BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahapan analisis kebutuhan dan perancangan sistem sebagai dasar pengembangan sistem *stock opname RFID*. Pembahasan menyoroti keputusan desain yang menjelaskan bagaimana sistem dibangun serta alasan teknis di balik setiap pilihan arsitektur.

Ruang lingkup perancangan mencakup arsitektur *hardware* dan *software*, mekanisme komunikasi antar komponen, pemodelan *database*, serta desain algoritma inti yang mengatur proses identifikasi dan verifikasi inventori. Seluruh rancangan pada bab ini disusun sebagai acuan desain yang akan direalisasikan dan diuji pada tahap implementasi di Bab IV.

Dengan struktur tersebut, Bab III berfungsi sebagai kerangka desain yang menjelaskan apa yang dibangun dan alasan pemilihan rancangan, tanpa membahas detail implementasi teknis secara langsung.

3.1 Analisis Kebutuhan Sistem

Tahap ini bertujuan untuk mengidentifikasi spesifikasi teknis yang harus dipenuhi agar sistem yang dirancang mampu menjawab permasalahan stock opname secara efektif pada lingkungan operasional toko vinyl. Analisis dilakukan berdasarkan observasi langsung terhadap alur kerja eksisting serta spesifikasi kebutuhan yang diperoleh melalui wawancara dengan pihak pengelola toko sebagai stakeholder sistem.

Wawancara dengan stakeholder dilakukan secara informal dan tanpa perekaman audio maupun visual. Hal ini dilakukan atas permintaan stakeholder yang menginginkan anonimitas serta tidak bersedia didokumentasikan secara langsung. Oleh karena itu, data hasil wawancara dicatat dalam bentuk ringkasan tertulis oleh peneliti dan digunakan sebagai dasar dalam perumusan kebutuhan fungsional dan non-fungsional sistem.

Selain wawancara, proses analisis juga mempertimbangkan observasi langsung terhadap praktik operasional toko, khususnya pada proses pengelolaan inventori dan pelaksanaan stock opname.

Berdasarkan hasil analisis tersebut, sistem dirancang untuk memenuhi kebutuhan-kebutuhan berikut.

- Sistem harus mampu terintegrasi dengan Moka POS untuk memperoleh data produk *vinyl* yang telah tersedia di toko sebagai referensi dalam proses *stock opname*, tanpa memerlukan *input* ulang.
- Sistem harus menyediakan *interface* untuk *assign* kode *EPC* (*Electronic Product Code*) dari *RFID tag* fisik ke entitas produk digital yang tersimpan di dalam *database*. Kebutuhan ini muncul karena setiap unit *vinyl* diperlakukan sebagai entitas fisik unik yang harus dapat dilacak secara individual.
- Sistem harus mampu mengendalikan *hardware RFID reader* untuk membaca aliran data *tag* secara *continuous scanning* selama sesi *opname* berlangsung. Proses pembacaan harus berjalan secara *real-time* agar operator dapat melakukan verifikasi stok tanpa interaksi manual per item.
- Sistem harus mampu menangani redundansi data akibat pembacaan tag yang berulang dalam interval waktu yang sangat singkat. Untuk itu, diperlukan mekanisme penyaringan berbasis algoritma deduplikasi yang efisien guna mencegah duplikasi data dan beban memori yang tidak perlu.
- Sistem harus memiliki logika komparasi inventori untuk membandingkan hasil pembacaan fisik dengan data ekspektasi yang tersimpan di dalam sistem. Hasil komparasi tersebut digunakan untuk mengklasifikasikan status setiap item secara otomatis ke dalam kategori *Found*, *Missing*, atau *Extra*.
- Sistem harus mampu menyajikan *visual feedback* mengenai progres dan hasil verifikasi stok kepada operator. Penyajian informasi ini bertujuan untuk mempermudah pemantauan jalannya proses *opname* serta mendukung pengambilan keputusan secara cepat dan akurat.

3.2 Perancangan Arsitektur Sistem

Perancangan arsitektur sistem bertujuan untuk menetapkan struktur *high-level* yang menjadi dasar interaksi antara komponen *hardware* dan *software*. Arsitektur ini dirancang untuk mendukung kebutuhan operasional *stock opname* berbasis *RFID*, dengan memperhatikan keterbatasan lingkungan toko serta kebutuhan integrasi dengan sistem *Point of Sale* yang telah berjalan.

Secara umum, sistem dibangun menggunakan pendekatan *host-based architecture* yang memusatkan pengolahan data dan logika aplikasi pada satu perangkat *host*. Rincian arsitektur sistem dijabarkan melalui pemisahan arsitektur *hardware*, *software*, serta mekanisme komunikasi antar komponen pada subbab berikutnya.

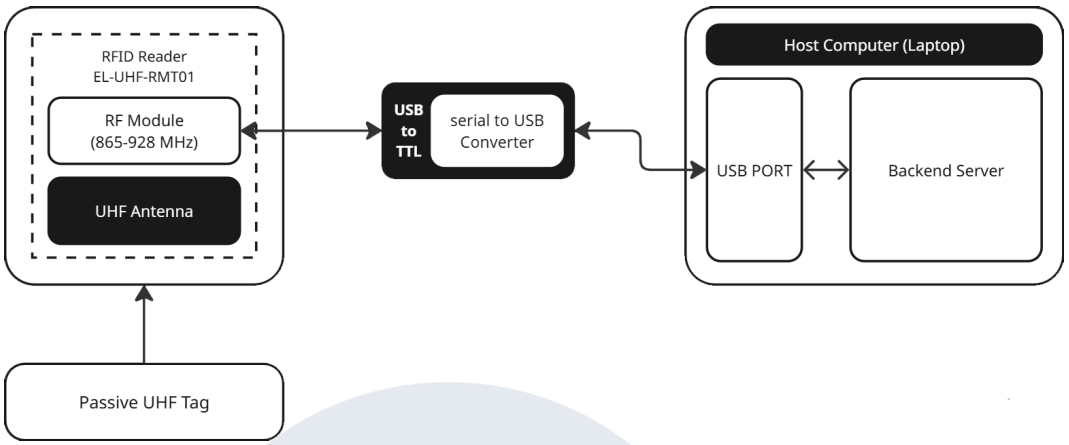
3.2.1 Arsitektur *Hardware*

Arsitektur *hardware* sistem terdiri dari satu unit *host* berupa laptop yang terhubung langsung dengan perangkat *UHF RFID reader*. *Host* berfungsi sebagai tempat dijalankannya aplikasi utama serta penyimpanan *database* lokal yang digunakan oleh sistem.

Koneksi fisik antara *host* dan *RFID reader* dilakukan melalui jalur komunikasi serial menggunakan *interface USB* yang terdeteksi sebagai *virtual serial port*. *RFID reader* terhubung langsung ke *host* tanpa perantara *microcontroller* tambahan.

Pada sisi pembacaan *tag*, antena *RFID reader* memancarkan gelombang radio untuk mengaktifkan *passive RFID tag*. *Tag* yang berada dalam jangkauan merespons dengan memantulkan sinyal yang membawa informasi identitasnya. Sinyal tersebut diterima dan diproses oleh modul *reader* menjadi data digital.

Data hasil pembacaan tag kemudian dikirimkan dari *RFID reader* ke *host* melalui komunikasi serial *UART* untuk diproses oleh *backend*.



Gambar 3.1 Rancangan Arsitektur *Hardware*

3.2.2 Arsitektur *Software*

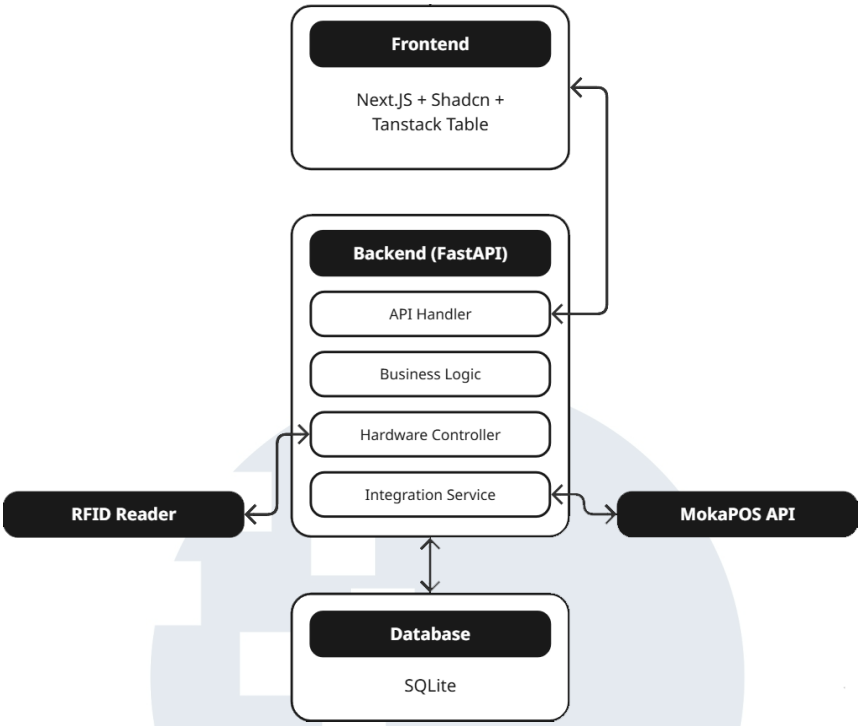
Arsitektur *software* sistem disusun menggunakan pendekatan *three-tier* yang dijalankan secara lokal pada perangkat *host*. Arsitektur ini membagi sistem ke dalam tiga lapisan utama, yaitu *presentation layer*, *logic layer*, dan *data layer*.

Presentation layer berfungsi sebagai *user interface* dan diakses melalui *browser*. Layer ini bertanggung jawab menampilkan informasi inventaris, status *stock opname*, serta interaksi pengguna dengan sistem.

Logic layer berfungsi sebagai pusat pemrosesan aplikasi. Layer ini menangani alur data, pengolahan hasil pembacaan *RFID*, komunikasi dengan *database*, serta penyediaan *endpoint API* yang diakses oleh *presentation layer*.

Data layer berfungsi sebagai media penyimpanan data inventaris dan data pendukung sistem lainnya. *Layer* ini menyimpan informasi produk, data *RFID*, serta hasil proses *stock opname* yang dihasilkan oleh sistem.

Hubungan antar *layer* bersifat terpisah secara logis, di mana *presentation layer* tidak berinteraksi langsung dengan *data layer*, melainkan melalui *logic layer* sebagai perantara.



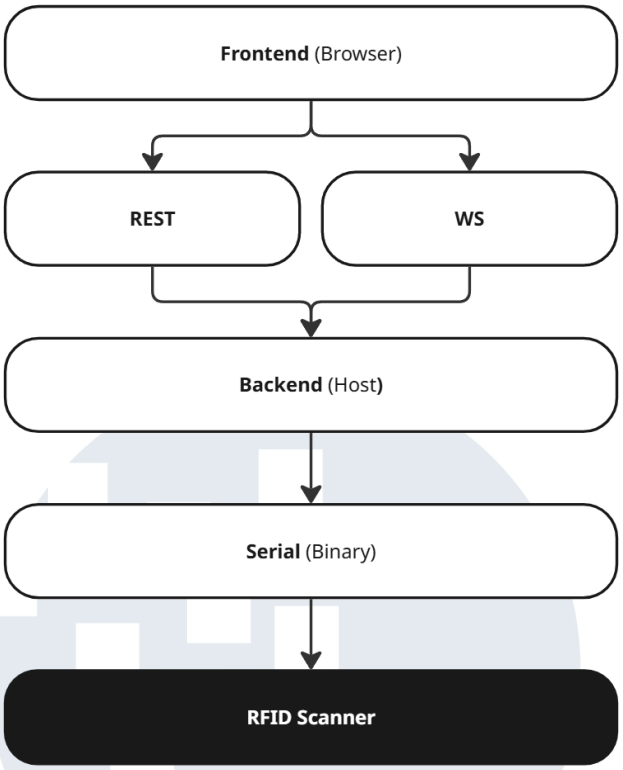
Gambar 3.2 Rancangan arsitektur *software*

3.2.3 Arsitektur Komunikasi

Arsitektur komunikasi sistem terdiri dari beberapa jalur komunikasi yang menghubungkan *frontend, backend*, dan *RFID reader*. Setiap jalur komunikasi ditempatkan sesuai dengan peran dan fungsi masing-masing komponen dalam sistem.

Komunikasi antara *frontend* dan *backend* dilakukan melalui jaringan lokal menggunakan protokol *HTTP* dan *WebSocket*. *Backend* menyediakan *interface* komunikasi yang digunakan oleh *frontend* untuk mengakses data inventaris serta menerima data hasil pembacaan *RFID*.

Komunikasi antara *backend* dan *RFID reader* dilakukan melalui jalur komunikasi serial dengan format *data binary*. *Backend* berperan sebagai perantara yang menerima data hasil pembacaan dari *hardware* dan meneruskannya ke *frontend* dalam format yang dapat diproses oleh aplikasi *web*.



Gambar 3.3 Rancangan Arsitektur Komunikasi

3.2.4 Distribusi *system logic* dan *Boundary Hardware-Software*

Distribusi *system logic* dibagi antara *frontend* dan *backend* dengan *backend* berperan sebagai pusat pemrosesan sekaligus penghubung antara *hardware* dan *software*. *RFID reader* berfungsi sebagai sumber data pembacaan *tag* dan mengirimkan data hasil pembacaan ke perangkat *host* melalui *serial communication*. Perangkat *hardware* tidak menjalankan logika aplikasi, tidak menyimpan *state* sistem.

Seluruh pemrosesan lanjutan dijalankan pada *backend* di perangkat *host*. *Backend* menangani komunikasi dengan *RFID reader*, pengolahan dan deduplikasi data pembacaan tag, pengelolaan *database*, serta integrasi dengan sistem eksternal seperti *Point of Sale*. *Backend* juga menyediakan *API* dan *WebSocket* yang digunakan oleh *frontend* untuk memperoleh data inventaris dan menerima pembaruan hasil pembacaan secara *real-time*.

Pada sisi *frontend*, *system logic* difokuskan pada pengelolaan sesi stock opname, komparasi antara daftar *expected item* dan hasil pembacaan *RFID*, serta penyajian status item kepada pengguna. *Frontend* tidak berinteraksi langsung dengan *hardware* maupun *database*, dan hanya berkomunikasi dengan *backend* melalui *interface* yang disediakan. Detail implementasi komunikasi dan pemrosesan data dibahas lebih lanjut pada Bab IV.

3.3 Perancangan Integrasi Data POS

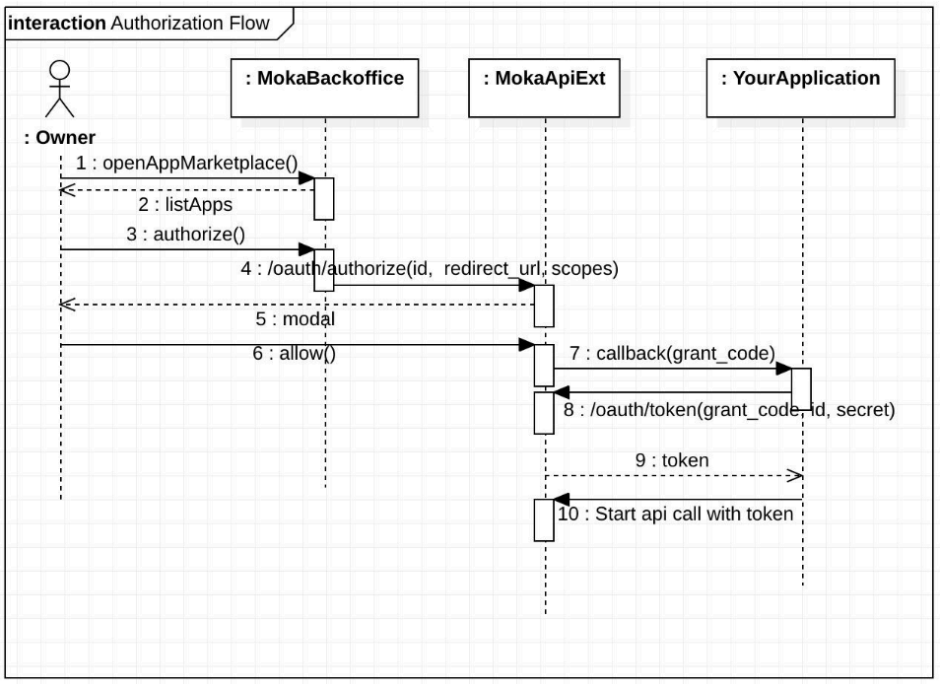
Sistem *stock opname* berbasis *RFID* yang dikembangkan memerlukan integrasi dengan sistem *Point of Sale (POS)* yang telah digunakan oleh toko untuk mengelola data produk dan transaksi penjualan. Integrasi ini dirancang agar sistem *RFID* dapat memanfaatkan data produk yang sudah ada tanpa menambah beban *input* manual, sekaligus menjaga konsistensi antara kondisi inventori fisik dan data stok pada sistem kasir.

Sistem *POS* yang digunakan pada penelitian ini adalah Moka POS, sebuah platform *cloud-based* yang menyediakan *REST API* untuk akses data. Sistem *RFID* memanfaatkan *API* tersebut untuk mengambil data produk dan stok yang sudah ada, tanpa memodifikasi data utama pada sistem *POS*.

Tabel 3.1 Kebutuhan Integrasi dengan Sistem POS

Kebutuhan	**Deskripsi**
Data produk	Mengambil daftar produk dan varian dari sistem *POS*
Data Stok	Mengetahui jumlah stok sebagai referensi inventori
Data Transaksi	Mendeteksi penjualan untuk pembaruan status inventori fisik
Autentikasi aman	Mengakses *API* tanpa menyimpan kredensial pengguna

3.3.1 Mekanisme Autentikasi dan Otorisasi



Gambar 3.4 Flow Otorisasi *API* MokaPOS

Integrasi sistem dengan Moka *Point of Sale (POS)* dirancang mengikuti mekanisme autentikasi resmi yang ditetapkan oleh penyedia layanan. Moka POS menyediakan *OAuth 2.0* sebagai satu-satunya mekanisme autentikasi untuk mengakses *REST API*, sehingga rancangan sistem mengadopsi mekanisme tersebut secara langsung tanpa mempertimbangkan metode autentikasi alternatif.

Sistem dirancang menggunakan *OAuth 2.0 Authorization Code Flow*, di mana proses otorisasi melibatkan interaksi langsung antara pengguna dan layanan Moka. Pada alur ini, pengguna diarahkan ke halaman otorisasi resmi Moka untuk memberikan persetujuan akses terhadap data POS yang diperlukan oleh sistem.

Setelah persetujuan diberikan, sistem menerima *authorization code* sebagai bukti otorisasi. *Authorization code* tersebut digunakan oleh *backend* untuk memperoleh *access token* yang dipakai dalam komunikasi selanjutnya dengan *API* MokaPOS.

3.3.2 Transformasi dan Pemetaan Data

Sinkronisasi data stok antara sistem *POS* dan sistem inventaris *RFID* menghadapi sejumlah tantangan, terutama perbedaan tingkat detail data dan sumber data. Sistem *POS* menyimpan stok dalam bentuk total jumlah barang, sementara sistem *RFID* melacak keberadaan unit fisik secara individual. Perbedaan ini memerlukan mekanisme transformasi dan pemetaan data.

Data produk dan varian dari Moka POS disesuaikan ke dalam struktur *database* lokal sesuai dengan kebutuhan operasional toko. Ringkasan aturan penyesuaian data ditunjukkan pada Tabel 3.2.

Tabel 3.2 Aturan Pemetaan Data POS ke Sistem Lokal

Atribut Moka POS	Entitas Sistem *RFID*	Keterangan
Item Name	*Product.Artist*	Adaptasi konvensi toko
Variant SKU	*Product.AlbumName*	Adaptasi konvensi toko
Variant Price	*Product.Price*	Data referensi
In Stock (Qty)	*Count (ProductCopy)*	Transformasi 1-ke-N *rows*

Selain entitas produk dan varian, sistem lokal menambahkan entitas *ProductCopy* yang merepresentasikan setiap unit fisik secara individual. Entitas ini tidak terdapat pada sistem *POS* dan menjadi komponen utama dalam pelacakan inventori berbasis RFID.

3.3.3 Strategi Sinkronisasi *Stock Bidirectional*

Sinkronisasi dilakukan dengan membandingkan jumlah stok dari POS dengan jumlah *ProductCopy* aktif di sistem lokal, kemudian menyesuaikan data berdasarkan selisih yang ditemukan.

Rekonsiliasi stok dalam sistem ini dirancang sebagai mekanisme untuk menjaga konsistensi antara data stok pada sistem *Point of Sale* dan representasi unit fisik pada sistem *RFID*. Karena MokaPOS menyimpan stok dengan jumlah total dan sistem *RFID* melacak unit secara individual, perlu ditetapkan sistem mana yang menjadi acuan utama untuk setiap jenis data.

Dalam rancangan ini, sistem mendukung alur sinkronisasi dua arah, di mana:

- Data produk dan stok ditarik dari *POS* ke sistem lokal
- Data penjualan pada *POS* digunakan untuk memperbarui status unit fisik di sistem *RFID*

Sistem menerapkan desain *nullable EPC* pada entitas *ProductCopy*. Unit fisik yang belum memiliki *RFID tag* tetap tercatat di dalam sistem dengan status *unassigned*, dan pengguna dapat melakukan proses *assign RFID tag* pada unit tersebut.

Selain itu, pembaruan status penjualan dirancang menggunakan pendekatan *First In, First Out (FIFO),* di mana unit yang lebih dulu masuk ke sistem akan diprioritaskan saat terjadi penjualan. Detail implementasi rekonsiliasi stok dan pembaruan status penjualan dibahas lebih lanjut pada Bab IV.

3.4 Perancangan *Database*

Database digunakan untuk menyimpan data inventaris, konfigurasi sistem, dan riwayat aktivitas *stock opname*. *Database* diakses oleh *backend* untuk memastikan data tersimpan dan dapat digunakan kembali secara konsisten.

Database dirancang menggunakan *relational model* dengan beberapa entitas yang saling terhubung. Skema *database* dirancang untuk merepresentasikan data produk, unit fisik barang, lokasi penyimpanan, serta sesi dan hasil *stock opname*. Relasi antar entitas menghubungkan data inventaris fisik dengan alur sistem.

Selain menyimpan data inventaris, *database* juga mencatat aktivitas sistem seperti perubahan status barang, hasil *stock opname*, dan *log* operasional lainnya. Data tersebut digunakan oleh *backend* untuk proses evaluasi inventaris dan pelacakan perubahan data dari waktu ke waktu.

Seluruh operasi baca dan tulis data dilakukan melalui *backend. Frontend* tidak berinteraksi langsung dengan *database*, melainkan melalui *API* yang disediakan oleh *backend*.

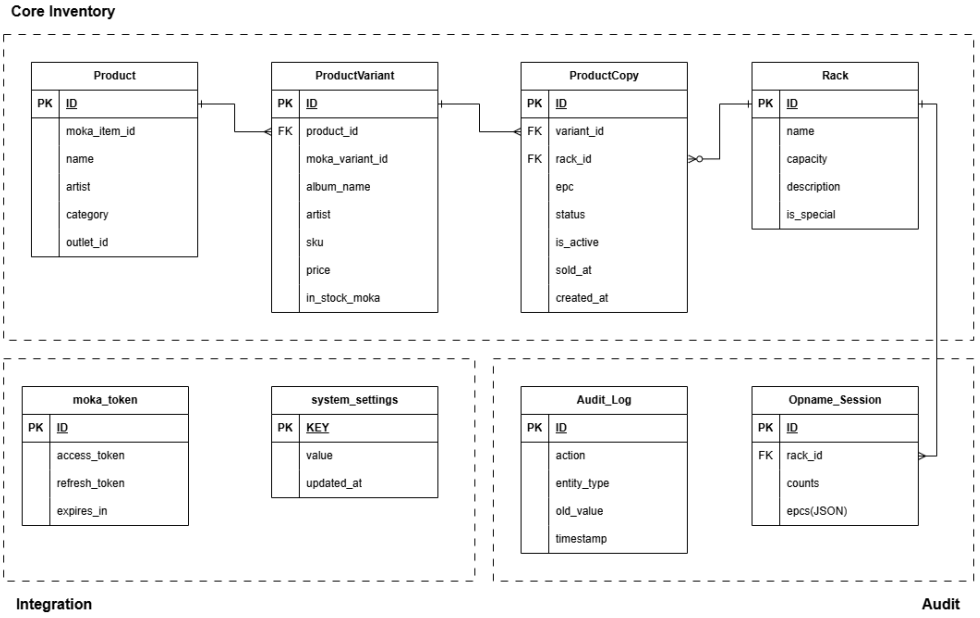
3.4.1 *Entity Relationship Diagram*

Skema database sistem terdiri dari delapan *entity* utama yang dikelompokkan berdasarkan fungsinya, sebagaimana dirangkum pada Tabel 3.3.

Tabel 3.3 Pengelompokan Entity Database

Kelompok	*Entity*	Fungsi Utama
Core Inventory	*Product, ProductVariant, ProductCopy, Rack*	Data inventaris utama
Integration	*MokaToken, SystemSettings*	Integrasi sistem eksternal
Audit	*OpnameSession, AuditLog*	Riwayat dan pelacakan aktivitas

Relasi *antar-entity* dalam sistem divisualisasikan dalam *Entity Relationship Diagram (ERD)* yang ditunjukkan pada Gambar 3.5



Gambar 3.5 Rancangan Entity Relationship Diagram

Tabel 3.4 Relasi Antar Entity

Relasi	Kardinalitas	Penjelasan
Product → *ProductVariant*	1:N	Satu artis memiliki banyak album
ProductVariant → *ProductCopy*	1:N	Satu album memiliki banyak unit fisik
Rack → *ProductCopy*	1:N	Satu rak berisi banyak *vinyl*
Rack → *OpnameSession*	1:N	Satu rak memiliki riwayat *opname*

3.4.2 Desain *Entity* Utama

Product

Entity Product merepresentasikan artis atau *band* sebagai pengelompokan tertinggi dalam inventaris.

Tabel 3.5 Struktur Entity Product

Atribut	Tipe	*Nullable*	Keterangam
id	Integer	*No*	Primary key

moka_item_id	Integer	*No*	Referensi ID item di MokaPOS
name	*String*	*No*	Nama artis
artist	*String*	*No*	Alias untuk konsistensi
category	*String*	*Yes*	Kategori dari *Moka*
outlet_id	Integer	*Yes*	ID outlet *Moka*

ProductVariant

Entity ProductVariant merepresentasikan album atau rilisan dari seorang artis.

Tabel 3.6 Struktur Entity ProductVariant

Atribut	Tipe	Nullable	Keterangan
id	Integer	No	Primary key
product_id	Integer	No	*Foreign key* ke Product
moka_variant_id	Integer	No	Referensi ID varian di *Moka*
album_name	*String*	No	Nama album
artist	*string*	No	Nama artist
sku	*String*	Yes	Stock Keeping Unit
price	*Decimal*	Yes	Harga jual
in_stock_moka	Integer	Yes	Stok referensi dari *POS*

ProductCopy

Entity ProductCopy merepresentasikan satu unit *vinyl* fisik dan menjadi penghubung antara data produk, lokasi fisik, dan identitas *RFID*.

Tabel 3.7 Struktur Entity ProductCopy

Atribut	Tipe	*Nullable*	Keterangan
id	Integer	*No*	Primary key
variant_id	Integer	*No*	*Foreign key* ke ProductVariant
epc	String	*Yes*	Identitas *RFID tag*
rack_id	Integer	*Yes*	*Foreign key* ke Rack
status	String	*No*	Status unit
is_active	Boolean	*No*	Penanda unit aktif
sold_at	Timestamp	*Yes*	Waktu penjualan
created_at	Timestamp	*No*	Waktu pembuatan record

Rack

Entity Rack merepresentasikan lokasi penyimpanan fisik seperti rak yang digunakan pada toko.

Tabel 3.8 Struktur Entity Rack

Atribut	Tipe	*Nullable*	Keterangan
id	Integer	No	*Primary key*
name	String	No	Nama rak
capacity	Integer	Yes	Kapasitas maksimal
description	String	Yes	Deskripsi
is_special	Boolean	No	Untuk *unplaced*

Sistem menyediakan satu *rack* khusus bernama *UNPLACED* yang berfungsi sebagai lokasi default untuk unit yang belum ditempatkan atau sebagai penampung sementara saat terjadi penghapusan rak.

3.4.3 Strategi *Nullable EPC*

Setelah proses sinkronisasi dari Moka POS, data *vinyl* yang masuk ke sistem belum memiliki *RFID tag*. Untuk menangani kondisi tersebut, atribut *EPC* pada entitas *ProductCopy* dirancang bersifat *nullable*.

Tabel 3.9 Penanganan Nullable EPC

Operasi	**Perlakuan Sistem**
Sinkronisasi stok	Membuat ProductCopy dengan *epc = NULL*
Assign RFID	*Assign* copy tanpa EPC
Stock opname	Hanya copy dengan EPC diproses
Perhitungan stok	Semua copy aktif dihitung

Konsistensi data dijaga melalui keunikan nilai EPC, validasi status unit, dan *foreign key* antar entitas.

3.5 Perancangan Algoritma dan *system logic*

Subbab ini membahas perancangan logika inti yang memproses data hasil pembacaan *RFID* dan menentukan status inventori. Pembahasan difokuskan pada pemisahan tanggung jawab antara pemrosesan data *low-level* dari *hardware* dan proses verifikasi stok pada level aplikasi.

3.5.1 Algoritma Deduplikasi Pembacaan Tag

RFID reader secara alami memancarkan identitas *tag* yang sama secara berulang dalam interval waktu yang sangat singkat. Tanpa adanya *filter*, perilaku ini berpotensi menghasilkan redundansi data yang tinggi dan membebani sistem.

Untuk mengatasi permasalahan tersebut, sistem dirancang dengan mekanisme deduplikasi pada sisi *backend*. Algoritma deduplikasi memanfaatkan struktur data berbasis *hash* yang memungkinkan identifikasi cepat terhadap tag yang sudah pernah terdeteksi dalam satu

sesi pemindaian. Kunci unik dibentuk dari kombinasi nilai *Electronic Product Code (EPC)* dan *Protocol Control (PC)* untuk membedakan setiap identitas fisik secara konsisten.

Dengan pendekatan ini, sistem hanya memproses perubahan status yang relevan, seperti kemunculan tag baru atau transisi kondisi tertentu, tanpa merespons setiap sinyal pembacaan yang berulang. Desain ini memastikan pemrosesan data tetap efisien dan hanya data yang relevan diteruskan ke lapisan berikutnya.

3.5.2 Logika Penentuan Status Opname

Proses verifikasi stok dirancang menggunakan perbandingan himpunan data untuk menentukan status inventori secara konsisten.

Dalam model ini, sistem mendefinisikan tiga himpunan utama:

- Himpunan ekspektasi (E), yang merepresentasikan daftar item yang sudah di *assign* ke dalam rak yang sedang di opname
- Himpunan hasil pemindaian (S), yang berisi identitas tag *RFID* yang terdeteksi oleh *hardware*
- Himpunan semesta data (D), yang merepresentasikan seluruh item yang terdaftar dalam sistem

Berdasarkan operasi antar-himpunan tersebut, status setiap item ditentukan sebagai berikut:

- *Found*, yaitu item yang berada dalam himpunan ekspektasi dan berhasil terdeteksi, didefinisikan sebagai $(E \cap S)$
- *Missing*, yaitu item yang diharapkan ada namun tidak terdeteksi, didefinisikan sebagai $(E - S)$
- *Extra*, yaitu item yang terdeteksi namun tidak termasuk dalam ekspektasi rak yang sedang diverifikasi, didefinisikan sebagai $((S \cap D) - E)$
- *Unknown*, yaitu tag yang terdeteksi oleh sistem tetapi tidak terdaftar dalam *database*, didefinisikan sebagai $(S - D)$

3.6 Perancangan *User Interface*

Perancangan *User Interface* difokuskan pada dukungan terhadap alur kerja operasional *stock opname* yang cepat, dan mudah dipahami oleh pengguna. Desain *User interface* tidak hanya berfokus pada aspek visual, tetapi juga memperkuat interaksi antara pengguna dan sistem melalui penyajian informasi yang jelas dan responsif.

Struktur *interface* aplikasi menggunakan pola navigasi yang konsisten, dengan *sidebar navigation* untuk perpindahan antar halaman utama dan *top navigation bar* untuk menampilkan konteks halaman aktif. *Top navigation bar* menyediakan akses ke sinkronisasi data dengan sistem *Point of Sale*, pengaturan tampilan (*light/dark mode*), serta konfigurasi sistem.

Pada halaman eksekusi *stock opname*, elemen *sidebar* dihilangkan agar perhatian pengguna fokus pada hasil pembacaan *RFID* dan status inventori yang ditampilkan secara *real-time*.

3.6.1 Halaman *Dashboard*

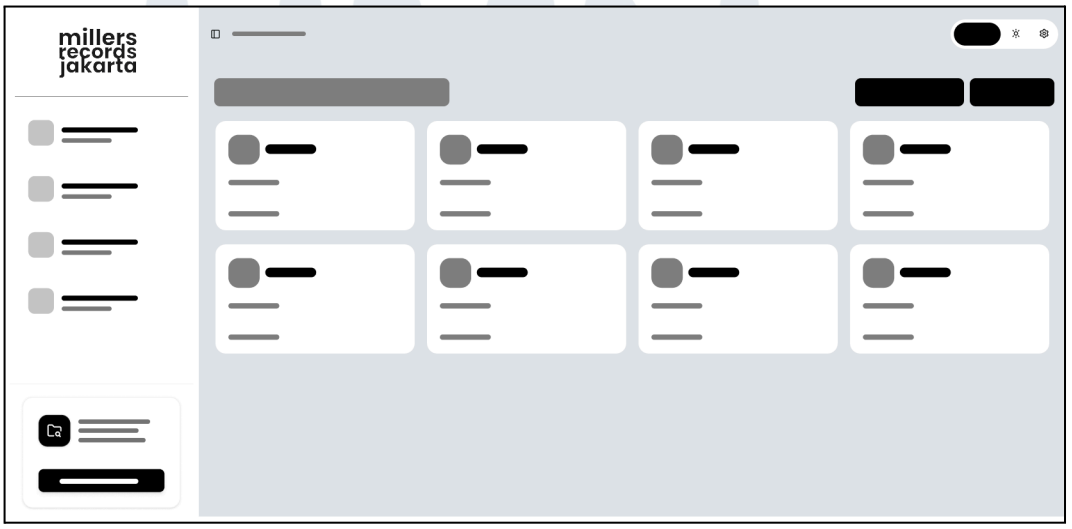
Halaman *Dashboard* berfungsi sebagai titik awal interaksi pengguna dengan sistem. *interface* ini dirancang untuk memberikan gambaran umum kondisi inventori dan status sistem secara ringkas. Informasi utama disajikan dalam bentuk *statistics cards* yang mudah dipahami sekilas, seperti total jumlah *item*, status konektivitas *RFID reader*, dan hasil sinkronisasi data.

Gambar 3.6 Rancangan halaman *dashboard*

3.6.2 Halaman *Rack Management*

Halaman *Rack Management* menggunakan tata letak berbasis *grid* untuk merepresentasikan struktur rak penyimpanan fisik di toko. Tata letak ini memudahkan pengguna dalam mengenali dan mengelola posisi barang secara fisik.

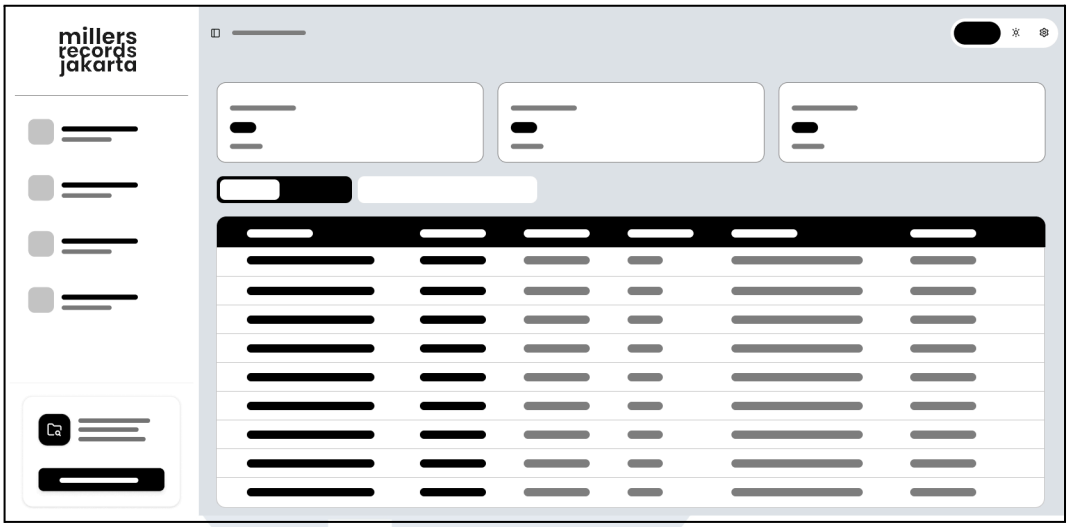
Interface ini menyediakan fungsi manajemen data (*Create, Read, Update, Delete*) untuk menambah, mengubah, dan mengatur konfigurasi rak.



Gambar 3.7 Rancangan halaman *Rack Management*

3.6.3 Halaman *Library*

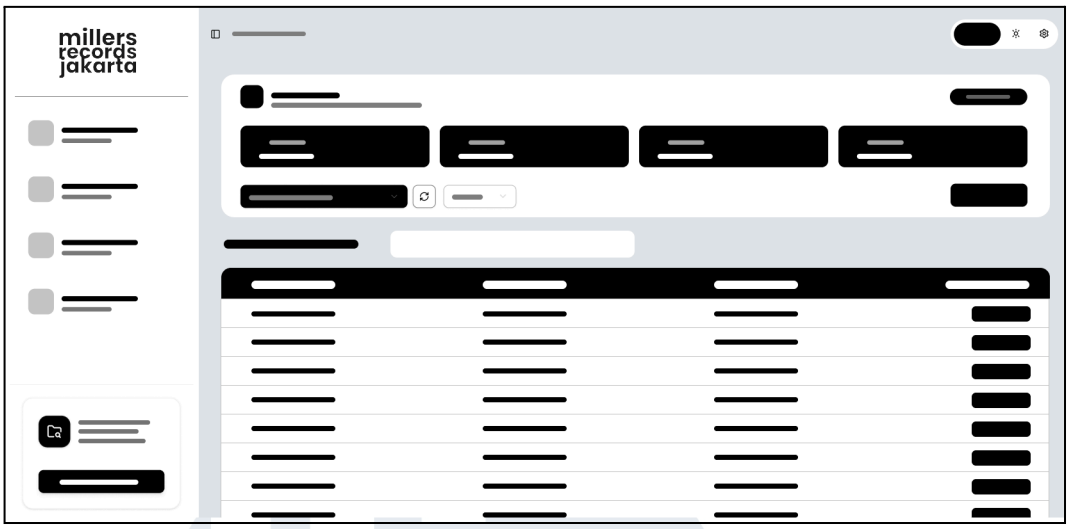
Halaman Library menampilkan data *vinyl* toko dengan navigasi tab untuk memisahkan produk yang sudah memiliki *RFID tag* (*assigned*) dan yang belum (*unassigned*).



Gambar 3.8 Rancangan halaman *Library*

3.6.4 Halaman *Assign RFID tag*

Halaman *Assign RFID Tag* digunakan untuk melakukan *assignment* RFID *tag* ke produk. Desain difokuskan pada efisiensi input, dengan dukungan fitur *batch selection* untuk meminimalkan langkah berulang saat memproses banyak *item*. Hal ini penting untuk efisiensi saat proses penempelan *RFID tag* dilakukan dalam jumlah besar.

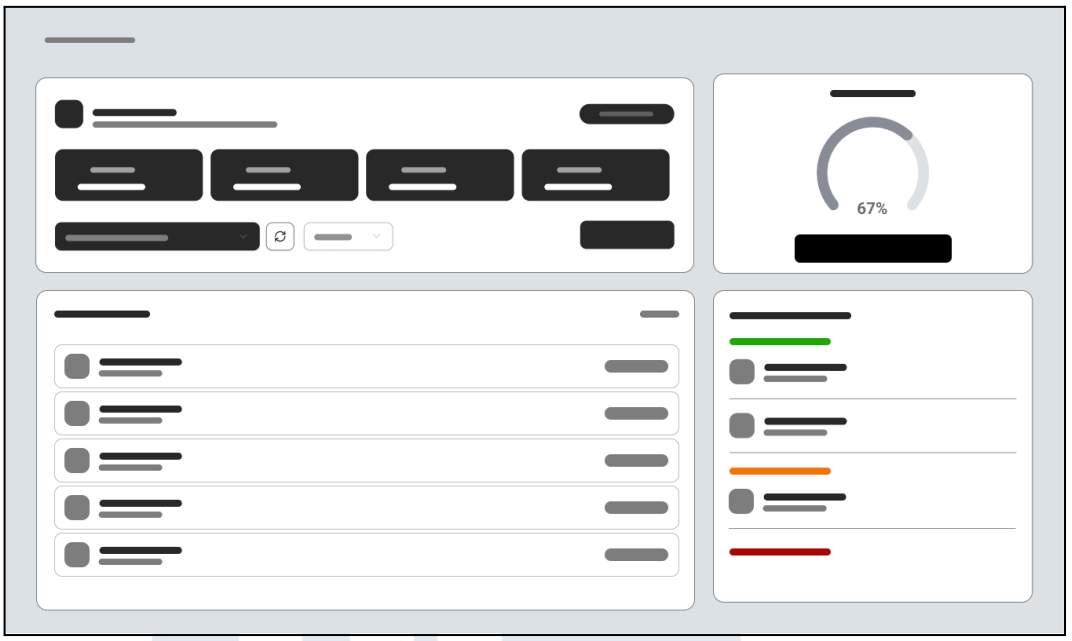


Gambar 3.9 Rancangan halaman *Assign RFID*

3.6.5 Halaman Opname

Interface stock opname dibagi ke dalam dua fase, yaitu halaman persiapan dan halaman eksekusi. Pada fase persiapan, pengguna memilih rak target yang akan diperiksa.

Pada fase eksekusi, *interface* dirancang secara minimalis dengan fokus pada daftar status item (Expected dan Scanned) serta indikator progres visual.



Gambar 3.11 (a) Rancangan halaman *opname dashboard.* (b) Rancangan halaman *opname session*

3.7 Rancangan Pengujian dan Evaluasi Sistem

Subbab ini membahas prosedur pengujian sistem untuk mengukur efisiensi waktu *stock opname* dan membandingkannya dengan metode manual.

3.7.1 Metodologi Pengujian

Pengujian dilakukan dengan membandingkan waktu yang dibutuhkan untuk menyelesaikan *stock opname* pada satu rak yang sama menggunakan metode manual dan sistem *RFID*.

- Prosedur manual yang menggunakan penglihatan yang menjadi standar operasional toko saat ini.
- Prosedur otomatis menggunakan sistem *stock opname* dengan *RFID* yang dikembangkan.

3.7.2 Objek dan Lingkungan Pengujian

Pengujian dirancang untuk dilaksanakan secara *on-site* pada lingkungan operasional toko.

- Pengujian difokuskan pada satu unit rak penyimpanan penuh yang berisi koleksi *vinyl*. Pengujian dilakukan pada satu rak penuh untuk merepresentasikan kondisi toko.
- Pengujian dilakukan pada kondisi tata letak toko standar tanpa modifikasi khusus, untuk mengukur performa sistem dalam kondisi sebenarnya.

3.7.3 Skenario Pengujian

Prosedur pengujian dibagi menjadi dua skenario utama yang akan dijalankan secara berurutan pada rak yang sama:

Skenario A

Pengukuran Metode Manual, Skenario ini bertujuan untuk mendapatkan data *baseline metrics*.

- Operator menyiapkan daftar stok referensi (dari sistem POS atau berkas Excel).
- Operator melakukan verifikasi fisik dengan mencocokkan judul/label pada setiap keping *vinyl* di rak satu per satu dengan daftar stok.
- Waktu pengukuran dimulai ketika operator mulai menyentuh item pertama dan dihentikan saat operator menyatakan telah selesai.

Skenario B

Pengukuran dengan *RFID*, Skenario ini bertujuan untuk mengukur kecepatan pemrosesan *opname stock* dengan *RFID*.

- Operator mengaktifkan mode *Stock Opname* pada aplikasi.
- Operator melakukan gerakan *sweeping* di depan rak menggunakan *RFID reader*.
- Waktu pengukuran dimulai saat operator menekan tombol *start* pada halaman opname dan dihentikan saat sistem memberikan indikator visual bahwa seluruh item dalam daftar ekspektasi rak tersebut telah *Found*.

3.7.4 Metrik Evaluasi

Data pengukuran dari kedua skenario akan dianalisis menggunakan satu metrik yaitu waktu. Evaluasi dilakukan dengan dua pendekatan perhitungan:

- Rata-rata Waktu Proses. Mengukur rata-rata durasi yang dibutuhkan untuk memverifikasi satu unit barang dalam detik.

$$Waktu\ per\ Item = \frac{Total\ Durasi\ (Detik)}{Jumlah\ Item}$$

- Persentase Reduksi Waktu. Mengukur seberapa besar penurunan durasi proses yang dihasilkan sistem usulan dibandingkan metode dasar.

$$Reduksi\ Waktu\ (\%) = \left(\frac{T_{Manual} - T_{RFID}}{T_{Manual}}\right) \times 100\%$$

Dimana:

T_{Manual} = Total waktu opname metode manual

T_{RFID} = Total waktu opname metode *RFID*