# Implementasi Xendit Payment Link dan Webhook untuk Otomatisasi Penagihan Langganan pada Bot Telegram

Renaldy Permana Sundawa[1], Agus Prihanto[2]

[1,2] Teknik Informatika, Universitas Negeri Surabaya
[1]renaldy.22112@mhs.unesa.ac.id
[2]agusprihanto@unesa.ac.id

***Abstrak*— Layanan Rafess, yaitu layanan automenfess yang menerapkan sistem berlangganan berbasis bot Telegram, menghadapi permasalahan pada proses penagihan yang masih dilakukan secara manual sehingga rawan kesalahan dan bergantung pada ketersediaan admin. Oleh karena itu, penelitian ini menawarkan solusi melalui implementasi sistem otomatisasi penagihan langganan dengan integrasi Xendit Payment Link dan webhook yang memungkinkan pembuatan tautan pembayaran otomatis, pembaruan status langganan secara real-time, serta pengiriman pengingat kepada pelanggan menjelang masa berakhir langganan. Hasil pengujian fungsional menggunakan metode Black Box Testing menunjukkan bahwa seluruh fitur utama sistem berjalan sesuai spesifikasi kebutuhan yang telah ditetapkan. Pengujian waktu respon terhadap 30 sampel transaksi menghasilkan rata-rata durasi end-to-end sebesar 1,963 detik (SD = 0,146 detik), menunjukkan sistem merespons secara konsisten di seluruh sampel pengujian. Komponen pemrosesan webhook internal mencatatkan rata-rata 0,535 detik dengan standar deviasi 0,012 detik, menunjukkan arsitektur FastAPI dan MongoDB mampu menangani pemrosesan secara stabil meskipun sistem berjalan pada infrastruktur terbatas. Sistem juga terbukti andal dalam menangani skenario gangguan melalui mekanisme retry webhook otomatis, sinkronisasi manual, serta idempotency untuk mencegah duplikasi data transaksi.**

***Kata Kunci*— Xendit Payment Link, Webhook, Bot Telegram, Otomatisasi Penagihan, FastAPI, MongoDB.**

## I. PENDAHULUAN

Perkembangan teknologi informasi yang pesat telah memberikan dampak signifikan terhadap berbagai sektor, termasuk dalam bidang layanan digital dan otomatisasi komunikasi. Salah satu platform yang banyak dimanfaatkan dalam berbagai bidang adalah Telegram, sebuah aplikasi pesan instan berbasis cloud yang menyediakan Application Programming Interface (API) untuk memungkinkan pengembang menciptakan sistem bot otomatis dengan berbagai fungsi [1]. Pemanfaatan bot Telegram telah banyak diterapkan untuk mendukung kegiatan akademik, bisnis, dan layanan publik karena bersifat ringan, multiplatform, serta memiliki tingkat keamanan yang tinggi melalui sistem enkripsi end-to-end [2].

Salah satu penerapan bot Telegram yang berkembang adalah sistem automenfess, yaitu layanan pengiriman pesan anonim yang diotomatisasi. Dalam penelitian ini, layanan yang menjadi objek adalah Rafess, yaitu layanan automenfess berbasis Telegram yang menggunakan bot untuk mengelola proses pengiriman pesan secara otomatis. Namun, seiring meningkatnya jumlah pengguna, proses penagihan langganan bulanan masih dilakukan secara manual. Pemilik layanan harus mengirim pengingat pembayaran, memverifikasi mutasi rekening, dan merekap data transaksi secara manual. Kondisi tersebut menyebabkan proses verifikasi menjadi lambat, bergantung pada ketersediaan admin, dan rawan terjadi kesalahan manusia.

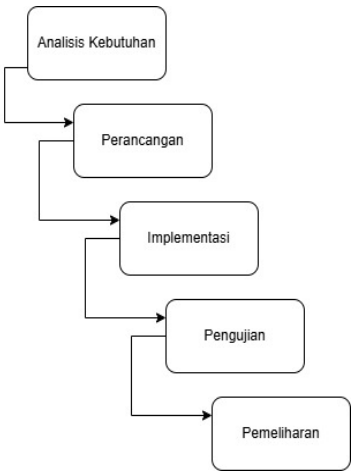
Permasalahan serupa juga ditemukan pada penelitian sebelumnya terkait sistem langganan digital. Proses penagihan manual menyebabkan keterlambatan dan membatasi jangkauan pelanggan pada sistem Software as a Service (SaaS). Implementasi payment gateway Stripe dengan metode otomatisasi terbukti mempercepat proses verifikasi pembayaran serta memperluas jangkauan pelanggan internasional [3]. Selain itu, integrasi payment gateway Midtrans dengan metode webhook pada bot Telegram untuk layanan reservasi dapat mempercepat konfirmasi pembayaran dan memperbarui status transaksi secara otomatis [4]. Penggunaan payment gateway Midtrans juga terbukti meningkatkan efisiensi transaksi dan mengurangi kesalahan input manual pada sistem pembayaran berbasis web [5]. Implementasi payment gateway pada sistem tiket pendakian juga menunjukkan bahwa integrasi webhook mampu memperbarui status pembayaran secara otomatis tanpa intervensi admin [6].

Metode webhook merupakan pendekatan komunikasi antara server dengan server yang hanya aktif ketika suatu peristiwa tertentu terjadi, seperti notifikasi pembayaran [7]. Pendekatan ini berbeda dengan metode long polling karena tidak memerlukan permintaan berulang dari server, melainkan langsung merespon ketika suatu event dipicu, sehingga sistem menjadi lebih efektif [8]. Penerapan metode webhook pada sistem berbasis bot Telegram terbukti mampu memberikan respons real-time kepada pengguna [9]. Selain itu, integrasi bot Telegram dengan metode webhook juga menunjukkan peningkatan efisiensi pemrosesan laporan hingga 75% dengan rata-rata waktu respons 2,3 detik [10]. Dalam konteks sistem pembayaran, webhook berfungsi untuk menerima notifikasi otomatis dari penyedia layanan seperti Xendit, Midtrans, atau Stripe setelah transaksi berhasil, kemudian memperbarui data di sisi aplikasi tanpa intervensi manual.

Berdasarkan kondisi tersebut, dibutuhkan sistem otomatisasi penagihan langganan untuk bot Telegram Rafess dengan mengintegrasikan Xendit Payment Link dan webhook. Xendit menyediakan sistem pembayaran yang mudah diintegrasikan dengan aplikasi pihak ketiga dan mendukung berbagai metode pembayaran. Integrasi webhook pada sistem memungkinkan proses verifikasi pembayaran berjalan otomatis dan status langganan diperbarui secara langsung di basis data tanpa campur tangan pemilik layanan.

## II. METODE PENELITIAN

Metode pengembangan sistem yang digunakan dalam penelitian ini adalah *System Development Life Cycle* (SDLC) dengan model Waterfall. Model ini dipilih sesuai dengan karakteristik pengembangan otomatisasi penagihan langganan. Alur tahapan penelitihan dapat dilihat pada Gbr 1.




Gbr. 1 Alur Penelitian

### A. Analisis Kebutuhan

Analisis kebutuhan dilakukan melalui pengamatan langsung terhadap proses penagihan pada layanan Rafess yang sebelumnya masih berjalan secara manual. Dari hasil pengamatan, ditemukan bahwa proses verifikasi pembayaran bergantung pada ketersediaan admin, rawan kesalahan manusia, dan tidak dapat berjalan di luar jam operasional. Berdasarkan kondisi tersebut, ditetapkan kebutuhan fungsional sistem sebagaimana ditunjukkan pada Tabel I.

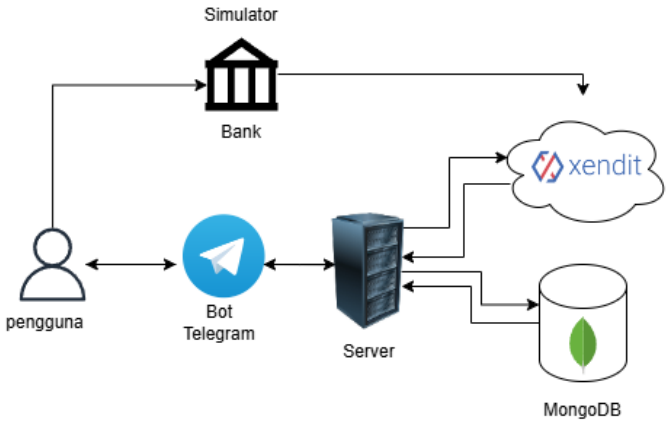
Tabel I
Kebutuhan Fungsional Sistem

| No | Kebutuhan Fungsional | Deskripsi |
|---|---|---|
| 1 | Pendaftaran Base | Sistem menyediakan perintah pendaftaran base untuk admin dan menyimpan data base ke MongoDB setelah validasi format |
| 2 | Pilihan Periode Langganan | Sistem menyediakan tiga pilihan periode langganan: harian (1 hari), mingguan (7 hari), dan bulanan (30 hari) dengan harga berbeda tiap paket |
| 3 | Pembuatan Tautan Pembayaran | Sistem membuat tautan pembayaran menggunakan API Xendit dan mengirimkannya ke pengguna melalui bot |
| 4 | Verifikasi Pembayaran | Sistem menerima notifikasi dari Xendit, memvalidasi callback token, memperbarui status invoice dan langganan di database, serta mengirimkan notifikasi ke pengguna |
| 5 | Notifikasi Pembayaran Kedaluwarsa | Sistem mendeteksi status EXPIRED dari webhook dan mengirimkan notifikasi kepada pengguna |
| 6 | Pengingat Penagihan | Sistem mengirimkan pengingat perpanjangan sebanyak tiga kali, yaitu pada H-3, H-2, dan H-1 sebelum masa aktif berakhir, disertai tautan pembayaran baru |
| 7 | Sinkronisasi Manual | Sistem menyediakan perintah untuk mengambil ulang status transaksi dari Xendit apabila webhook tidak diterima |
| 8 | Pencegahan Duplikasi | Sistem mencegah pemrosesan ulang notifikasi webhook yang sama menggunakan pengecekan event_id di MongoDB |
| 9 | Laporan Pendapatan | Sistem menyediakan perintah yang menampilkan ringkasan pendapatan 30 hari terakhir |

Dari sisi perangkat lunak, sistem dikembangkan menggunakan Python 3.10 dengan framework FastAPI, MongoDB sebagai basis data, serta pustaka PyTelegramBotAPI, PyMongo, APScheduler, dan Requests. Pada sisi perangkat keras, pengembangan dilakukan pada laptop AMD Ryzen 5 3550H RAM 16 GB, sedangkan penerapan sistem berjalan pada home server ARM RAM 2 GB yang terhubung ke jaringan publik melalui Cloudflare Tunnel.

### B. Rancangan Sistem

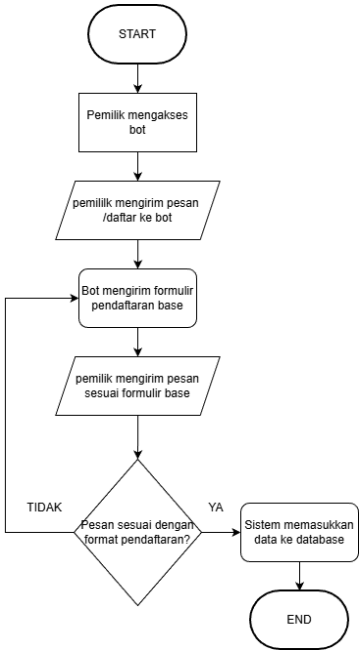
Sistem dirancang menggunakan arsitektur terpusat (*centralized architecture*) yang menempatkan server sebagai pusat kendali yang menghubungkan seluruh komponen sistem. Tiga komponen utama yang terhubung ke server adalah Bot Telegram sebagai antarmuka pengguna, Xendit sebagai payment gateway, MongoDB sebagai basis data. Arsitektur jaringan sistem ditunjukkan pada Gbr. 2.




Gbr. 2 Arsitektur Jaringan Sistem

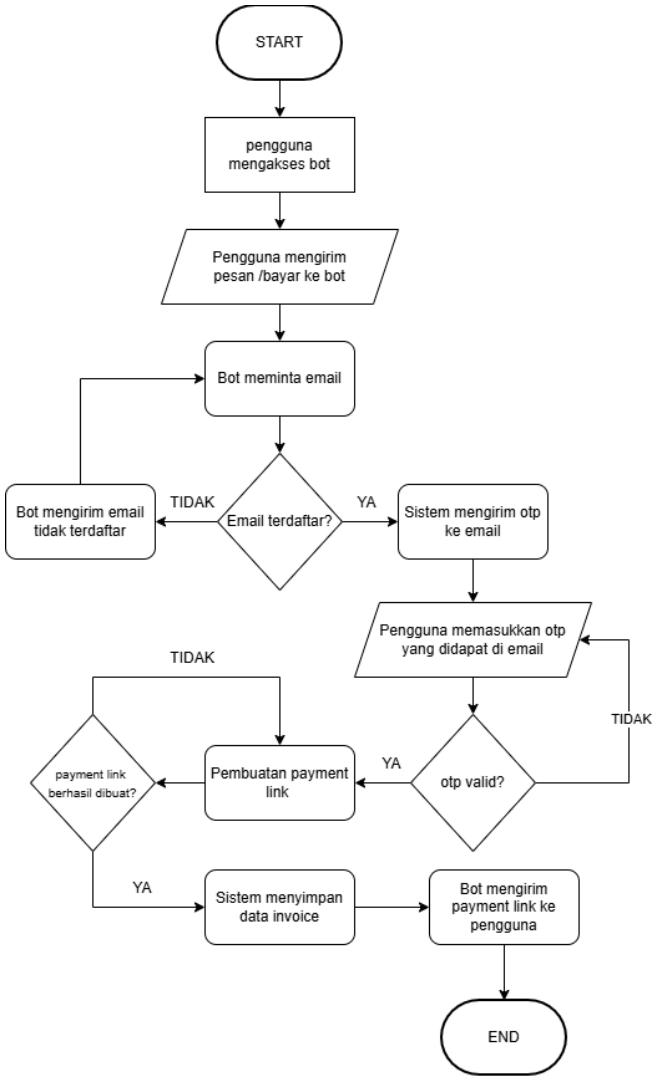
Alur proses dirancang dalam empat flowchart utama sebagai berikut.

*1) Alur Pendaftaran Base*: Alur pendaftaran base memfasilitasi admin dalam menambahkan base baru ke dalam sistem melalui bot Telegram. Proses dimulai ketika admin mengirimkan perintah /daftar, bot mengirimkan formulir pendaftaran, admin mengisi dan mengirimkan data, lalu sistem memvalidasi format input. Jika format tidak sesuai, bot mengirim ulang formulir; jika sesuai, sistem menyimpan data base ke MongoDB. Alur pendaftaran base ditunjukkan pada Gbr. 3.



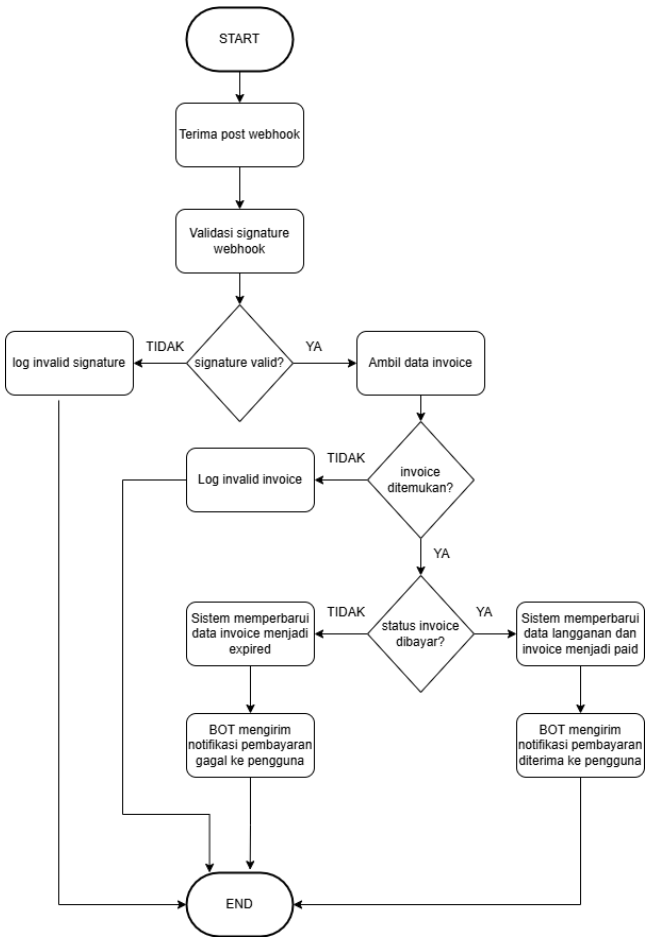

Gbr. 3 Alur Pendaftaran Base

*2) Alur Proses Pembayaran:* Alur proses pembayaran menangani permintaan pembayaran dari pengguna hingga tautan pembayaran berhasil dikirimkan. Proses dimulai ketika pengguna mengirim perintah /bayar, bot meminta input email, sistem memvalidasi email dan mengirimkan kode OTP. Setelah OTP terverifikasi, sistem membuat tautan pembayaran melalui API Xendit, menyimpan data invoice ke MongoDB, dan mengirimkan tautan ke pengguna melalui bot. Alur proses ditunjukkan pada Gbr. 4.



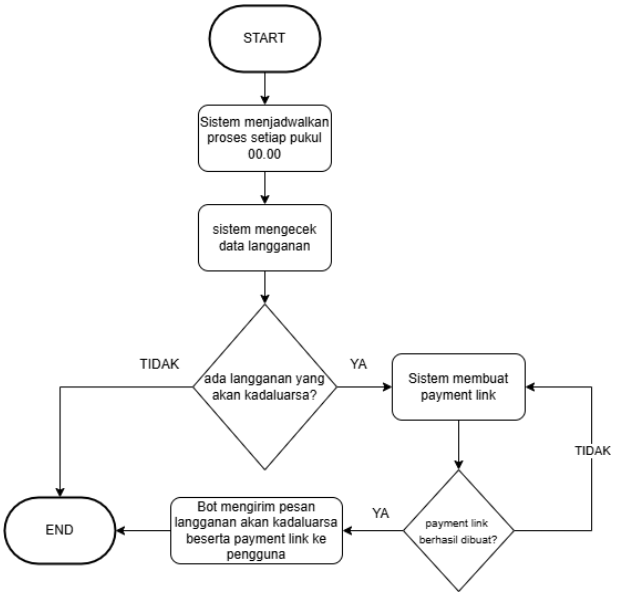

Gbr. 4 Alur Proses Pembayaran

*3) Alur Konfirmasi Pembayaran:* Alur konfirmasi pembayaran menangani notifikasi POST webhook yang dikirimkan oleh Xendit setelah pengguna menyelesaikan atau melewatkan pembayaran. Server memvalidasi callback token terlebih dahulu; jika tidak valid, sistem mencatat log kesalahan. Jika valid dan status *PAID*, sistem memperbarui data langganan dan invoice di MongoDB serta mengirimkan notifikasi berhasil ke pengguna. Jika status *EXPIRED*, sistem memperbarui status invoice dan mengirimkan notifikasi kedaluwarsa melalui bot. Alur konfirmasi pembayaran ditunjukkan pada Gbr. 5.




Gbr. 5 Alur Proses Konfirmasi Pembayaran

*4) Alur Penjadwalan Penagihan Otomatis:* Alur penjadwalan penagihan otomatis berjalan secara terjadwal setiap pukul 00.00 WIB oleh sistem. Sistem memeriksa data langganan di MongoDB untuk mencari pengguna yang masa aktifnya akan segera berakhir pada H-3, H-2, atau H-1. Jika ditemukan, sistem membuat tautan pembayaran baru melalui Xendit dan bot Telegram mengirimkan pesan pengingat beserta tautan perpanjangan kepada pengguna yang bersangkutan. Alur penjadwalan penagihan ditunjukkan pada Gbr. 6.




Gbr. 6 Alur Proses Penjadwalan Penagihan Otomatis

### C. Pengujian Sistem

Pengujian sistem dilakukan dalam tiga tahap, yaitu pengujian fungsionalitas, pengujian waktu respon, dan pengujian keandalan sistem.

*1) Pengujian Fungsionalitas:* Pengujian fungsionalitas dilakukan menggunakan metode Black Box Testing, yaitu metode pengujian yang berfokus pada keluaran sistem berdasarkan masukan yang diberikan tanpa memperhatikan struktur internal kode program. Metode ini dipilih karena sesuai untuk memvalidasi apakah setiap fitur sistem berjalan sesuai spesifikasi kebutuhan yang telah ditetapkan. Skenario pengujian mencakup enam fungsi utama sistem sebagaimana ditunjukkan pada Tabel II.

Tabel II
Skenario Pengujian Fungsional

| No | Skenario Pengujian | Kondisi Awal | Hasil yang Diharapkan |
|---|---|---|---|
| 1 | Pengguna mengirim perintah /start | Bot aktif dan terhubung ke jaringan | Bot mengirimkan pesan sambutan dan daftar perintah |
| 2 | Admin mengirim perintah /daftar | Bot aktif; pengirim adalah admin | Bot menampilkan panduan format pendaftaran |
| 3 | Pengguna mengirim perintah /bayar | Bot aktif; pengguna sudah terdaftar di database; Sudah memilih paket langganan | Bot mengirimkan pesan tautan pembayaran dari Xendit |
| 4 | Pengguna menyelesaikan transaksi | Invoice berstatus PENDING di database; pengguna menyelesaikan pembayaran di Xendit sandbox | Bot mengirim pesan notifikasi: "Pembayaran telah Berhasil". |
| 5 | Pengguna tidak menyelesaikan transaksi dalam batas waktu | Invoice berstatus PENDING di database; waktu invoice habis tanpa pembayaran | Bot mengirim pesan notifikasi: "Pembayaran telah Kedaluwarsa". |
| 6 | Admin mengirim perintah /report | Bot aktif; terdapat data transaksi dalam 30 hari terakhir di MongoDB | Bot mengirimkan pesan laporan pendapatan |

*2) Pengujian Waktu Respon:* Pengujian waktu respon dilakukan untuk mengukur durasi sistem secara *end-to-end*. Parameter pengukuran terbagi menjadi empat komponen waktu sebagaimana ditunjukkan pada Tabel III. Hasil pengujian dianalisis menggunakan statistik deskriptif meliputi rata-rata, standar deviasi, nilai minimum, dan maksimum. Semakin kecil nilai simpangan baku yang dihasilkan, semakin konsisten kinerja sistem dalam memberikan respons kepada pengguna [11].

Tabel III
Parameter Pengujian Waktu Respon

| Parameter | Keterangan |
|---|---|
| T1 | Durasi pembuatan payment link melalui Xendit API |
| T2 | Durasi pengiriman notifikasi dari Xendit ke server (webhook delivery) |
| T3 | Durasi pemrosesan webhook internal oleh server |
| T4 | Durasi pengiriman notifikasi ke pengguna melalui bot Telegram |
| Total | Total durasi end-to-end (T1 + T2 + T3 + T4) |

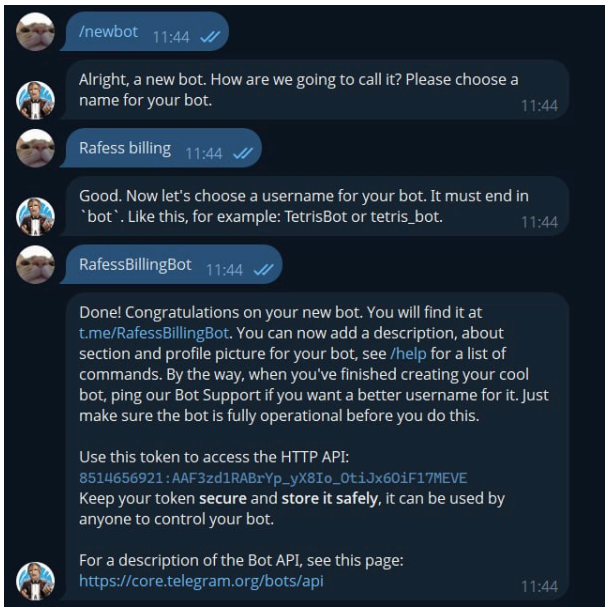
*3) Pengujian Keandalan Sistem:* Pengujian keandalan dilakukan melalui tiga skenario gangguan untuk memvalidasi ketahanan sistem dalam kondisi tidak normal. Skenario pertama adalah pengujian *retry webhook* otomatis saat server tidak dapat dijangkau, skenario kedua adalah sinkronisasi manual apabila notifikasi webhook tidak diterima, dan skenario ketiga adalah pengujian *idempotency* untuk memastikan tidak terjadi pemrosesan data transaksi yang sama secara berulang.

## III. HASIL DAN PEMBAHASAN

### A. Implementasi Sistem

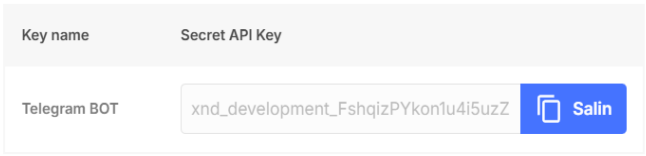
Implementasi sistem otomatisasi penagihan langganan dilakukan berdasarkan seluruh kebutuhan fungsional yang telah ditetapkan pada tahap analisis. Sistem dibangun menggunakan Python 3.10 dengan framework FastAPI sebagai server webhook dan MongoDB sebagai basis data utama.

*1) Konfigurasi Bot Telegram:* Pembuatan bot Telegram dilakukan melalui akun @BotFather pada aplikasi Telegram. @BotFather adalah bot yang disediakan oleh Telegram untuk membuat dan mengelola bot. Langkah pertama adalah mengirimkan perintah /newbot, kemudian mengisi nama dan username bot. Setelah proses selesai, @BotFather menerbitkan API Token berupa kode unik yang digunakan untuk setiap permintaan ke API Telegram. Pada Gbr. 7 menunjukkan proses pembuatan bot Telegram beserta perolehan API Token.

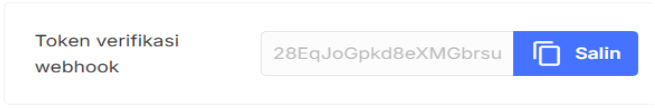



Gbr. 7 Pembuatan Bot Telegram

*2) Pembuatan Kredensial API Xendit:* Integrasi dengan Xendit memerlukan dua jenis kredensial utama. Pertama, Secret Key digunakan untuk mengautentikasi setiap *request* pembuatan Payment Link ke server Xendit. Kedua, Callback Token digunakan untuk memvalidasi keaslian payload webhook yang masuk ke server, memastikan bahwa notifikasi hanya diproses apabila berasal dari Xendit. Kedua kredensial tersebut diperoleh dari dashboard Xendit pada mode *sandbox* dan dikonfigurasi melalui env. Pada Gbr. 8 dan Gbr. 8 menampilkan *Secret Key* dan *Callback Token* pada dashboard Xendit.




Gbr. 8 Secret Key Xendit

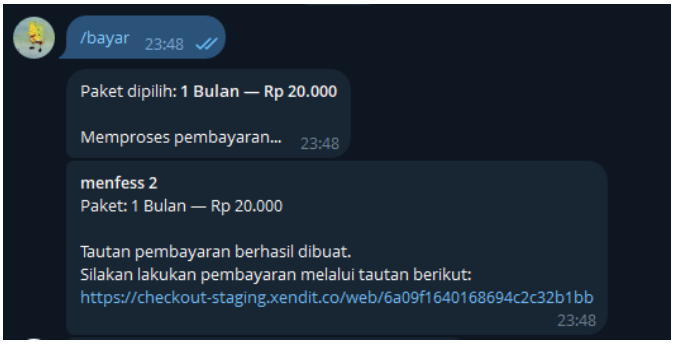



Gbr. 9 Callback Token Xendit

*3) Implementasi Pendaftaran Base:* Proses pendaftaran base diakses melalui perintah /daftar pada bot Telegram yang hanya dapat digunakan oleh admin. Bot merespons dengan menampilkan panduan format pendaftaran, kemudian memvalidasi input admin sesuai format yang ditentukan. Apabila validasi berhasil, sistem menyimpan data base ke koleksi bases di MongoDB dan mengirimkan notifikasi konfirmasi kepada admin. Pada Gbr. 10 menunjukkan tampilan proses pendaftaran base pada bot.

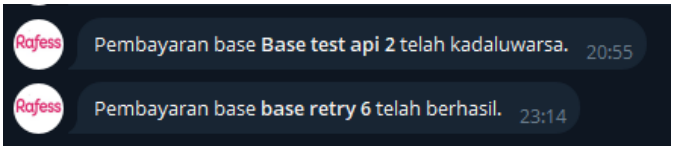

Gbr. 10 Pendaftaran base

*4) Implementasi Proses Pembayaran:* Proses pembayaran diawali ketika pengguna mengirimkan perintah /bayar pada bot. Sistem meminta pengguna memasukkan alamat email terdaftar, kemudian mengirimkan kode OTP untuk verifikasi identitas. Setelah OTP terverifikasi, sistem melakukan permintaan ke API Xendit untuk membuat tautan pembayaran, menyimpan data invoice ke MongoDB, dan mengirimkan tautan pembayaran kepada pengguna. Pengguna dapat menyelesaikan pembayaran melalui berbagai metode yang tersedia pada halaman checkout Xendit, meliputi transfer bank, e-wallet, dan QRIS. Pada Gbr. 11 menunjukkan tampilan tautan pembayaran yang diterima pengguna melalui bot.



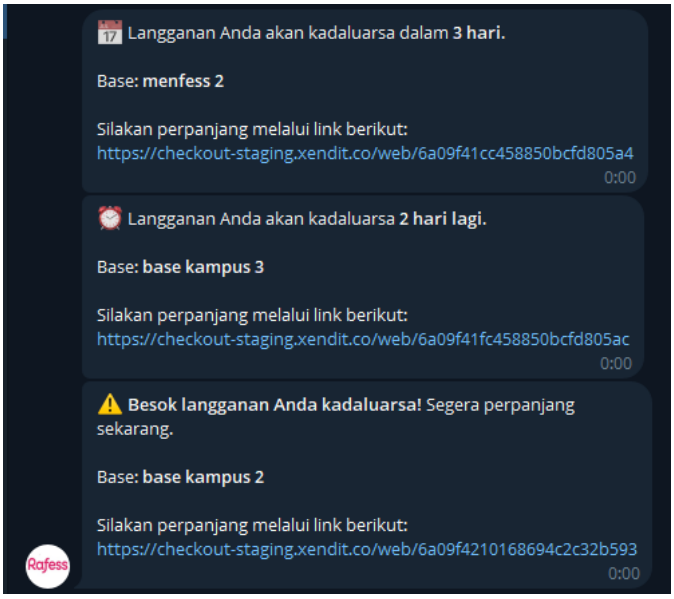

Gbr. 11 Proses Pembayaran

*5) Implementasi Konfirmasi Pembayaran:* Konfirmasi pembayaran diproses melalui mekanisme webhook. Ketika pengguna menyelesaikan pembayaran, Xendit mengirimkan notifikasi POST ke *endpoint* server. Apabila status yang diterima adalah *PAID*, sistem memperbarui status invoice dan data langganan di MongoDB menjadi aktif, lalu mengirimkan notifikasi konfirmasi kepada pengguna. Apabila status adalah *EXPIRED*, sistem memperbarui status invoice dan mengirimkan notifikasi kedaluwarsa. Pada Gbr. 12 ditunjukkan tampilan notifikasi pembayaran berhasil dan kedaluwarsa yang diterima pengguna.




Gbr. 12 Notifikasi Pembayaran

*6) Implementasi Penagihan Otomatis:* Proses penagihan otomatis dijalankan menggunakan APScheduler setiap pukul 00.00 WIB. Sistem memeriksa seluruh data langganan di basis data untuk mengecek pengguna yang masa aktif langganannya akan berakhir dalam H-3, H-2, atau H-1. Untuk setiap pengguna yang masa aktif langganannya akan kedaluwarsa, sistem membuat tautan pembayaran baru melalui API Xendit dan mengirimkan pesan pengingat perpanjangan beserta tautan pembayaran melalui bot Telegram. Pada Gbr. 13 ditunjukkan tampilan notifikasi penagihan otomatis yang diterima pengguna.

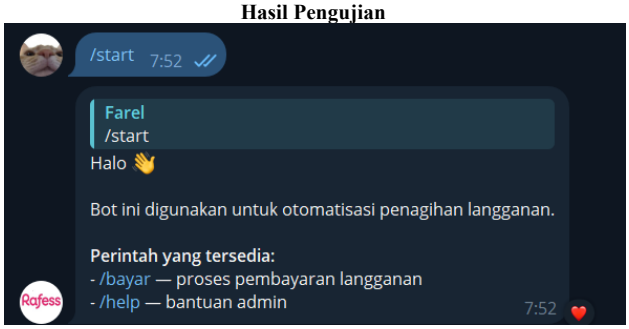



Gbr. 13 Notifikasi Penagihan

B. *Pengujian*

*1) Pengujian Fungsional:* Pengujian fungsional dilakukan menggunakan metode Black Box Testing. Black Box Testing merupakan prosedur pengujian yang berfokus pada proses input dan output sistem tanpa memperhatikan struktur logika internal kode. Metode ini dipilih karena sesuai untuk memvalidasi apakah setiap fitur sistem berjalan sesuai spesifikasi kebutuhan yang telah ditetapkan. Hasil pengujian ditampilkan pada tabel di bawah ini.

Tabel IV
Pengujian Perintah /start

| **Pengujian 1 perintah /start** |
|---|
| **Skenario pengujian:**<br>Pengguna mengirim perintah /start |
| **Harapan:**<br>Bot mengirimkan pesan fungsi bot disertai dengan daftar perintah yang tersedia |
| **Hasil Pengujian**<br>  |
| **Hasil:**<br>Valid |

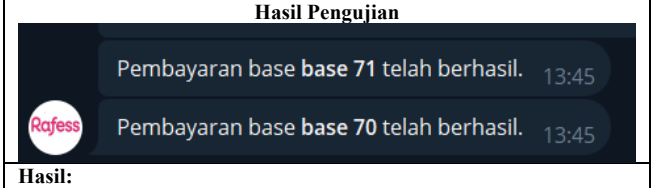
Tabel V
Pengujian Perintah /daftar

| **Pengujian 2 perintah /daftar** |
|---|
| **Skenario pengujian:**<br>Admin mengirim perintah /daftar |
| **Harapan:**<br>Bot mengirimkan pesan yang berisi informasi panduan format pendaftaran |
| **Hasil Pengujian**<br>  |
| **Hasil:**<br>Valid |

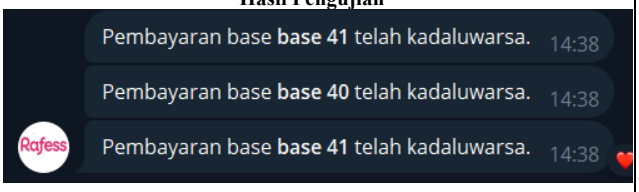
Tabel VI
Pengujian Perintah /bayar

| **Pengujian 3 perintah /bayar** |
|---|
| **Skenario pengujian:**<br>Pengguna mengirim perintah /bayar |
| **Harapan:**<br>Bot mengirimkan pesan yang berisi informasi nama base yang akan diperpanjang dan tautan pembayaran. |
| **Hasil Pengujian**<br>  |
| **Hasil:**<br>Valid |

Tabel VII
Pengujian Pembayaran Berhasil

| **Pengujian 4 Pembayaran Berhasil** |
|---|
| **Skenario pengujian:**<br>Pengguna menyelesaikan transaksi |
| **Harapan:**<br>Bot mengirimkan pesan fungsi bot disertai dengan daftar perintah yang tersedia |
| **Hasil Pengujian**<br>  |
| **Hasil:**<br>Valid |

Tabel VIII
Pengujian Pembayaran Kedaluwarsa

| **Pengujian 5 Pembayaran Kedaluwarsa** |
|---|
| **Skenario pengujian:**<br>Pengguna tidak menyelesaikan transaksi dalam batas waktu |
| **Harapan:**<br>Bot mengirimkan pesan yang berisi informasi nama base yang pembayarannya telah kedaluwarsa |
| **Hasil Pengujian**<br>  |
| **Hasil:**<br>Valid |

Tabel IX
Pengujian Perintah /report

| **Pengujian 6 Perintah /report** |
|---|
| **Skenario pengujian:**<br>Admin mengirim perintah /report |
| **Harapan:**<br>Bot mengirimkan pesan yang berisi informasi pendapatan 30 hari terakhir |

**Hasil Pengujian**

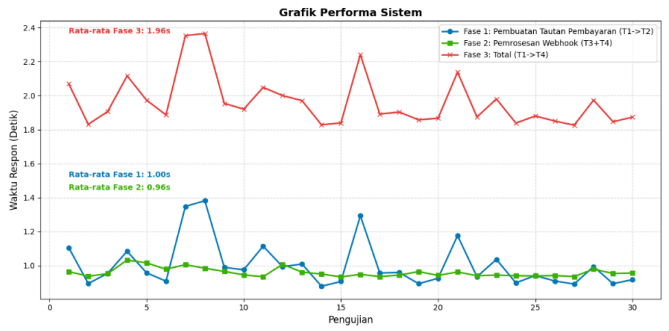

**Hasil:**
Valid

Berdasarkan hasil uji fungsional pada Tabel IV hingga Tabel IX, seluruh enam skenario pengujian menghasilkan keluaran yang sesuai dengan spesifikasi kebutuhan yang telah ditetapkan. Hasil ini membuktikan bahwa integrasi antara bot Telegram, API Xendit, dan MongoDB telah berjalan dengan baik, di mana setiap aksi pengguna mampu memicu rantai proses otomatis hingga notifikasi akhir tanpa intervensi manual.

*2) Pengujian Waktu Respon:* Pengujian waktu respon dilakukan untuk mengukur durasi sistem dalam mengeksekusi pembayaran secara end-to-end. Data dikumpulkan dari 30 transaksi menggunakan *logging* dengan mengukur empat titik waktu: T1 (request pembuatan Payment Link ke API Xendit), T2 (pengiriman tautan pembayaran ke pengguna), T3 (pemrosesan webhook internal), dan T4 (pengiriman notifikasi konfirmasi). Hasil pengujian dari 30 sampel disajikan pada Tabel X dan divisualisasikan pada Gbr. 14.

Tabel X
Statistik Waktu Respon

| Metrik | T1 | T2 | T3 | T4 | Total Durasi |
|---|---|---|---|---|---|
| Rata-rata | 0.587s | 0.417s | 0.535s | 0.424s | 1.963s |
| Std. Deviasi | 0,128 s | 0,022 s | 0,012 s | 0,026 s | 0,146 s |
| Tercepat | 0.432s | 0.384s | 0.516s | 0.393s | 1.826s |
| Terlambat | 0.929s | 0.477s | 0.563s | 0.495s | 2.365s |




Gbr. 14 Grafik Waktu Respons

Berdasarkan Tabel X dan Gbr. 14, total durasi end-to-end rata-rata sebesar 1,963 detik (SD = 0,146 detik) dengan rentang antara 1,826 detik dan 2,365 detik. T1 mencatatkan variabilitas tertinggi dengan standar deviasi 0,128 detik, yang mencerminkan ketergantungan pada kondisi jaringan. Sebaliknya, T3 mencatatkan standar deviasi terendah sebesar 0,012 detik, membuktikan bahwa arsitektur FastAPI dan MongoDB mampu memproses webhook secara stabil dan konsisten meskipun sistem berjalan pada infrastruktur dengan spesifikasi terbatas. T2 dan T4 merepresentasikan pengiriman pesan melalui Telegram Bot API memiliki rata-rata masing-masing 0,417 detik dan 0,424 detik dengan selisih waktu 0,007 detik, menunjukkan bahwa ukuran payload pesan tidak berpengaruh signifikan terhadap latensi pengiriman.

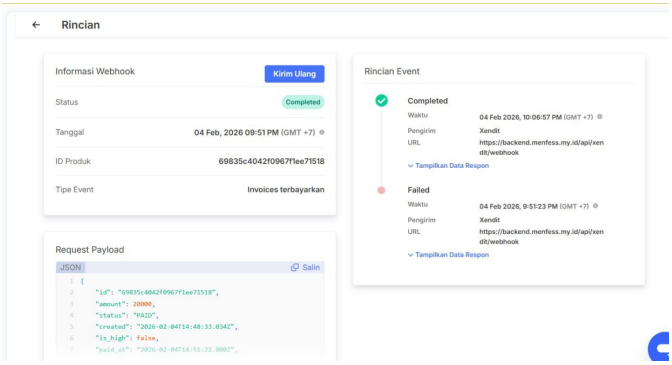
C. *Pengujian Keandalan Sistem*

Pengujian keandalan sistem dilakukan untuk memvalidasi kemampuan sistem dalam menjaga keutuhan data transaksi saat terjadi gangguan penerimaan webhook. Pengujian mencakup tiga skenario: retry webhook otomatis, sinkronisasi manual, dan penanganan idempotency.

*1) Pengujian Retry Webhook:* Pengujian ini bertujuan untuk memvalidasi kemampuan Xendit dalam mengirim ulang notifikasi secara otomatis ketika server tidak dapat merespons. Pengujian dilakukan dengan mematikan server sebelum transaksi dibayar, kemudian mengaktifkannya kembali pada jendela waktu setiap *retry*. Hasil pengujian ditunjukkan pada Tabel XI dan log retry ditunjukkan pada Gbr. 15.

Tabel XI
Pengujian Retry Webhook

| Percobaan | Dokumentasi | Interval Aktual | Status Hasil |
|---|---|---|---|
| Retry 1 | 15 menit | 15 menit | Valid |
| Retry 2 | 45 menit | 30 menit | Valid |
| Retry 3 | 120 menit | 75 menit | Valid |
| Retry 4 | 180 menit | 60 menit | Valid |
| Retry 5 | 360 menit | 180 menit | Valid |
| Retry 6 | 720 menit | 360 menit | Valid |




Gbr. 15 Percobaan Retry 1

Berdasarkan Tabel XI, sistem berhasil menerima dan memproses webhook pada seluruh enam jendela retry. Terjadi perbedaan variasi interval antara dokumentasi pada Xendit dan hasil aktual, namun tidak memengaruhi fungsionalitas karena pengiriman percobaan tetap dilakukan hingga 6 kali.

*2) Pengujian Sinkronisasi Manual:* Pengujian ini bertujuan untuk memvalidasi kemampuan sistem dalam melakukan rekonsiliasi data transaksi secara mandiri ketika webhook tidak berhasil diterima. Pengujian dilakukan dengan mensimulasikan kegagalan penerimaan webhook, kemudian admin mengirimkan perintah /sync untuk memicu pengambilan status terbaru dari API Xendit. Hasil pengujian ditunjukkan pada Tabel XII.

Tabel XII
Hasil Pengujian Sinkronisasi Manual

| Skenario Pengujian | Status Awal | Tindakan | Status Akhir | Status Hasil |
|---|---|---|---|---|
| Sinkronisasi transaksi sukses yang tertunda | *PENDING* | Admin mengirim /sync | *PAID* | Valid |
| Sinkronisasi transaksi yang sudah kedaluwarsa | *PENDING* | Admin mengirim /sync | *EXPIRED* | Valid |
| Sinkronisasi transaksi yang belum dibayar | *PENDING* | Admin mengirim /sync | *PENDING* | Valid |

Berdasarkan Tabel XII, sistem mampu melakukan rekonsiliasi data pada tiga kondisi berbeda tanpa intervensi langsung pada database, membuktikan bahwa sistem memiliki kapabilitas pemulihan aktif yang menjamin integritas data saat terjadi gangguan webhook.

*3) Pengujian Idempotency:* Pengujian ini bertujuan untuk memvalidasi kemampuan sistem dalam mencegah pemrosesan duplikat apabila Xendit mengirimkan notifikasi webhook yang sama lebih dari satu kali. Pengujian dilakukan dengan mengirimkan payload webhook yang identik secara berulang menggunakan fitur kirim ulang pada dashboard Xendit. Hasil pengujian ditunjukkan pada Tabel XIII.

Tabel XIII
Pengujian Idempotency

| Percobaan | Status Transaksi | Respon Sistem (HTTP) | Respon Body | Status Hasil |
|---|---|---|---|---|
| Pengiriman pertama | *PAID* | 200 OK | *{"ok": true}* | Valid |
| Pengiriman Kedua (Duplikat) | *PAID* | 200 OK | *{"ok": true, "duplicate": true}* | Valid |
| Pengiriman Ketiga (Duplikat) | *PAID* | 200 OK | *{"ok": true, "duplicate": true}* | Valid |

Berdasarkan Tabel XIII, sistem berhasil mencegah pemrosesan duplikat melalui pengecekan event_id di MongoDB. Notifikasi yang teridentifikasi sebagai duplikat langsung dilewati tanpa memperbarui data langganan, sehingga mencegah terjadinya billing ganda pada pengguna.

## IV. KESIMPULAN

Sistem otomatisasi penagihan langganan pada bot Telegram Rafess berhasil diimplementasikan melalui integrasi Xendit Payment Link dan webhook menggunakan framework FastAPI dan MongoDB. Sistem mampu menjalankan seluruh alur penagihan secara otomatis, meliputi pembuatan tautan pembayaran, verifikasi transaksi melalui webhook, pembaruan status langganan secara real-time, serta pengiriman pengingat perpanjangan pada H-3, H-2, dan H-1 sebelum masa aktif berakhir.

Hasil pengujian fungsional menggunakan metode Black Box Testing menunjukkan bahwa seluruh enam skenario pengujian berjalan sesuai spesifikasi kebutuhan Pengujian waktu respon terhadap 30 sampel transaksi menghasilkan rata-rata durasi end-to-end sebesar 1,963 detik (SD = 0,146 detik), dengan komponen pemrosesan webhook internal mencatatkan standar deviasi terendah sebesar 0,012 detik, membuktikan arsitektur FastAPI dan MongoDB mampu memproses notifikasi secara stabil pada infrastruktur terbatas. Sistem juga terbukti andal dalam menangani skenario gangguan melalui mekanisme retry webhook otomatis, sinkronisasi manual, dan idempotency untuk mencegah duplikasi data transaksi.

## REFERENSI

[1] A. Zubaidi, A. Z. Mardiansyah, W. Wedashwara, and A. H. Jatmika, "Integrasi Sistem Informasi Akademik dan BOT Telegram Sebagai Media Pengakesan Informasi di Universitas Mataram," *Jurnal Teknologi Informasi, Komputer, dan Aplikasinya (JTIKA)*, vol. 3, no. 2, pp. 251–260, 2021.

[2] J. K. Adangbain and E. S. Bata, "Pemanfaatan Bot Telegram Sebagai Media Informasi Dan Layanan Akademik Dengan Metode Webhook," in *Seminar Nasional & Konferensi Ilmiah Sistem Informasi, Informatika & Komunikasi*, 2021, pp. 106–112.

[3] H. Haeruddin, N. Ma'muriyah, and W. Wijaya, "PENGEMBANGAN SYSTEM BERLANGGANAN APLIKASI DI USERTIP MENGGUNAKAN STRIPE," *Journal of Information System Management (JOISM)*, vol. 5, no. 2, pp. 127–133, 2024.

[4] M. A. Akbar and P. Nerisafitra, "Penerapan Payment Gateway Menggunakan Metode Webhook Dalam Pengembangan Bot Reservasi Jiot Gadget Solution," *Journal of Informatics and Computer Science (JINACS)*, pp. 932–947, 2025.

[5] K. A. Nugraha and D. Sebastian, "Designing consultation chatbot using telegram api and webhook-based nodejs applications," in *7th International Conference on Education and Technology (ICET 2021)*, 2021, pp. 119–122.

[6] A. R. Syah and A. Prihanto, "Auto Response Messages pada Telegram Bot untuk Pelayanan Sistem Informasi Praktek Industri dan Skripsi dengan Metode Webhook," *Journal of Informatics and Computer Science (JINACS)*, vol. 3, no. 04, pp. 547–556, 2022.

[7] P. A. Safitri and A. Prapanca, "Sistem Layanan Pengaduan Kekerasan Seksual Berbasis Bot Telegram dengan Webhook Communication di Universitas Negeri Surabaya," *Journal of Informatics and Computer Science (JINACS)*, vol. 5, no. 03, pp. 272–281, 2024.

[8] A. Bintang, A. L. Hananto, and A. Hananto, "Telegram BOT Application Development Integration with Google Sheets for Sending Service Reporting," *Journal of Artificial Intelligence and Engineering Applications (JAIEA)*, vol. 4, no. 3, pp. 2208–2214, 2025.

[9] S. Alfarezy, M. F. Ridho, and J. Prayoga, "SISTEM PEMBAYARAN KOS BERBASIS WEB MENGGUNAKAN MIDTRANS PAYMENT GATEWAY," *Syntax: Journal of Software Engineering, Computer Science and Information Technology*, vol. 5, no. 2, pp. 580–585, 2024.

[10] F. Bahy, D. Arif, and M. S. Amin, "Implementing a Payment Gateway in the Mount Slamet Hiking Ticketing System," *Sinkron: jurnal dan penelitian teknik informatika*, vol. 10, no. 1, pp. 406–417, 2026.

[11] A. Setiawan, Y. Alexandro, and J. Parhusip, "ANALISIS DISTRIBUSI RATA-RATA WAKTU RESPON APLIKASI BERBASIS WEB MENGGUNAKAN KURVA NORMAL DAN SIMPANGAN BAKU," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 9, no. 1, pp. 211–216, 2025.