IMPLEMENTASI FRAMEWORK FLASK PADA PERANCANGAN SISTEM POINT OF SALE (POS) UNTUK MULTI CABANG BENGKEL PRATAMA MOTOR

Christopher Arvin Tungady, Sri Winarso Martyas Edi
Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana
Jalan Diponegoro No.52-60 Telp.(0298), Salatiga, 50711, Indonesia
672022279@student.uksw.edu

ABSTRAK

Perkembangan teknologi menuntut digitalisasi manajemen operasional pada sektor *UMKM*. Namun, bengkel dengan banyak cabang sering menghadapi kendala dalam sinkronisasi pencatatan transaksi dan manajemen stok antar-cabang yang tidak efisien. Penelitian ini bertujuan untuk merancang dan membangun sistem *Point of Sale (POS)* berbasis web guna mengatasi masalah tersebut. Metode penelitian yang digunakan adalah studi kasus dengan pendekatan kualitatif pada Bengkel Pratama Motor menggunakan data simulasi. Sistem dikembangkan menggunakan bahasa *Python dengan framework Flask*, *MongoDB* sebagai database *NoSQL*, dan *Redis* untuk penyimpanan session. Hasil implementasi menunjukkan bahwa sistem berhasil mencakup fitur *login multi-role*, pengelolaan stok per cabang, transaksi kasir, serta riwayat penjualan yang dapat *difilter* berdasarkan tanggal dan operator. Berdasarkan pengujian fungsional, sistem berjalan sesuai kebutuhan dan diharapkan menjadi solusi digitalisasi *POS* yang efektif bagi *UMKM* multi-cabang.

Kata kunci : *Point of Sale, Flask, MongoDB, Redis, RESTful API*

1. PENDAHULUAN

Perkembangan teknologi informasi seperti web saat ini menjadi sebuah sarana yang dimanfaatkan untuk mendukung operasional bisnis, Salah satu aspek yang krusial dalam kegiatan *UMKM* adalah proses transaksi penjualan dan pengelolaan stok barang[1]. Hanya sekitar 39,7 % *UMKM* di Indonesia yang telah terkoneksi digital melalui *website*, *e-commerce*, atau *platform online* per Juli 2024[2]. Padahal pemerintah menargetkan agar 63,7 % dari 60 juta *UMKM* sudah *on-boarding* digital berdasarkan program Bangga Buatan Indonesia pada Agustus 2022[3].

Ketidakefisienan dalam pencatatan transaksi dan ketidaksesuaian data stok dengan kondisi nyata merupakan isu umum dalam operasional bisnis ritel dan servis yang masih menggunakan sistem manual. Ketidakteraturan dalam manajemen inventori dapat menyebabkan pemborosan, keterlambatan layanan, hingga biaya tambahan akibat pemesanan ulang barang yang sebenarnya masih tersedia di gudang[4]. Selain itu, penggunaan pencatatan manual atau lembar Excel untuk pengelolaan stok dan transaksi sangat rentan terhadap kesalahan perhitungan dan manipulasi harga, yang berdampak pada kerugian baik bagi pelanggan maupun pemilik usaha[5]. Tidak adanya sistem monitoring yang terpusat juga memperburuk situasi, terutama bagi pemilik yang memiliki lebih dari satu cabang, karena ketidaksinkronan data antar lokasi dapat menyulitkan pengambilan keputusan[6].

Kondisi tersebut terjadi pada Bengkel Pratama Motor, sebuah usaha jasa servis dan penjualan komponen kendaraan yang memiliki dua cabang. Saat ini, sistem transaksi pada bengkel tersebut masih dilakukan secara manual. Ketiadaan sistem yang transparan membuka celah terjadinya ketidaksesuaian data audit stok dengan kondisi riil di lapangan. Selain itu, keterbatasan pemilik dalam melakukan pengawasan fisik secara simultan di kedua lokasi cabang memperbesar risiko kerugian finansial akibat kesalahan pencatatan nota transaksi.

Untuk mengatasi permasalahan tersebut, penelitian ini bertujuan untuk merancang dan membangun Sistem Informasi *Point of Sales (POS)* berbasis web yang terintegrasi. Sistem ini diharapkan dapat mengoptimalkan proses pencatatan stok, mengamankan transaksi penjualan, serta memfasilitasi monitoring multi-cabang secara *real-time* bagi pemilik Bengkel Pratama Motor.

Rencana penyelesaian masalah dilakukan dengan mengembangkan arsitektur sistem *POS* terpusat menggunakan bahasa pemrograman *Python* dan *framework Flask* yang ringan untuk kebutuhan performa web. Guna mengatasi masalah sinkronisasi data multi-cabang dan fleksibilitas struktur data komponen bengkel, digunakan *MongoDB* sebagai *database NoSQL*. Selain itu, Redis diintegrasikan sebagai penyimpanan *session* berbasis memori untuk menjamin kecepatan respons dan keamanan akses pengguna berdasarkan peran. Melalui pendekatan teknologi ini, celah manipulasi harga dapat ditekan dan pengawasan multi-cabang dapat dilakukan dalam satu platform tunggal.

2. TINJAUAN PUSTAKA

Point of Sale (POS) adalah sistem yang digunakan untuk mencatat transaksi penjualan secara digital, menggantikan metode manual. *POS* modern tidak hanya mencatat penjualan, tetapi juga dapat mengelola stok, laporan penjualan, serta terintegrasi dengan sistem pembayaran dan manajemen cabang. *POS* berbasis *web* memberikan keunggulan dari sisi aksesibilitas, pemantauan *real-time*, dan efisiensi operasional [2]. *POS* juga penting bagi *UMKM* sebagai sarana transformasi digital [3].

Penggunaan *MongoDB* sebagai database non-relasional menawarkan fleksibilitas dalam pengelolaan data dinamis dan tidak terstruktur[7], sangat cocok untuk aplikasi *POS* berbasis *web*. *MongoDB* adalah database *NoSQL(Not Only SQL)* yang dirancang untuk menyimpan data dalam bentuk dokumen *JavaScript object notation* (*JSON*). *MongoDB* berbasis dokumen yang berarti tidak memiliki kolom, baris dan tabel namun menggunakan dokumen dan koleksi[8].

Python adalah bahasa pemrograman berorientasi objek, multifungsi, kaya fitur, dan tersusun rapi. *Python* sangat mudah dipelajari, bahkan bagi pemula, dan memiliki komunitas yang besar serta aktif[9]. *Python* juga bersifat *open-source*, sehingga dapat diakses gratis dan mudah diintegrasikan dengan berbagai *platform* serta perangkat lunak lain. Fokus pada keterbacaan kode membuat sintaks *Python* mudah dipahami dan digunakan untuk berbagai tujuan, termasuk analisis data dan visualisasi data[10]. Selain itu, aplikasi *Python* yang terus berkembang menunjukkan peningkatan kebutuhan bahasa ini di berbagai bidang kerja. Dari pemula hingga tingkat lanjut, *Python* tetap efisien dalam menyelesaikan masalah kompleks tanpa membebani pengembang dengan kompleksitas implementasi, berkat tingkat abstraksi yang tinggi dan ketersediaan pustaka standar serta pihak ketiga yang melimpah[9].

Flask adalah sebuah *web framework* yang ditulis dengan bahasa *Python* dan tergolong sebagai jenis *microframework*. *Flask* termasuk *microframework* karena tidak memerlukan alat atau pustaka tertentu secara default, sehingga hanya komponen yang dibutuhkan saja yang dipasang, membuatnya ringan dan fleksibel[11]. *Flask* berfungsi sebagai kerangka kerja aplikasi dan tampilan dari suatu *website*. Dengan menggunakan *Flask* dan bahasa *Python*, pengembang dapat membuat sebuah *website* yang terstruktur dan dapat mengatur perilaku suatu *website* dengan lebih mudah[12]. Meskipun demikian, *Flask* tetap memiliki kemampuan fungsionalitas yang tinggi dan skalabilitas yang baik karena dapat diperluas dengan ekstensi pihak ketiga.

Redis adalah singkatan dari *Remote Dictionary Server*, yaitu penyimpanan data nilai utama di dalam memori yang sangat cepat, bersifat *open source*, dan digunakan sebagai *database, cache, message broker*, dan antrian[13]. Redis merupakan basis data berbasis *key-value* yang diciptakan oleh Salvatore Sanfilippo dan dirilis pada tahun 2009. *Redis* dirancang untuk kinerja tinggi dengan waktu respons sub-milidetik dan mendukung berbagai struktur data seperti *string, hash, list, set*, dan *sorted set*[14][15]. *Redis* adalah sistem penyimpanan data in-memory yang memungkinkan penyimpanan data secara cepat dan akses yang efisien, sangat berguna untuk mempercepat operasi relasional dan pengambilan data pada aplikasi dengan volume data besar[15]. *Redis* juga memiliki mekanisme *caching* yang kuat yang dapat mengurangi beban server dan mempercepat akses data, sehingga meningkatkan responsivitas aplikasi [16].

2.1. Penelitian Terdahulu

Penelitian berjudul *"Perancangan Aplikasi Web Manajemen Data Produk Bisnis Perhiasan Berbasis Flask dan MongoDB"* oleh Usman dan Martyas Edi[8] mengembangkan aplikasi manajemen produk berbasis *website* untuk bisnis perhiasan dengan memanfaatkan *framework Flask* dan *database MongoDB*. Penelitian ini bertujuan untuk mengatasi tantangan dalam pengelolaan stok, pemasaran, dan pemesanan produk yang sering berubah akibat dinamika tren di industri perhiasan.

Selanjutnya, penelitian berjudul *"Sistem Informasi Kasir Toko Kembar menggunakan Framework Webix dan MongoDB"* oleh Kamaz dan Beeh [17] membahas pengembangan aplikasi kasir berbasis *website* untuk mengatasi proses transaksi manual di Toko Kembar. Dalam penelitian ini, digunakan metode *Research and Development (R&D)* dan teknologi utama berupa *framework Webix* sebagai antarmuka pengguna, *Flask* sebagai *web framework* berbasis *Python*, serta *MongoDB* sebagai basis data *NoSQL* yang mendukung penyimpanan data secara fleksibel dan cepat.

Penelitian berjudul *"Implementasi Arsitektur Microservices pada Aplikasi Point of Sale Toko Flyover Stiker"* oleh Saidah dan kawan-kawan [18]membahas pengembangan aplikasi *POS* berbasis web dengan pendekatan arsitektur *microservices*. Tujuannya adalah untuk mengatasi kompleksitas pengembangan sistem *POS* berbasis monolitik yang menyulitkan perawatan dan pengembangan fitur baru. Sistem dikembangkan menggunakan metode *Software Development Life Cycle (SDLC)* dengan lima tahapan: analisis, perancangan, implementasi, pengujian, dan pemeliharaan. Teknologi yang digunakan adalah *JavaScript* dengan *Express.js* untuk *backend* dan *React.js* untuk frontend, serta *Cypress* untuk *automated end-to-end testing*.

Perbedaan pada penelitian ini adalah penggunaan *MongoDB* dengan skema fleksibel untuk mengelola data stok, transaksi, dan pengguna berdasarkan berdasarkan cabang melalui *branch_Id*, sehingga setiap cabang memiliki ruang data tersendiri di dalam satu sistem terpusat, memungkinkan sistem untuk mendukung multi cabang tanpa perlu membuat database terpisah untuk tiap lokasi.

Kebaruan penelitian ini terletak pada penggunaan *Framework Flask* sebagai *backend modular* dalam sistem *POS* multi cabang yang mengadopsi arsitektur *RESTful*, berbeda dari penelitian sebelumnya yang menerapkan *Flask* untuk manajemen data sederhana tanpa pemisahan antar cabang.

3. METODE PENELITIAN

3.1. Penerapan Metode

Penelitian ini menggunakan metode kualitatif dan studi kasus, dengan bengkel milik keluarga peneliti sebagai objek implementasi sistem. Data yang digunakan merupakan data simulasi (*dummy*) yang merepresentasikan transaksi harian toko, seperti data produk, stok, dan transaksi kasir. Tujuannya adalah merancang dan membangun sistem *Point of Sale (POS)* berbasis web yang dapat mengelola transaksi dan stok barang multi cabang secara efisien menggunakan teknologi *RESTful API* dan *MongoDB*. Dalam proses melakukan penelitian, peneliti menggunakan skema berikut:

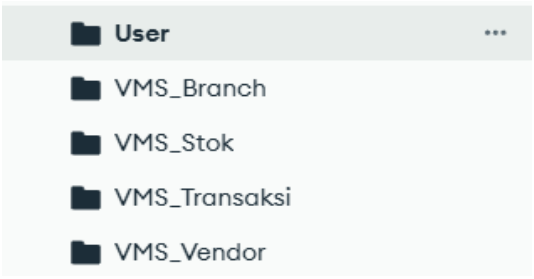
Gambar 1. Alur Penerapan Metode

3.2. Analisis Kebutuhan

Memahami segala kebutuhan *client* dan kebutuhan untuk pembuatan aplikasi Point of Sale dengan Arsitektur *Microservices* serta dilakukan pencarian informasi terkait Arsitektur *Microservices* dengan cara studi pustaka seperti mengumpulkan jurnal dan artikel.

3.3. Pengumpulan Data

Dalam penelitian ini, data transaksi, stok barang, dan informasi pengguna yang digunakan untuk implementasi dan pengujian sistem bersifat simulasi (*dummy*). Data ini dibuat untuk meniru kondisi nyata dan digunakan sebagai dasar dalam pengujian fungsionalitas sistem.



Gambar 2. Collection yang digunakan sebagai dataset

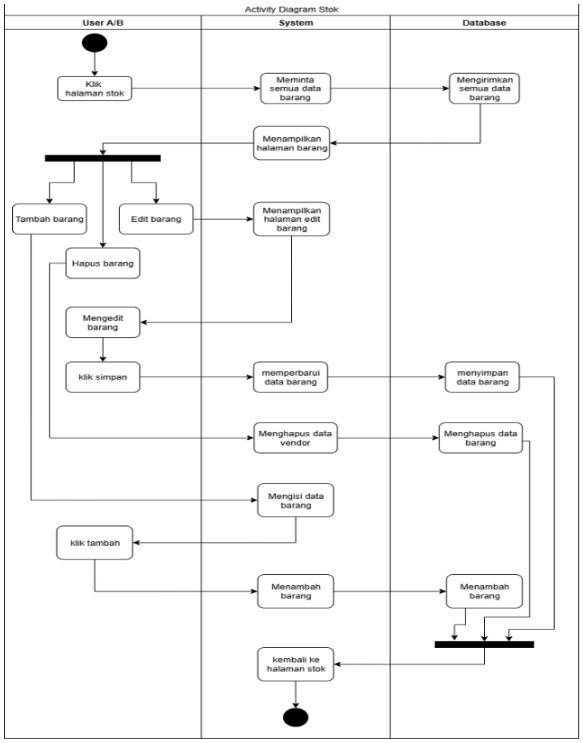
Pada gambar 2 menunjukkan beberapa *Collection* dataset yang digunakan pada penlitian ini meliputi User, VMS_Branch, VMS_Stok, VMS_Transaksi, VMS_Vendor. “User” berisi nama, alamat, nomor hp, serta id cabang. “VMS_Stok” berisi stok yang tersedia. “VMS_Branch” berisi detail cabang. “VMS_Transaksi” berisi riwayat transaksi yang telah dilakukan. “VMS_Vendor” berisi detail *Vendor* yang bekerjasama.

3.4. Perancangan Sistem

Setelah semua informasi yang dibutuhkan sudah didapat, langkah berikutnya adalah merancang sistem agar bisa memberikan *output* yang sesuai dengan kebutuhan client. Proses perancangan sistem dapat dilakukan dengan memanfaatkan *Unified Modeling Language (UML)*, suatu metode yang memanfaatkan representasi visual untuk mendokumentasikan, menetapkan spesifikasi, dan mengembangkan perangkat lunak[19]. Jenis diagram *UML* yang diterapkan dalam penelitian ini adalah *usecase diagram, activity diagram, dan class diagram*

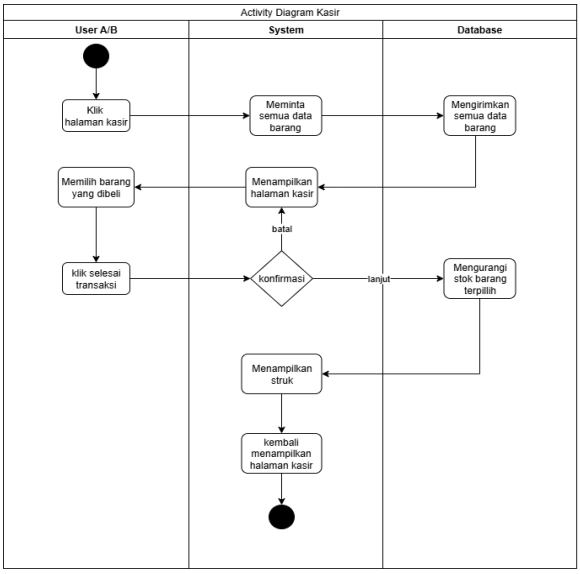
Gambar 3. *Usecase Diagram* menunjukkan akses untuk masing-masing *role*

Pada Gambar 3 Karyawan diidentifikasi sebagai *User*, *User* dari cabang A hanya bisa mengakses database dari cabang A, begitu juga sebaliknya. *Owner* sendiri diidentifikasi sebagai Admin, dapat mengakses stok, laporan, vendor, dan profil cabang dari kedua cabang sekaligus



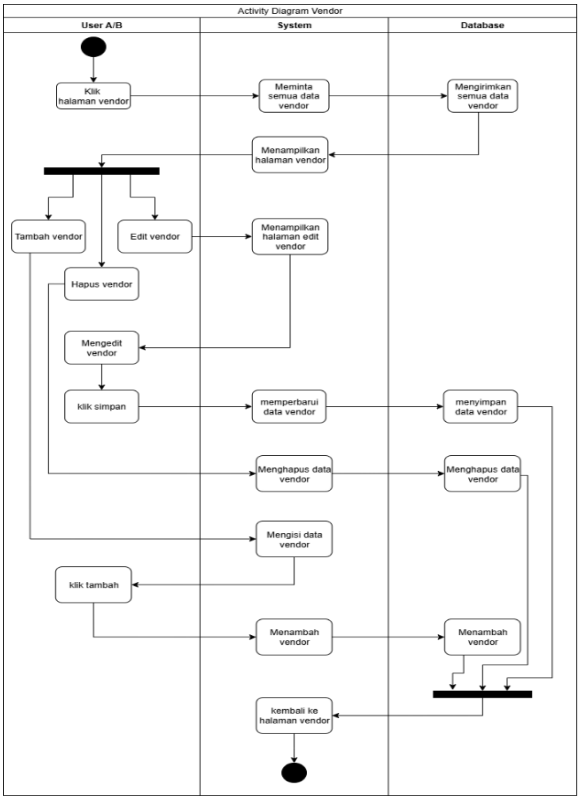
Gambar 4. *Activity Diagram* Stok

Pada Gambar 4 menunjukkan langkah-langkah aktivitas yang dapat dilakukan di halaman Stok. Pengguna dapat melihat semua barang yang tersedia di cabang dan dapat melakukan tambah, *edit*, dan hapus barang, khusus admin dapat mengganti cabang.



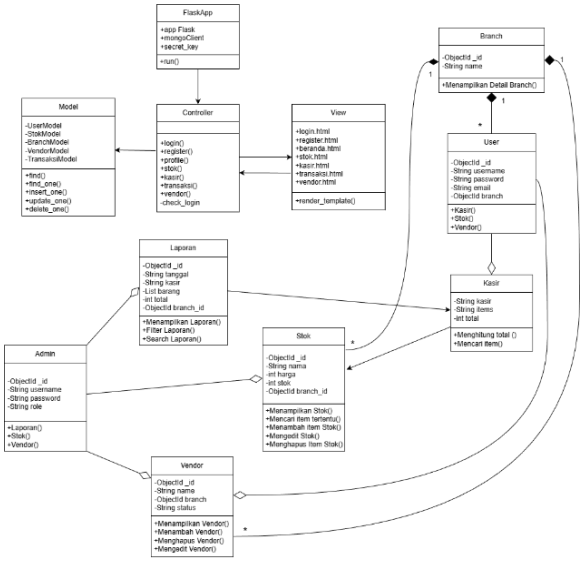
Gambar 5. *Activity Diagram* Kasir

Pada Gambar 5 menunjukkan langkah-langkah aktivitas yang bisa dilakukan pada halaman kasir. Pengguna bisa menambahkan barang yang akan dibeli, kemudian klik *button* selesai transaksi, maka akan keluar struk total pembelian beserta detailnya.



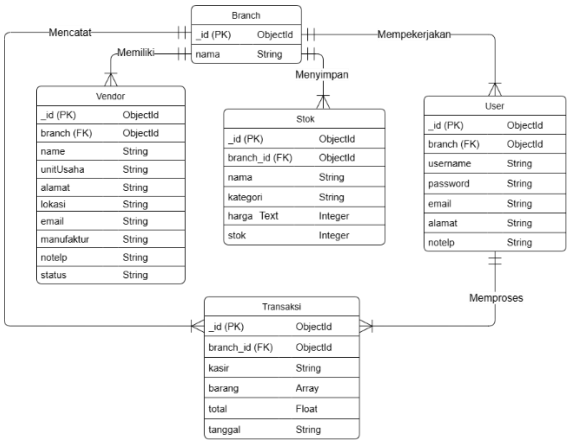
Gambar 6. *Activity Diagram* Vendor

Pada Gambar 6 menunjukkan langkah-langkah aktivitas yang bisa dilakukan pada halaman vendor. Pengguna dapat melakukan tambah, edit, dan hapus vendor, khusus untuk admin dapat mengganti cabang.



Gambar 7. *Class Diagram*

Pada Gambar 7 menggambarkan arsitektur aplikasi *Point of Sale (POS*) menggunakan pola *Model-View-Controller (MVC)* dengan *framework Flask*. Diagram ini terbagi menjadi beberapa komponen utama yang saling berinteraksi. Alur kerja *MVC* dalam aplikasi ini dimulai dari *Flask App* yang memuat semua *Controller*. Ketika *user* mengakses suatu *URL*, *Controller* menerima *request* dan menggunakan *Session Manager* untuk memvalidasi login. Jika valid, *Controller* mengakses *Model* untuk operasi data *(CRUD)* yang tersimpan di *Database MongoDB*. Setelah memproses data, *Controller* melakukan *render View* yang sesuai dan mengembalikan *response* berupa *HTML* ke *user*.



Gambar 8. Desain Database

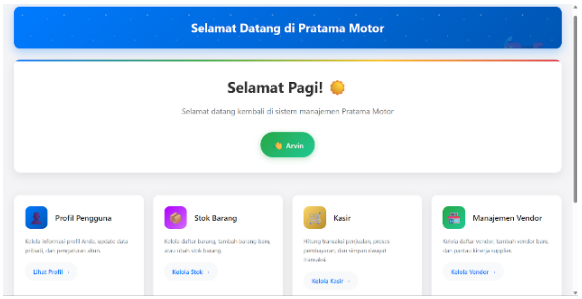
Gambar 8 merupakan desain database yang digunakan dalam aplikasi ini, setiap branch memiliki 1 atau lebih karyawan, dan karyawan memproses 1 atau lebih transaksi, branch menyimpan 1 atau lebih stok,

branch memiliki 1 atau lebih vendor, dan branch mencatat 1 atau lebih transaksi.

4. HASIL DAN PEMBAHASAN

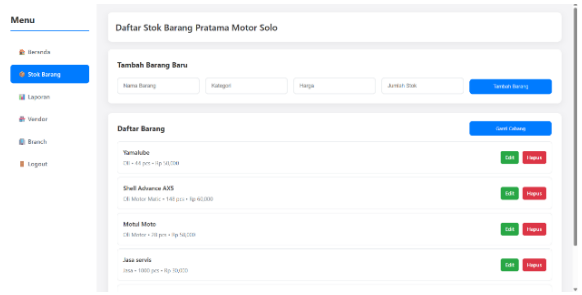
4.1. Implementasi

Melalui serangkaian langkah-langkah dalam perancangan perangkat lunak, penelitian ini berhasil menciptakan sebuah aplikasi web *Point of Sale (POS)* yang dapat mengoptimalkan proses pencatatan stok dan transaksi, serta monitoring cabang pada Bengkel Pratama Motor. Aplikasi web ini dikembangkan menggunakan bahasa pemrograman *Python*, *Framework Flask*, *Redis* dan *MongoDB*.



Gambar 8. Tampilan Halaman Utama

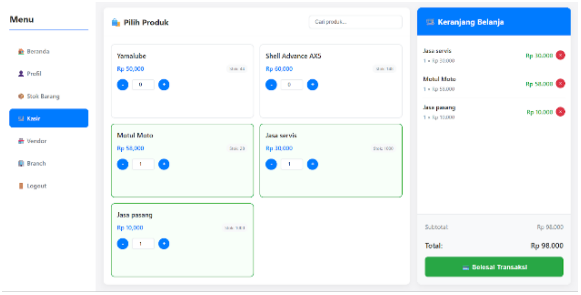
Pada gambar 8 terdapat menu Profil Pengguna, Stok Barang, Kasir, Vendor, Informasi Cabang, dan Logout. Bila pengguna menekan button yang ada di tiap menu, akan menuju halaman sesuai menu yang dituju.



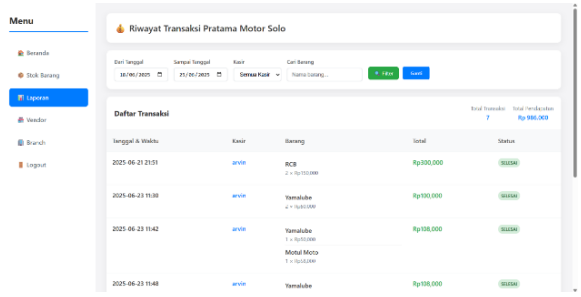
Gambar 9. Halaman Stok Barang

Pada Gambar 9 ini terdapat daftar barang yang berisi nama dan detail barang, kemudian terdapat button edit dan hapus barang. Diatas daftar barang terdapat kolom untuk input tambah barang. Pada Halaman ini *User* hanya bisa mengakses stok sesuai cabang yang dipilih pada saat register karena id_branch melekat pada *User*. Berbeda dengan *User* disini *Admin* bisa melihat stok dari kedua cabang dengan mengganti cabang.

Gambar 10 menunjukkan halaman Kasir yang berisi produk yang tersedia dan terdapat button untuk menambahkan ke keranjang. Pada kolom keranjang terdapat button untuk mengeluarkan barang dari keranjang. Pada bagian bawah terdapat button untuk menyelesaikan transaksi.

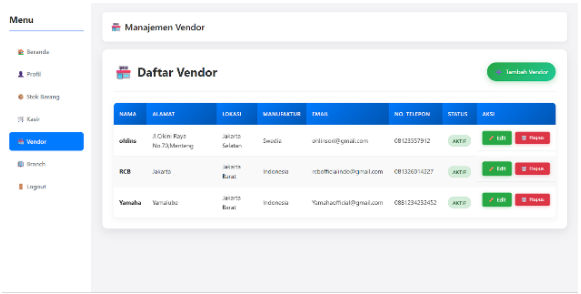


Gambar 10. Halaman Kasir



Gambar 11. Halaman Laporan

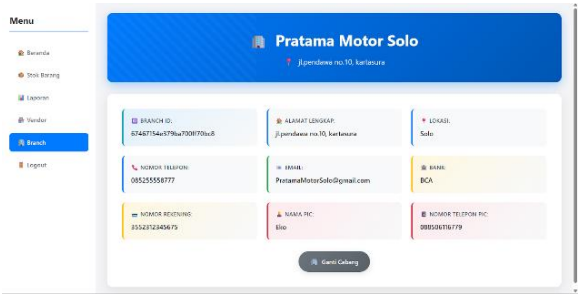
Pada Gambar 11 terdapat filter tanggal, operator kasir, dan barang tertentu. Pada daftar transaksi terdapat tanggal&waktu, operator kasir, nama barang, total, dan status. Disamping daftar transaksi terdapat total transaksi dan total pendapatan. Terdapat pula button ganti untuk mengganti tampilan menjadi riwayat transaksi cabang lain.



Gambar 12. Halaman Vendor

Pada Gambar 12 ini ditampilkan daftar vendor berupa nama, alamat, lokasi, manufaktur, email, nomor telepon, dan status. Pada kolom aksi pengguna bisa melakukan edit dan delete vendor. Terdapat pula button untuk menambah vendor di pojok kanan atas

.



Gambar 13. Halaman Branch

Pada Gambar 13 ini terdapat info seputar cabang seperti id cabang, alamat lengkap cabang, lokasi, nomor telepon cabang, email cabang, nomor rekening cabang, Nama PIC, dan nomor telepon PIC. Terdapat pula button ganti cabang supaya admin bisa mengganti tampilan menjadi tampilan cabang lain.

4.2. Pengujian

Tabel 1. Pengujian halaman beranda

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Status
1	Membuka halaman beranda setelah login	Menampilkan ringkasan data dan menu navbar dengan benar	Menampilkan ringkasan data dan menu navbar dengan benar	Valid
2	Mengklik menu navigasi	Berhasil diarahkan ke halaman yang sesuai	Berhasil diarahkan ke halaman yang sesuai	Valid

Tabel 2. Pengujian halaman stok

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Status
1	Menampilkan daftar stok barang	Tabel stok tampil dengan data yang sesuai dari database	Tabel stok tampil dengan data yang sesuai dari database	Valid
2	Mencari barang menggunakan fitur pencarian	Menampilkan data barang sesuai dengan kata kunci pencarian	Menampilkan data barang sesuai dengan kata kunci pencarian	Valid
3	Menambah barang baru (form tambah stok)	Barang baru tersimpan di database dan tampil di tabel stok	Barang baru tersimpan di database dan tampil di tabel stok	Valid
4	Mengedit data barang (halaman edit_stok)	Perubahan data barang tersimpan dan terupdate di tabel	Perubahan data barang tersimpan dan terupdate di tabel	Valid
5	Menghapus data barang	Barang hilang dari tabel dan terhapus di database	Barang hilang dari tabel dan terhapus di database	Valid

Tabel 3. Pengujian halaman kasir

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Status
1	Menambahkan barang ke keranjang	Barang masuk ke daftar belanja kasir dengan harga yang tepat	Barang masuk ke daftar belanja kasir dengan harga yang tepat	Valid
2	Mengubah kuantitas barang di keranjang	Total harga terhitung otomatis dengan benar	Total harga terhitung otomatis dengan benar	Valid
3	Memproses pembayaran (checkout)	Transaksi berhasil, stok berkurang, dan struk tercetak/tersimpan	Transaksi berhasil, stok berkurang, dan struk tercetak/tersimpan	Valid
4	Membatalkan keranjang belanja	Daftar belanja di kasir kosong	Daftar belanja di kasir kosong	Valid

Tabel 4. Pengujian halaman Laporan

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Status
1	Membuka halaman riwayat transaksi	Menampilkan daftar transaksi yang telah berhasil dilakukan	Menampilkan daftar transaksi yang telah berhasil dilakukan	Valid
2	Memfilter riwayat transaksi berdasarkan tanggal	Hanya menampilkan transaksi pada rentang tanggal yang dipilih	Hanya menampilkan transaksi pada rentang tanggal yang dipilih	Valid

Tabel 5. Pengujian halaman vendor

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Status
1	Menampilkan daftar vendor	Tabel vendor tampil dengan data yang sesuai dari database	Tabel vendor tampil dengan data yang sesuai dari database	Valid
2	Menambah vendor baru (halaman add_vendor)	Data vendor baru tersimpan dan tampil di tabel	Data vendor baru tersimpan dan tampil di tabel	Valid
3	Mengedit data vendor (halaman edit_vendor)	Perubahan tersimpan dan terupdate di tabel	Perubahan tersimpan dan terupdate di tabel	Valid
4	Menghapus data vendor	Data vendor terhapus dari sistem	Data vendor terhapus dari sistem	Valid

Tabel 6. Pengujian halaman branch

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Status
1	Menampilkan daftar cabang	Tabel cabang tampil dengan data dari database	Tabel cabang tampil dengan data dari database	Valid
2	Menambah/Mengedit cabang	Data cabang tersimpan dengan benar	Data cabang tersimpan dengan benar	Valid

5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian dan implementasi yang telah dilakukan, dapat disimpulkan bahwa sistem *Point of Sale (POS)* berbasis web yang dirancang menggunakan *framework Flask* dengan arsitektur *RESTful* dan database *MongoDB* berhasil dibangun dan berjalan sesuai tujuan. Fitur utama yang berhasil diimplementasikan meliputi pengelolaan data pengguna per cabang, manajemen stok barang berdasarkan cabang, pencatatan transaksi kasir, dan penyimpanan riwayat transaksi. Sistem juga menyediakan tampilan antarmuka berbasis web yang sederhana dan mudah digunakan. Seluruh pengujian dilakukan menggunakan data simulasi (dummy) yang merepresentasikan proses bisnis UMKM dan menunjukkan bahwa sistem berjalan secara fungsional sesuai dengan skenario yang dirancang. Dengan demikian, sistem ini telah memenuhi tujuan penelitian, yaitu merancang dan membangun sistem POS berbasis web yang dapat membantu pelaku usaha dalam mencatat penjualan, mengelola persediaan, dan memisahkan data per cabang secara efisien.

DAFTAR PUSTAKA

[1] F. Aziz, "IMPLEMENTASI SISTEM INFORMASI PENJUALAN MENGGUNAKAN METODE RAPID APPLICATION DEVELOPMENT (RAD) PADA TOKO MASTER," Advances in Computer System Innovation Journal, vol. 2, no. 2, pp. 71–77, 2024, doi: 10.51577/acsijournal.v2i2.577.

[2] A. Nugraheni, "Optimalisasi Platform Digital Memperkuat Pasar UMKM," Kompas. [Online]. Available: https://www.kompas.id/artikel/optimalisasi-platform-digital-memperkuat-pasar-umkm

[3] "Ciptakan UMKM Tangguh, Adopsi Teknologi Digital Dapat Dukungan Pemerintah," Kementerian Koordinator Bidang Perekonomian RI. [Online]. Available: https://ekon.go.id/publikasi/detail/4441/ciptakan-umkm-tangguh-adopsi-teknologi-digital-dapat-dukungan-pemerintah

[4] F. Anwar, "9 POS Integration Categories to Run Your Repair Shop Efficiency," ReapairDesk. [Online]. Available: https://blog.repairdesk.co/2022/09/23/9-pos-integration-categories-to-run-your-repair-shop-efficiently/

[5] N. Hashem, "The Hidden Costs of an Inefficient POS System-And How to Fix Them," bimpos. [Online]. Available: https://bimpos.com/blog/how-to-fix-the-hidden-costs-fees-of-an-inefficient-pos-system

[6] Zeeshandigi, "How do you handle discrepancies between POS inventory records and actual stock when dealing with parts suppliers in a wireless repair shop?," reddit. [Online]. Available: https://www.reddit.com/r/InventoryManagement/comments/1jessh1/how_do_you_handle_discrepancies_between_pos/

[7] S. Neeli, "A Comparative Analysis of SQL and NoSQL Database Management within Cloud Architectures for Mission-Critical Business Systems," International Journal of Advanced Computer Technology (IJACT), vol. 2, no. 4, pp. 140-149, 2025, doi: 10.56472/25838628/IJACT-V2I4P118.

[8] U. Syach and S. Winarso Martyas Edi, "PERANCANGAN APLIKASI WEB MANAJEMEN DATA PRODUK BISNIS PERHIASAN BERBASIS FLASK DAN MONGODB," IT-Explore: Jurnal Penerapan Teknologi Informasi dan Komunikasi, vol. 3, no. 2, pp. 162–176, 2024.

[9] M. H. Maulana, "Python Bahasa Pemograman Yang Ramah Bagi Pemula," JISCO: Journal of Information System and Computing, vol. 2, no. 2, pp. 73–78, 2024.

[10] A. Kencana Putri and D. Ichsanuddin Nur, "Penggunaan bahasa python untuk analisis dan visualisasi data penduduk di Desa Sumberjo, Nganjuk.," Karya: Jurnal Pengabdian kepada Masyarakat, vol. 3, no. 3, pp. 206–217, 2023.

[11] E. Haezer, Y. Kristianto, and N. Setiyawati, "PEMBANGUNAN APLIKASI VIRTUAL INVENTORY SYSTEM (VIS) BERBASIS WEB MENGGUNAKAN FLASK FRAMEWORK (STUDI KASUS : PT XYZ)," Jurnal MNEMONIC Vol 4.2 (2021)., vol. 4, no. 2, pp. 128–135, 2021.

[12] R. Y. Azhari, "Web service framework: Flask dan FastAPI," Technology and Informatics Insight Journal, vol. 1, no. 1, pp. 80–87, 2022.

[13] R. Ridhalri, "Pemanfaatan Caching System Menggunakan Redis Untuk Sistem Pengelolaan Informasi Ambalan Ashabul Kahfi Berbasis Web," Jurnal DIALOGIKA: Manajemen Dan Administrasi, vol. 4, no. 1, pp. 39–56, 2022.

[14] B. Haryanto, A. Ardiansyah, and M. Kurniasih, "PENGENALAN DATABASE NOSQL DAN PERBANDINGANNYA DENGAN DATABASE RELASIONAL," Insan Pembangunan Sistem Informasi Dan Komputer (IPSIKOM), vol. 12, no. 1, pp. 1-7, 2024.

[15] D. R. Prasetyo, F. Fatoni, M. Nasir, and I. Effendy, "Implementasi Redis Untuk Mempercepat Pengambilan Data (Studi Kasus : Sistem Dasawisma PKK Sumsel)," Jurnal Sistem EXPLORE: JURNAL SISTEM INFORMASI DAN TELEMATIKA Учредители: Universitas Bandar Lampung Publication Center, vol. 15, no. 2, p.214, 2024.

[16] I. N. Ramadhan et al., "Penerapan Database Redis Sebagai Optimalisasi Pemrosesan Kueri Data Pengguna Aplikasi SIRESMA Berbasis Laravel," Technomedia Journal, vol. 8, no. 3, pp. 64-67, 2024.

[17] R. P. Kamaz and Y. R. Beeh, "Sistem Informasi

Kasir Toko Kembar menggunakan Framework Webix dan MongoDB," Jurnal Indonesia: Manajemen Informatika dan Komunikasi, vol. 6, no. 1, pp. 1–13, 2025.

[18] S. Saidah et al., "Implementasi Arsitektur Microservices pada Aplikasi Point of Sale Toko Flyover Stiker," JIKI (Jurnal Ilmu Komputer dan Informatika), vol. 4, no. 1, pp. 77-88, 2023.

[19] F. Rosabel Ramli, F. Hakim, and R. Anggelina Hutabarat, "Perancangan Web Design Aplikasi E-Learning dengan Metode Prototype pada Tingkat SMA," Majalah Ilmiah UPI YPTK, vol. 28, no. 1, pp. 13-18, 2021.