi, privasi, dan pelacakan (*traceability*) melalui integrasi IoT.	Persamaan: Penggunaan teknologi blockchain sebagai instrumen *anti-counterfeiting*. Perbedaan: Menggunakan perangkat keras IoT (NFC/RFID), sedangkan penelitian ini

				menggunakan sistem pemindaian QR Code yang lebih efisien biaya.
2	*Blockchain Based Fake Product Detection*, (Ganapathy et al., 2022)	Mengatasi ancaman produk palsu melalui sistem identifikasi berbasis *blockchain* yang terintegrasi aplikasi.	QR Code yang terhubung dengan *blok* data unik mampu memberikan notifikasi autentikasi secara *real-time*.	Persamaan: Penggunaan QR Code, *smart contract*, dan penyimpanan data terdesentralisasi. Perbedaan: Fokus pada keamanan grafis fisik, sedangkan penelitian ini menerapkan algoritma ECDSA untuk keamanan tanda tangan digital kriptografis.
3	*Verification of Ph.D. Certificate Using QR Code on Blockchain Ethereum (Noorhizam et al., 2023)*	Mengatasi tantangan verifikasi ijazah Ph.D secara *real-time*, mengurangi biaya, birokrasi, dan mencegah pemalsuan sertifikat menggunakan teknologi *blockchain*.	Pengembangan sistem berbasis web yang terintegrasi dengan Ethereum. Sistem berhasil mengimplementasikan fitur tambah, update, hapus, serta verifikasi	Persamaan: Menggunakan teknologi *Blockchain* (Ethereum) dan *Smart Contract* (Solidity) sebagai instrumen utama untuk menjamin keamanan dan imutabilitas data.

			ijazah secara instan melalui pemindaian QR Code.	Perbedaan: Fokus spesifik pada dokumen akademik (Ph.D), menggunakan hashing SHA-256 untuk data teks, dan pemanfaatan QR Code sebagai jembatan verifikasi antara fisik sertifikat dengan data di *blockchain*.
4	*Fake Product Detection Using Blockchain Technology(Ganapathy et al., 2022)*	Mendeteksi produk fisik palsu di pasar guna melindungi reputasi *brand* dan kerugian finansial manufaktur serta keamanan konsumen.	Aplikasi Android yang memungkinkan produsen mendaftarkan produk dan konsumen melakukan pemindaian QR Code di toko untuk cek keaslian secara langsung.	Persamaan: Memanfaatkan imutabilitas *blockchain* untuk menyimpan kode unik produk dan menggunakan QR Code untuk akses cepat. Perbedaan: Objeknya adalah barang konsumen (ritel); platform berbasis Mobile (Android); menggunakan Firebase sebagai *database*; memiliki fitur pelacakan lokasi jika ditemukan produk palsu.

5	*Blockchain-Based E-Certificate System: Secure and Transparent Credential Management(Chotijah et al., 2024)*	Mengembangkan sistem e-sertifikat yang aman dan transparan di lingkup universitas menggunakan *private blockchain*.	Sistem yang mengotomatisasi penerbitan, penyimpanan, dan verifikasi sertifikat kegiatan mahasiswa menggunakan *Smart Contract* untuk efisiensi operasional.	Persamaan: Menggunakan Ethereum, *Smart Contract*, dan berfokus pada kredensial akademik. Perbedaan: Menggunakan metodologi *Waterfall* dalam pengembangannya; fokus pada sertifikat kegiatan (HIMATIF); memiliki mekanisme pencabutan (*revocation*) sertifikat jika terjadi kecurangan.

2.3 Kerangka Teoritis *(Theoritical Framework)*



Gambar 2.1 Kerangka Teoritis