BAB III
METODE PELAKSANAAN

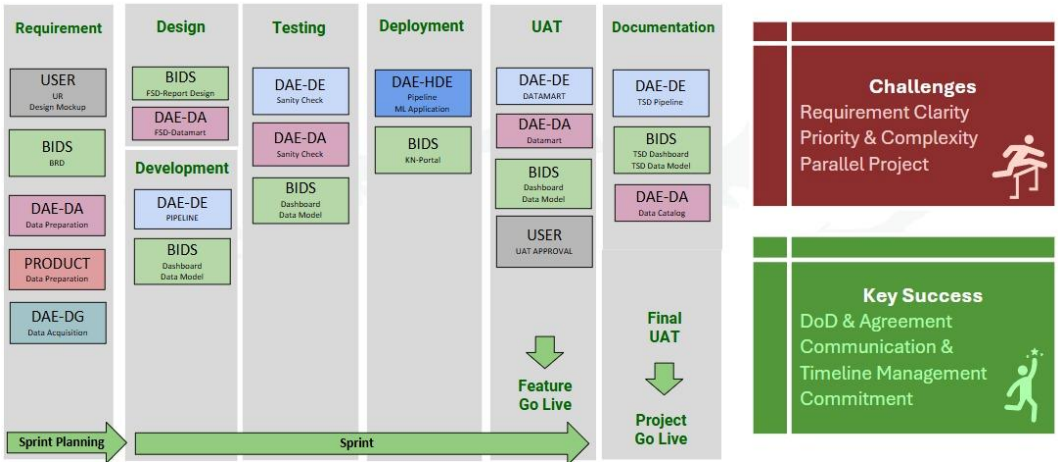
3.1 Agile-Scrum *Methodology*

Dalam pengembangan sistem informasi, pemilihan metode kerja yang tepat menjadi faktor penting dalam menentukan keberhasilan implementasi. Metode yang digunakan harus mampu menyesuaikan diri dengan dinamika kebutuhan pengguna, mendorong kolaborasi lintas tim, serta memungkinkan perbaikan sistem dilakukan secara bertahap. Hal ini sangat relevan pada proyek *pengembangan Dashboard Data Inventory* di Kalbe *Nutritionals* yang memiliki tingkat kompleksitas tinggi dan melibatkan berbagai pihak, mulai dari tim teknis hingga pemangku kepentingan bisnis. Untuk menjawab tantangan tersebut, penelitian ini menerapkan pendekatan Agile-Scrum, sebuah kerangka kerja pengembangan sistem yang bersifat iteratif, adaptif, dan berfokus pada pencapaian hasil secara bertahap. Dengan metode ini, proses pengembangan dapat lebih responsif terhadap perubahan kebutuhan sekaligus memastikan kualitas dan ketepatan fungsi sistem yang dihasilkan.

Metode ini memungkinkan pengembangan dilakukan secara bertahap dalam siklus waktu pendek (*sprint*), yang pada setiap siklusnya menghasilkan produk kerja yang dapat langsung diuji dan dievaluasi oleh pengguna. Penerapan *Agile-Scrum* juga memungkinkan perubahan kebutuhan direspons secara cepat, sekaligus menjaga kesinambungan proses pengembangan melalui dokumentasi dan *review* yang sistematis. Sebelum membahas secara rinci mengenai tahapan siklus kerja dalam *Agile-Scrum,* terlebih dahulu akan dijelaskan bagaimana alur proses pengembangan *dashboard* dirancang dan dijalankan dalam konteks organisasi. Penjelasan ini mencakup peran masing-masing tim dalam pengembangan sistem serta struktur pelaksanaan *sprint* yang digunakan. Dengan pemahaman menyeluruh terhadap alur kerja dan model *sprint*, pembaca dapat melihat bagaimana metodologi yang digunakan tidak hanya bersifat teoritis, tetapi juga telah diimplementasikan secara nyata dalam pengembangan sistem ini.

3.1.1 Alur Metode Proses Pengembangan

Proses ini melibatkan beberapa peran, termasuk tim *Business Intelligence*, Data *Architect*, Data *Engineering*, serta pihak pengguna (*user*). Masing-masing tahap memiliki peran spesifik yang mendukung keberhasilan implementasi *project*. Alur metode scrum *agile* di *project* kalbe *nutritionals* dapat dilihat pada Gambar 3.1 berikut.

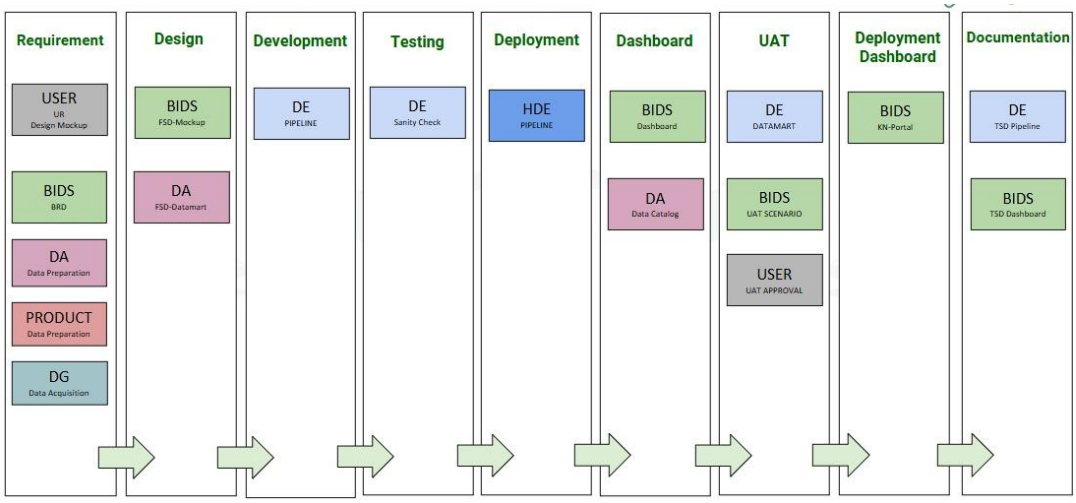


Gambar 3.1 Implementasi metode *scrum* di Kalbe *Nutritionals*.

Sumber: Kalbe *Nutritionals*.

3.1.2 Model *Sprint* dalam Proyek *Dashboard*

Dalam implementasinya, pengembangan *dashboard* dilakukan secara iteratif dan incremental menggunakan kerangka kerja *Agile-Scrum*. Metode ini memudahkan tim beradaptasi terhadap perubahan kebutuhan sambil menjaga kualitas hasil. Setiap siklus sprint mencakup tahap *Sprint Planning* untuk menetapkan *backlog* dan prioritas, Eksekusi *Sprint* untuk membangun fitur, UAT (*User Acceptance Test*) untuk memastikan kesesuaian dengan kebutuhan pengguna, serta *Project Go Live* sebagai peluncuran resmi *dashboard*. *Product Owner* menetapkan prioritas, *Scrum Master* memfasilitasi proses, dan *Development Team* mengimplementasikan serta menguji fitur. Pendekatan ini memastikan pengembangan berjalan terstruktur, kolaboratif, dan berfokus pada pencapaian tujuan. Berikut adalah visualisasi proses *sprint* di *project* data *inventory* beserta pembagian peran pada setiap tahapannya di gambar 3.2 :



Gambar 3.2 Implementasi *Scrum* di Kalbe *Nutritionals* dalam *project dashboard* data *inventory*

Sumber : Kalbe *Nutritionals*

Model ini menunjukkan bahwa keberhasilan proyek sangat dipengaruhi oleh kolaborasi antar peran, kejelasan *requirement*, dan pengelolaan waktu yang efektif. Tantangan utama dalam pelaksanaannya adalah banyaknya proyek paralel dan kebutuhan untuk sinkronisasi lintas tim secara konsisten. Dengan siklus iteratif berupa *sprint*, *Scrum* memungkinkan proses *feedback* berjalan cepat dan terus-menerus dari *stakeholder*, sehingga risiko kesalahan dalam pengembangan sistem dapat diminimalkan sedini mungkin. Ini sangat sesuai dengan lingkungan pengembangan berbasis data, yang rentan terhadap perubahan kebutuhan dan kompleksitas tinggi terhadap struktur aset. Metodologi ini memungkinkan tim proyek untuk merespons perubahan kebutuhan secara dinamis, melakukan evaluasi secara berkala bersama pemangku kepentingan, serta menghasilkan solusi yang fungsional secara *incremental* melalui mekanisme *sprint* atau iterasi bertahap. Hal ini sangat krusial dalam proyek berbasis data yang kompleks, di mana kebutuhan bisnis dapat berubah seiring pemahaman yang berkembang terhadap aset data.

3.1.2.1 *Sprint 1 –Requirements Refinement*

Sprint ini mencakup proses identifikasi kebutuhan, eksplorasi awal data, serta

penyelarasan ekspektasi antar tim teknis dan bisnis. Kegiatan utama pada *Sprint* 0 mencakup:

1. Penyesuaian *timeline* proyek (*timeline adjustment*).
2. Eksplorasi website *openmetadata* serta struktur data dan aset dalam data *warehouse* (*data exploration*),
3. Sesi meeting bersama stakeholder untuk sinkronisasi kebutuhan (*requirement refinement sessions*).
4. Penyusunan dokumen *Business Requirement Document* (BRD), dan
5. Perumusan *Functional Specification Document* (FSD) sebagai acuan desain sistem.

Tahap *Sprint* 0 bertujuan sebagai fondasi awal dalam menyusun *backlog* utama yang mengacu pada kebutuhan bisnis yang telah diidentifikasi. Dilakukan proses sinkronisasi ekspektasi antar pemangku kepentingan, khususnya antara *Data Architect* dan tim *Data Management*. Sebagai bagian dari pemenuhan *Key Performance Indicator (KPI)* Divisi Teknologi Informasi tahun 2025, di mana salah satu indikator utamanya adalah keberhasilan implementasi *Data Inventory*, maka dibutuhkan sistem yang mampu memfasilitasi proses pemantauan menyeluruh terhadap kondisi *Data Warehouse* yang dikelola oleh *tim Data Architecture & Engineering.* Untuk itu, diperlukan beberapa komponen fungsional utama sebagai berikut:

1. Pengembangan dashboard ini bertujuan memberikan visibilitas komprehensif terhadap seluruh aset data di bawah pengelolaan tim data. *Dashboard* berfungsi memantau kuantitas dan status data dalam Data Warehouse berdasarkan *domain, confidentiality, availability, layer*, dan parameter penting lainnya, sehingga mendukung pengambilan keputusan strategis secara efektif dan berbasis data.
2. Dibutuhkan implementasi platform data *profiling* untuk mengidentifikasi dan mengukur aset data secara kuantitatif di dalam *Data Warehouse*. Sebagai

contoh dilakukan klasifikasi terhadap *data sources* untuk mengetahui sumber data mana saja yang telah ditarik dan diintegrasikan ke dalam ekosistem *Data Warehouse*. Informasi ini penting untuk mengoptimalkan pemanfaatan aset data yang tersedia.

3. Diperlukan sistem pemantauan progres standarisasi dan katalogisasi aset data, melalui pengembangan fitur *Asset Progress Tracking for Standardization and Cataloging.* Proses ini mencakup evaluasi terhadap kepatuhan aset data terhadap standar metadata serta integrasi katalogisasi metadata melalui platform seperti website *OpenMetadata*. Dengan pendekatan ini, tim data dapat memastikan bahwa aset data tersusun secara sistematis, sesuai dengan standar yang berlaku, dan siap digunakan dalam proses analitik lanjutan.
4. Diperlukan proses *housekeeping Data Warehouse* yang berfungsi untuk melakukan pemantauan dan pengelolaan penggunaan penyimpanan data baik pada infrastruktur *on-premise* maupun *cloud*. Pemantauan ini bertujuan untuk memastikan efisiensi penggunaan sumber daya penyimpanan dan mendukung keberlanjutan operasional infrastruktur data di Kalbe *Nutritionals*.

3.1.2.2 *Sprint* 1 – *Dashboard Inventory Design*

Milestone tahap ini mencakup beberapa aktivitas utama, yaitu *Design Mockup* yang memuat seluruh requirement hasil diskusi pada tahap sebelumnya, dilanjutkan dengan UAT (*User Acceptance Testing*) untuk memastikan kesesuaian fungsional dan visual. Output *Sprint* 1 adalah prototipe dashboard pertama yang tervalidasi, menampilkan informasi awal aset data seperti domain, pemilik (*owner*), dan readiness status. Sumber data berasal dari integrasi Data *Warehouse* Kalbe *Nutritionals*, baik berbasis *cloud* maupun *on-premise*. Prototipe ini menjadi acuan pengembangan berikutnya, memastikan desain dan fitur telah sesuai kebutuhan pengguna sebelum tahap implementasi lanjutan.

Sprint 1 bertujuan untuk merancang dan memvalidasi desain awal *Dashboard Data Inventory* yang akan dikembangkan. Proses ini mencakup pengumpulan kebutuhan

fungsional dari stakeholder, pembuatan *mockup dashboard* berdasarkan kebutuhan tersebut, serta pelaksanaan UAT (*User Acceptance Test*ing) awal terhadap desain visual dan struktur halaman *dashboard*. Dengan adanya prototipe yang tervalidasi, diharapkan tim pengembang memiliki acuan yang jelas dan terarah sebelum memasuki tahap pengembangan teknis.

3.1.2.3 Sprint 2 – *Development*

Pada *Sprint* 2, fokus utama adalah pembuatan data *pipeline* dan *dashboard* yang bersumber dari database website openmetadata yang sudah dieksplorasi dan diidentifikasi. Tahap awal dari *sprint* ini dimulai dengan membuat *flow* terhadap tabel-tabel sumber metadata yang mencakup informasi penting seperti *tag* proyek, domain, status *availability*, tipe aset, layanan data (*service*), serta tingkat *confidentiality*. Metadata tersebut digunakan sebagai dasar dalam merancang struktur *dashboard* yang bersifat relevan, dinamis, dan terukur, sesuai kebutuhan tim data di Kalbe *Nutritionals*. *Sprint* 2 difokuskan pada pengembangan sistem secara teknis, mencakup pembangunan data *pipeline*, integrasi metadata dari *OpenMetadata* ke data *warehouse*, serta penyusunan model data *conceptual* (CDM) dan proses transformasi data.

Tujuan utama dari *sprint* ini adalah menghasilkan *dashboard* yang telah terintegrasi dengan data aktual melalui tiga *layer* data *(Staging, Refine, dan Datamart),* dengan dukungan orkestrasi *Prefect* dan transformasi data menggunakan *Python Pandas*. Selain itu, sprint ini juga bertujuan untuk memastikan bahwa struktur metadata dapat ditelusuri secara informatif dan akurat dalam *dashboard*. Sebagai bagian dari proses desain arsitektur, disusun *Conceptual Data Model* (CDM) yang menggambarkan bagaimana informasi dari *OpenMetadata* akan dialirkan dan diolah menuju data *warehouse*. Model ini mencakup jalur integrasi data mulai dari tahap identifikasi, transformasi, hingga ke visualisasi *dashboard*, dengan memastikan bahwa setiap atribut seperti *data owner, project, confidentiality, shareability, dan availability* dapat ditelusuri dan dikonsumsi oleh pengguna secara informatif. Untuk mendukung proses *ETL (Extract, Transform, Load)*, *pipeline* data dibangun menggunakan

pendekatan berbasis otomatisasi. Data dari *OpenMetadata* dialirkan ke dalam *data warehouse on-premise* Kalbe *Nutritionals* melalui tiga lapisan transformasi, yaitu:

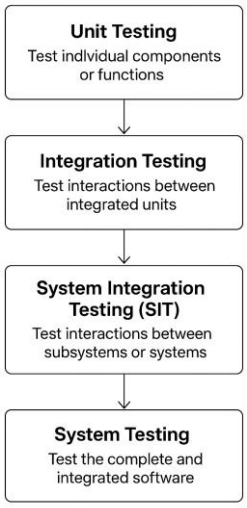
1. STG (*Staging Layer*) – Menampung data mentah dari sumber metadata.
2. *Refine Layer* – Melakukan pembersihan dan transformasi awal terhadap struktur data.
3. *Datamart Layer* – Menyediakan data yang telah dikurasi dan siap digunakan untuk visualisasi *dashboard*.

Proses orkestrasi data *pipeline* akan dilakukan menggunakan *Prefect Orkestrator*, sebuah platform orkestrasi work*flow* berbasis *Python* yang memungkinkan kontrol dan pengawasan terhadap *task* data *pipeline* secara modular dan terjadwal. Untuk proses transformasi dan rekonstruksi struktur metadata, digunakan *library Pandas*, yang memberikan fleksibilitas tinggi dalam pembersihan dan manipulasi data tabular. Selain itu, untuk identifikasi pola metadata, khususnya dalam hal standar penamaan, klasifikasi, dan evaluasi kelengkapan atribut, digunakan teknik pencocokan berbasis *Regular Expression* (*Regex*). Output akhir dari alur ini adalah visualisasi dashboard berbasis *Table* yang mampu menyajikan data hasil integrasi dan transformasi secara visual, sesuai dengan struktur halaman yang telah dirancang pada fase desain sebelumnya *(Executive Summary, Catalog Completeness, Asset Standardization,* dan *Storage.*

3.1.2.4 *Sprint 3 Tahap Pengujian Sistem*

Testing merupakan tahapan penting dalam pengujian sistem terintegrasi yang dilakukan setelah development *pipeline* data serta *dashboard* dan sebelum UAT. Tujuan utama dari SIT adalah untuk memastikan bahwa seluruh komponen yang terlibat dalam proses transformasi, dan visualisasi telah berfungsi secara terpadu dan konsisten di sepanjang alur sistem.

Flow Testing bisa dilihat pada gambar 3.3 berikut :



Gambar 3.3 Tahap Pengujian Sistem

Pengujian ini memvalidasi sistem secara menyeluruh guna memastikan bahwa seluruh komponen mulai dari ekstraksi data, transformasi, penyimpanan di data *warehouse*, hingga visualisasi di *Tableau* telah berfungsi dengan baik dan konsisten. *Sprint* ini juga mencakup s*anity check*, verifikasi alur data secara end-to-end, dan evaluasi kelayakan sistem. Tujuan akhirnya adalah memastikan sistem siap untuk dioperasikan dalam lingkungan produksi dengan tingkat kestabilan dan akurasi yang memadai, perancangan untuk tahap pengujian bisa dilihat di tabel 3.1 :

Tabel 3.1 Perancangan Pengujian

Tahap Testing	Tujuan	Implementasi di Project Data	Level Scope	Output
Unit Testing (UT)	Menguji fungsi/komponen terkecil secara terisolasi.	- Cek fungsi *Python* untuk *parsing* file json apakah hasil *parsing* mengalami *missing value*	*function/table query*	Test Case
Integration Testing (IT)	Menguji interaksi antar unit yang sudah digabung.	- Pastikan fungsi *extract* dari *postgre* + *transform* di *Pandas* bisa jalan bareng.- *Join SQL* antar *table* hasil sesuai *requirement*.	2–3 komponen yang sudah saling terhubung	Test Case
System Integration Testing (SIT)	Menguji alur *end-to-end* antar *subsistem*, biasanya cross-layer.	- *Test* alur ETL penuh: *Source Postgre → Staging → Refine → Datamart.*- Pastikan data dari sumber masuk ke	*(ETL pipeline, storage, DWH*)	Test Case

		DWH dengan struktur yang benar.		
System Testing (ST)	Menguji keseluruhan sistem apakah sesuai kebutuhan bisnis (black box).	- Jalankan dashboard Tableau pakai data dari datamart, cek apakah angka count(1) sama persis.	*Full system (ETL, DWH, BI tools, user requirement)*	Test Case

Tahapan pertama yang dilakukan adalah *Unit Testing (UT)* yang berfokus pada pengujian fungsi atau komponen terkecil dalam sistem secara terisolasi. Tujuan dari pengujian ini adalah untuk memastikan bahwa setiap potongan kode, seperti fungsi *Python* maupun *query SQL*, dapat berjalan sesuai dengan spesifikasi sebelum digabungkan dengan modul lain. Misalnya, dilakukan pengujian terhadap fungsi parsing file *JSON* untuk mendeteksi adanya nilai yang hilang (*missing value*), serta verifikasi struktur hasil parsing apakah sudah sesuai dengan format data yang diharapkan. Output dari tahap ini berupa *test case* yang mendokumentasikan hasil pengujian fungsi tunggal, sehingga dapat dijadikan acuan apabila terjadi error pada tahap berikutnya. Setelah seluruh komponen lolos Unit Testing, tahap berikutnya adalah *Integration Testing* (IT) yang bertujuan menguji interaksi antar unit yang telah digabungkan. Fokus utama dari tahap ini adalah memastikan modul-modul yang telah teruji secara individual dapat saling terhubung dengan benar ketika dijalankan secara bersamaan. Implementasi nyata pada proyek data adalah menguji alur *extract* dari *PostgreSQL* yang kemudian diproses menggunakan Pandas untuk transformasi data. Selain itu, dilakukan pengujian terhadap hasil *join* tabel SQL guna memverifikasi kesesuaian data antar entitas. Hasil dari tahap ini juga terdokumentasi dalam bentuk *test case* yang menggambarkan keberhasilan integrasi antar modul kecil. Tahap selanjutnya adalah *System Integration Testing* (SIT) yang menguji alur *end-to-end* antar subsistem yang berbeda dalam lingkungan data. Pengujian ini memiliki cakupan yang lebih luas dibanding Integration Testing karena melibatkan beberapa lapisan dalam pipeline data. Contoh implementasi SIT adalah pengujian alur *ETL* penuh, mulai dari sumber data *PostgreSQL*, pemuatan ke Staging *Layer*, transformasi di *Refine Layer*, hingga penyimpanan akhir pada *Datamart*. Pada

tahap ini, sistem diuji untuk memastikan bahwa data dari sumber dapat dipindahkan ke data warehouse dengan struktur yang benar serta konsisten. Setiap skenario SIT disusun dalam *test case* agar proses verifikasi antar subsistem terdokumentasi secara sistematis.

Tahapan terakhir dari metodologi pengujian adalah *System Testing* (ST) yang memvalidasi sistem secara keseluruhan dengan pendekatan *black box*. Tujuan dari pengujian ini adalah mengevaluasi apakah sistem telah memenuhi kebutuhan bisnis dan ekspektasi pengguna. Implementasi dalam proyek data dilakukan dengan menjalankan *dashboard Tableau* yang menggunakan data dari *Datamart*. Proses pengujian dilakukan dengan membandingkan hasil agregasi, misalnya perintah *COUNT*(1), antara dashboard dan query manual pada data warehouse. Konsistensi hasil menunjukkan bahwa sistem telah berfungsi sesuai dengan kebutuhan bisnis. Dokumentasi hasil pengujian pada tahap ini berupa *test case* yang berfungsi sebagai bukti kesiapan sistem untuk melangkah ke tahap produksi. Secara keseluruhan, metode perancangan pengujian ini dirancang untuk memastikan kualitas sistem data melalui pendekatan berlapis, mulai dari unit terkecil hingga sistem secara utuh. *Unit Testing* memastikan fungsi dasar berjalan sesuai ekspektasi, *Integration Testing* menjamin modul yang terhubung dapat berkolaborasi dengan benar, sementara *System Integration Testing* memvalidasi alur antar subsistem yang lebih kompleks.

Tahap akhir, yaitu System Testing, berfokus pada verifikasi kebutuhan bisnis dan kepuasan pengguna sebelum sistem diimplementasikan penuh. Dengan adanya alur pengujian berlapis ini, risiko kegagalan dapat diminimalkan sekaligus menjamin bahwa sistem yang dikembangkan memiliki tingkat stabilitas, keakuratan, dan reliabilitas yang tinggi sebelum memasuki fase UAT.

3.1.2.5 *Sprint* 4 *User Acceptance Test* (UAT) & *Go-Live*

User Acceptance Test (UAT) dinyatakan sukses jika tidak ditemukan kendala mayor. Secara umum, semua jika semua *requirement* didalam dokumen test case yang berisi dari semua requirement di *Bussines Requirements Design* sudah terimplementasi yang dibuktikan dengan hasil kepuasan pengguna > 80%, informatif, dan memberikan

nilai tambah yang signifikan terhadap pengelolaan aset data. *Dashboard* dinilai efektif dalam memberikan visibilitas yang mendukung pengambilan keputusan, dan menjadi alat bantu dalam pelaksanaan kebijakan *housekeeping* serta standarisasi metadata. Berikut adalah table 3.1 definition of done dari *User Acceptance Test* yang akan dilakukan bersama tim *Data Architecture & Engineering* :

Tabel 3.2 Final *User Acceptance Test* (UAT)

No	Task	Definition of Done
1	*Demo Dashboard*	Menyusun list *Traceability* serta form UAT untuk menilai kepuasan dari sisi *requirement*.
2	*Test Case Question*	Hasil pengisian form UAT *Treshold* 80% Kepuasan *User.*

Setelah proses *User Acceptance Test* (UAT) selesai, maka sistem dinyatakan siap untuk *Go-Live*. Pada fase ini, seluruh komponen sistem mulai dari *ELT pipeline* yang dibangun dengan *Prefect*, proses transformasi menggunakan *Pandas*, hingga *dashboard* interaktif di *Table* di-*deploy* secara resmi ke production environment Kalbe *Nutritionals*. Ditetapkan juga prosedur operasional pasca-*Go-Live*, antara lain:

1. Monitoring berkala terhadap performa sistem dan proses *refresh* data.
2. Mekanisme pelaporan *error* dan anomali integrasi *pipeline*.
3. Evaluasi rutin terhadap masukan pengguna guna mendukung *continuous improvement* dan scalability sistem di masa mendatang.

Dengan selesainya tahap *User Acceptance Test* (UAT) dan implementasi sistem secara menyeluruh, maka dilanjutkan untuk *sprint go-live* atau *publish dashboard* di *server tableau* kalbe *nutritionals*.

3.1.2.6 *Sprint* Dokumentasi

Dokumentasi yang dimaksud di sprint ini merupakan *Data Engineering Documentation* yang disusun sebagai hasil akhir dari proses pengembangan dan evaluasi sistem, khususnya setelah dilakukannya *Go-Live* Dokumen ini berfungsi sebagai bentuk pencatatan teknis atas proses pengelolaan data *inventory* yang telah dikembangkan di Kalbe *Nutritionals*, serta mencerminkan hasil akhir dari proses validasi yang dilakukan oleh pengguna akhir terhadap *Dashboard Data Inventory*. Penyusunan dokumentasi dilakukan setelah *User Acceptance Test* (UAT) selesai dilaksanakan. Dengan demikian, dokumentasi ini menjadi acuan resmi yang dapat digunakan untuk keperluan *handover*, pelatihan pengguna, maupun pengembangan lebih lanjut. Dokumentasi proyek *Dashboard Data Inventory* ini disusun sebagai bagian dari tahapan akhir pengembangan sistem, tujuannya yaitu untuk memberikan acuan teknis dan operasional yang dapat dimanfaatkan dalam proses pemeliharaan maupun pengembangan lanjutan sistem di masa mendatang. Secara umum, dokumentasi ini memiliki beberapa tujuan sebagai berikut:

1. Mencatat seluruh proses pengembangan sistem secara sistematis, mulai dari tahap perencanaan, desain, implementasi, hingga pengujian, sehingga seluruh tahapan proyek dapat ditelusuri secara transparan dan akuntabel.
2. Menyediakan referensi teknis bagi tim pengelola dan pengembang berikutnya, terutama dalam hal struktur pipeline, alur data, konfigurasi ELT, tools yang digunakan, serta struktur *dashboard* yang telah diimplementasikan.
3. Mendukung proses handover dan pelatihan pengguna, dengan memberikan informasi menyeluruh mengenai sistem yang telah dikembangkan agar proses adaptasi dan pemanfaatan dapat dilakukan secara efektif oleh pihak yang berkepentingan.
4. Merepresentasikan hasil evaluasi dari pengguna akhir melalui UAT, sebagai bentuk validasi bahwa sistem telah memenuhi kebutuhan operasional dan sesuai dengan ekspektasi stakeholder.

5. Menjadi dasar bagi *continuous improvement*, di mana dokumentasi ini dapat dijadikan acuan dalam mengevaluasi performa sistem serta mengidentifikasi potensi pengembangan di masa mendatang, termasuk skalabilitas dan penambahan fitur baru.

3.2 Alat dan Bahan Penelitian

Pada penelitian ini perangkat atau alat digunakan untuk keberlangsungan dalam penelitian ini adalah sebagai berikut:

Tabel 3.3 Tabel Spesifikasi Perangkat yang Digunakan

Komponen	**Spesifikasi**
Laptop	Lenovo Thinkpad X395
Prosesor	Ryzen 5 PRO 3500U (2,1 → 3,7 GHz)
Random Access Memory (RAM)	4 GB DDR4
Perangkat penyimpanan	SSD 256 GB
Sistem operasi	Microsoft Windows 11 Home

Adapun perangkat lunak beserta library yang digunakan untuk menunjang pengembangan model pada penelitian ini dicantumkan pada Tabel 3.2

Tabel 3.4 Tabel Perangkat lunak dan *library* yang digunakan

Perangkat Lunak	**Fungsi / Kegunaan**
PostgreSQL	Pengolahan metadata dari repository internal dan *data warehouse on-premise*
Visual Studio Code	Editor kode
DBeaver	Aplikasi untuk koneksi *database*

Perangkat Lunak	**Fungsi / Kegunaan**
Python	Automasi proses ETL (*Extract, Transform, Load*) dan analisis data
Prefect	*Orkestrator workflow*
Apache Spark	Proses *Big Data*
GitLab CI/CD	*Repository Development*
DBT (Data Build Tool)	Transformasi antar *schema*
Tableau	*Platform* visualisasi data
OpenMetadata	Website untuk mengisi roster aset metadata
Microsoft Excel	Pengolahan dan analisis data tambahan