Jurnal Ilmiah Wahana Pendidikan, Juli 2026, 12(7.D), 171-191
DOI: https://jurnal.peneliti.net/index.php/JIWP/article/view/13639
p-ISSN: 2622-8327 e-ISSN: 2089-5364
Accredited by Directorate General of Strengthening for Research and Development
Available online at https://jurnal.peneliti.net/index.php/JIWP

Aplikasi Mobile Point Of Sales Berbasis Android Dengan Pemesanan Qr Code Dan Analisis Prediksi Stok Menggunakan Algoritma Linear Regression

Ziyat Akhsani[1], Yudi Santoso[2]

[1,2] Sekolah Tinggi Ilmu Komputer Cipta Karya Informatika (STIKOMCKI) Jakarta

Received: 19 Januari 2026
Revised: 29 Januari 2026
Accepted: 9 Februari 2026

Abstrak

Perkembangan teknologi digital yang semakin mendorong banyak pelaku usaha telah memberikan kemudahan dalam berbagai bidang, termasuk sektor penjualan di bidang kuliner. Kedai kopi Mocks Street masih menggunakan sistem pencatatan manual sehingga sering terjadi kesalahan dalam pengelolaan pesanan dan stok barang. Penelitian ini bertujuan mengembangkan Sistem Informasi *Point of Sales* (POS) berbasis Android dengan fitur pemesanan menggunakan QR Code serta prediksi stok menggunakan algoritma Linear Regression. Aplikasi memungkinkan pelanggan memesan secara mandiri melalui pemindaian QR Code di meja dan pesanan langsung tercatat di sistem tanpa campur tangan kasir. Sistem ini dirancang untuk mempermudah proses transaksi, meningkatkan akurasi pemesanan, serta membantu pemilik usaha dalam memperkirakan kebutuhan persediaan karena dalam sistem mampu memperbarui stok secara realtime menggunakan database lokal, yang dapat memprediksi kebutuhan stok berdasarkan data penjualan sebelumnya. Hasil implementasi menunjukkan bahwa sistem POS yang dikembangkan pada aplikasi ini membantu menjalankan proses transaksi melalui QR Code, mengurangi kesalahan pencatatan dan mempermudah pengelolaan stok, sehingga meningkatkan efisiensi operasional di Mocks Street Coffee. Sistem ini dinilai membantu dan mendukung operasional bisnis serta meningkatkan manajemen persediaan.

Kata Kunci: Point of Sales (POS), Android, QR Code, Algoritma Linear Regression, Prediksi Stok, Implementasi dalam kehidupan.

(*) Corresponding Author: akhsaniziyat@gmail.com, yudisantoso@stikomcki.ac.id

How to Cite: Akhsani, Z., & Santoso, Y. (2026). APLIKASI MOBILE POINT OF SALES BERBASIS ANDROID DENGAN PEMESANAN QR CODE DAN ANALISIS PREDIKSI STOK MENGGUNAKAN ALGORITMA LINEAR REGRESSION. *Jurnal Ilmiah Wahana Pendidikan*, *12*(7.D), 171-191. Retrieved from https://jurnal.peneliti.net/index.php/JIWP/article/view/13639.

PENDAHULUAN

Perkembangan teknologi informasi yang pesat telah membawa perubahan yang signifikan dalam beberapa sektor, terutama pada sektor perdagangan dan jasa. Proses transaksi yang sebelumnya dilakukan secara manual kini akan beraih ke sistem yang lebih terintegrasi untuk meningkatkan efisiensi dalam pelayanan. Salah satu inovasi yang banyak diterapkan dalam dunia bisnis adalah Sistem Informasi Point of sales (POS) yang berfungsi membantu proses transaksi penjualan secara cepat dan akurat. Sistem POS berfungsi untuk mengelola data stok, laporan pejualan serta mendukung keputusan melalui analisis data.

Di sisi lain, perubahan perilaku konsumen yang semakin mengutamakan kemudahan dalam berinteraksi menuntut pelaku usaha untuk beradaptaasi dengan teknologi digital. Pemanfaatan perangkat mobile berbasis Android memberikan

kemudahan akses bagi pengguna. Dengan integrasi pemesanan melalui QR Code, pelanggan dapat melakukan pemesanan secara mandiri tanpa harus menunggu pelayan, sehingga meningkatkan efisien dan kenyamanan dalam proses transaksi jual beli.

Permasalahan yang sering muncul dalam sistem penjualan dan pembelian adalah ketidaktepatan dalam pengelolaan stok barang yang dapat menyebabkan kelebihan atau kekurangan stok. Stok yang menumpuk juga bisa mengakibatkan kerugian. Untuk mengatasi hal tersebut, diperlukan metode analisis yang mampu memprediksi kebutuhan stok berdasarkan data penjualan sebelumnya. Salah satu metode yang dapat digunakan adalah Algoritma Linear Regression, yang dapat mengestimaasi tren penjualan serta membantu penjual untuk memanajemen dan menentukan jumlah stok yang optimal dan akurat.

Berdasarkan permasalahan tersebut, peneliti mengembangakan Aplikasi Mobile Point of Sales berbasis Android dengan pemesanan QR Code dan Analisis Prediksi Stok menggunakan Algoritma Linear Reggresion. Dimana diharapkan pada sistem ini dapat memberikan solusi yang lebih efisien dalam mengelola transaksi serta menghitung persediaan stok barang bagi kedai Mocks Street yang bertempat di Jl. Pejompongan IV no 136, Jakarta Pusat.1.

METODE

Penelitian ini menggunakan pendekatan kualitatif deskriptif dengan sumber data dari aktivitas operasional Kedai Kopi Mocks Street, yang berfokus pada analisis kebutuhan, alur kerja (*workflow*), serta permasalahan transaksi dan pengelolaan stok sebelum dan sesudah penerapan sistem. Data dikumpulkan melalui observasi langsung, wawancara dengan kasir dan pengelola, serta dokumentasi catatan penjualan dan stok, kemudian disusun menjadi *synthetic private dataset* untuk kebutuhan pengujian dan analisis prediksi guna menjaga kerahasiaan bisnis. Tahapan penelitian dilakukan secara berurutan mulai dari pengolahan dan pembersihan data (*data cleaning*), pengelompokan penjualan berdasarkan periode, hingga penyesuaian data agar dapat diolah secara numerik. Selanjutnya diterapkan algoritma *Linear Regression* untuk memprediksi penjualan periode berikutnya, dengan variabel X sebagai urutan waktu dan Y sebagai jumlah produk terjual, sehingga hasil prediksi dapat digunakan sebagai acuan kebutuhan stok. Sistem kemudian diimplementasikan menjadi aplikasi *Point of Sales* (POS) berbasis Android menggunakan Android Studio dan bahasa pemrograman Kotlin yang terhubung dengan basis data, dengan fitur utama pencatatan transaksi, pemesanan menggunakan QR Code, pengelolaan stok, dan penyajian prediksi stok. Pengujian sistem dilakukan menggunakan *Black Box Testing* untuk memastikan fungsi aplikasi berjalan sesuai kebutuhan, mencakup pengujian fungsional POS, mekanisme pemesanan QR Code, serta pengujian prediksi stok. Evaluasi ketepatan prediksi dilakukan dengan membandingkan hasil prediksi dan data aktual menggunakan metrik MSE (*Mean Squared Error*), RMSE (*Root Mean Squared Error*), MAE (*Mean Absolute Error*), dan MAPE (*Mean Absolute Percentage Error*) guna mengukur tingkat kesalahan prediksi secara kuantitatif.

HASIL & PEMBAHASAN

1. Implementasi dan Pengujian

Pada tahap implementasi dan pengujian, penulis melakukan beberapa perancangan dan alur flow guna melakukan pembuatan aplikasi POS berbasis Android dengan menggunakan QR Code dan Analisis Prediksi Stok.

Aplikasi POS Android dibangun menggunakan Kotlin dengan Pattern MVVM dan *clean architecture* (core, data, feature). Aplikasi terdiri dari 9 fitur utama yang menangani lifecycle dari pemesanan hingga transaksi. Berikut untuk struktur package pada implementasinya:

```
app/src/main/java/com/example/skripsi/
├── core/          (utilities, constants, ext)
├── data/          (DAO, Entity, Model, API, Repository)
└── feature/       (ui, viewmodel, fragments/activities)
    ├── auth       (login/autentikasi)
    ├── barang     (manajemen produk)
    ├── order      (pembuatan pesanan)
    ├── qr         (QR code scanning)
    ├── checkout   (proses pembayaran)
    ├── stok       (manajemen stok)
    ├── prediksi   (prediksi stok)
    ├── admin      (dashboard admin)
    └── transaksi  (riwayat transaksi)
```

Gambar 1. Struktur Package

Penggunaan aplikasi POS memiliki alur flow sebagai tujuan atas pengimplementasian pada fitur yang tersedia.

Aktor	Alur
Customer	Scan QR meja → Pilih menu → Add to cart → Checkout → Konfirmasi pembayaran (QRIS) → Transaksi selesai
Admin	Login → Dashboard → Kelola produk → Lihat pesanan masuk → Update status order → Lihat prediksi stok

Gambar 2. Alur Flow Aplikasi

2. Penjelasan code atas pembuatan sistem

```
    vm = ViewModelProvider( owner = this, factory = LoginViewModelFactory( ctx = this))[LoginViewModel::class.java]
    vm.seed()  // seed data user pertama kali

    binding.btnLogin.setOnClickListener {
        val username = binding.etUsername.text.toString().trim()
        val password = binding.etPassword.text.toString().trim()

        if (username.isEmpty() || password.isEmpty()) {
            Toast.makeText( context = this, text = "Username dan Password tidak boleh kosong", duration = Toast.LENGTH_SHORT).show()
            return@setOnClickListener
        }

        vm.login(username, password,
            onSuccess = {
                Toast.makeText( context = this, text = "Login berhasil", duration = Toast.LENGTH_SHORT).show()
                startActivity(Intent( packageContext = this, cls = MainActivity::class.java))
                finish()
            },
            onError = { msg ->
                Toast.makeText( context = this, text = msg, duration = Toast.LENGTH_SHORT).show()
            }
        )
    }
}
```

Gambar 3. Auth untuk mengetahui Pembeli / Penjual

Validasi ketika login jika error, maka akan memunculkan toast dan jika berhasil maka device akan save data dan nantinya akan langsung masuk ke halaman dashboard.

```
fun login(username: String, password: String, onSuccess: () -> Unit, onError: (String) -> Unit) {
    viewModelScope.launch( context = Dispatchers.IO) {
        val u = repo.login(username, password)
        if (u == null) {
            launch( context = Dispatchers.Main) { onError("Username/password salah") }
        } else {
            session.saveUser(u.username, u.role)
            launch( context = Dispatchers.Main) { onSuccess() }
        }
    }
}
```

Gambar 4. Validasi ketika login

Beberapa validasi ketika input barang:

a. Validasi Input, artinya jika ada kolom yang tidak terisi maka akan memunculkan toast
b. Validasi duplikat, jika menambahkan data barang yang sudah ada sebelumnya maka akan memunculkan alert dialog untuk mengarahkan restock barang

```
binding.btnTambah.setOnClickListener {
    val nama = binding.etNama.text.toString().trim()
    val hargaText = binding.etHarga.text.toString()
    val stokText = binding.etStok.text.toString()

    // Validasi input
    val hargaDigits = hargaText.replace( regex = "\\D".toRegex(), replacement = "")
    val harga = hargaDigits.toLongOrNull()
    val stok = stokText.toIntOrNull()

    if (nama.isEmpty() || harga == null || harga <= 0L || stok == null || stok <= 0) {
        Toast.makeText( context = this, text = "Input tidak valid", duration = Toast.LENGTH_SHORT).show()
        return@setOnClickListener
    }

    // CEK DUPLIKAT
    viewModel.getBarangByNama(nama) { existingBarang ->
        if (existingBarang != null) {
            // Barang sudah ada - tampilkan dialog restock
            showRestockDialog(existingBarang, newStok = stok, newHarga = harga)
        } else {
            // Barang baru - insert langsung
            val now = System.currentTimeMillis()
            viewModel.insert(BarangEntity(nama = nama, harga = harga, stok = stok, createdAt = now))
            clearFormInputs()
            Toast.makeText( context = this, text = "Barang berhasil ditambahkan", duration = Toast.LENGTH_SHORT).show()
        }
    }
}
```

Gambar 5. Validasi input barang

Class barang adapter berguna untuk memunculkan data barang yang sudah diinput serta pengecekan duplikat jika data sama maka akan ditaruh di kolom yang sama dan di update, jika data baru maka akan membuat kolom yang baru.

```
inner class ViewHolder(private val binding: ItemBarangBinding) :
    RecyclerView.ViewHolder( itemView = binding.root) {
    fun bind(item: BarangEntity) {
        binding.tvNamaBarang.text = item.nama
        binding.tvHargaBarang.text =
            "Rp %,d".format(Locale( language = "in",  country = "ID"),  ...args = item.harga)
        binding.tvStokBarang.text = "Stok: ${item.stok}"
    }
}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
    val binding = ItemBarangBinding.inflate(
        inflater = LayoutInflater.from(parent.context),
        parent,  attachToParent = false
    )
    return ViewHolder(binding)
}

override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    holder.bind( item = getItem(position))
}

1 Usage
class DiffCallback : DiffUtil.ItemCallback<BarangEntity>() {
    override fun areItemsTheSame(oldItem: BarangEntity, newItem: BarangEntity) =
        oldItem.id == newItem.id

    override fun areContentsTheSame(oldItem: BarangEntity, newItem: BarangEntity) =
        oldItem == newItem
}
```

Gambar 6. Class barang adapter

Sistem ini melakukan permintaan perizinan kepada user untuk mengakses kamera hp yang bertujuan untuk melakukan scan qr.

```
private val requestPermission =
    registerForActivityResult( contract = ActivityResultContracts.RequestPermission()) { granted ->
        if (granted) startCamera() else Toast.makeText(
            context = this,
            text = "Izin kamera diperlukan",
            duration = Toast.LENGTH_SHORT
        ).show()
    }

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ActivityQrBinding.inflate(layoutInflater)
    setContentView(binding.root)

    if (ContextCompat.checkSelfPermission(
            context = this,
            permission = Manifest.permission.CAMERA
        ) == PackageManager.PERMISSION_GRANTED
    ) {
        startCamera()
    } else {
        requestPermission.launch( input = Manifest.permission.CAMERA)
    }
}
```

Gambar 7. Permohonan Aplikasi untuk mengakses kamera

Membuat kondisi untuk kamera dapat membaca barcode, ketika berhasil maka akan menjalankan function handlePayload yang nanti akan di proses isi data berdasarkan barcode. Jika gagal maka kamera akan keluar dan tidak dapat membaca barcode.

```
analysis.setAnalyzer(cameraExecutor) { imageProxy: ImageProxy ->
    if (handledOnce) {
        imageProxy.close(); return@setAnalyzer
    }
    val mediaImage = imageProxy.image
    if (mediaImage != null) {
        val image =
            InputImage.fromMediaImage(mediaImage, imageProxy.imageInfo.rotationDegrees)
        scanner.process( p0 = image)
            .addOnSuccessListener { barcodes ->
                if (barcodes.isNotEmpty()) {
                    val raw = barcodes.first().rawValue ?: ""
                    handlePayload(raw)
                    handledOnce = true
                }
            }
            .addOnFailureListener {
                // optional log
            }
            .addOnCompleteListener { imageProxy.close() }
    } else {
        imageProxy.close()
    }
}
```

Gambar 8. Kondisi kamera ketika scan barcode

Membuat function untuk mengambil data dari QR dan kemudian dipross lalu menghasilkan 2 kondisi berhasil dan gagal

```
private fun handlePayload(raw: String) {
    Log.d( tag = "QR",  msg = "raw=$raw")
    when {
        raw.startsWith( prefix = "item:") -> {
            val id = raw.removePrefix( prefix = "item:").toIntOrNull()
            if (id != null) {
                val result = Intent().apply {
                    putExtra( name = "qr_type",  value = "item")
                    putExtra( name = "qr_value",  value = id.toString())
                }
                setResult( resultCode = RESULT_OK,  data = result)
                finish()
            } else {
                Toast.makeText( context = this,  text = "QR item tidak valid",  duration = Toast.LENGTH_SHORT).show()
                handledOnce = false
            }
        }
raw.startsWith( prefix = "table=") -> {
    // format: table=1&status=success
    val params = raw.split( ...delimiters = "&")
    val tableId = params
        .firstOrNull { it.startsWith( prefix = "table=") }
        ?.substringAfter( delimiter = "table=")

    val statusDemo = params
        .firstOrNull { it.startsWith( prefix = "status=") }
        ?.substringAfter( delimiter = "status=")
        ?: "success"

    if (!tableId.isNullOrBlank()) {
        val result = Intent().apply {
            putExtra( name = "qr_type",  value = "table")
            putExtra( name = "qr_value",  value = tableId)
            putExtra( name = "qr_status",  value = statusDemo)
        }
        setResult( resultCode = RESULT_OK,  data = result)
        finish()
    } else {
        Toast.makeText( context = this,  text = "QR meja tidak valid",  duration = Toast.LENGTH_SHORT).show()
        handledOnce = false
    }
}
 raw.startsWith( prefix = "cart:") -> {
     val payload = raw.removePrefix( prefix = "cart:")
     val result = Intent().apply {
         putExtra( name = "qr_type",  value = "cart")
         putExtra( name = "qr_value",  value = payload)
     }
     setResult( resultCode = RESULT_OK,  data = result)
     finish()
 }
 else -> {
     Toast.makeText( context = this,  text = "Format QR tidak dikenali",  duration = Toast.LENGTH_SHORT).show()
     handledOnce = false
 }
```

Gambar 9. Fungsi membaca isi dari barcode

Pada gambar berikut merupakan data class yang berisi informasi keseluruhan pesanan. Terdiri dari field total (total harga) dan items yang merupakan List dari Order Item. Setiap OrderItem merepresentasikan satu produk yang dipesan dengan detail: product_id (ID produk), qty (jumlah), harga (harga satuan), dan subtotal (qty × harga). Data class ini digunakan untuk serialization ke JSON dan pengiriman pesanan ke backend via API endpoint /api/orders.

```
data class OrderRequest(
    val total: Double,
    val items: List<OrderItem>
)

3 Usages
data class OrderItem(
    val product_id: Int,
    val qty: Int,
    val harga: Double,
    val subtotal: Double
)
```

Gambar 10. Data class order

Kode ini mengimplementasikan observer pattern menggunakan LiveData untuk memantau perubahan list pesanan secara real-time. Ketika list pesanan berubah, function orderAdapter.submit(list) mengirimkan data ke adapter untuk update RecyclerView. Total harga dihitung dengan sumOf { it.qty.toLong() * it.hargaSatuan } yang menjumlahkan subtotal dari setiap item, kemudian diformat menggunakan CurrencyFormatter.rupiah() dan ditampilkan pada tvTotal. Pendekatan ini memastikan total harga selalu tersinkronisasi dengan data pesanan.

```
// Observe list pesanan
vm.items.observe( owner = this) { list ->
    orderAdapter.submit(list)
    val totalAmount = list.sumOf { it.qty.toLong() * it.hargaSatuan }
    binding.tvTotal.text = "Total: ${CurrencyFormatter.rupiah( v = totalAmount)}"

// Muat menu dari DB
loadMenuFromDb()
```

Gambar 11. Observe data list pesanan

Ketika customer ingin menyelesaikan transaksi, sistem melakukan serangkaian validasi sebelum memproses pembayaran. Proses ini diimplementasikan dalam setOnClickListener pada tombol "Kirim" yang bertujuan memastikan order valid dan metode pembayaran telah dipilih.

Ketika customer mengklik tombol "Kirim", sistem pertama-tama mengambil orderId dari ViewModel. Validasi pertama yang dilakukan adalah mengecek apakah orderId sudah ada atau masih null. Jika orderId adalah null (berarti order belum berhasil dibuat di backend), aplikasi akan menampilkan Toast notification dengan pesan "Order belum sign up" dan mengakhiri proses menggunakan return@setOnClickListener tanpa melanjutkan ke tahap pembayaran. Hal ini memastikan bahwa tidak ada pembayaran yang diproses untuk order yang tidak valid.

Setelah validasi orderId berhasil, sistem menggunakan when statement untuk mengecek metode pembayaran mana yang dipilih customer. Jika customer memilih metode pembayaran CASH (tunai), sistem memanggil function vm.markUnpaid dengan parameter:

- Method = "CASH": Menunjukkan metode pembayaran adalah CASH
- OnDone: Callback yang dijalankan jika request ke backend berhasil. Di dalamnya menampilkan pesan sukses "Pembayaran dikirim (CASH). Tunjukkan ke kasir." selama durasi LENGTH_LONG, lalu menutup activity dengan finish
- OnError: Callback yang dijalankan jika request ke backend gagal. Di dalamnya menampilkan error message dari server dengan detail error: ${e.message}

Ini memastikan proses pembayaran yang robust dengan validasi berlapis dan error handling yang proper di setiap tahap.

```
binding.btnKirim.setOnClickListener {
    val id = vm.orderId
    if (id == null) {
        Toast.makeText( context = this, text = "Order belum siap", duration = Toast.LENGTH_SHORT).show()
        return@setOnClickListener
    }
    when {
        binding.rbCash.isChecked -> {
            vm.markUnpaid(
                method = "CASH",
                onDone = {
                    Toast.makeText(
                        context = this,
                        text = "Pesanan dikirim (CASH). Tunjukkan ke kasir.",
                        duration = Toast.LENGTH_LONG
                    ).show()
                    finish()
                },
                onError = { e ->
                    Toast.makeText(
                        context = this,
                        text = "Gagal kirim: ${e.message}",
                        duration = Toast.LENGTH_LONG
                    ).show()
                }
            )
        }
binding.rbEwallet.isChecked -> {
    // E-Wallet = QRIS demo
    startQrisPayment()
}
else -> {
    Toast.makeText( context = this, text = "Pilih metode bayar dulu", duration = Toast.LENGTH_SHORT).show()
}
```

Gambar 12. Implementasi payment menthod handling

Function startQrisPayment mengimplementasikan alur pembayaran QRIS secara end-to-end mulai dari validasi order ID dan cart items, transformasi data cart

ke format OrderItem yang sesuai API spesifikasi, kalkulasi total harga menggunakan sumOf { it.subtotal }, hingga pengiriman order ke backend melalui HTTP POST request ke endpoint /api/orders. Setelah backend mengembalikan order_id yang unik, function melanjutkan dengan mengirim request ke endpoint /api/create-qris beserta order ID tersebut untuk meminta backend memanggil Midtrans API dan generate QR code QRIS. Seluruh proses komunikasi dengan backend dilakukan dalam coroutine dengan lifecycleScope.launch dan withContext (Dispatchers.IO) untuk menghindari blocking UI thread. Setelah mendapatkan QR code string dari response Midtrans, function memanggil show Qris Dialog untuk menampilkan QR code kepada customer dalam bentuk dialog sehingga customer dapat scan menggunakan e-wallet. Jika terjadi error pada salah satu tahap (validasi gagal, network error, invalid response), aplikasi akan menampilkan Toast notification dengan pesan error detail untuk memberikan feedback kepada user.

```
private fun startQrisPayment() {
    val draftId = vm.orderId
    if (draftId == null) {
        Toast.makeText( context = this, text = "Order belum siap", duration = Toast.LENGTH_SHORT).show()
        return
    }

    val cartItems = vm.items.value.orEmpty()
    if (cartItems.isEmpty()) {
        Toast.makeText( context = this, text = "Belum ada item di pesanan", duration = Toast.LENGTH_SHORT).show()
        return
    }

    val orderItems = cartItems.map { item ->
        OrderItem(
            product_id = item.barangId,
            qty = item.qty,
            harga = item.hargaSatuan.toDouble(),
            subtotal = (item.qty * item.hargaSatuan).toDouble()
        )
    }
```

```
val total = orderItems.sumOf { it.subtotal }
val request = OrderRequest(
    total = total,
    items = orderItems
)
lifecycleScope.launch {
    try {
        // 1. Kirim order ke backend (mock)
        val orderRes = withContext( context = Dispatchers.IO) {
            ApiClient.apiService.createOrder( body = request)
        }
        val remoteOrderId = orderRes.order_id

        // 2. Minta QRIS untuk order ini
        val qrisRes = withContext( context = Dispatchers.IO) {
            ApiClient.apiService.createQris( body = QrisRequest( order_id = remoteOrderId))
        }

        val qrString = qrisRes.qris_data.qr_string

        // 3. Tampilkan QR & gunakan statusDemo untuk hasil
        showQrisDialog(qrString, remoteOrderId)
```

```
} catch (e: Exception) {
    e.printStackTrace()
    Toast.makeText(
        context = this@OrderActivity,
        text = "Gagal memproses pembayaran online: ${e.message}",
        duration = Toast.LENGTH_LONG
    ).show()
}
```

Gambar 13. Function alur pembayaran via qris

ApiService adalah interface Kotlin yang mendefinisikan contract/blueprint semua HTTP endpoint yang tersedia di backend Laravel. Interface ini menggunakan Retrofit annotations seperti @POST, @GET, @Path, dan @Body untuk menspesifikasi HTTP method, endpoint URL, dan parameter yang diperlukan. Setiap function dalam interface merepresentasikan satu endpoint API (contoh: createOrder() untuk POST ke /api/orders, createQris() untuk POST ke /api/create-qris, getOrderStatus() untuk GET dari /api/order-status/{order_id}), dengan parameter function yang menjadi payload atau path parameter dan return type yang merupakan response model (contoh: OrderResponse, QrisResponse, StatusResponse). Pada saat runtime, Retrofit secara otomatis generate implementasi dari interface ini (menggunakan Java reflection dan dynamic proxy), sehingga ketika function dari apiService dipanggil, Retrofit langsung melakukan HTTP request ke endpoint yang sesuai, auto-serialize parameter menjadi JSON, auto-deserialize response JSON menjadi model object yang ditentukan, dan return hasilnya kepada caller.

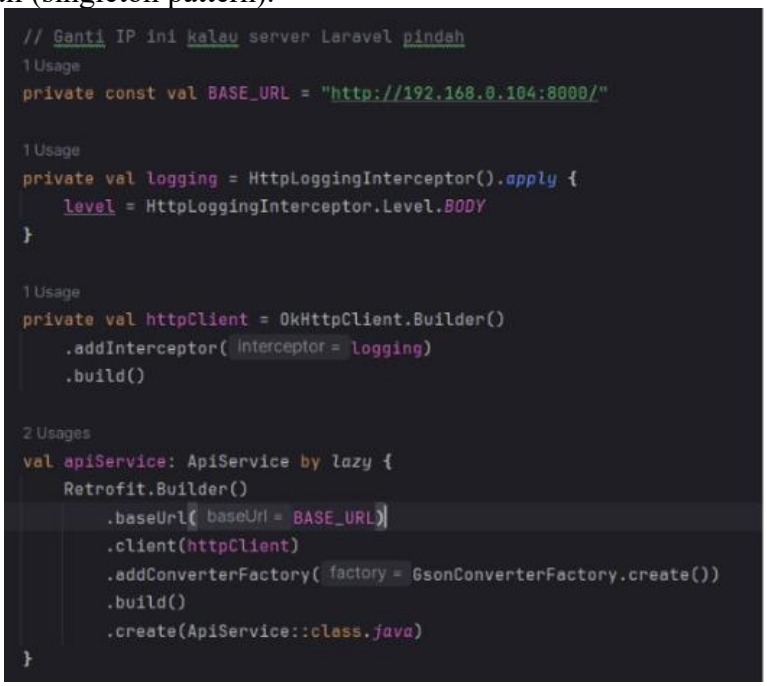
```
1 Usage
@POST( value = "api/orders")
suspend fun createOrder(
    @Body body: OrderRequest
): OrderResponse

1 Usage
@POST( value = "api/create-qris")
suspend fun createQris(
    @Body body: QrisRequest
): QrisResponse
```

Gambar 14. ApiService

ApiClient adalah singleton object yang bertanggung jawab untuk mengkonfigurasi dan menginisialisasi Retrofit HTTP client yang digunakan untuk komunikasi dengan backend Laravel. Object ini mendefinisikan BASE_URL yang merupakan alamat IP dan port backend (contoh: "http://000.000.0.000:8000/"),

mengatur HttpLoggingInterceptor dengan level BODY untuk logging HTTP request-response pada saat developmet, membangun OkHttpClient dengan interceptor logging, kemudian membuat instance Retrofit dengan configuration: base URL, OkHttpClient, dan GsonConverterFactory untuk automatic JSON serialization-deserialization. Selain itu, ApiClient menyediakan lazy-initialized property apiService yang merupakan instance dari interface ApiService, sehingga setiap kali aplikasi membutuhkan memanggil API endpoint, cukup menggunakan ApiClient.apiService.methodName() tanpa perlu membuat ulang Retrofit instance berulang kali (singleton pattern).



```
// Ganti IP ini kalau server Laravel pindah
1 Usage
private const val BASE_URL = "http://192.168.0.104:8000/"

1 Usage
private val logging = HttpLoggingInterceptor().apply {
    level = HttpLoggingInterceptor.Level.BODY
}

1 Usage
private val httpClient = OkHttpClient.Builder()
    .addInterceptor( interceptor = logging)
    .build()

2 Usages
val apiService: ApiService by lazy {
    Retrofit.Builder()
        .baseUrl( baseUrl = BASE_URL)
        .client(httpClient)
        .addConverterFactory( factory = GsonConverterFactory.create())
        .build()
        .create(ApiService::class.java)
}
```

Gambar 15. ApiClient

Backend Laravel mengimplementasikan 3 endpoint utama yang diakses oleh aplikasi Android untuk mengelola alur pemesanan dan pembayaran QRIS. Pertama, endpoint POST /api/orders menerima request OrderRequest dari aplikasi yang berisi total harga dan daftar items yang dipesan. Endpoint ini melakukan validasi data, kemudian menyimpan order baru ke tabel orders dengan generate ID unik (format: "ORD-YYYYMMDD-XXXXX"), menyimpan detail setiap item ke tabel order_details, dan mengembalikan OrderResponse berisi order_id, total, dan status awal "pending". Kedua, endpoint POST /api/create-qris menerima request QrisRequest yang berisi order_id dari response sebelumnya. Endpoint ini memanggil Midtrans API untuk generate QRIS/QR code berdasarkan order amount, menyimpan transaction details di tabel transactions, dan mengembalikan QrisResponse yang berisi QR code data (qr_string, payment_url, deep_link) serta durasi validitas QR code (15 menit). Ketiga, endpoint GET /api/order-status/{order_id} menerima order_id sebagai path parameter dan mengembalikan StatusResponse yang berisi status terkini pesanan (success/failed/expired/pending) dan pesan deskriptif status untuk ditampilkan di aplikasi. Ketiga endpoint ini bekerja bersama membentuk alur pembayaran QRIS yang lengkap dari pembuatan order, generate QR, hingga pengecekan status pembayaran.

```php
Route::get('/orders', function() {
    return response()->json([
        'success' => true,
        'orders' => [
            [
                'order_id' => 'ORD-20260110090000',
                'total' => 100000,
                'status' => 'completed',
            ],
        ]
    ]);
});

Route::post('/orders', function (Request $request) {
    $orderId = 'ORD-' . now()->format('YmdHis');
    return response()->json([
        'success' => true,
        'order_id' => $orderId,
        'message' => 'Order berhasil dibuat (mock)',
        'data' => [
            'order_id' => $orderId,
            'total' => $request->input('total', 0),
            'status' => 'pending',
            'timestamp' => now()->format('Y-m-d H:i:s'),
        ],
    ], 201);
});
Route::post('/create-qris', function (Request $request) {
    $orderId = $request->input('order_id', 'ORD-UNKNOWN');
    return response()->json([
        'success' => true,
        'order_id' => $orderId,
        'payment_method' => 'QRIS',
        'qris_data' => [
            'payment_url' => url("/mock-qris/{$orderId}"),
            'qr_string' => "MOCK-QRIS-{$orderId}",
            'deep_link' => "mockqris://pay?order_id={$orderId}",
            'status' => 'pending',
            'expires_at' => now()->addMinutes(15)->format('Y-m-d H:i:s'),
        ],
    ], 201);
});

Route::get('/order-status/{order_id}', function ($order_id) {
    return response()->json([
        'success' => true,
        'order_id' => $order_id,
        'status' => 'settlement',
        'message' => 'Pembayaran berhasil (mock)',
        'updated_at' => now()->format('Y-m-d H:i:s'),
    ]);
});

Route::get('/api/order-status/{order_id}', function ($order_id) {
    $allStatus = ['success', 'failed', 'expired', 'pending'];
    $status = $allStatus[array_rand($allStatus)];

    return response()->json([
        'success'   => true,
        'order_id'  => $order_id,
        'status'    => $status,
        'message'   => match ($status) {
            'success' => 'Pembayaran berhasil.',
            'failed'  => 'Pembayaran gagal.',
            'expired' => 'QR sudah kedaluwarsa.',
            'pending' => 'Menunggu pembayaran.',
        },
    ]);
});
```

Gambar 16. Service yang digunakan

3. Fitur Prediksi menggunakan Linear Regression

Fitur ini digunakan untuk memproyeksikan kebutuhan stok berdasarkan data penjualan historis. Algoritma yang digunakan:

- Input: Data penjualan per hari/periode (X= hari, Y= qty terjual)
- Proses : Hitung slope (m) dan itercept (b) dari rumus: Y = m*X+b
- Output: Prediksi qty untuk periode berikutnya

```
m = (n*Σ(X*Y) - Σ(X)*Σ(Y)) / (n*Σ(X²) - (Σ(X))²)
b = (Σ(Y) - m*Σ(X)) / n
prediksi_Y = m * X_baru + b
```

Gambar 17. Rumus Linear Regression

4. Hasil Pembuatan Aplikasi

Aplikasi yang dikembangkan dirancang untuk membantu proses pengelolaan transaksi penjualan, pencatatan produk serta pengelolaan data stok secara terkomputerisasi. Aplikasi ini dibuat sebagai salah satu solusi atas permasalahan yang sering terjadi saat pencatatan secara manual.

Saat aplikasi digunakan sistem menditeksi auntetikasi login dimana memberikan dampak positif terhadap akuntabilitas sistem karena setiap transaksi dapat ditelusuri berdasarkan pengguna yang melakukan aktivitas tersebut.

Dengan adanya penggunaan aplikasi ini dapat membantu pengelola memantau seluruh transaksi yang terdapat di dalamnya secara *real time*. Berdasarkan implementasi yang telah dilakukan sehingga menghasilkan aplikasi POS berbasis Android dengan pemesanan QR Code dan analisis prrediksi stok menggunakan algoritma Linear Regression.

a. Halaman Kelola Barang

Flow “admin tambah barang” pada tampilan ini dapat dijelaskan sebagai proses ketika admin menambahkan barang baru sekaligus menangani kondisi jika barang yang dimasukkan ternyata sudah ada di sistem. Prosesnya melibatkan pengisian form, pengecekan ke database, konfirmasi restock, dan pembaruan daftar barang pada halaman kelola barang.

Kondisi awal :

1) Admin sudah login dan berada pada halaman “Kelola Barang” di aplikasi kasir/inventory.
2) Sistem menampilkan form tambah barang berisi field Nama Barang, Harga, dan Stok, serta daftar kartu barang yang sudah tersimpan (misalnya teh, susu, kopi).

Langkah input data :

1) Admin menekan tombol “Tambah” sehingga sistem menampilkan form tambah barang jika sebelumnya belum tampil penuh.
2) Admin mengisi Nama Barang (contoh: kopi), Harga (contoh: 10000), dan Stok awal yang ingin ditambahkan (contoh: 10), lalu menekan tombol Tambah untuk mengirim data ke sistem.

Proses validasi di sistem :

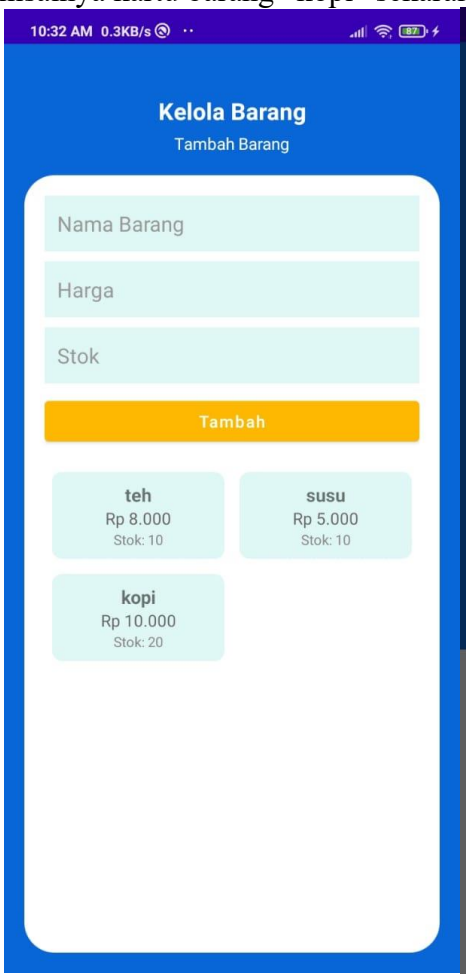
1) Setelah tombol Tambah ditekan, sistem melakukan pengecekan apakah nama barang yang dimasukkan sudah ada pada database barang.
2) Jika nama barang belum terdaftar, sistem langsung menyimpan data sebagai barang baru dengan stok sesuai input dan memperbarui tampilan daftar barang.

Barang sudah ada (restock) :

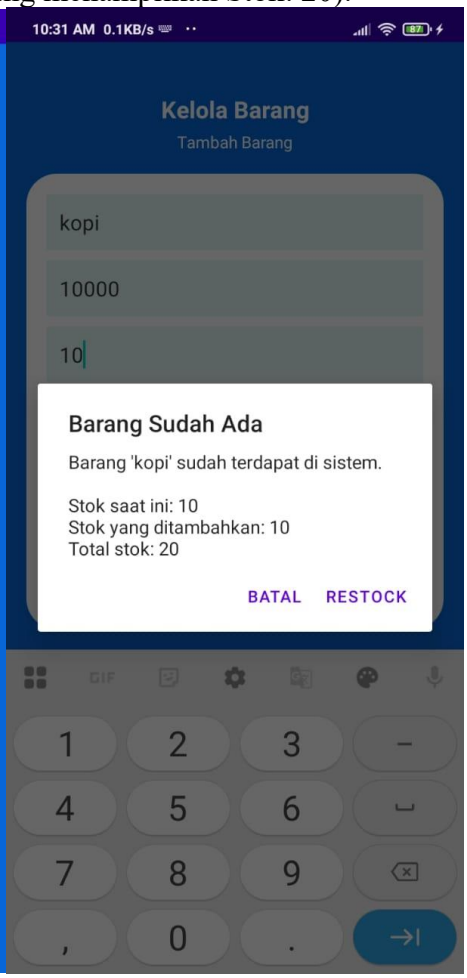
1) Jika nama barang yang dimasukkan sudah ada (misalnya “kopi”), sistem tidak langsung menambah data baru, tetapi menampilkan dialog konfirmasi “Barang Sudah Ada”.
2) Dialog tersebut menjelaskan bahwa barang dengan nama tersebut sudah terdapat di sistem dan menampilkan informasi stok saat ini, stok yang akan ditambahkan, serta total stok setelah penambahan (contoh: stok saat ini 10, stok yang ditambahkan 10, total stok 20).

Keputusan admin dan hasil akhir :

1) Pada dialog, admin diberi dua pilihan: “Batal” untuk membatalkan proses jika data dirasa tidak sesuai, atau “Restock” untuk menyetujui penambahan stok barang yang sudah ada.
2) Jika admin memilih “Restock”, sistem memperbarui nilai stok barang di database menjadi total stok baru, menutup dialog, mengosongkan form input, dan menampilkan kembali daftar barang dengan stok yang telah terupdate (misalnya kartu barang “kopi” sekarang menampilkan Stok: 20).



Gambar 18.
Halaman Admin tambah barang



Gambar 19.
Admin tambah barang

b. Flow User Beli

Flow ini menggambarkan proses user melakukan pembelian menu melalui aplikasi POS mulai dari scan QR sampai pembayaran selesai. Alurnya terdiri dari tiga bagian utama: akses ke halaman user, proses pemesanan, dan penyelesaian transaksi.Akses ke menu pembelian :

1) Setelah login, user berada pada halaman "Point Of Sales User" yang menampilkan dua tombol: "Scan QR" dan "Logout".
2) User menekan tombol "Scan QR" sehingga aplikasi membuka tampilan kamera dengan instruksi "Arahkan ke QR code" untuk memindai kode QR yang disediakan.

Hasil scan dan tampilan order :

1) Ketika QR berhasil discan, aplikasi mengarahkan user ke halaman "Order Menu" yang berisi daftar menu yang dapat dipilih (misalnya teh, susu, kopi beserta harga).
2) User memilih menu yang diinginkan sehingga item tersebut masuk ke bagian "Pesanan (Draft)" lengkap dengan nama barang, jumlah, harga satuan, dan subtotal per item.

Pengelolaan pesanan draft :

1) Pada setiap item di bagian Pesanan (Draft), user dapat mengurangi jumlah dengan tombol "–", menambah jumlah dengan tombol "+", atau menghapus item dari pesanan dengan tombol "X".
2) Sistem otomatis menghitung dan menampilkan total harga pesanan di bagian bawah (contoh Total: Rp 15.000) sesuai perubahan jumlah item.

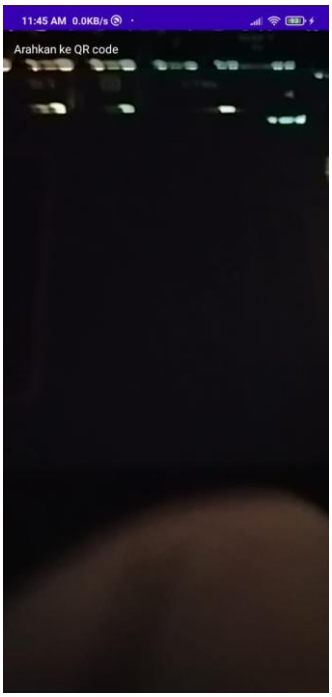
Pemilihan metode pembayaran :

1) Setelah pesanan sesuai, user memilih metode pembayaran dengan opsi radio button, misalnya "Tunai" atau "Non-Tunai".
2) Pemilihan metode ini menentukan cara pembayaran yang akan diproses kasir (misalnya pembayaran cash langsung atau via QRIS/e-wallet).

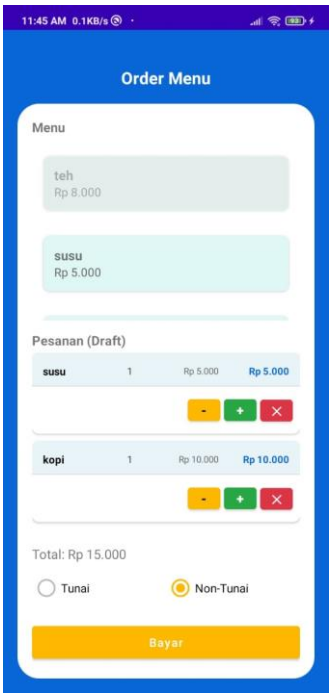
Penyelesaian transaksi :

1) User menekan tombol "Bayar" sehingga aplikasi mengirim data pesanan dan metode pembayaran ke sistem POS untuk dicatat sebagai transaksi penjualan.
2) Sistem kemudian memproses pembayaran, mengurangi stok barang sesuai jumlah yang dibeli, dan menyimpan data transaksi sebagai dasar pembuatan struk atau laporan penjualan.

Gambar 20.
Dashboard User



Gambar 21.
User Scan



Gambar 22.
User Input

c. Flow Terima Pesanan, Safety Stok, Prediksi Stok, Notifikasi

Flow ini dimulai dari admin menerima pesanan masuk dari user, memantau stok dan safety stock, hingga menyelesaikan proses dengan pembaruan riwayat transaksi. Admin berperan dalam konfirmasi pesanan, validasi stok, dan pembayaran untuk memastikan transaksi lancar.

Halaman dashboard admin :

1) Admin login ke aplikasi POS dan melihat halaman utama dengan menu: Kelola Barang, Prediksi Stok, Riwayat Transaksi, Safety Stock, Generate QR, Pesanan Masuk, dan Logout.
2) Menu "Pesanan Masuk" menampilkan daftar order pending dari user, masing-masing dengan nomor order, metode pembayaran (QRIS/Cash), status (PENDING/SUCCESS), detail item, qty, harga, subtotal, dan tombol "KONFIRM".

Monitoring safety stock :

1) Admin menekan "Safety Stock" untuk melihat daftar barang dengan status stok: hijau (aman), kuning (batas), merah (WAN/habis).
2) Tampilan menunjukkan nama barang, stok saat ini, safety stock minimum, dan status warna (contoh: kopi stok 15 safety 5 WAN; susu stok 5 safety 15 WAN).

Notifikasi stok hampir habis :

1) Ketika stok mencapai level kritis, sistem menampilkan notifikasi "Stok hampir habis" dengan detail barang (contoh: teh sisa 2 item) dan tombol untuk debug atau tindakan lanjutan.
2) Admin dapat menekan "X" untuk tutup notifikasi atau ambil aksi seperti restock melalui menu Kelola Barang.

Konfirmasi pesanan masuk :

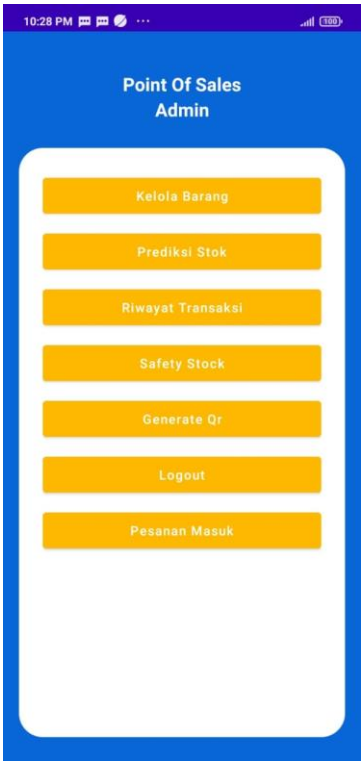
1) Admin membuka “Pesanan Masuk” dan memilih order pending (contoh Order #6 Cash PENDING atau Order #8 QRIS SUCCESS), lalu menekan tombol “KONFIRM” untuk mengonfirmasi pesanan.
2) Setelah konfirmasi, status berubah menjadi SUCCESS, stok barang dikurangi otomatis sesuai qty pesanan, dan order dipindahkan ke riwayat.

Riwayat dan detail transaksi :

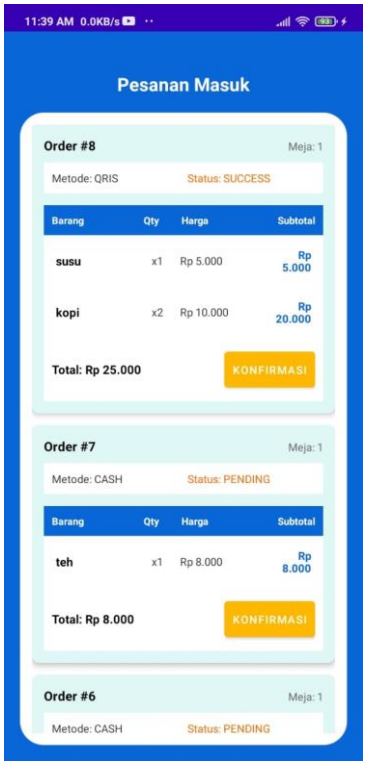
1) Admin menekan “Riwayat Transaksi” untuk melihat daftar transaksi hari ini beserta tanggal, waktu, dan total nominal (contoh: 12 Jan Rp 23.000).
2) Memilih item riwayat membuka “Detail Riwayat” dengan breakdown: metode (QRIS/Cash), item (teh 1x Rp8.000, susu 1x Rp5.000, kopi 1x Rp10.000), subtotal, dan total.

Prediksi dan menu terjual :

1) Menu “Prediksi Stok” menampilkan daftar menu dengan terjual Rp dan stok tersisa (contoh: kopi terjual Rp10.000 stok 2; susu terjual Rp10.000 stok 5).
2) Data ini membantu admin merencanakan restock berdasarkan penjualan terkini dan prediksi kebutuhan.



Gambar 23.
Dashboard Admin



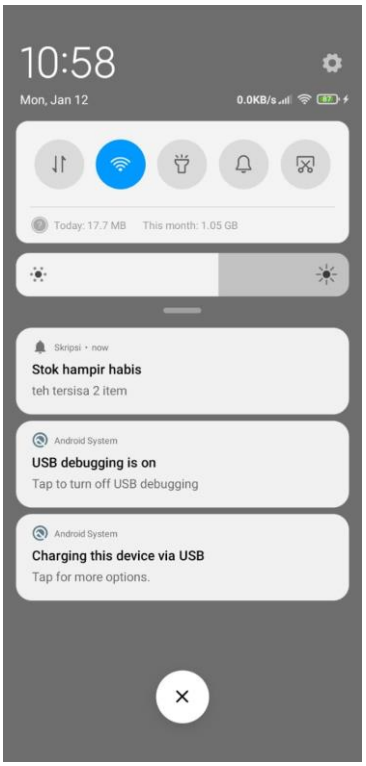
Gambar 24.
Admin terima pesanan

Gambar 25.
Halaman riwayat transaksi

Gambar 26.
Detail riwayat transaksi

Gambar 27.
Halaman prediksi stok

Gambar 28.
Safety stok



Gambar 29 admin terima notifikasi

KESIMPULAN

Berdasarkan hasil perancangan, pembuatan, dan pembahasan aplikasi Point of Sales (POS) berbasis Android yang telah dilakukan, maka dapat ditarik beberapa kesimpulan sebagai berikut:

1. Aplikasi Point of Sales (POS) berbasis Android berhasil dirancang dan diimplementasikan sesuai dengan kebutuhan pengguna, khsusnya dalam membantu proses pengelolaan transaksi penjualan, pencatatan produk, serta pengelolaan data stok secara terkomputerisasi.
2. Penerapan aplikasi POS berbasis Android mampu mengatasi permasalahan yang terdapat pada sistem pencatatan manual, seperti kesalahan perhitungan transaksi, duplikasi data, serta keterlambatan dalam pembuatan laporan penjualan.
3. Aplikasi yang dikembangkan telah dilengkapi dengan fitur-fitur utama, antara lain manajemen pengguna, pengelolaan data produk, transaksi penjualan, laporan penjualan, dan pengelolaan stok, yang seluruhnya dapat berjalan dengan baik berdasarkan hasil pengujian fungsional.
4. Penggunaan aplikasi ini dapat meningkatkan efisiensi dan efektivitas kerja pengguna, karena proses transaksi dan pencatatan data dilakukan secara otomatis oleh sistem, sehingga dapat menghemat waktu dan meminimalkan kesalahan yang disebabkan oleh faktor manusia (human error).
5. Laporan penjualan yang dihasilkan oleh aplikasi dapat membantu pemilik usaha dalam memantau kinerja penjualan serta mendukung pengambilan keputusan yang lebih tepat dan berbasis data.

6. Secara keseluruhan, aplikasi POS berbasis Android yang dikembangkan dinilai layak untuk diterapkan pada usaha kecil dan menengah sebagai solusi sistem informasi penjualan yang praktis, mudah digunakan, dan efisien.

DAFTAR PUSTAKA

1. Hamdanah, Fitria H dan Fitrianah, D. 2021. Analisis Performansi Algoritma Linear Regression dengan Generalized Linear Model Untuk Prediksi Penjualan Pada Usaha Mikro, Kecil, dan Menengah. *Janapati*. Volume 10 Nomor 1
2. Ervnisari, Y. Putri dan Muhamad, Koyimatu dan Kristine, A. Simanjuntak. Perancangan Sistem Pemesanan Makanan dan Minuman Menggunakan *QR Code* Berbasis *Website* pada *Cafe* Sudut Temu. *Jurnal Inovasi Kewirausahaan.* Volume 1. No 3 Oktober 2024. 26-46.
3. Athallah, Ariq dan M Fadhil, Bayhaqi dan Rizza, Indah M. M. 2025. Toko Pintar: Aplikasi Point of Sale dengan Fitur Prediksi Penjualan Berbasis Machine Learning untuk UMKM. *e-Proceeding of Applied Science.* Vol 11 no 4 Agustus 2025. 729
4. Suryawan, M Arif dan Ery, M. Hasiri dan Kensino, Ode. 2020. Implementasi Sistem QR Code dan Barcode Pada Sistem Pembayaran di Toko Perbelanjaan Menggunakan Aplikasi Android. *Jurnal Informatika.* Volume 9 no 2 Desember 2020.
5. Eunice Keith. 2023. Optimizing Inventory Managament trough Advanced Forecasting Techniques in Supply Chains. *European Jurnal of Supply Chain Management.* Vol 1 No 1. 22-30.
6. Damayanti, A. & Marleny, Fink Dona & Ningrum, Ayu Ahadi. Implementasi Regresi Linear Berganda Untuk Prediksi Penjualan Pada PT Trimandiri Sarana Propetindo Banjarmasin. *JITET (Jurnal Informatika dan Teknik Elektro Terapan).* Volume 13 No 3.
7. Pradipta, Rishon Krischnan Tegar & Widiono, S. 2024. Perancangan Aplikasi Point of Sales Berbasis Android Pada Komedi Café. *DJtechno: Jurnal Teknologi Informasi.* Vol. 5, No. 3 Desember 2024. 10.46576/djtechno
8. Hinostroza, E. & dkk. Linear Regression Model to Predict the Use of Artificial Intelligence in Experimental Science Students. *International Electronic Journal of Mathematics*. 2025.
9. Rasuli, A & dkk. 2025. Prediksi Stok Produk Susu Pada PT Greenfileds Dairy Indonesia Menggunakan Metode Regresi Linear. *CONTEN: Computer and Network Technology.* Vol. 5, No. 1 Juni 2025. 23-28.
10. Syam, Muhammad L. & Erdisna. Sistem Informasi Stok Barang Menggunakan QR Code Berbasis Android. *Jurnal Informatika Ekonomi Bisnis.* Vol. 4, No. 1.
11. Martanto, Ichlas A. & Dikananda, A. R & Mulyawan. 2025. Implementasi Algoritma Regresi Linear Untuk Model Prediksi Penjualan di Toko Amanda Brownies. *JITET (Jurnal Informatikan dan Teknik Elektro Terapan)*. Vol. 3 No. 2.
12. Adrian, A. dan Setyawati, E. 2024. Aplikasi Mobile Point of Sales (POS) Pada Usaha Mikro Kecil Menengah (UMKM) Retro di SMK Kesatrian

Purwokerto Menggunakan Barcode Untuk Membaca Identitas Barang Berbasis Android. J*otika Journal in Education*. Vol. 4, No. 1 Agustus 2024.

13. Verdianto, R. dan Dwi, H. dan Eko, P. 2025. Pengembangan Aplikasi Point of Sales untuk Prediksi Penjualan Harian Usaha Minuman Menggunakan Algoritma Random Forest Regression. *Infotek: Jurnal Informatika dan Teknologi.* Vol. 8, No. 1 Januari 2025.
14. Fatimah, D dan Wijaya, Herry D. 2022. enerapan QR Code pada Website E-commerce PT. Bravo Satria Perkasa dengan Algoritma Reed-Solomon Code dan Regression Linear. *Faktor Exacta.* Vol. 15, No. 1, March 2022. 10-27.
15. Pratiwi, Nabilah A. dan Triayudi, A. dan Endah, T. 2021. Analysis and Design of Mobile Web-Based Menu E-Order Systems Using The Pieces Method (Case Study: Cafe 50/50 Coffee). *Jurnal Riset Informatika*. Vol. 4, No. 1 Desember 2021.
16. Sulistyo Dwi S. 2024. Penerapan Sistem Point of Sales Berbasis Android Untuk Peningkatan Kinerja Usaha. *Infotek : Jurnal Informatika dan Teknologi*. Vol. 7 No. 1, Januari 2024. 195-204. 10.29408/jit.v7i1.23934
17. Pamungkas, G. dan Yuliansyah, H. 2017. Rancangan Bangun Aplikasi Kasir *Portable* Android Point of Sales (POS) yang Terintegrasi dengan Printer di Kafe Kantin S15 Yogyakarta. *Jurnal Sarjana Teknik Informatika.* Volume 05 Nomor 3.
18. Hanif. A. dan Agus, Suhendar dan Rr. Hajar P., Sejati. 2024. Design and Development of an Android-Based Point of Sale Application: A Case Study of Warung Dapur Barokah, Pangkalpinang. *International Journal Software Engineering and Computer Science (IJSECS)*. Vol. 4 No. 1. 293-300.
19. Faaldiansyah, Reza dan Utami, W. Sri. 2024. Pengembangan Aplikasi Pemantau Stok Barang Menggunakan QR Code Secara Real-Time Tracking dengan Metode Prototype Berbasis Android. *ILKOMNIKA: Journal of Computer Science and Applied Informatics.* Vol. 6, No. 3.
20. Fatimah, Dian dan Wijaya, H. Derajad. 2022. Penerapan QR Code pada Website E-commerce PT. Bravo Satria Perkasa dengan Algoritma Reed-Solomon Code dan Regression Linear. *Faktor Exacta.* Vol. 15, No. 1. 10-27.