

Design of a Mobile Application-Based Digital Health Solution Using the Agile Method: A Case Study in Annisa Farma Pharmacy

M. Adi Siswanto¹, Susanna Dwi Yulianti Kusuma²

^{1,2}Universitas Pamulang, Tangerang Selatan, Indonesia

Email: adi15siswanto@gmail.com

Abstract

Annisa Farma Pharmacy still manages drug stock data, sales transactions, and customer information manually through an application that can only be accessed from a single device, making flexible data monitoring difficult and limiting payment options. This research aims to design and build a digital health solution in the form of a mobile application and an admin dashboard named PharmaLink ID using the Agile development method. The system was designed using an Entity Relationship Diagram (ERD), Logical Record Structure (LRS), and Unified Modeling Language (UML), while the implementation used React Native and Expo for the mobile application, react for the admin dashboard, and Express, Prisma, and MySQL on the backend, integrated with the Xendit payment gateway for non-cash payments. The system consists of 44 pages and 22 database tables, equipped with security mechanisms such as prescription verification, a ten-stage transaction state machine, and idempotent financial records. Testing was conducted using the black box method on 22 scenarios and the white box method on three critical modules. The black box testing results show that all scenarios are valid, and the white box testing produced twelve independent paths that were all executed according to the designed logic. Thus, PharmaLink ID can improve the operational efficiency of the pharmacy, enable real-time data monitoring, and provide digital payment options for customers.

Keywords: *Digital Health Solution, Pharmacy, Mobile Application, Agile, React Native, Payment Gateway, Black Box Testing, White Box Testing.*

A. INTRODUCTION

Annisa Farma Pharmacy is situated in a strategic location in Parungpanjang and experiences a high volume of daily customer visits. As a healthcare provider, the pharmacy not only facilitates direct medication sales but also offers the public information and basic consultations regarding medication use. In its day-to-day operations, data management—such as tracking medication stock, sales transactions, and customer records—is handled using a simple application accessible from only a single device. This limitation complicates management, particularly when the owner or manager needs to monitor data flexibly.

The current application lacks online capabilities, hindering the pharmacy's sales efficiency—particularly when real-time sales data from multiple locations is required. Furthermore, customers face limitations such as being restricted to in-store transactions, difficulty locating the nearest pharmacy, and the need to visit in person to collect

medication. The sale of prescription-only drugs also necessitates a mechanism for pharmacists to verify prescriptions online. Addressing these challenges, this study aims to design a digital health solution—a mobile application developed using the Agile methodology—intended to improve upon the previous system.

The proposed solution is an online, multi-device compatible application named PharmaLink ID, featuring integrated e-commerce capabilities to streamline transactions. Key features include medication management, transaction processing, pharmacy location services, customer education, and sales reporting. By employing an Agile approach, the development process remains responsive to user needs, ensuring the final product delivers tangible improvements to the operational efficiency of Annisa Farma Pharmacy.

Several previous studies serve as references. Sihombing (2024) applied Agile methodology to develop a prescription medication management system aimed at enhancing the pharmacy user experience; Kaharu (2023) built a mobile-based application for managing medication sales data; Mulyani et al. (2022) designed an Android-based application for medication sales and inventory; Mas'udia et al. (2022) analyzed the Quality of Service (QoS) of a medication sales information system featuring delivery services; and Yuyun et al. (2024) developed a geographic information system for pharmacy locations. Unlike these studies, PharmaLink ID integrates three interfaces—a patient app, a pharmacist app, and an admin dashboard—into a single backend, featuring online prescription verification and cashless payment capabilities via a payment gateway.

Based on the above, the research problem is formulated as follows: (1) how to design an integrated and efficient system for managing medication stock, sales transactions, and customer information that does not rely on a single-device application; (2) how to develop a solution that enables flexible, real-time data monitoring from various locations; and (3) how to incorporate cashless payment features to offer customers a wider range of transaction options. The research aims to design and build a digital health solution—comprising a mobile application and an admin dashboard—capable of enhancing pharmacy operational efficiency, supporting multi-device access, providing real-time operational information, and offering cashless payment options through integration with the Xendit payment gateway.

B. METHOD

Data was collected using three techniques. Direct observation of Annisa Farma Pharmacy's operational activities was conducted to understand workflows, constraints, and user needs. Interviews were held with the pharmacy owner/manager to gather in-depth information regarding existing issues and expectations for the new system. A document study involved analyzing current transaction records, report formats, and medication data structures to formulate the requirements for the new system.

The Agile method was selected for system development due to its iterative and flexible nature, allowing for phased development and responsiveness to changing requirements. The development stages included: Product Backlog (formulating a list of requirements and features based on data collection results); Sprint Planning (prioritizing features for each 1–2 weeks iteration); Sprint Development (the process of coding and testing features incrementally); Daily Scrum (daily self-evaluations to ensure features met targets); and Sprint Review (evaluating features at the end of each sprint).

System design was modeled using Entity Relationship Diagrams (ERD) and Logical Record Structures (LRS) for the database, and Unified Modeling Language (UML), including use case, activity, sequence, class, and state machine diagrams to model system behavior. The database was implemented using MySQL. System testing employed two approaches: black-box testing to verify functional compliance with requirement specifications, and white-box testing using the basis path testing technique to validate the internal logic structure of critical modules via cyclomatic complexity calculations.

C. RESULT AND DISCUSSION

1. System Architecture

The PharmaLink ID system is built with three interfaces sharing a single backend: a mobile application for patients and pharmacists (React Native + Expo), a web-based admin dashboard (React + Vite), and an API backend (Express + Prisma + MySQL). The overall architecture is illustrated in Figure 1. All client requests pass through the routes, controllers, and service layers before reaching the database via Prisma ORM. Business logic, such as status transition rules, prescription gating, total calculations, platform commissions, and wallet idempotency is centralized within the service layer to ensure consistent application across all interfaces. External services are utilized for payments and fund disbursements (Xendit), OTP delivery via email (Mailjet) and WhatsApp (Fonnte), and push notifications (Firebase Cloud Messaging).

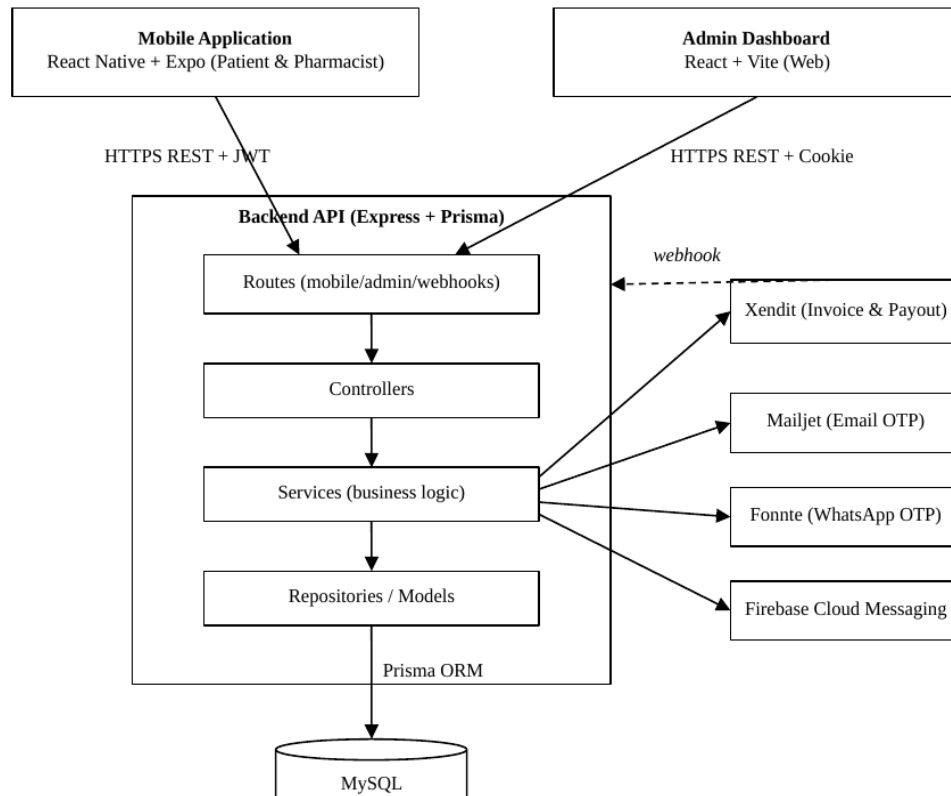


Figure 1. PharmaLink ID System Architecture

2. Database Design

The system's conceptual data model was designed using an ERD and derived into 22 interrelated entities (tables). To facilitate readability, the entities are grouped into five modules, User and Master Data, Medication Catalog and Stock, Transactions and Prescriptions, Finance, and Notifications as illustrated in Figure 2. The **User** and **Pharmacy** entities serve as the core of the data model: the **User** entity encompasses all accounts (admin, pharmacist, patient), distinguished by their relationships with the **Role** entity, while the **Pharmacy** entity acts as the parent for catalog, stock, transaction, and financial data. A summary of the table groupings by module is presented in Table 1.

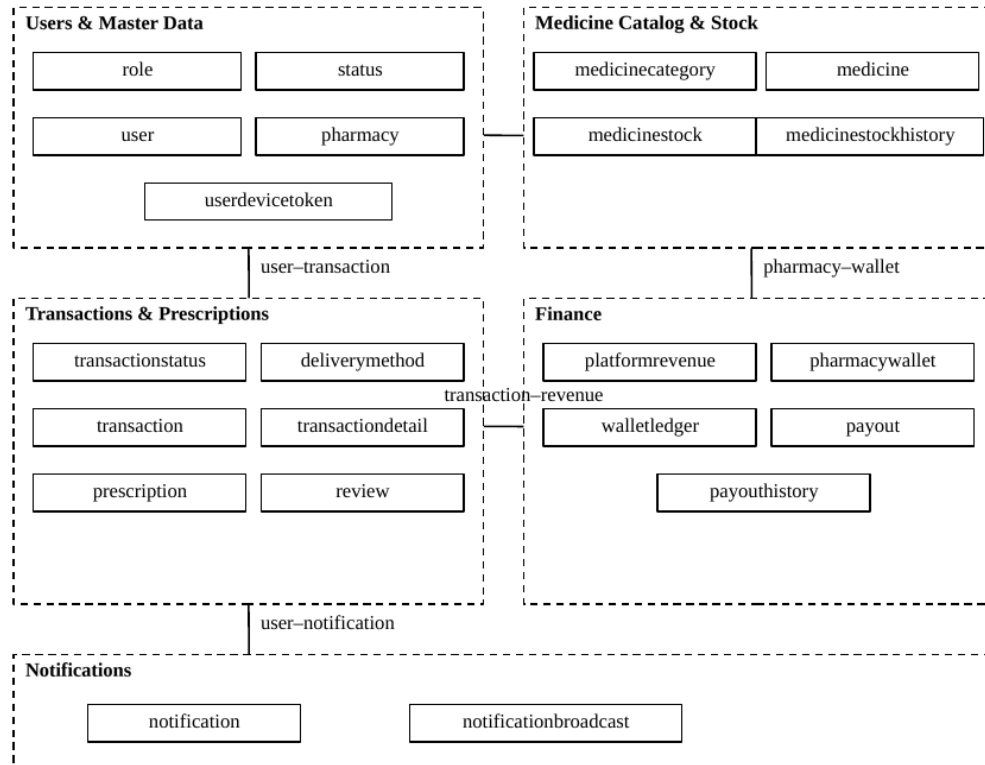


Figure 2. Summary of PharmaLink ID Database Entity Grouping

Table 1. Grouping of 22 Database Tables by Module

Module	Table (Entity)	Amount
Users & Master Data	role, status, user, pharmacy, user device token	5
Medication Catalog & Stock	Medicine category, medicine, medicine stock, medicine stock history	4
Transactions & Prescriptions	Transaction status, delivery method, transaction, transaction detail, prescription, review	6
Finance	Platform revenue, pharmacy wallet, wallet ledger, payout, pay out history	5
Notifications	notification, notification broadcast	2
Total		22

3. Functional Design

The system's functional requirements are modeled using a use case diagram involving four actors—patients, pharmacists, administrators, and the external actor Xendit—as shown in Figure 3. Patients can register and perform OTP verification, log in, search for pharmacies and medicines, create and track orders, make payments, upload prescriptions, and submit reviews. Pharmacists manage medicines, categories, and stock; process orders; review prescriptions; manage their wallets; and request fund withdrawals. Administrators monitor platform transactions and revenue, manage users

and pharmacies, moderate reviews, and send broadcast notifications. The Xendit actor is involved in invoice generation and the receipt of payment status and payout webhooks.

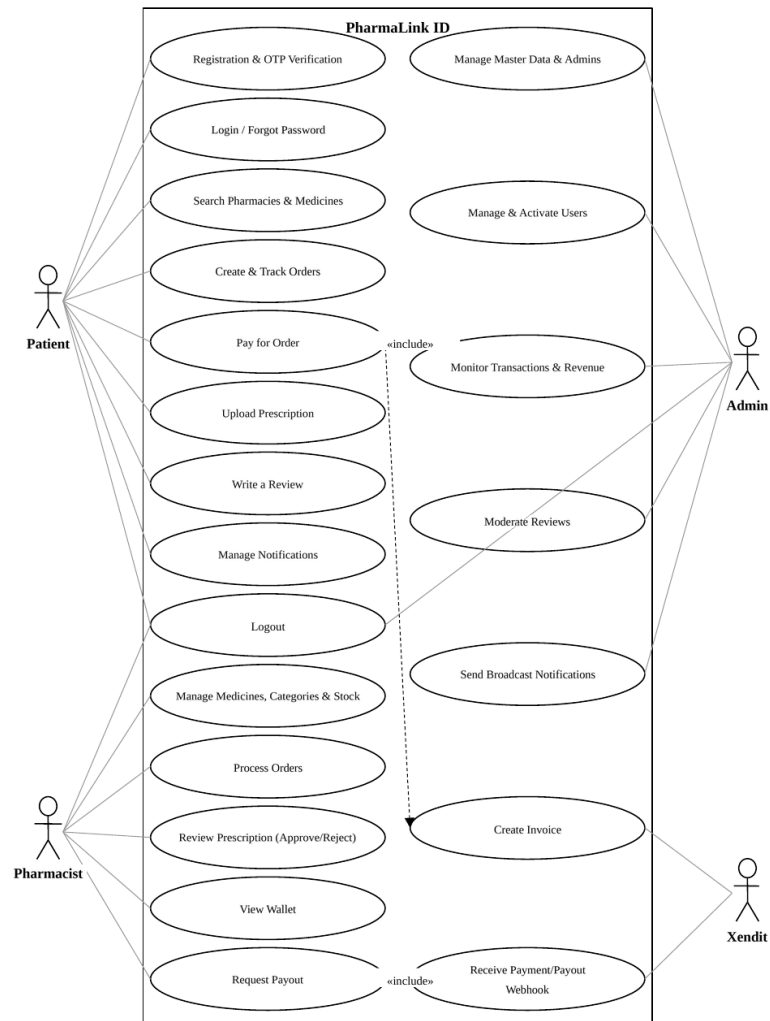


Figure 3. PharmaLink ID Use Case Diagram

4. Transaction Status Engines

Transaction reliability is maintained by a ten-stage server-validated transaction state machine, as illustrated in Figure 4. Transactions progress from **Draft** to **Waiting Payment**, then to **Payment Verified** upon receipt of the **PAID** webhook, followed by **Processing** once the prescription is approved, and finally to **Completed** via either **Ready for Pickup** or **On Delivery**. The **Cancelled**, **Failed**, and **Refunded** states handle scenarios involving cancellations, payment failures or expirations, and refunds. Forward transitions can only be triggered by a pharmacist, whereas shifts to **Payment Verified** and **Failed** are triggered by Xendit webhooks, ensuring that transaction statuses cannot be manipulated from the client side.

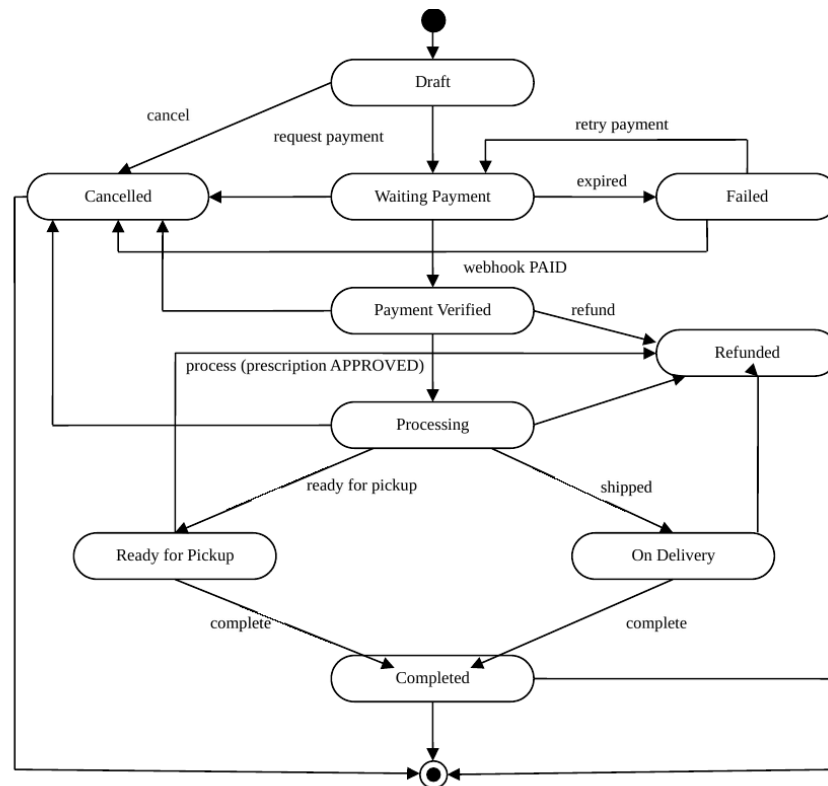


Figure 4. Transaction Status State Machine Diagram

5. Interface Implementation

The system is implemented across 44 pages categorized by user role: seven shared authentication pages, sixteen patient pages, thirteen pharmacist pages on the mobile application, and eight web-based admin dashboard pages. The patient interface includes a home page, pharmacy and medication search, cart, checkout, prescription upload, payment via Xendit invoices, and order tracking, as shown in Figure 5. The pharmacist interface covers incoming order processing, prescription review, and pharmacy wallet management, as shown in Figure 6. The admin dashboard provides platform revenue monitoring and user management, as shown in Figure 7.

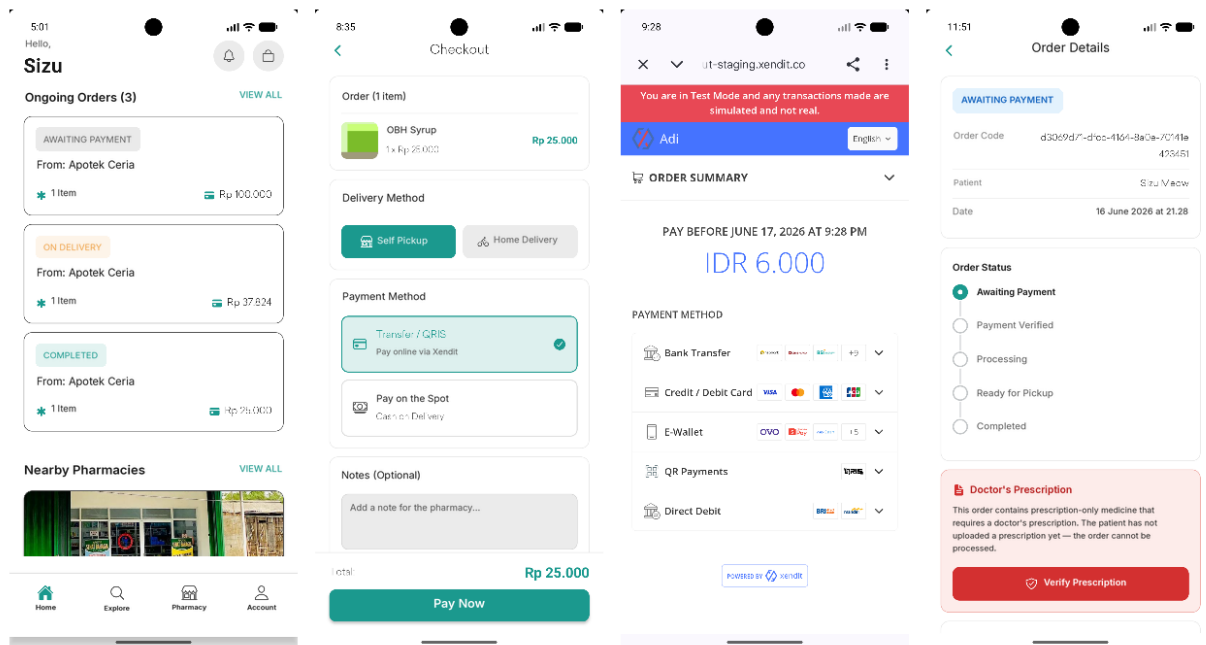


Figure 5. Patient Interface (Home, Checkout, Xendit Payment, and Order Details)

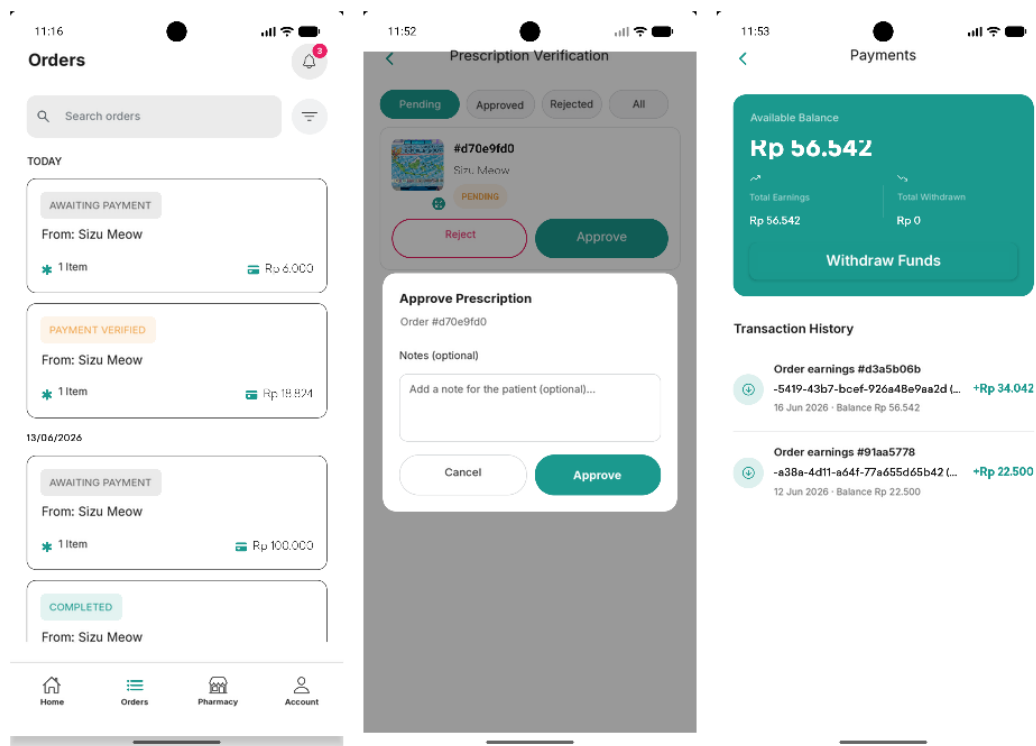


Figure 6. Pharmacist Interface (Incoming Orders, Prescription Review, and Pharmacy Wallet)

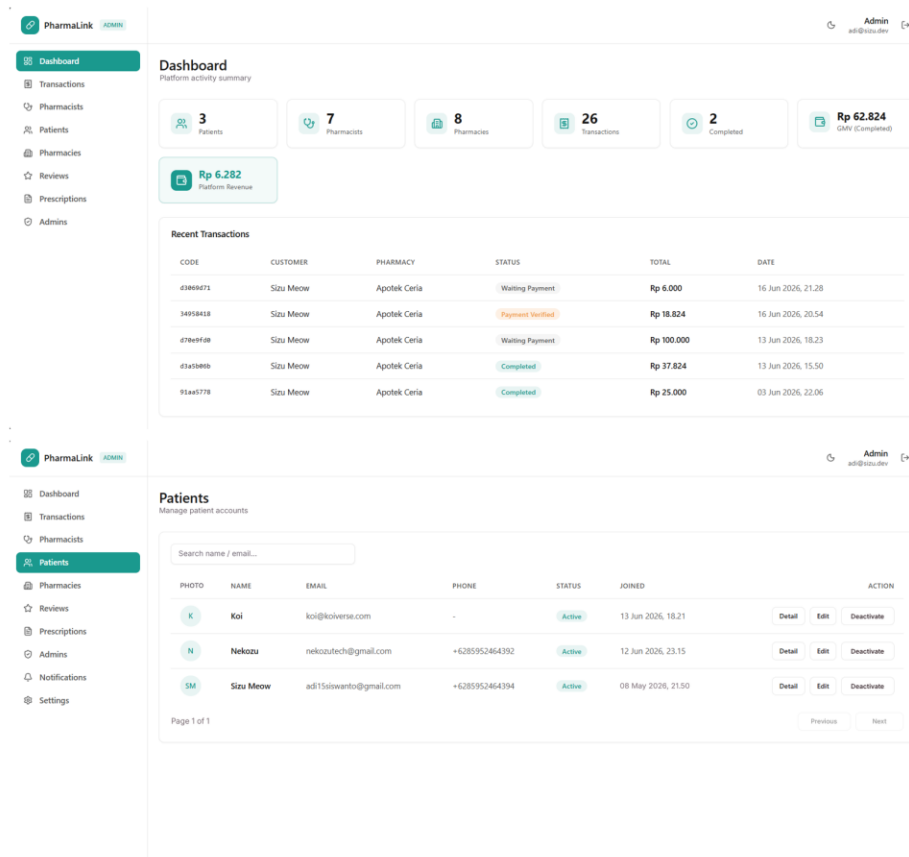


Figure 7. Admin Interface (Revenue Dashboard and User Management)

6. Black Box Testing

Functional testing was conducted using the black-box method across 22 scenarios representing the system's core functions, including authentication, search and ordering, payment, prescription verification, wallet management and fund withdrawal, reviews, notifications, and user administration. The scenarios encompassed both positive cases (normal workflows) and negative cases (input validation, access denial, and failure handling). Detailed test results are presented in Table 2. All scenarios (22 out of 22) yielded the expected outputs—resulting in a "Valid" status with a 100% success rate—and no critical functional defects were found.

Table 2. Black Box Testing Scenarios and Results

No.	Test Scenario	Expected Result	Conclusion
1	Login with a valid email address and password	The user is redirected to the homepage; a token is issued	Valid
2	Login with an incorrect password	The system rejects the request (401) and displays a failure message	Valid
3	Patient registration with complete data	The account is created; an OTP is sent via email and WhatsApp	Valid

4	Verification of the correct OTP through both channels	The account becomes active (registration completed)	Valid
5	Verification with an incorrect or expired OTP	Verification is rejected; a resend option is available	Valid
6	Forgot password	A password reset link is sent to the user's email	Valid
7	Search for pharmacies using the open-status filter	Only pharmacies that are currently open are displayed	Valid
8	Checkout with sufficient stock	A Waiting Payment transaction is created; the stock is reserved; an invoice is issued	Valid
9	Checkout exceeding available stock	The order is rejected with an insufficient stock message	Valid
10	Checkout for prescription-only medication	The system requests the user to upload a prescription image (status: PENDING)	Valid
11	Successful invoice payment	PAID webhook received; payment status becomes Payment Verified; commission is recorded	Valid
12	Invoice allowed to expire	The status changes to Failed; the reserved stock is returned	Valid
13	Processing an order before the prescription is approved	The system rejects the request (409); the status remains unchanged	Valid
14	Processing an order after the prescription is approved	The status changes to Processing; a notification is sent	Valid
15	Complete the order (Completed)	The pharmacy wallet is credited once with the net amount (idempotent)	Valid
16	Submit a review for a completed order	The review is saved; the pharmacy rating is updated	Valid
17	Submit a second review for the same order	The system rejects the duplicate review	Valid
18	Request a payout with sufficient balance	Payout status becomes REQUESTED; the balance is debited; the transaction history is recorded	Valid
19	Request a payout exceeding the available balance	The system rejects the request with an insufficient balance message	Valid

20	Payout FAILED webhook	The balance is restored and the reversal is recorded	Valid
21	Order status change	Notifications are received by both the patient and the pharmacist	Valid
22	Administrator disables a user	The inactive user cannot log in to the system	Valid

7. White Box Testing

White-box testing was conducted using the basis path testing technique to examine the program's internal logic structure. The flow of each module was depicted as a flowchart and subsequently converted into a flow graph to calculate cyclomatic complexity, $V(G)$, using three formulas that must yield the same value: $V(G) = E - N + 2$, $V(G) = P + 1$, and $V(G) = \text{number of regions (where } E \text{ is the number of edges, } N \text{ is the number of nodes, and } P \text{ is the number of predicate nodes)}$. Three modules exhibiting the most representative branching logic were selected as samples: User Login, Order Creation, and Payout Request. The flow graphs for these three modules are presented in Figures 8, 9, and 10, while a summary of the calculations is provided in Table 3.

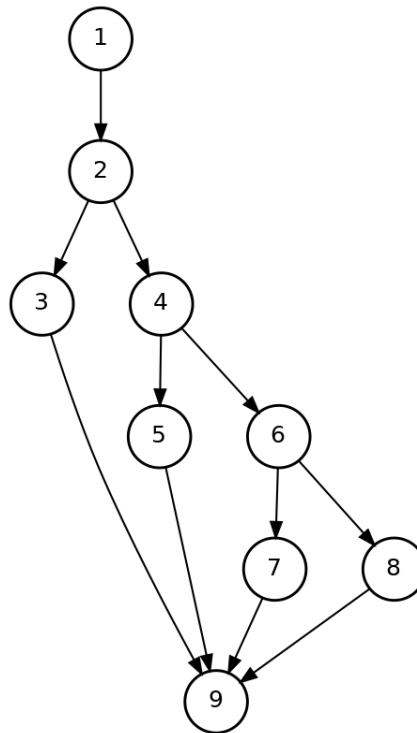


Figure 8. User Login Module Flowchart

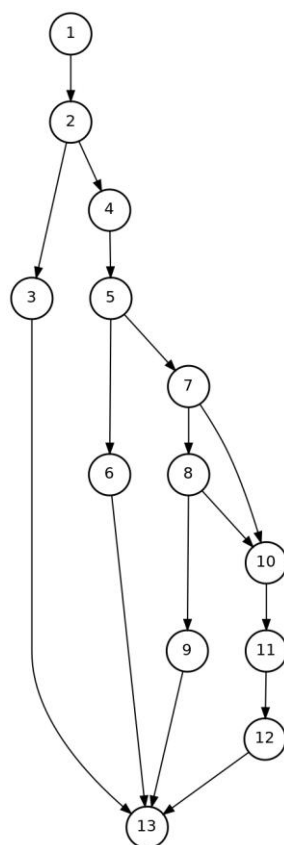


Figure 9. Order Creation Module Flowchart

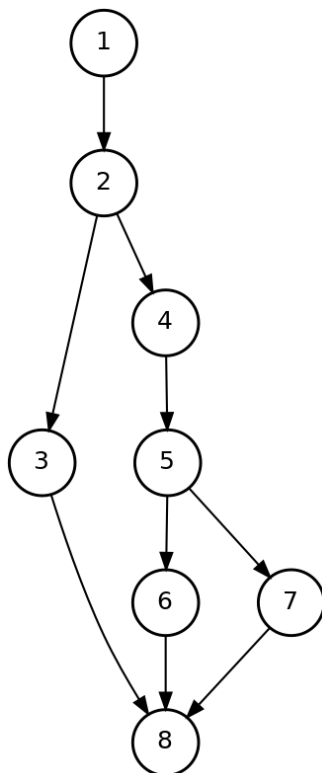


Figure 10. Payout Request Module Flowcharts

Table 3. Summary of Cyclomatic Complexity Calculation

Module	N	E	P	V(G)	Number of Lanes
User Login	9	11	3	4	4
Order Creation	13	16	4	5	5
Payout Request	8	9	2	3	3
Total					12

All three modules yielded consistent V(G) values across the three formulas, and every independent path (totaling $4 + 5 + 3 = 12$ paths) was successfully executed, producing outputs that aligned with the design logic. Thus, the program's logical structure was proven to be free of dead paths and unhandled decision branches. The combined black-box and white-box results demonstrate that the system's functionality meets the requirement specifications and its internal logic is correct.

D. CONCLUSION

Based on the analysis, design, implementation, and testing phases, the PharmaLink ID system was successfully developed with three interfaces sharing a unified backend: mobile applications for patients and pharmacists, a web-based admin dashboard, and an API backend utilizing a MySQL database comprising 22 tables. This online architecture decouples the management of medication stock, sales transactions, and customer data from reliance on a single device. Flexible data monitoring is facilitated through the admin dashboard and pharmacist application, which provide real-time data on orders, stock levels, revenue, and the pharmacy wallet. Cashless payment capabilities are enabled via integration with the Xendit payment gateway—covering invoice issuance, webhook-based payment confirmation, and fund disbursement—while transaction reliability is ensured through a ten-stage state machine, authoritative server-side total calculations, a prescription approval gateway, and idempotent financial logging. Black-box testing across 22 scenarios achieved a 100% success rate, and white-box testing identified 12 independent paths, all of which executed according to the design.

Future research should aim to complement existing testing with non-functional tests such as load testing and security assessments; integrate third-party delivery services and courier tracking; expand integration with external systems like BPJS Kesehatan or pharmaceutical distributors; publish the application to official app stores and introduce features for health education and online pharmacist consultations; and conduct periodic security audits, including dependency updates and the rotation of external service secret keys.

REFERENCES

1. Aditiya, M. R., Setiadi, R. D. P., & Santoso, B. (2025). Implementasi JSON Web Token (JWT) untuk sistem autentikasi dan otorisasi pada aplikasi Android native dan Spring Boot di CV Karya Belia Nusantara. *BINER: Jurnal Ilmu Komputer, Teknik dan Multimedia*, 3(3), 267–274.
2. Al Fajri, M., & Asqia, M. (2026). Pengembangan backend platform konseling mental menggunakan Express JS. *DBESTI: Journal of Digital Business and Technology Innovation*, 3(1), 57–64. <https://doi.org/10.54914/dbesti.v3i1.2198>
3. Aulia, R., Candra, D. G. A., & Fauziyyah, A. (2025). Analisa perancangan permodelan basis data pada pengembangan sistem informasi pendaftaran menggunakan Entity Relationship Diagram. *Inventor: Jurnal Inovasi dan Tren Pendidikan Teknologi Informasi*, 3(3), 100–111. <https://doi.org/10.37630/inventor.v3i3.2682>
4. Djuwitaningrum, E. R., & Jati, I. B. W. (2025). Implementasi payment gateway Midtrans pada website e-commerce toko buah dan sayur. *Jurnal Ilmu Pengetahuan dan Teknologi (IPTEK)*, 9(1). <https://doi.org/10.31543/jii.v9i1.390>
5. Expo. (2024). *Expo documentation*. <https://docs.expo.dev/>
6. Google. (2023). *Meet Android Studio*. <https://developer.android.com/studio/intro>
7. Kaharu, N. A. (2023). Aplikasi pengelolaan data penjualan obat pada apotek berbasis mobile. *Jurnal Pendidikan Tambusai*, 7(2), 10613–10621. <https://doi.org/10.31004/jptam.v7i2.8045>
8. Laragon. (2023). *Laragon documentation*. <https://laragon.org/docs/>
9. Mas'udia, P. E., Pratama, C. A., Purwati, D., Ratnawati, Y., Sarosa, M., & Hidayati, N. (2022). Rancang bangun dan analisis QoS pada sistem informasi penjualan obat dengan layanan antar-jemput berbasis Android. *Techno.COM*, 21(3), 633–643.
10. Meta Platforms. (2023). *React Native documentation*. <https://reactnative.dev/>
11. Microsoft. (2023). *Visual Studio Code documentation*. <https://code.visualstudio.com/docs>
12. Mulyani, A., Septiana, Y., & Helmi, R. (2022). Rancang bangun aplikasi penjualan dan persediaan obat pada apotek berbasis Android. *Jurnal Algoritma*, 19(2), 639–648. <https://doi.org/10.33364/algoritma/v.19-2.1180>
13. Naresvari, E., & Susetyo, Y. A. (2025). Penerapan JavaScript React pada perancangan front-end website UMKM Jemari Ragil. *IT-Explore: Jurnal Penerapan Teknologi Informasi dan Komunikasi*, 4(1), 16–32. <https://doi.org/10.24246/itexplore.v4i1.2025.pp16-32>
14. Nurandani, D., Sitorus, S. P., Manoppo, A. R., Adrian, D., & Ayuda, M. I. (2025). Penggunaan OTP (One Time Password) sebagai lapisan keamanan tambahan dalam aplikasi AY Pulsa. *Jurnal Minfo Polgan*, 14(2), 3304–3314. <https://doi.org/10.33395/jmp.v14i2.15724>
15. Pratiwi, Y., & Widiarti, L. W. (2025). Implementasi whitebox testing dengan teknik basis path pada pengujian halaman pencarian program promo. *Jurnal Kecerdasan Buatan dan Teknologi Informasi*, 4(2), 173–180. <https://doi.org/10.69916/jkbt.v4i2.280>
16. Prisma. (2024). *Prisma ORM documentation*. <https://www.prisma.io/docs>

17. Ronal, P. (2022). Desain Unified Modeling Language (UML) dalam perancangan aplikasi hauling trip di industri tambang batubara. *Jurnal Teknik Informatika dan Sistem Informasi (JATISI)*, 9(4), 3038–3050.
18. Sihombing, D. J. C. (2024). Implementation of agile in developing prescription drug management system to enhance user experience in pharmacies. *Jurnal Info Sains: Informatika dan Sains*, 14(1), 1036–1045.
19. Syahputri, A. Z., Fallenia, F. D., & Syafitri, R. (2023). Kerangka berfikir penelitian kuantitatif. *Tarbiyah: Jurnal Ilmu Pendidikan dan Pengajaran*, 2(1), 160–166.
20. Uminingsih, Ichsanudin, M. N., Yusuf, M., & Suraya. (2022). Pengujian fungsional perangkat lunak sistem informasi perpustakaan dengan metode black box testing bagi pemula. *STORAGE: Jurnal Ilmiah Teknik dan Ilmu Komputer*, 1(2), 1–8. <https://doi.org/10.55123/storage.v1i2.270>
21. Vite. (2024). Vite documentation. <https://vitejs.dev/guide/>
22. Wahyudin, A., Anisyah, A., & Achmaddifa, D. (2025). Pengembangan sistem notifikasi real time untuk aplikasi manajemen persuratan multiplatform menggunakan Firebase Cloud Messaging dan Application Programming Interface. *Jurnal Teknik Informatika UNIKA Santo Thomas*, 10(1), 75–87.
23. Yuyun, E., Mukramin, M., & Muhallim, M. (2024). Aplikasi sistem informasi geografis lokasi apotek di Masamba berbasis Android. *Jurnal Riset Sistem Informasi*, 1(3), 45–53. <https://doi.org/10.69714/npvk4k95>