BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Kajian terhadap penelitian terdahulu dilakukan untuk mengidentifikasi posisi penelitian ini dalam konteks literatur yang ada, serta menemukan celah penelitian (*research gap*) yang perlu diisi. Berikut adalah sintesis penelitian-penelitian yang relevan:

Tabel 2.1 Penelitian Terdahulu

Judul Jurnal	Nama Jurnal	Penulis / Tahun	Hasil	Relevansi
Rancangan Implementasi *Enterprise Resource Planning (ERP)* pada Sistem Pengelolaan Sales Order PT Jaya Mandiri Indotech	Jurnal Teknologi dan Manajemen	T. Fatmawati, Ridzky Kramanandita, Rabiathul Miza (2022)	ERP berhasil mengintegrasikan proses Sales Order dari inquiry hingga invoicing secara terstruktur dan mengurangi ketidakefisienan operasional [3].	Membuktikan kelayakan implementasi ERP Sales Order berbasis web pada konteks perusahaan lokal.
Business Process Re-engineering of KE and ES	*Proceedings of the International Conference*	Daniele De Guzman, Samuel Mirasol, King	Penerapan BPR pada SAP S/4HANA meningkatkan	Menguatkan urgensi restrukturisasi proses bisnis Sales Order

Judul Jurnal	Nama Jurnal	Penulis / Tahun	Hasil	Relevansi
Marketing Sales and Distribution Process ultilizing S4 Hana	*on Industrial Engineering and Operations Managemen t*	Perez, Grace Lorraine D. Intal / 2022	produktivitas proses *sales & distribution* hingga ±82% [7].	melalui sistem digital.
Reengineering Business Process Manufacturng Company Sales Module Using Odoo V12.0 Application	Jurnal Ilmiah Merpati (Menara Penelitian Akademika Teknologi Informasi)	Rika Wahyuni, I. M. Sukarsa, D. M. S. Arsa / 2021	*Reengineering* modul *sales* menggunakan Odoo menghasilkan tingkat *usability* 82% dan kepuasan pengguna 88% [8].	Mendukung penggunaan metode *prototype* dan UAT dalam pengembanga n modul *Sales Order.*
Design Of Customized Business Process Management With Implementatio	*Edelweis Applied Science and Technology*	Bayu Prabowo Sutjiatmo, Siti Nurkomariyah , Avia Enggar Tyasti, Fajar Ciputra Daeng Bani / 2024	Desain BPM terintegrasi berasil menghubungkan CRM, *warehouse,* hingga *finance*	Memperkuat pentingnya integrasi alur order-to-cash dalam prototype yang dikembangkan.

Judul Jurnal	Nama Jurnal	Penulis / Tahun	Hasil	Relevansi
n In The Apparel Industry As A Solution To Achieve Operational Excellence			dalam satu alur *order-to-cash [4].*	
Improvement of an order-to-cash business process by deploying lean six sigma tools: a case study	*International Journal of Productivity and Performance Management*	Emilia Kääriä, Ahm Shamsuzzoha / 2023	Bottleneck teridentifikasi pada pricing dan invoicing yang menyebabkan inefisiensi dan error [9].	Memberikan justifikasi kebutuhan digitalisasi proses quotation dan invoice dalam sistem.
Modernizing Opportunity-To-Order Workflows Through SAP BTP Integration Architecture	*International Journal of Applied Mathematics*	Padmanabhan Venkiteela / 2025	Modernisasi *workflow* meningkatkan *touchless orders* dari 26% menjadi 50% dan menurunkan *latency* hingga 35% [10].	Menguatkan manfaat otomatisasi workflow Sales Order berbasis teknologi modern.

Judul Jurnal	Nama Jurnal	Penulis / Tahun	Hasil	Relevansi
A PLS-SEM for ERP Adoption Readiness in Government Organizations: A TOEPP Conceptual Model	*IEEE Access*	R. Witjaksono, T. K. Priyambodo, Mardhani Riasetiawan / 2025	Faktor *Techonology, Organization, Environment, People,* dan *Process* semuanya berpengaruh signifikan terhadap *readiness* ERP [11].	Menunjukkan pentingnya kesiapan organisasi sebagai konteks latar belakang penelitian, meskipun bukan fokus utama.
Enhancing Readiness: A Comprehensive Study on ERP Implementation Readiness Factors	*International Conference on Information Management and Technology*	Bhre Bramantyo, Evelyn Halim, Salsabila Oktariana Andanti Arring, Dyah Wahyu Sukmaningsih / 2024	Faktor utama *readiness* meliputi *training,* kompetensi IT, manajemen proyek, komunikasi, dan *change management [12].*	Mendukung pentingnya UAT dan pelatihan pengguna dalam implementasi prototype.

Berdasarkan sintesis penelitian terdahulu, dapat diidentifikasi bahwa sebagian besar penelitian yang ada berfokus pada implementasi ERP menggunakan *platform enterprise* yang sudah jadi seperti Odoo atau SAP, atau pada studi evaluasi kesiapan implementasi ERP (*readiness assessment*). Penelitian yang secara spesifik mengembangkan *prototype* fungsional ERP modul *Sales Order* berbasis web dari awal (*custom development*) menggunakan *Next.js* dan *Firebase Firestore* pada perusahaan distribusi skala menengah di Indonesia masih terbatas. Inilah celah penelitian yang diisi oleh penelitian ini, yaitu pengembangan *prototype* fungsional ERP *Sales Order* yang dibangun secara *custom*, mencakup alur *order-to-cash end-to-end*, dilengkapi *approval workflow*, dan diuji melalui *black-box testing* serta UAT.

2.2 Teori yang berkaitan

2.2.1 *Enterprise Resource Planning*

Enterprise Resource Planning (ERP) adalah sistem informasi terintegrasi yang digunakan untuk mengelola dan mengoordinasikan seluruh proses bisnis dalam suatu organisasi [13]. Selain sebagai perangkat lunak, ERP sering kali juga digunakan sebagai *platform* strategis yang mendukung efisiensi operasional, koordinasi lintas fungsi, serta pengambilan keputusan berbasis data [1]. ERP memungkinkan integrasi berbagai aktivitas bisnis ke dalam satu sistem yang terpusat, sehingga seluruh informasi dapat diakses secara *real-time* dan konsisten.

Secara konseptual, ERP didefinisikan sebagai pendekatan modern dalam menjalankan organisasi dengan dukungan sistem informasi yang komprehensif untuk mendukung proses bisnis utama [14]. ERP juga dipahami sebagai perangkat lunak yang mampu mengintegrasikan seluruh proses dalam perusahaan, termasuk pemasaran, keuangan, produksi, dan sumber daya manusia [1]. ERP merupakan evolusi dari sistem *Material Requirements Planning* (MRP) pada tahun 1960-an, yang berkembang menjadi MRP II dan akhirnya menjadi ERP yang mencakup seluruh fungsi bisnis [9].

Dalam implementasinya, ERP terdiri dari berbagai modul yang saling terintegrasi. Modul *Sales and Distribution* (SD) merupakan salah satu modul

penting yang berfungsi untuk mengelola proses penjualan, mulai dari pengelolaan pesanan, penentuan harga, pengiriman barang, hingga penagihan, serta terintegrasi dengan modul *Material Management* (MM) dan *Production Planning* (PP) [8]. Selain manfaatnya, implementasi ERP juga menghadapi berbagai tantangan berupa resistensi terhadap perubahan, biaya implementasi yang tinggi, serta kompleksitas dalam integrasi data [15].

2.2.2 *Sales Order* dan *Order-to-Cash*

Proses *Sales Order* merupakan bagian inti dari siklus operasional perusahaan yang berkaitan dengan pengelolaan transaksi penjualan kepada pelanggan. Dalam konteks ERP, proses ini dikenal sebagai *Order-to-Cash* (O2C), yaitu rangkaian aktivitas yang dimulai dari penerimaan permintaan pelanggan hingga penerimaan pembayaran.

Secara rinci, alur O2C dalam sistem ERP meliputi: (1) *pre-sales*, yaitu pengelolaan data pelanggan dan pembuatan *quotation*; (2) sales order processing, yaitu konfirmasi dan pemrosesan pesanan; (3) *delivery management*, yaitu pengelolaan pengiriman barang; (4) *billing*, yaitu pembuatan *invoice*; dan (5) *payment collection*, yaitu penerimaan dan verifikasi pembayaran [15]. Setiap tahapan didukung oleh modul yang berbeda dan terintegrasi satu sama lain melalui referensi data yang konsisten.

Penelitian menunjukkan bahwa digitalisasi proses O2C melalui sistem ERP mampu mengurangi kesalahan data secara signifikan, mempercepat proses pembuatan *sales order*, dan meningkatkan konsistensi informasi antar bagian [3]. ERP *Sales Order* pada perusahaan distribusi menggunakan metode *prototyping* menghasilkan sistem yang lebih mudah diterima pengguna karena dikembangkan secara iteratif berdasarkan kebutuhan bisnis yang teridentifikasi.

2.2.3 *Business Process Management*

Business Process Management (BPM) merupakan pendekatan sistematis yang digunakan untuk menganalisis, merancang, mengimplementasikan, serta mengoptimalkan proses bisnis dalam suatu organisasi agar lebih efisien dan terintegrasi [4]. Dalam konteks ERP, BPM berperan penting dalam mengelola

alur proses bisnis secara *end-to-end*, khususnya pada modul *Sales Order* yang mencakup tahapan *inquiry, quotation, sales order, delivery*, hingga *billing*.

Penerapan BPM dalam sistem ERP terbukti mampu meningkatkan integrasi antar proses bisnis serta mengurangi ketergantungan pada aktivitas manual [4]. Selain itu, BPM sering dikombinasikan dengan pendekatan *Business Process Reengineering* (BPR) untuk melakukan perbaikan proses secara radikal. Penerapan BPR dalam sistem ERP seperti pada SAP S/4HANA menunjukkan bahwa pemetaan proses bisnis mampu mengidentifikasi ketidakefisienan dan menghasilkan peningkatan produktivitas yang signifikan [16]. Pemilihan metodologi pengembangan sistem informasi yang tepat, termasuk pendekatan iteratif seperti *prototyping*, sangat menentukan keberhasilan implementasi sistem.

Dalam praktiknya, BPM berperan penting dalam pengelolaan proses *Order-to-Cash* (O2C). Optimalisasi proses ini dengan pendekatan *Lean Six Sigma* mampu mengidentifikasi *bottleneck* pada aktivitas *pricing* dan *invoicing* yang sering menyebabkan kesalahan dan keterlambatan. Modernisasi sistem melalui integrasi teknologi juga mampu meningkatkan tingkat otomatisasi proses [2], [16].

2.2.4 *Unified Modelling Language*

Unified Modeling Language (UML) adalah standar pemodelan visual yang digunakan untuk menggambarkan desain dan arsitektur sistem perangkat lunak. Dalam penelitian ini, beberapa diagram UML digunakan untuk mendokumentasikan rancangan sistem, antara lain: *Use Case Diagram* untuk menggambarkan interaksi pengguna dengan sistem; *Activity Diagram* untuk menggambarkan alur proses bisnis: dan *Entity Relationship Diagram* (ERD) untuk menggambarkan struktur basis data [17], [18].

Pemilihan ketiga diagram tersebut didasarkan pada kebutuhan untuk mendokumentasikan tiga aspek utama sistem, yaitu interaksi aktor dengan sistem (perspektif fungsional), alur proses bisnis lintas peran pengguna (perspektif prosedural), dan struktur data yang mendasari seluruh transaksi (perspektif data). Ketiga perspektif ini saling melengkapi, sehingga rancangan

sistem dapat dipahami secara komprehensif baik oleh pengembang maupun pihak non-teknis seperti perwakilan PT Artha Guna Sejati.

Dalam penelitian ini, notasi UML yang digunakan mengacu pada standar UML 2.0, dengan penyesuaian seperlunya agar mudah dipahami oleh pengguna yang tidak memiliki latar belakang teknis. Use Case Diagram dirancang untuk menggambarkan interaksi tiga peran pengguna (Admin, Manager, Operator) dengan fitur-fitur utama sistem, sedangkan Activity Diagram dirancang secara terpisah untuk setiap modul transaksi guna memudahkan penelusuran alur proses pada tahap implementasi dan pengujian.

2.2.5 *Black-Box Testing*

Black-box testing adalah metode pengujian perangkat lunak yang berfokus pada perilaku eksternal sistem tanpa memperhatikan struktur internal kode program [19]. Pengujian dilakukan berdasarkan skenario uji yang ditetapkan untuk memverifikasi apakah setiap fitur sistem berjalan sesuai dengan spesifikasi yang telah ditentukan. Dalam penelitian ini, *black-box testing* digunakan untuk menguji seluruh fitur utama sistem secara fungsional, meliputi autentikasi, CRUD *master data*, pembuatan *quotation*, pemrosesan *sales order, approval workflow*, pengelolaan *delivery*, *invoice*, dan *payment*.

Pengujian *black-box* berfokus pada validasi terhadap input dan output sistem, tanpa perlu memahami logika pemrograman di baliknya. Teknik yang umum digunakan dalam *black-box testing* antara lain *equivalence partitioning, boundary value analysis*, dan *error guessing*, yang bertujuan untuk memastikan bahwa sistem memberikan hasil yang sesuai dengan spesifikasi kebutuhan pada berbagai kondisi input, termasuk kondisi normal maupun kondisi tidak biasa (*edge case*) [20].

Pada penelitian ini, *black-box testing* diterapkan pada seluruh modul fungsional sistem Sales ERP, mulai dari autentikasi, *master data, quotation, sales order, delivery, invoice*, hingga *payment*. Setiap skenario uji disusun berdasarkan kebutuhan fungsional (*user requirements*) yang telah diidentifikasi dengan hasil uji dikategorikan sebagai Valid atau Tidak Valid berdasarkan kesesuaian antara hasil aktual dan hasil yang diterapkan.

2.2.6 *User Acceptance Testing*

User Acceptance Testing (UAT) adalah tahapan pengujian yang dilakukan oleh pengguna akhir untuk memastikan bahwa sistem yang dikembangkan telah memenuhi kebutuhan bisnis dan dapat diterima sebagai solusi yang layak digunakan [21]. UAT menggunakan skenario penggunaan yang mencerminkan aktivitas nyata pengguna dalam pekerjaan sehari-hari. Hasil UAT digunakan untuk mengidentifikasi kesenjangan antara ekspektasi pengguna dengan implementasi sistem, serta menjadi dasar evaluasi kelayakan sistem sebelum diterapkan secara penuh [10].

UAT memiliki peran penting dalam siklus pengembangan sistem karena menjadi tahap akhir validasi sebelum sistem dinyatakan siap digunakan oleh pengguna sesungguhnya. Berbeda dengan *black-box testing* yang berfokus pada kebenaran fungsi secara teknis, UAT menitikberatkan pada persepsi dan pengalaman pengguna terhadap sistem, termasuk kemudahan pengguna, kejelasan alur kerja, serta kesesuaian sistem terhadap kebutuhan bisnis sehari-hari [21].

Dalam penelitian ini, UAT dilakukan dengan menggunakan kuesioner skala *Likert* (1–5) yang mencakup aspek kemudahan penggunaan, kesesuaian fitur, kejelasan alur, dan kelengkapan informasi [6]. Skor rata-rata UAT digunakan sebagai indikator penerimaan sistem.

2.2.7 SDLC *Model Prototyping*

Software Development Life Cycle (SDLC) adalah kerangka kerja yang mendefinisikan tahapan-tahapan dalam proses pengembangan perangkat lunak secara sistematis [22]. Model *Prototyping* merupakan salah satu pendekatan dalam SDLC yang berfokus pada pembuatan prototipe awal sistem secara cepat untuk kemudian dievaluasi oleh pengguna, sehingga kebutuhan sistem dapat diklarifikasi dan disempurnakan secara iteratif [21]. Pendekatan ini sangat sesuai digunakan ketika kebutuhan pengguna belum terdefinisi secara lengkap pada awal proyek.

Tahapan dalam model *Prototyping* meliputi: (1) pengumpulan kebutuhan awal dari pengguna; (2) perancangan *prototype* cepat berdasarkan kebutuhan yang teridentifikasi; (3) evaluasi *prototype* oleh pengguna; (4) perbaikan *prototype* berdasarkan umpan balik; dan (5) pengembangan sistem final setelah *prototype*

disetujui [23]. Proses evaluasi dan perbaikan dilakukan secara berulang hingga *prototype* memenuhi kebutuhan pengguna. Keunggulan utama model ini adalah tingginya keterlibatan pengguna dalam setiap iterasi pengembangan, sehingga risiko ketidaksesuaian antara sistem yang dikembangkan dengan kebutuhan bisnis dapat diminimalkan.

Dibandingkan dengan model *Waterfall* yang bersifat linier dan sekuensial, model *Prototyping* lebih fleksibel dalam mengakomodasi perubahan kebutuhan yang muncul selama proses pengembangan. Sementara itu, dibandingkan dengan pendekatan *Agile/Scrum* yang menekankan pengiriman perangkat lunak fungsional dalam *sprint* pendek dengan tim pengembang yang terorganisir, model *Prototyping* lebih sesuai untuk penelitian akademis berskala kecil dengan pengguna tunggal yang belum memiliki gambaran sistem yang jelas sejak awal [23]. Dalam konteks penelitian ini, model *Prototyping* dipilih karena proses bisnis *Sales Order* di PT Artha Guna Sejati belum sepenuhnya terdefinisi secara formal, sehingga pendekatan iteratif berbasis prototipe memungkinkan sistem dikembangkan dan disempurnakan secara bertahap sesuai masukan pengguna.

2.2.8 Arsitektur Aplikasi Web

Arsitektur aplikasi web merujuk pada struktur komponen dan cara komponen-komponen tersebut berinteraksi untuk membentuk sebuah sistem berbasis web yang fungsional [18]. Salah satu pola arsitektur yang umum digunakan adalah *three-tier architecture* (arsitektur tiga lapis), yang memisahkan sistem menjadi tiga komponen utama: (1) *presentation layer* atau lapisan antarmuka pengguna; (2) *business logic layer* atau lapisan logika bisnis; dan (3) *data layer* atau lapisan penyimpanan data [18].

Pemisahan antar lapisan ini memberikan beberapa keuntungan, di antaranya kemudahan dalam pemeliharaan sistem, fleksibilitas dalam penggantian komponen tanpa memengaruhi komponen lain, serta peningkatan keamanan karena setiap lapisan hanya dapat berkomunikasi dengan lapisan yang berdekatan. Dalam pengembangan sistem SalesERP ini, arsitektur tiga lapis diimplementasikan dengan Next.js sebagai *presentation layer*, Node.js sebagai *business logic layer,* dan *Firebase Firestore* sebagai *data layer*, dengan komunikasi antar lapisan menggunakan REST API.

Dalam penelitian ini, arsitektur *three-tier* diterapkan dengan Next.js sebagai *presentation layer* yang menangani antarmuka pengguna, Node.js/Express.js sebagai *application layer* yang menangani logika bisnis dan API *endpoint*, serta *Firebase Firestore* sebagai *data layer* yang menyimpan seluruh data transaksi. Pemisahan ini memungkinkan pengembangan *frontend* dan *backend* dilakukan secara independent, serta memudahkan proses pengujian pada masing-masing lapisan serta terpisah sebelum diintegrasikan secara keseluruhan.

2.2.9 *Approval Workflow*

Approval workflow adalah alur proses yang mengatur bagaimana sebuah dokumen atau transaksi bisnis harus melalui serangkaian persetujuan dari pihak-pihak yang berwenang sebelum dapat diproses lebih lanjut [17]. Dalam konteks sistem ERP, *approval workflow* memastikan bahwa setiap transaksi telah melewati pemeriksaan dan validasi yang diperlukan sesuai prosedur organisasi, sehingga meningkatkan kontrol internal dan mengurangi risiko kesalahan atau penyimpangan.

Komponen utama dalam *approval workflow* meliputi: (1) *initiator*, yaitu pihak yang mengajukan dokumen atau transaksi untuk disetujui; (2) *approver*, yaitu pihak yang berwenang memberikan persetujuan atau penolakan; (3) *status tracking*, yaitu mekanisme pencatatan status setiap tahapan persetujuan; dan (4) *audit log*, yaitu catatan historis seluruh aktivitas persetujuan yang telah dilakukan [17]. Dalam penelitian ini, *approval workflow* diimplementasikan pada proses *Sales Order*, di mana Admin berperan sebagai *initiator* dan Manager berperan sebagai *approver*, dengan status transaksi yang berpindah secara otomatis dari *Draft* → *Submitted* → *Approved/Rejected*.

Penerapan *approval workflow* pada sistem ERP memberikan manfaat berupa peningkatan akuntabilitas dan transparansi proses bisnis, karena setiap keputusan persetujuan tercatat secara digital lengkap dengan identitas penyetuju dan waktu persetujuannya (*approval log*). Pada penelitian ini, *approval workflow* diterapkan khususnya pada proses perubahan status *sales order* dari *draft* menjadi *submitted*, kemudian dari hari *submitted* menjadi *approved* atau *rejected* oleh pengguna dengan *role* Manager, sehingga proses persetujuan yang sebelumnya dilakukan

secara manual dapat digantikan dengan mekanisme yang terstruktur dan dapat ditelusuri.

2.2.10 ERP *Modul Sales and Distribution*

Modul *Sales and Distribution* (S&D) dalam sistem ERP merupakan komponen yang bertanggung jawab atas pengelolaan seluruh proses penjualan dan distribusi produk, mulai dari manajemen hubungan pelanggan, pemrosesan pesanan, pengelolaan pengiriman, hingga penagihan dan penerimaan pembayaran [8]. Modul ini terintegrasi erat dengan modul ERP lainnya seperti *Material Management* (MM) untuk pengelolaan stok, *Finance and Accounting* (FI/CO) untuk pencatatan transaksi keuangan, serta *Production Planning* (PP) untuk pemenuhan pesanan produksi.

Dalam konteks *prototype* yang dikembangkan pada penelitian ini, cakupan modul dibatasi pada proses *Sales Order* secara *end-to-end*, meliputi pengelolaan *data master (customer, produk, price list), quotation, sales order* dengan *approval workflow, delivery, invoice,* dan *payment.* Modul-modul ERP lainnya seperti *inventory management, procurement*, dan *finance* belum diintegrasikan dalam *prototype* ini, sehingga sistem yang dikembangkan berfungsi sebagai *prototype* ERP modul *Sales Order*, bukan sebagai sistem ERP yang lengkap dan terintegrasi penuh [14].

Pembatasan cakupan tersebut sejalan dengan prinsip pengembangan *prototype* fungsional, yaitu memfokuskan sumber daya penelitian pada area proses bisnis yang memberikan nilai dan dampak paling signifikan bagi perusahaan dalam jangka waktu penelitian yang tersedia, sebelum dikembangan lebih lanjut menjadi modul ERP yang lebih lengkap seperti *inventory, procurement,* dan *finance* pada penelitian selanjutnya.

2.2.11 Research and Development (R&D)

Research and Development (R&D) merupakan metode penelitian yang digunakan untuk menghasilkan produk tertentu serta menguji efektivitas produk tersebut melalui proses pengembangan sistematis dan bertahap [19]. Berbeda dengan penelitian deskriptif yang hanya bertujuan mendeskripsikan suatu fenomena, R&D menekankan pada aktivitas menghasilnnya (*developing*) sebuah

artefak atau produk, baik berupa perangkat lunak, model, maupun prosedur kerja, yang kemudian divalidasi melalui pengujian terhadap pengguna atau ahli.

Secara umum, tahapan R&D meliputi: (1) potensi dan masalah, yaitu identifikasi kesenjangan antara kondisi ideal dan kondisi nyata yang menjadi dasar kebutuhan pengembangan produk; (2) pengumpulan data, yaitu pengumpulan informasi pendukung melalui studi pustaka, observasi, dan wawancara; (3) desain produk, yaitu perancangan spesifikasi dan arsitektur produk yang akan dikembangkan; (4) pengembangan produk, yaitu implementasi rancangan menjadi produk fungsional; (5) uji coba produk, yaitu pengujian produk baik secara teknis maupun oleh pengguna; serta (6) revisi produk, yaitu perbaikan produk berdasarkan hasil evaluasi pada tahap uji coba [19].

Dalam penelitian ini, pendekatan R&D dipilih karena tujuan penelitian bukan sekadar mendeskripsikan permasalahan proses bisnis *Sales Order* pada PT Artha Guna Sejati, melainkan juga menghasilkan sebuah produk nyata berupa prototype fungsional sistem ERP yang dapat diuji langsung oleh pengguna. Tahapan R&D pada penelitian ini diselaraskan dengan model SDLC *Prototyping*, di mana tahap potensi dan masalah serta pengumpulan data pada R&D sejalan dengan tahap identifikasi kebutuhan pada SDLC, sedangkan tahap desain dan pengembangan produk pada R&D sejalan dengan tahap perancangan dan pembangunan *prototype*, dan tahap uji coba serta revisi produk pada R&D sejalan dengan tahap evaluasi dan perbaikan *prototype* pada SDLC.

2.2.12 Digital Tranformation

Digital transformation atau transformasi digital didefinisikan sebagai proses penggunaan teknologi digital untuk menciptakan perubahan signifikan pada model bisnis, proses operasional, serta pengalaman pelanggan suatu organisasi [24]. Transformasi digital tidak hanya sebatas menggantikan proses manual dengan aplikasi digital, tetapi juga mencakup perubahan cara kerja, budaya organisasi, dan pengambilan keputusan berbasis data, sehingga organisasi mampu beradaptasi secara berkelanjutan terhadap perubahan lingkungan bisnis.

Delapan elemen kunci dalam proses transformasi digital, di antaranya penggunaan teknologi digital sebagai pemicu, perubahan pada rantai nilai (*value creation path*), perubahan struktural organisasi, hambatan (*barriers*) yang perlu

diatasi, serta dampak positif maupun negatif yang ditimbulkan terhadap organisasi [24]. Bagi perusahaan skala menengah seperti PT Artha Guna Sejati, transformasi digital sering kali dimulai dari digitalisasi proses inti yang paling berdampak terhadap operasional dan pengalaman pelanggan, sebelum berkembang menuju transformasi yang lebih menyeluruh pada seluruh lini bisnis.

Dalam konteks penelitian ini, pengembangan *prototype* sistem ERP modul *Sales Order* merupakan langkah awal transformasi digital pada PT Artha Guna Sejati, yaitu digitalisasi proses inti *Order-to-Cash* yang sebelumnya bergantung pada dokumen fisik, *spreadsheet*, dan aplikasi perpesanan. Digitalisasi pada tahap ini diharapkan menjadi fondasi bagi perusahaan untuk melanjutkan transformasi digital pada proses bisnis lain, seperti *inventory, procurement,* dan *finance*, menuju sistem ERP yang terintegrasi secara penuh di masa mendatang.

2.3 Framework/Algoritma yang digunakan

2.3.1 *Next.js*

Next.js adalah *framework React* berbasis *JavaScript/TypeScript* untuk pengembangan aplikasi web yang mendukung rendering sisi server (*Server-Side Rendering*/SSR) dan rendering sisi klien (*Client-Side Rendering*/CSR). *Next.js* dipilih sebagai *framework frontend* dalam penelitian ini karena menawarkan beberapa keunggulan, antara lain: (1) routing berbasis *file system* yang mempermudah navigasi antar halaman; (2) dukungan *TypeScript* yang meningkatkan keandalan kode; (3) ekosistem *React* yang kaya dengan komponen *reusable*; dan (4) performa yang optimal untuk aplikasi web berbasis data dinamis [25].

Dalam sistem yang dikembangkan, *Next.js* digunakan pada sisi *frontend* untuk membangun antarmuka pengguna yang responsif dan interaktif, termasuk halaman *login, dashboard, master data, quotation, sales order, delivery, invoice, payment*, dan *approval*. *Frontend* berkomunikasi dengan *backend* melalui REST API.

Selain itu, Next.js menyediakan fitur file-based routing yang menyederhanakan proses pembuatan halaman baru, serta mendukung optimasi otomatis seperti code splitting dan image optimization, sehingga performa

aplikasi web dapat tetap terjaga meskipun jumlah modul yang dikembangkan terus bertambah seiring kebutuhan sistem SalesERP.

2.3.2 *Node.js* dan *Express.js*

Node.js adalah *platform runtime JavaScript* sisi server yang memungkinkan pengembangan aplikasi *backend* dengan *JavaScript/TypeScript*. Dalam sistem yang dikembangkan, *Node.js* digunakan sebagai runtime untuk *backend* HTTP server yang mengelola *routing* API, autentikasi, dan logika bisnis [26]. Implementasi *backend* menggunakan *TypeScript* untuk meningkatkan keandalan kode melalui sistem tipe yang ketat.

Backend sistem dibangun menggunakan *Node.js built-in* HTTP server yang diperkuat dengan arsitektur modular berbasis *Express.js pattern*, mencakup modul *auth, dashboard, quotations, sales-orders, deliveries, invoices*, dan *payments*. Setiap modul terdiri dari *controller* (penanganan *routing* dan HTTP), *service* (logika bisnis), dan *repository* (akses data *Firestore*).

Pendekatan modular ini mempermudah proses pemeliharaan kode program, karena setiap modul memiliki tanggung jawab yang jelas dan terpisah (separation of concerns), serta memudahkan proses pengujian unit pada masing-masing modul tanpa memengaruhi modul lainnya.

2.3.3 *Firebase Firestore*

Firebase Firestore adalah layanan basis data NoSQL berbasis *cloud* yang dikembangkan oleh Google [12]. *Firestore* menggunakan model dokumen-koleksi (*document-collection*) yang fleksibel, mendukung *query* yang kuat, dan menyediakan sinkronisasi data *real-time*. Dalam sistem yang dikembangkan, *Firestore* digunakan sebagai basis data utama yang menyimpan seluruh koleksi data, meliputi *users, customers, products, price_lists, quotations, sales_orders, deliveries, invoices*, dan *payments*.

Dibandingkan dengan *Relational Database Management System* (RDBMS) seperti MySQL atau PostgreSQL, *Firestore* menawarkan fleksibilitas skema yang lebih tinggi dan skalabilitas *horizontal* yang mudah. Namun, *Firestore* memiliki keterbatasan dalam hal relasi data yang kompleks dan transaksi *multi-collection* yang belum sefleksibel sistem RDBMS. Keterbatasan ini perlu

dipertimbangkan dalam pengembangan modul ERP yang lebih kompleks di masa mendatang.

Kemampuan sinkronisasi data secara real-time pada Firestore juga dimanfaatkan pada sistem SalesERP untuk memastikan bahwa perubahan status transaksi, seperti status sales order dari draft menjadi approved, dapat langsung tercermin pada dashboard tanpa memerlukan proses refresh manual oleh pengguna.

2.4 *Tools/software* yang digunakan

2.4.1 *Visual Studio Code*

Visual Studio Code merupakan aplikasi *code editor* yang dikembangkan oleh perusahaan Microsoft. *Visual Studio Code* banyak digunakan oleh pengembang perangkat lunak karena memiliki fitur yang lengkap, ringan, serta mendukung berbagai bahasa pemrograman.

Dalam pengembangan sistem ERP pada penelitian ini, *Visual Studio Code* digunakan sebagai lingkungan pengembangan utama (*Integrated Development Environment* / IDE) untuk menulis dan mengelola kode program. Beberapa bahasa pemrograman yang digunakan dalam sistem ini antara lain HTML, CSS, *JavaScript*, dan PHP**.**

Visual Studio Code juga menyediakan berbagai *extension* yang membantu mempercepat proses pengembangan sistem, seperti extension untuk manajemen kode, debugging, serta integrasi dengan sistem kontrol versi. Dengan menggunakan *Visual Studio Code*, proses pengembangan sistem dapat dilakukan secara lebih efisien dan terstruktur.

2.4.2 *Firebase Firestore*

Firebase Firestore adalah sistem manajemen basis data NoSQL berbasis *cloud* yang dikembangkan oleh Google sebagai bagian dari *platform Firebase* [12]. *Firestore* menggunakan model penyimpanan berbasis dokumen (*document-based*) yang diorganisasikan dalam koleksi (*collection*), sehingga memungkinkan penyimpanan data yang fleksibel dan terstruktur tanpa skema yang kaku (*schemaless*).

Dalam pengembangan sistem ERP modul *Sales Order* ini, *Firebase Firestore* digunakan sebagai basis data utama yang berfungsi untuk menyimpan seluruh data transaksi penjualan perusahaan. Data disimpan dalam sembilan koleksi utama, yaitu *users, customers, products, price_lists, quotations, sales_orders, deliveries, invoices*, dan *payments*. Penggunaan *Firestore* memungkinkan sistem melakukan akses data secara *real-time*, mendukung *query* kompleks, serta menyediakan kemampuan sinkronisasi data yang andal untuk mendukung monitoring transaksi secara langsung.

Sebagai tools pengelolaan basis data, Firebase Console (antarmuka web resmi Firebase) digunakan selama proses pengembangan untuk memantau struktur data secara langsung, menguji query, mengelola aturan keamanan (security rules), serta melakukan debugging data tanpa perlu menulis kode tambahan, sehingga mempercepat proses pengembangan dan pengujian modul-modul sistem SalesERP.

2.4.3 *Firebase Authentication*

Firebase Authentication adalah layanan autentikasi berbasis cloud yang disediakan oleh Google melalui platform *Firebase* [11]. Layanan ini mendukung berbagai metode autentikasi, termasuk autentikasi berbasis email dan *password* yang digunakan dalam sistem ini.

Dalam sistem ERP yang dikembangkan, *Firebase Authentication* digunakan untuk mengelola proses *login* pengguna. Ketika pengguna berhasil *login* menggunakan *email* dan *password*, *Firebase Authentication* mengeluarkan *Firebase* ID *Token* yang bersifat JWT (JSON *Web Token*). Token ini kemudian dikirimkan dalam setiap permintaan API dari *frontend* ke *backend* sebagai *Bearer Token* dalam *header authorization*, sehingga *backend* dapat memverifikasi identitas pengguna sebelum memproses setiap permintaan.

Selain autentikasi berbasis email dan password, Firebase Authentication turut menyediakan fitur manajemen sesi seperti refresh token untuk memperpanjang masa berlaku sesi pengguna tanpa perlu login ulang, serta kemampuan menonaktifkan akun pengguna secara terpusat, yang dimanfaatkan pada penelitian ini untuk mendukung mekanisme status aktif/nonaktif pada data pengguna sistem.

2.4.4 *Firebase Admin* SDK

Firebase Admin SDK adalah *library* resmi yang disediakan oleh Google untuk mengintegrasikan layanan *Firebase* ke dalam aplikasi *server-side* (*backend*). Admin SDK memungkinkan backend mengakses layanan *Firebase* dengan hak akses penuh (*privileged access*) tanpa melalui proses autentikasi sisi klien. Dalam sistem ERP yang dikembangkan, *Firebase* Admin SDK digunakan pada sisi *backend* (*Node.js*) untuk dua fungsi utama: (1) memverifikasi *Firebase* ID *Token* yang dikirimkan oleh *frontend* pada setiap permintaan API, proses ini diimplementasikan dalam *middleware authenticateRequest*() yang memvalidasi token sebelum permintaan diproses lebih lanjut; dan (2) mengakses *Firebase Firestore* secara langsung dari sisi server menggunakan *Firestore Admin Client* untuk operasi baca dan tulis data.

Penggunaan Firebase Admin SDK pada sisi backend juga memungkinkan penerapan Role-Based Access Control (RBAC) secara aman, karena hanya server yang memiliki kredensial admin yang dapat memverifikasi dan mengubah data sensitif seperti status akun pengguna atau data transaksi keuangan, sementara sisi frontend hanya berinteraksi melalui API yang telah diverifikasi.

Dibandingkan dengan pendekatan yang mengizinkan frontend mengakses Firestore secara langsung menggunakan Firebase Client SDK, penggunaan Admin SDK pada backend memberikan kontrol yang lebih ketat terhadap validasi data dan logika bisnis sebelum data disimpan, sehingga mengurangi risiko manipulasi data dari sisi klien yang tidak sah.

2.4.5 *TypeScript*

TypeScript adalah *superset* dari *JavaScript* yang menambahkan sistem tipe statis (*static typing*) pada bahasa pemrograman *JavaScript*. *TypeScript* dikompilasi menjadi *JavaScript* standar yang dapat dijalankan di *browser* maupun lingkungan server seperti *Node.js*. Dalam sistem ERP yang dikembangkan, *TypeScript* digunakan secara konsisten pada seluruh lapisan sistem, baik *frontend* (*Next.js*/TSX) maupun *backend* (*Node.js*). Penggunaan *TypeScript* memberikan beberapa keunggulan, antara lain: (1) deteksi kesalahan pada tahap kompilasi, sehingga mengurangi *bug* pada saat *runtime*; (2) kode

yang lebih mudah dibaca dan dipelihara dengan definisi tipe yang eksplisit; serta (3) dukungan IDE yang lebih baik melalui fitur *auto-complete* dan *type checking*. Versi *TypeScript* yang digunakan dalam proyek ini adalah *TypeScript* 5.7.2.

Penggunaan *TypeScript* juga mendukung konsistensi tipe data antara *frontend* dan *backend*, misalnya struktur data *SalesOrder* atau *Invoice* yang didefinisikan sebagai *interface* dapat dipakai bersama pada kedua sisi *sistem*, sehingga mengurangi risiko ketidaksesuaian format data ketika frontend mengirim atau menerima data dari *backend* melalui REST API.

Pada penelitian ini, disiplin penggunaan *TypeScript* juga membantu proses *debugging* selama pengembangan, karena kesalahan tipe data (misalnya *field* yang seharusnya berupa angka tetapi diisi teks) dapat terdeteksi sejak tahap penulisan kode, sebelum aplikasi dijalankan atau diuji secara fungsional pada tahap *black-box testing*.

2.4.6 *Node Package Manager* (NPM)

npm (*Node Package Manager*) adalah manajer paket resmi untuk ekosistem *Node.js* yang digunakan untuk mengelola dependensi proyek *JavaScript*/*TypeScript*. npm memungkinkan pengembang untuk menginstal, memperbarui, dan mengelola *library* pihak ketiga yang digunakan dalam proyek dengan mudah. Dalam pengembangan sistem ERP ini, npm digunakan untuk mengelola seluruh dependensi proyek pada sisi *backend* maupun *frontend*. Dependensi utama *backend* meliputi *firebase-admin* untuk integrasi *Firebase*, *dotenv* untuk manajemen *environment variables*, dan *zod* untuk validasi skema data. Dependensi utama *frontend* meliputi *framework Next.js* beserta *library React* dan *TypeScript* yang dibutuhkan.

Selain mengelola instalasi dependensi, npm juga digunakan untuk menjalankan *script* pengembangan yang telah dikonfigurasi pada file *package.json*, seperti perintah untuk menjalankan server pengembangan (*development server*), melakukan build produksi, serta menjalankan proses linting untuk memastikan konsistensi gaya penulisan kode di seluruh proyek.

Penggunaan npm sebagai *package manager* tunggal untuk *frontend* dan *backend* juga menyederhanakan proses instalasi ulang lingkungan pengembangan, karena seluruh dependensi proyek tercatat secara eksplisit pada file *package.json* dan *package-lock.json*, sehingga versi library yang digunakan dapat direproduksi secara konsisten pada komputer pengembang lain.

2.4.7 *Postman*

Postman merupakan aplikasi yang digunakan untuk menguji, mendokumentasikan, dan men-debug REST API secara interaktif tanpa perlu membangun antarmuka pengguna terlebih dahulu. Postman memungkinkan pengembang mengirim permintaan HTTP (GET, POST, PUT, DELETE) ke *endpoint backend* beserta *header* dan *body request* yang dapat dikustomisasi.

Pada penelitian ini, *Postman* digunakan secara intensif selama tahap pengembangan *backend* untuk menguji setiap *endpoint* API *SalesERP*, seperti *endpoint* autentikasi, pengelolaan *quotation, sales order, delivery, invoice,* dan *payment*, sebelum *endpoint* tersebut diintegrasikan dengan *frontend*. Pengujian ini membantu memastikan bahwa response API sesuai dengan format yang diharapkan sebelum pengujian black-box testing pada Bab IV dilakukan.

Selain pengujian manual, Postman juga digunakan untuk menyimpan koleksi (*collection*) permintaan API yang telah disusun berdasarkan modul sistem, sehingga pengujian ulang terhadap endpoint tertentu dapat dilakukan secara cepat setiap kali terdapat perubahan pada logika backend.

2.4.8 *Web Browser*

Web browser merupakan perangkat lunak yang digunakan untuk mengakses dan menampilkan halaman website yang telah dikembangkan. *Web browser* berfungsi sebagai media bagi pengguna untuk berinteraksi dengan sistem ERP berbasis web.

Dalam proses pengembangan dan pengujian sistem pada penelitian ini, web browser digunakan untuk menjalankan aplikasi ERP yang telah dibuat. Melalui *web browser*, pengguna dapat mengakses berbagai fitur sistem ERP seperti

pengelolaan data pelanggan, pembuatan *sales order*, pengelolaan *invoice*, serta melihat laporan transaksi penjualan.

Selain sebagai media akses, *web browser* modern seperti Google Chrome juga menyediakan *Developer Tools* yang dimanfaatkan peneliti untuk melakukan *debugging* pada sisi *frontend*, memeriksa permintaan (*request*) dan respons (*response*) API, serta menguji tampilan sistem pada berbagai ukuran layar (*responsive design*) selama proses pengembangan dan pengujian *black-box testing* berlangsung.