Perancangan dan Implementasi *Back-End* Manajemen Persediaan Obat pada Apotek Menggunakan Metode *Extreme Programming* Berbasis *Website*

Aditya Setiawan
Teknik Informatika
Universitas Telkom Purwokerto,
Purwokerto, Indonesia
21102034@ittelkom-pwt.ac.id

Iqsyahiro Kresna A
Teknik Informatika
Universitas Telkom Purwokerto,
Purwokerto, Indonesia
hiroka@telkomuniversity.ac.id

Maryona Septiara
Rekayasa Perangkat Lunak
Universitas Telkom Purwokerto,
Purwokerto, Indonesia
septiara@telkomuniversity.ac.id

***Abstrak* — Ketersediaan obat-obatan sangat penting bagi masyarakat guna menjaga kesehatan mental dan tubuh. Obat yang dikonsumsi masyarakat merupakan hasil distribusi dari berbagai perusahaan di bidang farmasi dan kesehatan, salah satunya adalah apotek independen yang memberikan akses kepada masyarakat untuk mendapatkan obat. Namun apotek independen dihadapkan pada tantangan serius, salah satunya adalah ketersediaan obat yang ada pada apotek itu sendiri. Kekosongan dan kekurangan persediaan obat menyebabkan masyarakat sulit mendapatkan obat. Selain itu kelebihan persediaan obat mengakibatkan obat kadaluarsa dan rusak akibat terlalu lama tersimpan. Penelitian ini bertujuan untuk mengembangkan solusi berbentuk sistem manajemen dan pengelolaan persediaan obat berbasis *website*. Dengan adanya sistem ini memungkinkan apotek dapat mengelola persediaan obat dengan baik. Persediaan obat dapat diatur mulai dari pemasukan dan pengeluaran obat, notifikasi kondisi obat, hingga laporan data persediaan obat dengan metode *first expired first out* (FEFO). Penelitian ini menerapkan metode *Extreme Programming* dalam pengembangan sistem manajemen dan pengelolaan persediaan obat. Menggunakan *framework Express.js* sebagai pengembangan *back-end* sistem tersebut yang dimplementasikan dalam bentuk *website*. Manfaat dari penelitian ini dapat meminimalisir *human error*, ketersediaan obat di apotek dan menghemat waktu dan tenaga.**

Kata kunci*: Apotek, Persediaan obat, *Website*, *Back-end*, *Extreme Programming*, *Express.js

I. PENDAHULUAN

Ketersediaan obat pada setiap unit apotek perlu diperhatikan karena merupakan suatu komponen masalah yang kompleks. Dari hal tersebut diperlukan jaminan ketersediaan obat baik dalam hal jumlah, jenis, dan ketepatan waktu yang sesuai dengan kebutuhan sehingga dapat memprediksi adanya kekurangan dan kelebihan persediaan obat. Menurut Peraturan Menteri Kesehatan Republik Indonesia Nomor 9 Tahun 2017 tentang apotek yaitu mengelola ketersediaan farmasi berupa obat, bahan obat, obat tradisional, kosmetika, bahan medis habis pakai, dan alat kesehatan. Dari peraturan tersebut apotek memiliki peran yang sangat penting dalam mengelola ketersediaan obat [1].

Persediaan obat yang terorganisir dengan baik diperlukan manajemen yang baik pula. Namun manajemen persediaan obat tersebut mengalami tantangan yang kompleks. Mulai dari pengarsipan dan pendataan yang buruk, penyimpanan obat yang tidak baik, distribusi obat yang tertunda dan masih banyak faktor lainnya. Kendala seperti itu pada manajemen obat dapat mengalami berbagai masalah, seperti *human error*, aliran persediaan obat terhambat, menghabiskan banyak waktu dan tenaga, dan pengelolaan keuangan yang rumit [2]. Masalah tersebut perlu adanya sistem yang membantu memanajemen ketersediaan obat.

System Inventory merupakan salah satu cara yang dapat menangani masalah tersebut. Cara ini digunakan untuk membantu perusahaan yaitu mendata barang-barang inventaris dengan berbagai cara seperti metode *First In First* Out (FIFO), *First Expired First Out* (FEFO), dan *Last In First Out* (LIFO). Pada perusahaan apotek umumnya menggunakan metode FIFO dan FEFO sehingga dapat mengatur dan mengelola persediaan obat secara terstruktur [3]. Dengan metode tersebut dapat membantu mengelola persediaan obat dan detail persediaan dapat dimuat dalam bentuk laporan.

Apotek Dian Brata Medika yang berlokasi di Dukuh Mekar, Karang Tengah, Kecamatan Kembaran, Kabupaten Purbalingga, Jawa Tengah merupakan badan usaha di bidang perdagangan farmasi milik perorangan yang menyediakan berbagai jenis obat-obatan. Prosedur dalam persediaan obat masih menggunakan arsip manual. Pengarsipan dilakukan mulai dari mingguan hingga bulanan mengakibatkan persediaan obat tidak dapat ditangani dengan baik sehingga muncul masalah lain seperti kekurangan atau kelebihan persediaan obat mengakibatkan obat rusak dan obat kadaluarsa, kurangnya pemantauan aktivikas karyawan yang berimbas pada pengelolaan keuangan apotek. Bila tidak segera tertangani dapat menghambat masyarakat untuk

memperoleh obat. Dampak lainnya adalah dapat mengalami penurunan kualitas pelayanan dan penurunan finansial.

Berdasarkan kasus pada apotek yang telah dijabarkan, penelitian ini bertujuan untuk mengembangkan solusi dalam bentuk sistem *back-end* yang dapat membantu memanajemen persediaan obat pada apotek tersebut dengan berbasis *website* menggunakan metode *Extreme Programming (XP)*. Metode ini sangat relevan digunakan pada pengembangan projek. Metode ini menekankan komunikasi, kesederhanaan, umpan balik, keberanian, dan menghargai kerjasama tim. Komunikasi yakni kolaborasi tim dengan klien sehingga antara tim dan pengguna terjalin dengan baik. Kesederhanaan atau disebut juga *simplicity* merupakan perancangan yang sederhana sesuai kebutuhan dan dilakukan secara terus-menerus untuk diperbaiki dan disederhanakan agar menjaga kualitas dan fungsionalitas program. Umpan balik atau *feedback* yang cepat yang didapat dari pengujian berbagai sumber. Keberanian atau *courage* yakni berani menghadapi tantangan dan masalah dan diselesaikan sesegera mungkin. Menghargai atau *respect* merupakan sikap menghargai kerjasama setiap anggota tim dan mendukung satu sama lain sehingga sangat membantu dalam memanajemen proyek [4].

Express.js digunakan sebagai kerangka dari *Node.js* dalam pengembangan sistem *back-end* yang dinamis berbasis *website*[5]. *Express.js* merupakan kerangka kerja yang menyediakan pola dan praktik untuk mengorganisir kode, mengatur *routing*, dan berbagai aspek lain pada pengembangan *website*. *Express.js* dapat menggunakan arsitektur *model view controllers* (MVC) sehingga memberikan rute sistem yang jelas. Kerangka kerja ini juga mendukung *middleware* dan menangani *request* dan *respons* HTTP berbasis *RESTful API* memungkinkan sistem berjalan terstruktur dan mudah untuk dipantau pada saat pengembangan [6].

Pengujian pada *back-end* menggunakan metode *white box testing* agar mengetahui fungsionalitas dan kode internal sistem dapat berjalan dengan baik sesuai dengan *test case*. Pengujian ini menerapkan berbagai cara yaitu *coverage test, flowchart, flowgraph, independent path,* uji skenario dan masih banyak lagi cara melakukan *testing* menggunakan metode *white box* [7].

Fitur yang ditawarkan kepada pengguna meliputi input dan output data obat berdasarkan metode FEFO, *dashboard* sebagai tampilan mengindikasi total obat, status obat, berserta ketersediaan obat. Fitur notifikasi yang dapat memberikan pemberitahuan dan peringatan kondisi obat. Fitur Laporan obat berdasarkan persediaan dan laba dari penjualan obat sehingga dapat mengetahui kondisi obat dan finansial pada apotek. Fitur riwayat untuk mengetahui aktifitas karyawan dan fitur manajemen pengguna untuk mengontrol akses karyawan [8].

II. KAJIAN TEORI

a. Manajemen Persediaan Obat

Manajemen persediaan obat adalah sistem farmasi terpadu guna mengelola obat-obatan dan mengotomatiskan operasi seperti pengendalian ketersediaan obat, pengeluaran obat, dan pelaporan. Informasi diperoleh dari berbagai dokumen dan catatan kesehatan. Manajemen persediaan obat dapat berupa metode manajemen secara manual atau alat bantu manajemen seperti aplikasi berbasis *website*, *mobile apps*, dan *desktop apps*. Manajemen persediaan obat yang baik digunakan untuk pengelolahan data meliputi notifikasi kondisi dan persediaan obat, pendataan secara *real-time*, dan laporan persediaan [9].

b. Website

Website merupakan kumpulan halaman *web* yang saling terhubung antara satu dengan yang lain dan dapat diakses melalui jaringan internet. Halaman web berisi informasi berupa teks, gambar, video, atau animasi. *Website* sangat berguna untuk berbagai keperluan mulai dari bisnis, pendidikan, hiburan dan masih banyak lagi. *Website* dibagi menjadi dua yaitu statis dan dinamis. *Website* statis mengandalkan pengembangan *front-end* berupa CSS dan HTML. Website jenis ini sering diguanakan untuk profil perusahaan, protofolio, atau halaman informasi sederhana. Sedangkan *website* dinamis memiliki konten yang berbeda dan menggunakan *back-end* dan *database* sebagai media penyimpanan [10]

c. Extreme Programming

Extreme Programming adalah salah satu jenis metode *Agile Development* yang digunakan tim mengontrol proses pengembangan agar mencapai kesuksesan pada pengembangan proyek. *Extreme Programming* menekankan *communication*, *simplicity*, *feedback*, *courage*, dan *respect* sehingga sangat membantu dalam memanajemen proyek. Tahapan dalam metode *Extreme Programming* mulai dari data yang didapat dari objektif berupa observasi dan wawancara yang menjadi *User Stories* sebagai fondasi riset dan pengembangan. Selanjutnya yaitu *Architectural Spike* dengan tahap ini melakukan studi literatur untuk memperoleh data seperti metode dan teknologi apa yang akan digunakan kemudian disusun menjadi rancangan dasar dalam riset dan pengembangan sehingga dapat memperirakan *Release Planning* atau perkiraan lamanya riset dan pengembangan. Setelah itu masuk ke tahap *Iteration* untuk melakukan riset dan pengembangan pada proyek berdasarkan rancangan yang sudah dibuat dan disepakati dari pihak apotek dengan peneliti. Di tahap ini terjadi perulangan yang didasarkan oleh pengujian dan *user stories* yang baru. Tahapan selanjutnya adalah *Acceptance Test* merupakan tahapan yang -ulang sama berulang karena merupakan pengujian dari tahapan sebelumnya yakni *Iteration.* Disini dilakukan *white-box testing.* Setelah lulus akan diteruskan ketahap akhir yaitu *Deployment* [11]

d. Back-End

Back-End merupakan bagian tertutup atau belakang dari sebuah proyek. *Back-end* berisi komponen-komponen yang bertugas dalam *server* yakni menerima permintaan (*request*) dan tanggapan (*response*) dari pengguna. *Back-end* mengandung logika bisnis untuk mengelola proses permintaan sepeti validasi data, pengolahan pesanan, dan perhitungan. *Back-end* meyediakan *Application Programming Interface* (API) yang bertanggung jawab atas komunikasi antara *front-end*, *back-end*, dan *database*. *Back-end* berinteraksi dengan *database* untuk menyimpan, mengambil dan memperbarui data. Ada beberapa jenis *back-end* berdasarkan bahasa pemrogramman namun yang digunakan pada penelitian ini berbasis *Node.js* dengan kerangka kerja *Express.js* [12]

e. Framework Express.js

Express.js merupakan kerangka kerja *Node.js* yang didesain untuk membantu mempercepat dalam pengembang

proyek. Kerangka kerja ini bertanggungjawab dalam alat dan struktur *Back-End*. Arsitektur kerangka kerja ini merupakan *Model View Controller* (MVC). Model berguna mengatur logika bisnis dan interaksi dengan *database*. *View* berfungsi menampilkan data ke pengguna yang sesuai. *Controller* berfungsi menangani permintaan, memproses data, dan menghasilkan respon. *Express.JS* juga mengatur *routing* dan *middleware* untuk akses ke *database* guna memproses permintaan dan tanggapan berupa *create, read, update, delete* (CRUD) sesuai intruksi *Controller* dengan menghubungkan fungsi penanganan (*handler*). Penelitian ini direncanakan akan menggunakan kerangka kerja ini sebagai alat bantu pengembangan *back-end* [6] Kinerja *Express.js* juga sangat baik bila dikombinasikan dengan RESTful API secara performa dan penggunaan sumber daya optimal [13].

f. RESTful API

RESTful API merupakan salah satu jenis arsitektur *Application Programming Interface* (API) yang menggunakan metode komunikasi protokol HTTP seperti POST, GET, PUT, PATCH, dan DELETE untuk pertukaran data. *RESTful* API bersifat *Statelessness* yang artinya informasi status tidak disimpan di *sever* antara permintaan. Penggunaan *RESTful* API sangat penting karena mempengaruhi peforma, kecepatan respon, dan penggunaan sumber daya sehingga dapat mempengaruhi proses pada pertukaran data kepada pengguna [14]. API ini juga direncanakan akan digunakan pada penelitian ini.

g. MariaDB

MariaDB merupakan media penyimpanan berbasis MySQL yang populer untuk digunakan pada pengembangan berbagai aplikasi berbasis *website* karena kecepatan membaca data yang baik. Tidak hanya itu MariaDB juga bersifat *open-source* sehingga biaya implementasi menjadi lebih rendah. Berdasarkan penelitian kinerja menggunakan *relational database management system* (RDBMS) MariaDB membuktikan bahwa mendapatkan performa lebih baik dibandingkan dengan jenis lainnya dari segi kecepatan. Ini membuktikan MariaDB cocok digunakan dalam pengembangan aplikasi berbasis *website* [15].

h. White Box Testing

White box testing merupakan salah satu metode pengujian dalam pengembangan sistem informasi yang bertujuan memastikan program tersebut berjalan semestinya sebelum dibagikan kepada pengguna. Ada berbagai jenis cara *white box testing* seperti seperti *coverage test* yang terdiri dari pengujian *statement, branch, function, lines,* dan *uncovered lines* [7] *White box* juga terdapat pengujian lain seperti *flowchart, flowgraph, independent path,* uji skenario, dan *cyclomatic complexity* untuk menghitung ukuran kuantitatif dari kompleksitas logika program [16] dengan rumus sebagai berikut.

Rumus *Cyclomatic Complexity*:

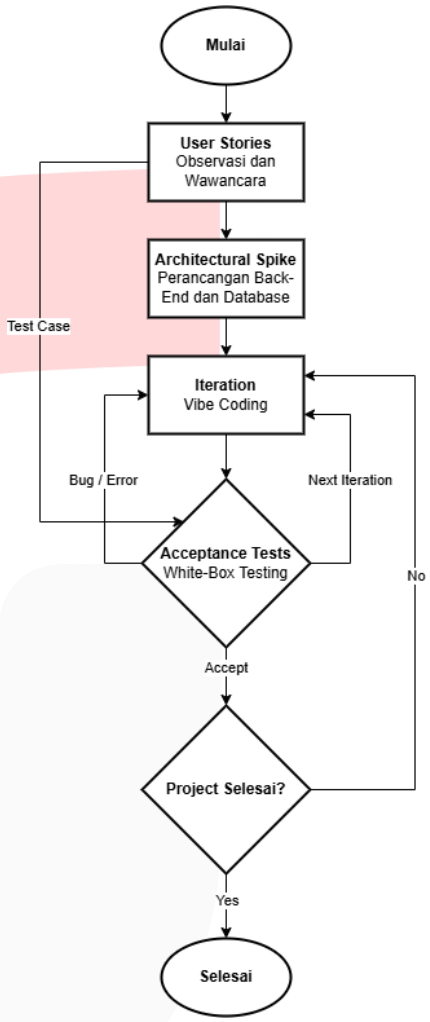
$$v(G) = E - N + 2$$

i. Unified Modeling Language (UML)

Unified Modeling Language (UML), adalah bahasa yang digunakan untuk memvisualisasikan pemodelan dan komunikasi terkait sistem dengan penggunaan skrip dan skema. Fungsi UML adalah mempermudah pengartian dari bahasa pemrograman menjadi bahasa model yang divisualisasikan. Penggunaan UML mengikuti pedoman dan standar yang telah ditentukan. Sekarang UML menggunakan versi 2.0 keatas sebagai pedoman utama dalam penggunaan. Pengimplementasian menggunakan empat jenis diagram sebagai pemodelan, yaitu: *Use Case Diagram*, *Activity Diagram*, dan *Sequence Diagram* [17].

III. METODE

Penelitian ini menggunakan pendekatan metode *Extreme Programming* dalam proses perancangan dan pengembangannya.



GAMBAR 1
(DIAGRAM ALIR)

a. Observasi dan Wawancara (*User Stories*)

Langkah pertama yang dilakukan oleh peneliti yaitu *user stories* untuk mendapatkan informasi yang kredibel berdasarkan yang terjadi di lapangan dan wawancara dari pemilik apotek. Dari hasil observasi dan wawancara dapat menyimpulkan permasalahan apa yang dihadapai, kebutuhan apa yang harus dikembangkan, dan solusi apa yang dapat ditawarkan kepada pemilik apotek Dian Brata Medika. Dengan mehami situasi dan kondisi di lingkungan apotek hasil wawancara dapat dikombinasikan dengan hasil observasi.

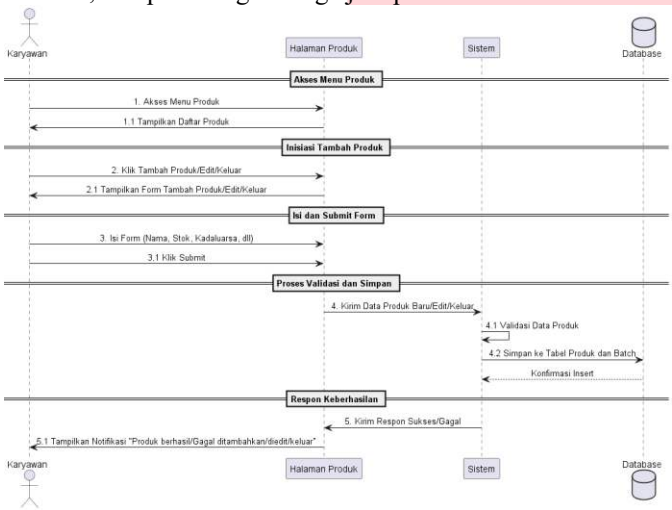
b. Studi Literature dan Perencanaan Back-End (*Architectural Spike*)

Langkah kedua yaitu mengumpulkan data dengan melakukan literature yakni penelitian terdahulu, dokumentasi, jurnal ilmiah yang disusun dan dibuat menjadi rancangan pengembangan *back-end* sistem manajemen

persediaan obat. Dalam rancangan tersebut solusi ditemukan dengan menawarkan sistem manajemen persediaan berbasis *website* dengan menggunakan metode *Extreme Programming* dan riset berfokus pada sistem *back-end* dari *website* tersebute dengan menggunakan kerangka kerja dari *Express.js*. Kemudian diperoleh fitur-fitur apa saja yang diperlukan oleh apotek Dian Brata Medika. Perancangan melibatkan penggunaan diagram sebagai berikut.

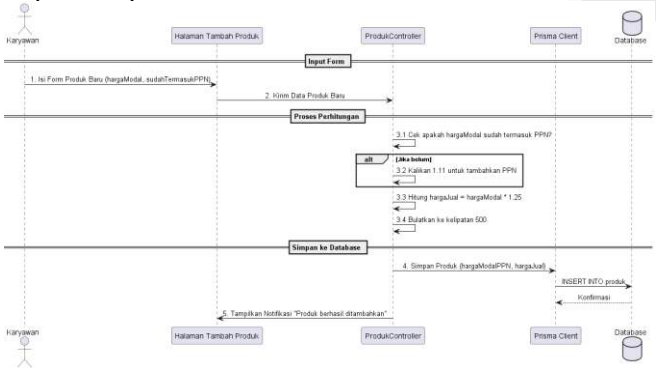
Analisa mendalam pada fitur-fitur sudah dilakukan dengan pengambaran *sequence* diagram. Langkah selanjutnya adalah memperdalam bagaimana sistem dapat menangani permintaaan dari pengguna. Sistem yang detail menangani permintaan pengguna dapat gambarkan pada sequence diagram. Pada sequence diagram terdapat logika program dari pengguna hingga *database* yang dapat menangani permintaan.

Sequence diagram yang paling utama adalah berdasarkan logika fitur produk masuk untuk perhitungan produk berdasarkan PPN, produk keluar untuk menerapkan prinsip FEFO, dan perhitungan harga jual produk.



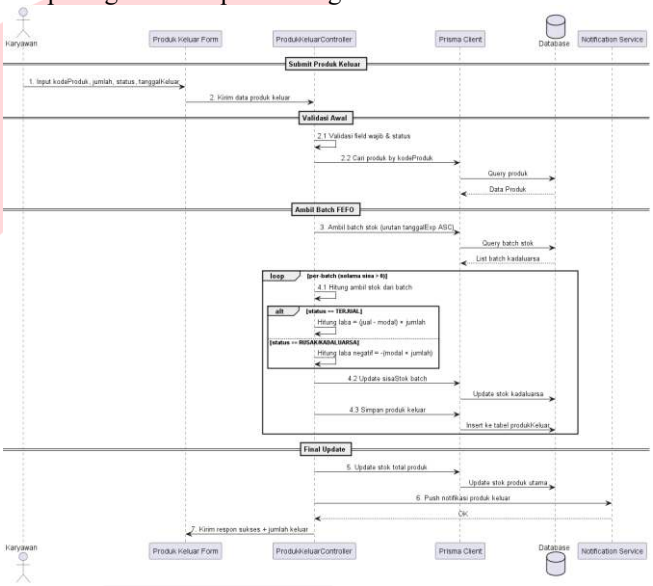
GAMBAR 2
(SEQUENCE PRODUK MASUK)

Pada fitur tersebut terdapat logika program yang khusus untuk menghitung nilai harga modal dan harga jual dengan melakukan perhitungan harga beli yang ditambahkan dengan pajak PPN 11% kemudian ditambahkan lagi dengan nilai keuntungan adalah 25% untuk harga jual. Detail alurnya terjadi ketika pengguna melakukan *submit* pada formulir pendaftaran. Ketika berhasil data yang dimasukan akan disimpan pada database yang kemudian dilanjutkan dengan memberikan respon berupa notifikasi.



GAMBAR 3
(SEQUENCE PERHITUNGAN HARGA JUAL)

Selanjutnya masih pada fitur produk yakni produk obat yang keluar yang kurang lebih sama dengan penambahan produk obat. Namun terdapat logika program yang berbeda yakni logika FEFO dan perhitungan laba. Karyawan memilih obat mana yang akan keluar kemudian mengirim permintaan untuk pengeluaran produk. Sistem pada kontroler melakukan validasi mengecek apakah ada produk serupa yang diminta kemudian mengambil datanya dan diurutkan batch produk berdasarkan tanggal kadaluarsa. Lalu dicek perbatch terdapat sisa persediaan yang dapat dikeluarkan lalu melakukan perhitungan laba yaitu harga jual dikurangi harga modal dikali jumlah produk sehingga menjadi laba keuntung. Jika produk yang keluar bersetatus rusak atau kadaluarsa akan melakukan perhitungan harga dengan melakukan pengurangan modal dikali produk obat yang keluar. Detail ada pada gambar sequence diagram berikut.



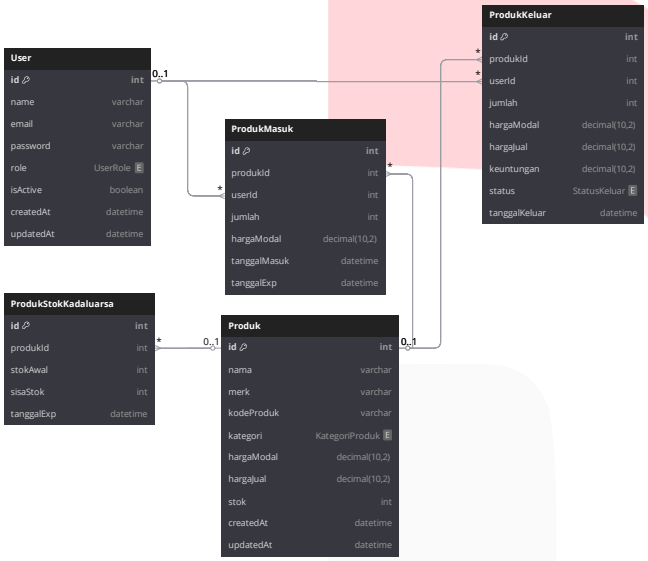
GAMBAR 4
(SEQUENCE PRODUK KELUAR FEFO)

Pada tahap perancangan database, sistem manajemen persediaan obat pada Apotek Dian Brata Medika digambarkan melalui satu skema *Entity Relationship Diagram* (ERD) yang mengintegrasikan lima entitas utama: *User*, Produk, Produk Stok Kadaluarsa, Produk Masuk, dan Produk Keluar. Entitas *User* merepresentasikan akun pengguna sistem baik karyawan dengan atribut dasar seperti nama, email, kata sandi, *role*, status aktif, serta tanggal update waktu pembuatan dan pembaruan data. Setiap pengguna karyawan dapat melakukan transaksi masuk dan keluar barang, sehingga tabel Produk Masuk dan Produk Keluar memuat kunci (*userId*) yang ditautkan riwayat aktivitas ke identitas pengguna yang bersangkutan.

Entitas Produk menyimpan data detail obat, mencakup nama, merek, kode unik, kategori (seperti obat bebas, obat keras, konsi, atau alkes), harga modal, harga jual, dan jumlah stok keseluruhan. Relasi *one-to-many* antara produk dan produk stok kadaluarsa memungkinkan pencatatan stok dalam batch berdasarkan tanggal kedaluwarsa. Konsep *first expired first out* (FEFO) dimana setiap baris Produk Stok Kadaluarsa mengandung stok awal, sisa stok, dan tanggal kedaluwarsa untuk satu batch. Selain itu, Produk Masuk dan Produk Keluar juga menautkan ke Produk lewat kunci (*produkId*), sehingga setiap peristiwa penambahan maupun

pengurangan stok tercatat dengan detail jumlah, harga modal, dan informasi tanggal.

Lebih jauh, tabel Produk Keluar dilengkapi kolom khusus untuk mencatat keuntungan per transaksi positif jika obat terjual dan negatif jika rusak atau kedaluwarsa serta status keluaran yang diatur melalui enum status keluar ("terjual", "kadaluarsa", "rusak", atau "tidak sesuai"). Dengan demikian, ERD ini tidak hanya menjamin integritas referensial antar tabel, tetapi juga mendukung implementasi logika program seperti perhitungan laba rugi dan pemilihan stok batch terdekat kedaluwarsa. Skema relasional ini selanjutnya akan diterjemahkan ke dalam struktur tabel MySQL dan diakses melalui sistem, menyediakan fondasi kokoh bagi implementasi backend berbasis *Express.js*. Untuk diagram dapat dilihat di bawah ini.



GAMBAR 5
(DIAGRAM ERD)

c. Iteration

Langkah ketiga yaitu langkah yang berkelanjutan atau *iteration*. Pada langkah ini peneliti membagi pengembangan sistem *back-end* dan *database* sesuai target yang ditentukan berdasarkan *story* yang telah disusun pada tahap *planning*. Sehingga peneliti dapat mengetahui apa yang harus dikerjakan terlebih dahulu dan bagian modul pengembangan yang harus diselesai terlebih dahulu. Ditahap ini peneliti sudah mulai melakukan pengembangan sistem *back-end* dan *database.* Langkah ini akan terus dilakukan selama modul yang telah dibuat lolos pada langkah *Acceptance tests*.

d. White Box Testing (Acceptance tests)

Pengujian white box meliputi berbagai macam metode seperti *coverage test* untuk menguji kode program internal secara langsung apakah dapat berjalan atau tidak sesuai *test case. Flowchart* dan *flowgraph* sebagai media penggamabaran alur program berdasarkan simpul dan panah. *Independent path* sebagai pengukuran jalur. Uji skenario berguna untuk menguji jalur, dan *cyclomatic complexity* untuk menghitung nilai jalur minium yang akan diuji.

e. Penyesuaian dan Perbaikan *(Retrospective)*

Pada tahap ini merupakan tahap yang lakukan bila pada tahap *Testing* tidak berhasil sesuai *test case.* Ditahap ini modul diperbaiki dan disesuaikan. Jika sudah modul diuji lagi pada tahap *Testing.*

IV. HASIL DAN PEMBAHASAN

a. Hasil Pengujian White Box

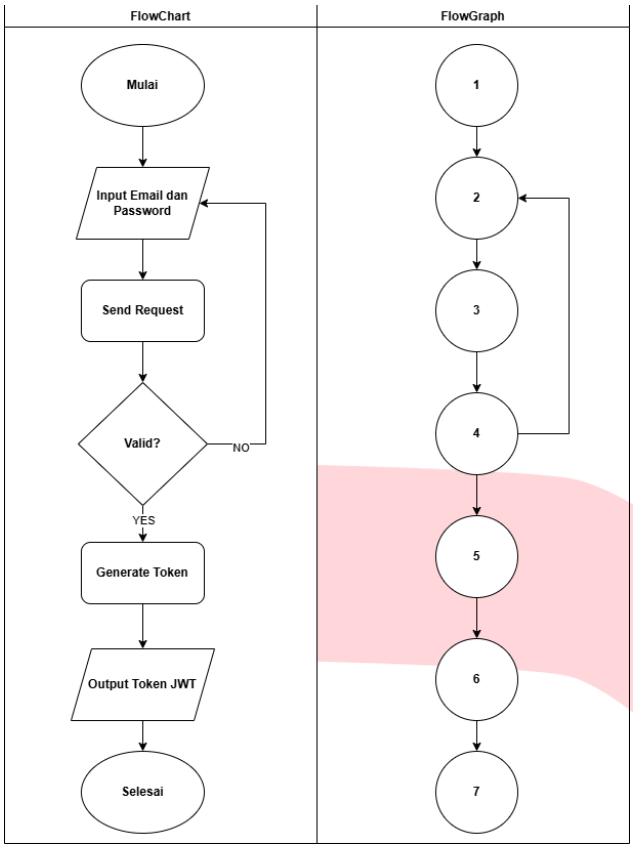
Pengujian *white box* dilakukan menggunakan berbagai metode yang digunakan dalam penelitian ini adalah *coverage test* untuk menguji kode program internal secara langsung apakah dapat berjalan atau tidak sesuai *test case. Flowchart* dan *flowgraph* sebagai media penggamabaran alur program berdasarkan simpul dan panah. *Independent path* sebagai pengukuran jalur. Uji skenario berguna untuk menguji jalur, dan *cyclomatic complexity* untuk menghitung nilai jalur minium yang akan diuji.

Pada *coverage test* dilakukan berbagai cara dalam menguji kode internal program. Cara pertama adalah *coverage statement* untuk menguji statement atau pernyataan dari kode program. Jika *statement* tersebut berjalan sesuai *test case* maka kemungkinan *error* jarang terjadi. Kedua adalah *coverage branch* untuk menguji percabangan dalam kode program. Percabangan harus diuji coba secara menyeluruh untuk memastikan bahwa semua kasus telah berjalan dan tidak *error*. Ketiga adalah *coverage function* untuk menguji fungsi kode program secara menyeluruh dari yang diuji sebelumnya yakni *statement* dan *branch.* Terakhir adalah *coverage lines dan uncovered lines.* Cara ini digunakan untuk mengecek baris kode program yang tereksekusi. Perbedaanya dengan *statement* adalah *coverage lines* menguji apakah semua baris program tersebut tereksekusi setidaknya satu kali. Jika tidak tereksekusi maka akan tampil pada *uncovered lines* dan diberi keterangan baris mana yang belum tereksekusi.

TABEL 1
(COVERAGE METRIC)

File	Statement	Branch	Func	Lines
auth	100%	100%	100%	100%
dashboard	100%	100%	100%	100%
laporan	100%	100%	100%	100%
notifikasi	100%	100%	100%	100%
product	100%	100%	100%	100%
product keluar	96%	100%	100%	95%
product masuk	97%	95%	100%	97%
riwayat	100%	100%	100%	100%
user	100%	100%	100%	100%

Dari hasil pengujian *coverage test* pada tabel diatas terlihat bahwa hampir semua pengujian sempurna. Kode program yang diuji berhasil berjalan sesuai *test case*. Terdapat beberapa baris kode program yang belum tereksekusi oleh test case tapi tidak menghambat atau mengganggu jalannya program hingga membuat *bug* atau *error.* Langkah selanjutnya adalah pengujian dalam menentukan kompleksitas pada alur program berdasarkan jalur-jalur yang dihasilkan dari perhitungan *cyclomatic complexity.* Perhitungan ini dilakukan pada setiap fitur yang telah dikembangkan. Berikut hasil pengujianya dimulai dari *flowchar*t dan *flowgraph* dari fitur *login*.



GAMBAR 6
(FLOW LOGIN)

Pengujian white box pada *endpoint login* ini dilakukan untuk menentukan jumlah jalur menggunakan rumus *cycleomatic complexity* atau (v(G)) dan dilakukan perhitungan sebagai berikut.

V(G) = E – N + 2
V(G) = 7 – 7 + 2
V(G) = 2

Dari hasil perhitungan *cycleomatic complexity* pada *endpoint login* mendapatkan 2 jalur untuk pengujian skenario. Jalur-jalur tersebut sebagai berikut.

Jalur 1 = 1-2-3-4-5-6-7

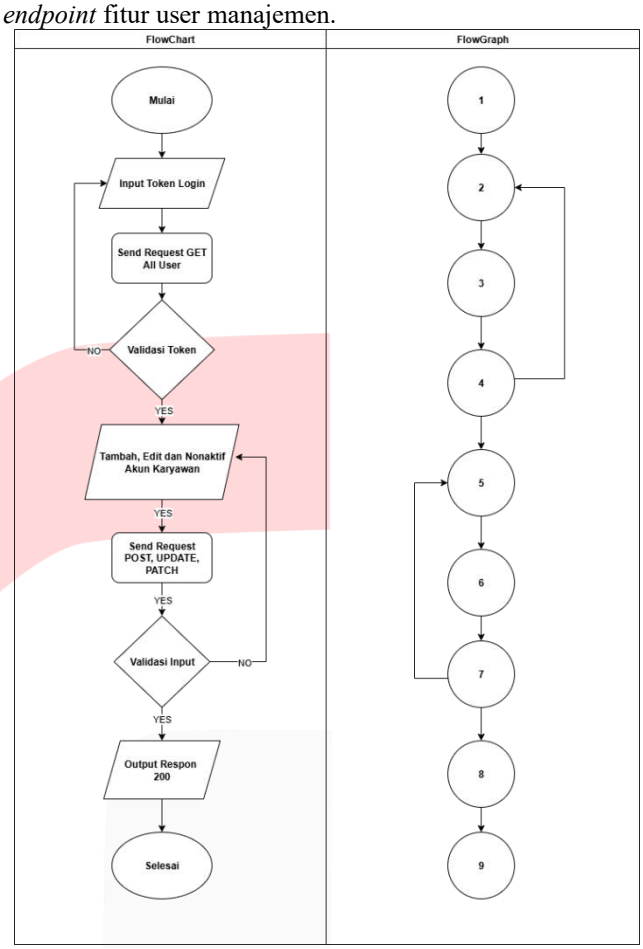
Jalur 2 = 1-2-3-4-2-3-4-5-6-7

Pengujan skenario dapat dilihat dari tabel berikut.

TABEL 2
(TEST SKENARIO LOGIN)

Path	**1**	**2**
Jalur	1-2-3-4-5-6-7	1-2-3-4-2-3-4-5-6-7
Skenario	1. Mulai 2. Input Email dan Password 3. Send Request 4. Validasi Token 5. Generate Token 6. Output Token JWT 7. Selesai	1. Mulai 2. Input Email dan Password 3. Send Request 4. Validasi Token 2. Input Email dan Password 3. Send Request 4. Validasi Token 5. Generate Token 6. Output Token JWT 7. Selesai
Hasil	Berhasil	Berhasil

Pada tabel pengujian *white box* skenario *endpoint login* pada kedua jalur berhasil dijalankan. Terbukti dari kedua jalur tersebut berjalan sesuai *test case.* Selanjutnya ke *endpoint* fitur user manajemen.



GAMBAR 7
(FLOW USER MANAJEMEN)

Pengujian white box pada *endpoint user* manajemen ini dilakukan untuk menentukan jumlah jalur menggunakan rumus *cycleomatic complexity* atau (v(G)) dan dilakukan perhitungan sebagai berikut.

V(G) = E – N + 2
V(G) = 10 – 9 + 2
V(G) = 3

Dari hasil perhitungan *cycleomatic complexity* pada *endpoint login* mendapatkan 3 jalur untuk pengujian skenario. Jalur-jalur tersebut sebagai berikut.

Jalur 1 = 1-2-3-4-5-6-7-8-9
Jalur 2 = 1-2-3-4-2-3-4-5-6-7-8-9
Jalur 3 = 1-2-3-4-5-6-7-5-6-7-8-9

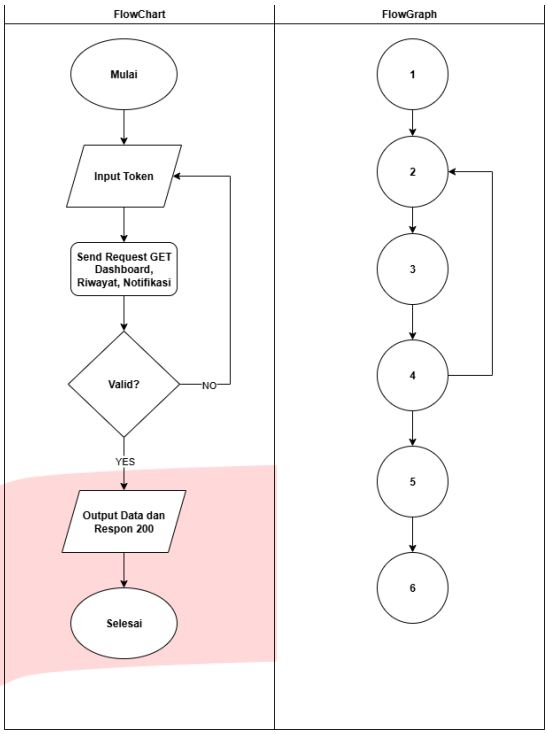
Pengujan skenario dapat dilihat dari tabel berikut.

TABEL 3
(TEST SKENARIO USER MANAJEMEN)

Path	**1**	**2**	**3**
Jalur	1-2-3-4-5-6-7-8-9	1-2-3-4-2-3-4-5-6-7-8-9	1-2-3-4-5-6-7-5-6-7-8-9
Skenario	1. Mulai 2. Input Token Login 3. Send Request	1. Mulai 2. Input Token Login	1. Mulai 2. Input Token Login

	GET All User 4. Validasi Token 5. Tambah, Edit dan Nonaktif Akun Karyawan 6. Send Request POST, UPDATE, PATCH 7. Validasi Input 8. Output Respon 200 9. Selesai	3. Send Request GET All User 4. Validasi Token 2. Input Token Login 3. Send Request GET All User 4. Validasi Token 5. Tambah, Edit dan Nonaktif Akun Karyawan 6. Send Request POST, UPDATE, PATCH 7. Validasi Input 8. Output Respon 200 9. Selesai	3. Send Request GET All User 4. Validasi Token 5. Tambah, Edit dan Nonaktif Akun Karyawan 6. Send Request POST, UPDATE, PATCH 7. Validasi Input 8. Output Respon 200 5. Tambah, Edit dan Nonaktif Akun Karyawan 6. Send Request POST, UPDATE, PATCH 7. Validasi Input 9. Selesai
Hasil	Berhasil	Berhasil	Berhasil

Pada tabel pengujian *white box* skenario *endpoint user* manajemen pada kedua jalur berhasil dijalankan. Selanjutnya ke *endpoint* fitur publik seperti *dashboard*, riwayat, dan notifikasi.



GAMBAR 8
(FLOW FITUR PUBLIK)

Pengujian white box pada *endpoint* fitur publik ini dilakukan untuk menentukan jumlah jalur menggunakan rumus *cycleomatic complexity* atau (v(G)) dan dilakukan perhitungan sebagai berikut.

V(G) = E – N + 2
V(G) = 6 – 6 + 2
V(G) = 2

Dari hasil perhitungan *cycleomatic complexity* pada *endpoint* fitur publik mendapatkan 2 jalur untuk pengujian skenario. Jalur-jalur tersebut sebagai berikut.

Jalur 1 = 1-2-3-4-5-6
Jalur 2 = 1-2-3-4-2-3-4-5-6

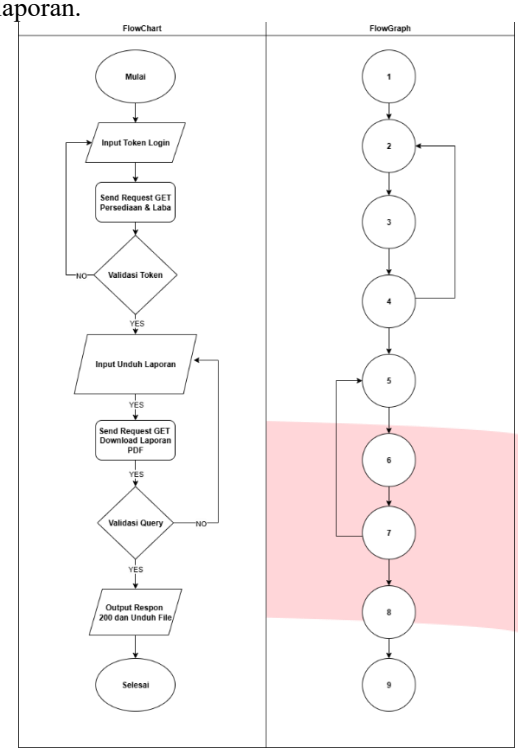
Pengujan skenario dapat dilihat dari tabel berikut.

TABEL 9
(TEST SKENARIO FITUR PUBLIK)

Path	**1**	**2**
Jalur	1-2-3-4-5-6	1-2-3-4-2-3-4-5-6
Skenario	1. Mulai 2. Input Token 3. Send Request GET Dashboard, Riwayat, Notifikasi 4. Validasi Token 5. Output Data respon 200 6. Selesai	1. Mulai 2. Input Token 3. Send Request GET Dashboard, Riwayat, Notifikasi 4. Validasi Token 2. Input Token 3. Send Request GET Dashboard, Riwayat, Notifikasi 4. Validasi Token 5. Output Data respon 200 6. Selesai
Hasil	Berhasil	Berhasil

Pada tabel pengujian *white box* skenario *endpoint* fitur publik

pada kedua jalur berhasil dijalankan. Selanjutnya ke *endpoint* fitur laporan.



GAMBAR 11
(FLOW LAPORAN)

Pengujian white box pada *endpoint* fitur laporan ini dilakukan untuk menentukan jumlah jalur menggunakan rumus *cycleomatic complexity* atau (v(G)) dan dilakukan perhitungan sebagai berikut.

V(G) = E – N + 2
V(G) = 10 – 9 + 2
V(G) = 3

Dari hasil perhitungan *cycleomatic complexity* pada *endpoint* fitur laporan mendapatkan 3 jalur untuk pengujian skenario. Jalur-jalur tersebut sebagai berikut.

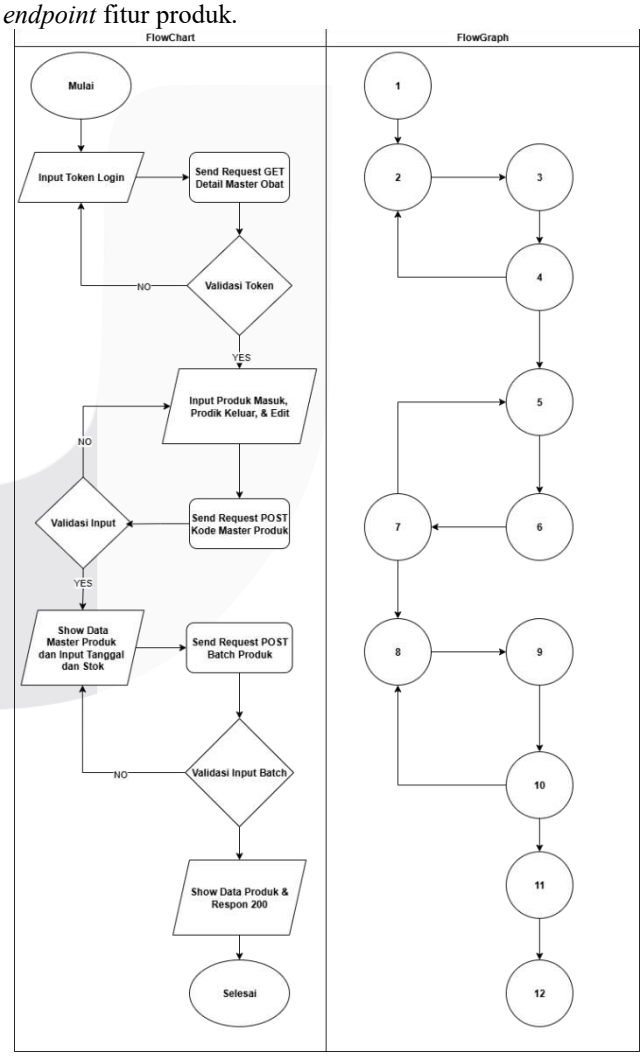
Jalur 1 = 1-2-3-4-5-6-7-8-9
Jalur 2 = 1-2-3-4-2-3-4-5-6-7-8-9
Jalur 3 = 1-2-3-4-5-6-7-5-6-7-8-9

Pengujan skenario dapat dilihat dari tabel berikut.

TABEL 10
(TEST SKENARIO LAPORAN)

Path	**1**	**2**	**3**
Jalur	1-2-3-4-5-6-7-8-9	1-2-3-4-2-3-4-5-6-7-8-9	1-2-3-4-5-6-7-5-6-7-8-9
Skenario	1. Mulai 2. Input Token Login 3. Send Request GET Persediaan & Laba 4. Validasi Token 5. Input Unduh Laporan 6. Send Request GET Download Laporan 7. Validasi Query 8. Output Respon 200 dan Unduh File 9. Selesai	1. Mulai 2. Input Token Login 3. Send Request GET Persediaan & Laba 4. Validasi Token 2. Input Token Login 3. Send Request GET Persediaan & Laba 4. Validasi Token 5. Input Unduh Laporan 6. Send Request GET Download Laporan 7. Validasi Query 8. Output Respon 200 dan Unduh File 9. Selesai	1. Mulai 2. Input Token Login 3. Send Request GET Persediaan & Laba 4. Validasi Token 5. Input Unduh Laporan 6. Send Request GET Download Laporan 7. Validasi Query 5. Input Unduh Laporan 6. Send Request GET Download Laporan 7. Validasi Query 8. Output Respon 200 dan Unduh File 9. Selesai
Hasil	Berhasil	Berhasil	Berhasil

Pada tabel pengujian *white box* skenario *endpoint* fitur laporan pada ketiga jalur berhasil dijalankan. Selanjutnya ke *endpoint* fitur produk.



GAMBAR 12
(FLOW PRODUK0

Pengujian white box pada *endpoint* fitur produk ini dilakukan untuk menentukan jumlah jalur menggunakan rumus *cycleomatic complexity* atau (v(G)) dan dilakukan perhitungan sebagai berikut.

V(G) = E – N + 2
V(G) = 14 – 12 + 2
V(G) = 4

Dari hasil perhitungan *cycleomatic complexity* pada *endpoint* fitur laporan mendapatkan 4 jalur untuk pengujian skenario. Jalur-jalur tersebut sebagai berikut.

Jalur 1 = 1-2-3-4-5-6-7-8-9-10-11-12
Jalur 2 = 1-2-3-4-2-3-4-5-6-7-8-9-10-11-12
Jalur 3 = 1-2-3-4-5-6-7-5-6-7-8-9-10-11-12
Jalur 4 = 1-2-3-4-5-6-7-8-9-10-8-9-10-11-12

Pengujan skenario dapat dilihat dari tabel berikut.

TABEL 1
(TEST SKENARIO PRODUK)

Path	**1**	**2**	**3**	**4**
Jalur	1-2-3-4-5-6-7-8-9-10-11-12	1-2-3-4-2-3-4-5-6-7-8-9-10-11-12	1-2-3-4-5-6-7-5-6-7-8-9-10-11-12	1-2-3-4-5-6-7-8-9-10-8-9-10-11-12
Skenario	1. Mulai 2. Input Token Login 3. Send Request GET Detail Master Obat 4. Validasi Token 5. Input Produk Masuk, Keluar, dan Edit 6. Send Request POST Kode Master Produk 7. Validasi Input 8. Show Data Master Produk dan Input Tanggal dan Stok 9. Send Request Post Batch Produk 10. Validasi Input Batch 11. Show Data Produk dan Respon 200 12. Selesai	1. Mulai 2. Input Token Login 3. Send Request GET Detail Master Obat 4. Validasi Token 2. Input Token Login 3. Send Request GET Detail Master Obat 4. Validasi Token 5. Input Produk Masuk, Keluar, dan Edit 6. Send Request POST Kode Master Produk 7. Validasi Input 8. Show Data Master Produk dan Input Tanggal dan Stok 9. Send Request Post Batch Produk 10. Validasi Input Batch 11. Show Data Produk dan Respon 200 12. Selesai	1. Mulai 2. Input Token Login 3. Send Request GET Detail Master Obat 4. Validasi Token 5. Input Produk Masuk, Keluar, dan Edit 6. Send Request POST Kode Master Produk 7. Validasi Input 5. Input Produk Masuk, Keluar, dan Edit 6. Send Request POST Kode Master Produk 7. Validasi Input 8. Show Data Master Produk dan Input Tanggal dan Stok 9. Send Request Post Batch Produk 10. Validasi Input Batch 11. Show Data Produk dan Respon 200 12. Selesai	1. Mulai 2. Input Token Login 3. Send Request GET Detail Master Obat 4. Validasi Token 5. Input Produk Masuk, Keluar, dan Edit 6. Send Request POST Kode Master Produk 7. Validasi Input 8. Show Data Master Produk dan Input Tanggal dan Stok 9. Send Request Post Batch Produk 10. Validasi Input Batch 11. Show Data Produk dan Respon 200 12. Selesai	1. Mulai 2. Input Token Login 3. Send Request GET Detail Master Obat 4. Validasi Token 5. Input Produk Masuk, Keluar, dan Edit 6. Send Request POST Kode Master Produk 7. Validasi Input 8. Show Data Master Produk dan Input Tanggal dan Stok 9. Send Request Post Batch Produk 10. Validasi Input Batch 11. Show Data Produk dan Respon 200 12. Selesai
Hasil	Berhasil	Berhasil	Berhasil	Berhasil

Secara keseluruhan dari hasil pengujian white box pada kode internal program dari website apotek Dian Brata Medika berhasil berjalan dan berfungsi semestinya. Berdasarkan hasil tersebut juga program bisa diimplementasikan segera dan dilanjutkan ketahap *deployment*.

b. Hasil Implementasi

Setelah berhasil diuji dengan hasil yang sesuai *test case* dilanjutkan dengan proses *deployment*. Deploy pada sistem *back-end* ini dilakukan dengan menggunakan layanan *Cloud VPS* sebagai *server vitual* tempat sistem agar bisa berjalan secara otomatis. Kemudian VPS dihubungkan ke alamat domain yang sudah tersedia yakni www.dianbratamedika.com dan terindex oleh google search.

V. KESIMPULAN

Berdasarkan hasil penelitian dan implementasi sistem informasi manajemen persediaan obat pada apotek independen Dian Brata Medika, maka dapat disimpulkan hasil pengujian *white box* pada sistem *back-end* berhasil dikembangkan dan memenuhi kebutuhan apotek. Manajemen menerapkan prinsip FEFO dengan baik. Sistem ini juga membantu dalam mengkontrol aktivitas karyawan dan alur keuangan apotek.

REFERENSI

[1] Menteri Kesehatan RI, *Peraturan Menteri Kesehatan Republik Indonesia Nomor 9 Tahun 2017 Tentang Apotek*. Indonesia, 2017, pp. 1–36.

[2] M. Rasyidan and I. Fikri, "Perancangan Aplikasi Pengelolaan Data Penyaluran Obat Dan Pemusnahan Obat Kadaluarsa Berbasis Web Dengan Metode Scrum," *JSSI*, vol. 1, no. 3, 2023, [Online]. Available: https://ojs.uniska-bjm.ac.id/index.php/JSSI
[3] A. Khan, S. Faisal, and D. Aboud, "An Analysis of Optimal Inventory Accounting Models - Pros and Cons," *http://www.eajournals.org/wp-content/uploads/An-Analysis-of-Optimal-Inventory-Accounting-Models-Pros-and-Cons.pdf*, vol. Vol.6, pp. 65–77, Apr. 2018.
[4] A. Shrivastava, I. Jaggi, N. Katoch, D. Gupta, and S. Gupta, "A Systematic Review on Extreme Programming," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jul. 2021. doi: 10.1088/1742-6596/1969/1/012046.
[5] O. Chaplia and H. Klym, *Node.js Framework for Automated Code Scaling and Execution on Multiple Distributed Machines*. 2023. doi: 10.1109/CSIT61576.2023.10324052.
[6] V. Sahni, A. Chopde, M. Goswami, and A. Kumar, "Mern (Mongodb , Express-Js, React-Js, Node-Js) Stack Web-Based Themefied Education Platform For Placement Preparation," *Educational Administration: Theory and Practice*, 2024, doi: 10.53555/kuey.v30i5.3035.
[7] A. Martin-Lopez, A. Arcuri, S. Segura, and A. Ruiz-Cortés, "Black-Box and White-Box Test Case Generation for RESTful APIs: Enemies or Allies?," *IEEE Softw*, pp. 1–11, 2022, doi: https://doi.org/10.1109/ISSRE52982.2021.00034.
[8] F. S. Ilahi, A. M. Yumassik, and M. M. A. Saputera, "Pembuatan Aplikasi Penyimpanan Obat (Apo) Dalam Pengontrolan Masa Kadaluarsa Obat Di Apotek Berbasis Web," *Jurnal Insan Farmasi Indonesia*, vol. 6, no. 1, pp. 72–81, May 2023, doi: 10.36387/jifi.v6i1.1301.
[9] J. Triputra, D. Mulyanti, J. Phh, M. No, and J. B. 41 Bandung, "Pengembangan Sistem Manajemen Farmasi Di Pelayanan Kesehatan Rumah Sakit: Studi Teoritis," *Jurnal Kesehatan Amanah*, vol. 7, no. 1, 2023.
[10] Quvvatov, Behruz, and U. Son, "Web Front-End And Back-End Technologies In Programming," *THEORETICAL ASPECTS IN THE FORMATION OF PEDAGOGICAL SCIENCES*, vol. 3, no. 1, pp. 208–215, 2024, doi: 10.5281/zenodo.10518360.
[11] R. Sasono, S. Kristanto, L. Hakim, and D. Yusuf, "Optimasi Web Service REST Pada Backend Aplikasi Prospect Menggunakan Metode Extreme Programming," *JOURNAL ZETROEM*, vol. 7, pp. 96–103, Mar. 2025, doi: 10.36526/ztr.v7i1.4202.
[12]J. Khatib Sulaiman Dalam No, N. Aqil Himawan, T. Fabrianti Kusumasari, and N. Ichsan Utama, "Enhancing the Performance of Desk Evaluation: Final Project Web Application through Back-End Maintenance," *Indonesian Journal of Computer Science Attribution*, vol. 12, no. 4, p. 1811, 2023.
[13]L. Mulana, K. Prihandani, A. Rizal, U. Singaperbanga, and K. Abstract, "Analisis Perbandingan Kinerja Framework Codeigniter Dengan Express.Js Pada Server RESTful Api," *Jurnal Ilmiah Wahana Pendidikan*, vol. 8, no. 16, pp. 316–326, 2022, doi: 10.5281/zenodo.
[14]M. Jonsson, E. Qvarnström, R. Lindell, and J. Gustafsson, "A Performance Comparison On Rest-Apis In Express.Js, Flask And Asp.Net Core," *Digitala Vetenskapliga Arkivet*, pp. 1–33, 2022, [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1669487&dswid=2315
[15] J. Fakultas Matematika dan Ilmu Pengetahuan Alam, "Analisis Kinerja Web Server pada SIM Manajemen Diklat Poltekpel Sorong Menggunakan RDBMS MySQL dan MariaDB," 2020.
[16] D. Bagus Sumantry, M. Lulu Latif Usman, and Y. Setiya Rafika Nur, "Pengembangan Rest API Untuk Monitoring Daerah Rawan Narkoba Menggunakan Framework ExpressJs Dengan Metode Scrum."
[17] F. Ciccozzi, I. Malavolta, and B. Selic, "Execution of UML models: a systematic review of research and practice," *Softw Syst Model*, vol. 18, no. 3, pp. 2313–2360, Jun. 2019, doi: 10.1007/s10270-018-0675-4.