Jutisi: Jurnal Ilmiah Teknik Informatika dan Sistem Informasi
Jl. Ahmad Yani, K.M. 33,5 - Kampus STMIK Banjarbaru
Loktabat – Banjarbaru (Tlp. 0511 4782881), e-mail: puslit.stmikbjb@gmail.com
e-ISSN: 2685-0893
p-ISSN: 2089-3787

Optimasi Dokumentasi Faktur Pembelian Menggunakan Pendekatan *Robotic Process Automation* Berbasis Selenium *Webdriver*

Siesilia Angelina[1*], Tony[2]
Sistem Informasi, Universitas Tarumanagara, DKI Jakarta, Indonesia
*e-mail *Corresponding Author:* Siesilia02@gmail.com

Abstract

High transaction volume at PT Enseval Putera Megatrading has increased employee working hours. This is due to manual input of purchase invoices which are prone to errors and delays. This has had a negative impact on business relationships and fulfillment partners. This study proposes implementing a Python-based RPA (Robotic Process Automation) application to address this issue. This RPA application is designed to retrieve information from a database operated with PostgreSQL and automatically input data on a website using Selenium Webdriver. This study aims to improve the company's operational efficiency by implementing Robotic Process Automation (RPA) and Selenium Webdriver. The study follows the Software Development Life Cycle (SDLC) approach with a waterfall model to ensure an effective and reliable application. The results include reduced input time, reduced risk of human error, and elimination of repetitive work. With this implementation, the company can focus on tasks that require human interaction, increase productivity, deliver invoices on time, and strengthen business relationships with the company's fulfillment partners.
Keywords: *RPA; Purchase Invoice; SDLC; Automation; Selenium Webdriver.*

Abstrak

Volume transaksi yang tinggi di PT Enseval Putera Megatrading telah meningkatkan waktu kerja karyawan. Hal itu disebabkan penginputan manual *purchase invoice* yang rentan terhadap kesalahan dan keterlambatan. Sehingga berdampak negatif pada hubungan bisnis dan *partner fulfillment*. Untuk mengatasi permasalahan tersebut, penelitian ini mengusulkan implementasi aplikasi RPA (*Robotic Process Automation*) berbasis Python. Aplikasi RPA ini dirancang untuk mengambil informasi dari *database* yang dioperasikan dengan PostgreSQL dan secara otomatis menginput data pada *website* menggunakan Selenium *Webdriver*. Tujuan penelitian ini adalah meningkatkan efisiensi operasional perusahaan melalui implementasi *Robotic Process Automation* (RPA) dan Selenium *Webdriver*. Studi ini mengikuti pendekatan *Software Development Life Cycle* (SDLC) dengan model *waterfall* untuk memastikan aplikasi yang efektif dan handal. Hasilnya mencakup pengurangan waktu penginputan, pengurangan risiko kesalahan manusia, dan menghilangkan pekerjaan repetitif. Dengan implementasi ini, perusahaan dapat fokus pada tugas yang memerlukan interaksi manusia, meningkatkan produktivitas, memberikan *invoice* secara tepat waktu, dan memperkuat hubungan bisnis dengan *partner fulfillment* perusahaan.
Kata kunci: RPA; *Purchase Invoice*; SDLC; Automatisasi; Selenium *Webdriver.*

1. Pendahuluan

Enseval, sebuah perusahaan distribusi produk farmasi dan konsumen yang terpisah dari Kalbe Farma sejak Oktober 1973, telah menjadi salah satu pemain kunci dalam industri distribusi produk farmasi di Indonesia. Dalam menghadapi kondisi geografis kepulauan yang kompleks, Enseval telah mengembangkan solusi inovatif untuk memenuhi berbagai tantangan, salah satunya adalah layanan *fulfillment center* yang bertujuan untuk menangani pesanan *e-commerce*. Saat ini, terdapat 76 cabang yang tersebar untuk menunjang kelancaran operasional logistik [1].

Operasi bisnis yang melibatkan banyak transaksi dan berbagai pihak membuat proses pembuatan *invoice* menjadi aspek yang sangat penting bagi perusahaan. Salah satu masalah yang dihadapi saat ini adalah penginputan manual *invoice* melalui *website* internal. Proses ini

memakan waktu yang cukup lama dan berpotensi menyebabkan keterlambatan dalam menghasilkan *invoice* akhir. Potensi kesalahan dalam proses manual juga menjadi kendala yang perlu diatasi, karena kesalahan penginputan dan duplikasi data dapat meningkatkan risiko *human error*, yang pada gilirannya dapat menghambat pekerjaan di berbagai aspek serta meningkatnya jam kerja karyawan.

Dalam era teknologi yang terus berkembang, terdapat potensi besar untuk mengatasi masalah tersebut dengan memanfaatkan otomatisasi. Salah satunya adalah Teknologi *Robotic Process Automation* (RPA), teknologi yang memungkinkan pembuatan *software* atau "robot" yang memiliki kemampuan untuk mengidentifikasi dan berinteraksi dengan tombol, bidang, teks, dan gambar di sebagian besar program dan sistem dengan cara yang mirip dengan manusia [2]. Teknologi RPA ini juga memiliki potensi besar dalam mengotomatisasi tugas-tugas yang sebelumnya memakan waktu dan berisiko kesalahan manusiawi dalam berbagai proses bisnis.

Penelitian ini bertujuan untuk mengembangkan aplikasi RPA yang akan mengatasi masalah penginputan *invoice* secara otomatis di Enseval. Tujuan utama adalah meningkatkan produktivitas karyawan, memastikan pengiriman *invoice* tepat waktu, serta mendukung kelancaran operasional dan hubungan bisnis dengan mitra *fulfillment* perusahaan. Penelitian ini akan membahas langkah-langkah pengembangan aplikasi RPA, hasil yang diharapkan, serta implikasi positifnya dalam operasi perusahaan. Dalam naskah ini, kami akan membahas latar belakang, tinjauan Pustaka, metode, hasil penelitian, dan simpulan yang diharapkan.

2. Tinjauan Pustaka

Quaum et al [2]. menghadirkan solusi inovatif dalam konteks *invoice processing* dan mailing. Tugas ini mencakup *export* data dari beragam faktur dan penyimpanannya dalam format file Excel, diikuti dengan pengiriman *email* secara otomatis ke alamat yang tercantum dalam faktur tersebut. persamaan dengan penelitian ini yaitu berfokus pada pembuatan sistem RPA menggunakan metode Selenium *Webdriver*. Tetapi dalam penelitian Quaum et al terdapat penggunaan workfusion untuk mengintegrasikan teknologi tersebut. Di sisi lain, dalam penelitian ini berfokus pada desain sistem RPA untuk otomatisasi pengolahan faktur pembelian dengan menerapkan metodologi pengembangan perangkat lunak dengan metode SDLC *Waterfall*.

Avrianto et al [3]. memilih pendekatan metodologi *Agile* karena fokusnya pada fleksibilitas dan kemampuan untuk merespons perubahan yang mungkin diperlukan. Persamaan sistem yang dikembangkan dalam penelitian Avrianto et al yaitu memanfaatkan library Selenium dengan bahasa pemrograman Python, perbedaannya adalah pendekatan ini mengambil data dari tim IT di sebuah perusahaan MNC, khususnya divisi *Quality Control* (QC) untuk proses penilaian kualitas situs *website* RCTI+. Sementara itu, penelitian ini memiliki fokus pada perancangan sistem RPA untuk otomatisasi pengolahan faktur pembeliandengan mengambil data dari *database* Perusahaan menggunakan postgreSQL.

Rizkiyani et al [4]. mengembangkan sistem RPA yang berkolaborasi dengan kecerdasan buatan (*Artificial Intelligence*) yang memanfaatkan *Natural Language Processing* (NLP) untuk menciptakan untuk menggantikan proses bisnis manual yang ada. Sedangkan dalam peneltian ini berfokus pada desain sistem *Robotic Process Automation* (RPA) untuk otomatisasi pengolahan faktur pembelian. Dalam rangka mencapai tujuan ini, penulis menggunakan Selenium *Webdriver* untuk mengakses *website* dan menerapkan metodologi pengembangan perangkat lunak dengan metode SDLC *Waterfall*.

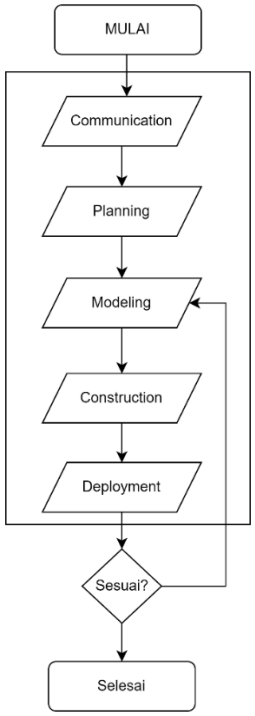
Abadi et al [5]. menggunakan *Artificial Intelligence* (AI) yang didukung oleh *Robotic Process Automation* (RPA) untuk meningkatkan *export* informasi dari faktur. Mereka memanfaatkan robot berbasis RPA yang terintegrasi dengan *Machine Learning* (ML) *Document Understanding* melalui UiPath AI *Center* untuk melakukan *export* secara otomatis. Hasilnya adalah robot yang mampu melatih model ML *Document Understanding* dan mencapai tingkat akurasi yang tinggi, sekitar 97% dalam bidang data dan 96% dalam ketepatan isi. Di sisi lain, penelitian ini merancang sistem RPA untuk mengotomatisasi penginputan faktur pembelian dengan memanfaatkan Selenium *Webdriver* untuk mengakses situs *website* dan menerapkan metodologi pengembangan perangkat lunak SDLC *Waterfall*.

Sahu et al [6]. Penelitian dari pendekatan Sahu et al memiliki kesamaan dengan penelitian mengembangkan sebuah aplikasi RPA untuk mengotomatisasi yang dapat mengubah proses pengolahan faktur dalam operasi keuangan. Aolkasi ini berupa *DataBot* dan memanfaatkan teknologi *Optical Character Recognition* dan kemampuan pemrosesan bahasa alami untuk membaca informasi faktur dengan akurat dengan metodologi yang digunakan yaitu

SDLC *Waterfall*. Di sisi lain, dalam penelitian ini berfokus pada desain sistem RPA untuk otomatisasi pengolahan faktur pembelian dengan menggunakan Selenium *Webdriver* untuk mengakses *website* dan menerapkan metodologi pengembangan perangkat lunak.

3. Metodologi

Dalam pembuatan aplikasi ini, metode yang digunakan adalah pendekatan SDLC (*Software Development Life Cycle*). Dalam rekayasa sistem dan pengembangan perangkat lunak, SDLC adalah proses yang digunakan untuk menciptakan dan mengubah sistem, serta model dan metodologi yang digunakan dalam pengembangan sistem-sistem tersebut. SDLC juga mewakili suatu kerangka kerja untuk pengembangan perangkat lunak yang melibatkan tahapan-tahapan, yaitu perencanaan, analisis desain, implementasi, pengujian, dan pemeliharaan [3]. Dalam penelitian ini, tahapan yang dilakukan menggunakan model *waterfall*, proses pengembangan akan berlangsung secara bertahap dan berkelanjutan, dimana setiap langkah akan dilanjutkan secara berurutan. Pendekatan ini tidak memungkinkan untuk kembali atau mengulang tahap-tahap sebelumnya setelah dimulai, dan langkah selanjutnya hanya akan dimulai setelah penyelesaian langkah sebelumnya [7]. Model *waterfall* ini meliputi tahapan seperti yang ditunjukkan pada **Gambar 1**.



Gambar 1. Tahapan Penelitian *[8]*

1) Tahapan *Communication*

Pada tahapan Analisis Kebutuhan dalam proses pengembangan sistem RPA yaitu melalui identifikasi dan pengumpulan kebutuhan agar dapat memahami dengan baik proses yang bisa diotomatisasi dengan RPA. berdasarkan hasil wawancara bersama salah satu staf *store operation*. Hasil analisis tersebut mengungkapkan bahwa otomatisasi proses ini memiliki potensi besar untuk mengatasi beberapa tantangan dalam pembuatan *invoice* manual, seperti pengisian data pesanan secara manual, potensi kesalahan manusia, dan keterlambatan dalam penyusunan *invoice*. Berikut *script* wawancara, seperti yang dapat dilihat pada **Tabel 1**.

Tabel 1. *Script Interview user*

No	Pertanyaan	Jawaban
1.	Bagaimana proses penginputan *invoice* secara manualnya?	- Pertama, kami memeriksa pesanan yang masuk dari *marketplace* setiap harinya. - Kemudian, kami mengambil data dari kolom yang perlu dikirim dan diproses pada hari itu. - Setelah itu, kami mengklik pada setiap *invoice* pesanan yang ada, mengambil data pesanan, dan memasukkan data tersebut satu per satu ke dalam formulir *input invoice* di *website*. - Dan yang terakhir, kami save data *invoice* yang telah dinput tersebut.
2	Berapa jumlah kolom yang perlu diisi untuk pembuatan *invoice purchase* ini?	ada sekitar 25 kolom yang perlu di isi untuk pembuatan *invoice purchase* ini.
3	Apakah dari *partner* mempunyai jam *cut off* untuk penginputan *invoice* hariannya?	Jam *cut off* berbeda untuk setiap mitra, seperti contoh mitrasana, yang memiliki jam *cut off*, *invoice* yang dinput hanya sampai pesanan pada pukul 16.00, setelah melewati waktu tersebut, transaksi akan diproses pada hari berikutnya.
4	Apa kendala utama yang dialami dalam proses ini?	Tantangannya terletak pada proses yang memakan waktu, sehingga meningkatnya jam kerja, risiko kesalahan *input* yang tinggi terutama untuk pemula, potensi duplikasi, serta rutinitas yang monoton karena melakukan tugas yang sama berulang. Selain itu, memerlukan waktu yang cukup lama untuk terbiasa dengan cara penginputan di situs *website* karena pemula harus sering beralih antar *tab*.
5	Bagaimana penanganan jika terjadi kesalahan dalam penginputan *invoice* setelah nomor *invoice* sudah keluar?	Apabila terjadi kesalahan dalam penginputan *invoice* setelah nomor *invoice* diterbitkan dan tersimpan, langkah yang harus diambil adalah menghubungi staf gudang yang bertanggung jawab atas pesanan tersebut. Kemudian kami akan menyebutkan nomor pesanan dan meminta konfirmasi apakah pesanan tersebut sudah dikemas. Jika sudah, kami akan meminta staf gudang untuk membongkarnya.
6	Siapa saja yang terlibat dalam proses ini?	Pihak *store operation*.
7	Apa harapan anda terkait masa depan *invoice process* ini?	Harapan kami adalah agar semua SKU yang aktif dapat diintegrasikan ke dalam sistem ini dan proses penginputannya dapat dilakukan secara otomatis. Hal ini akan membantu mengurangi kesalahan penginputan dan menghilangkan pekerjaan yang bersifat berulang.
8	Apa permintaan khusus dari *store operation* terkait pengembangan aplikasi RPA ini?	harapan saya terkait sistem ini adalah memiliki tampilan sederhana dan minimalis, telah terhubung dengan *database* pesanan sehingga ketika pembuatan *invoice* hanya perlu menginput tanggal mulai dan berakhirnya *invoice*.

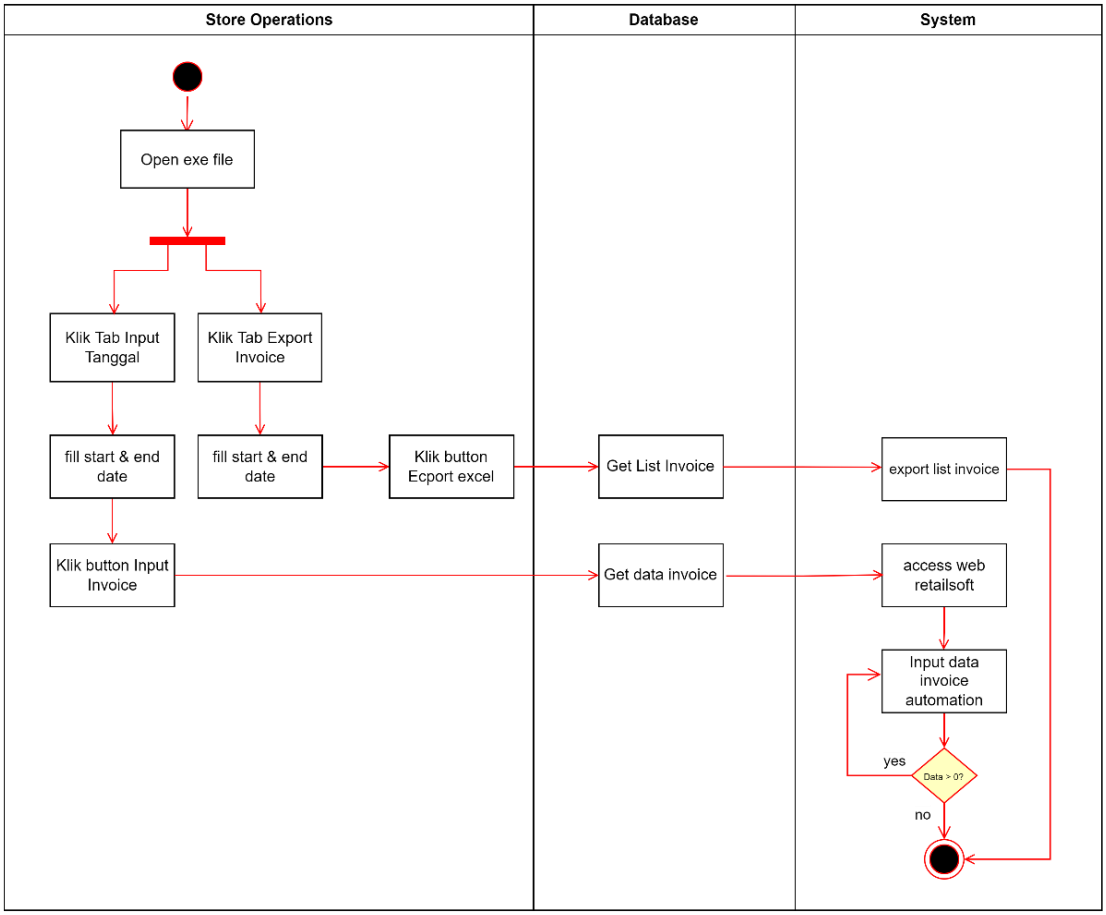
2) Tahapan *Planning*

Setelah analisis dilakukan, selanjutnya adalah tahap pengumpulan data yang sesuai dengan kebutuhan atau *user requirement* dalam pembuatan *software* akan dimasukkan dalam rencana selanjutnya yang akan dikerjakan. Pada proses ini, data dikumpulkan dari hasil analisis dan menjadi bahan untuk merancang alur sistem aplikasi RPA yang disesuaikan dengan kebutuhan *user*. Data tersebut juga menjadi acuan dalam perancangan pembuatan aplikasi pada tahap selanjutnya. Dalam tahap ini, diperlukan perencanaan yang terstruktur agar dapat menyelesaikan tahap pemodelan dengan tepat waktu.

3) Tahapan *Modeling*

Proses ketiga dalam pengembangan adalah dimulai dari merancang arsitektur dan komponen-komponen utama sistem dengan menggunakan *Unified Modeling Language* (UML), yang membantu dalam membuat *diagram* dan model yang menggambarkan struktur, interaksi, dan aliran data dalam sistem. Salah satu model yang dilampirkan adalah *Activity Diagram*, *Activity Diagram* adalah representasi visual yang digunakan untuk mengilustrasikan tingkah laku objek-

objek yang beroperasi secara independen dalam sebuah proses bisnis. *Activity Diagram* dapat menggambarkan berbagai hal, mulai dari alur kerja dalam sebuah bisnis yang kompleks dengan banyak *use case* yang berbeda hingga ke tingkat detail dengan fokus pada *use case* per individu [9]. berikut ini merupakan *activity diagram* keseluruhan dari aplikasi RPA yang ditunjukkan pada **Gambar 2**.

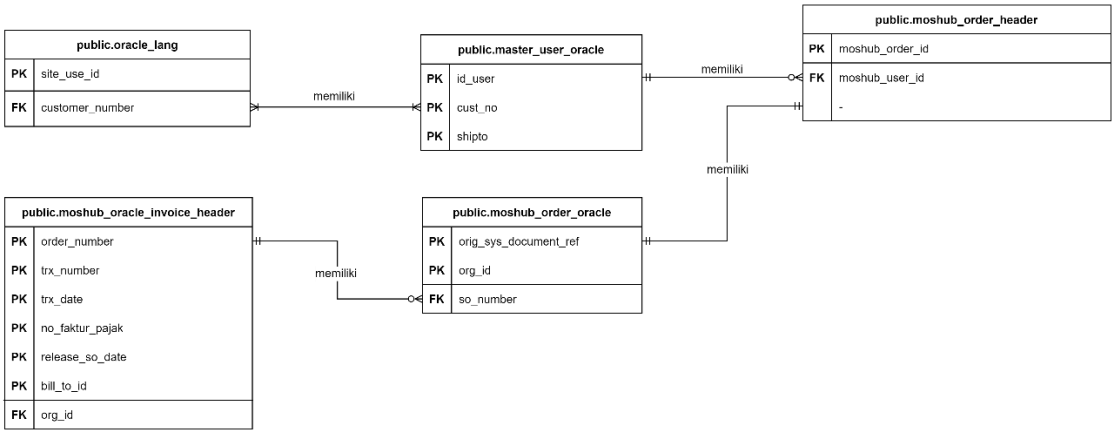


Gambar 2. Alur kerja Aplikasi RPA

User store operations dapat membuka aplikasi Python dalam bentuk exe yang memiliki dua *tab*, "*Input* Tanggal" dan "*Export Invoice*". Pengguna memulai dengan memilih *tab* "*Input* Tanggal" untuk memasukkan tanggal faktur yang akan diproses. Setelah tanggal dimasukkan, aplikasi secara otomatis mengambil data faktur yang sesuai dengan tanggal tersebut dari *database* internal. Selanjutnya, dengan menggunakan Selenium *Webdriver*, aplikasi secara otomatis mengakses situs *website* eksternal, memasukkan tanggal secara otomatis, dan mengumpulkan informasi faktur. Setelah robot selesai menginputkan data untuk semua faktur, pengguna dapat mengekspor *invoice* tersebut melalui *tab* "*Export Invoice*" dalam format Excel yang dapat diunduh.

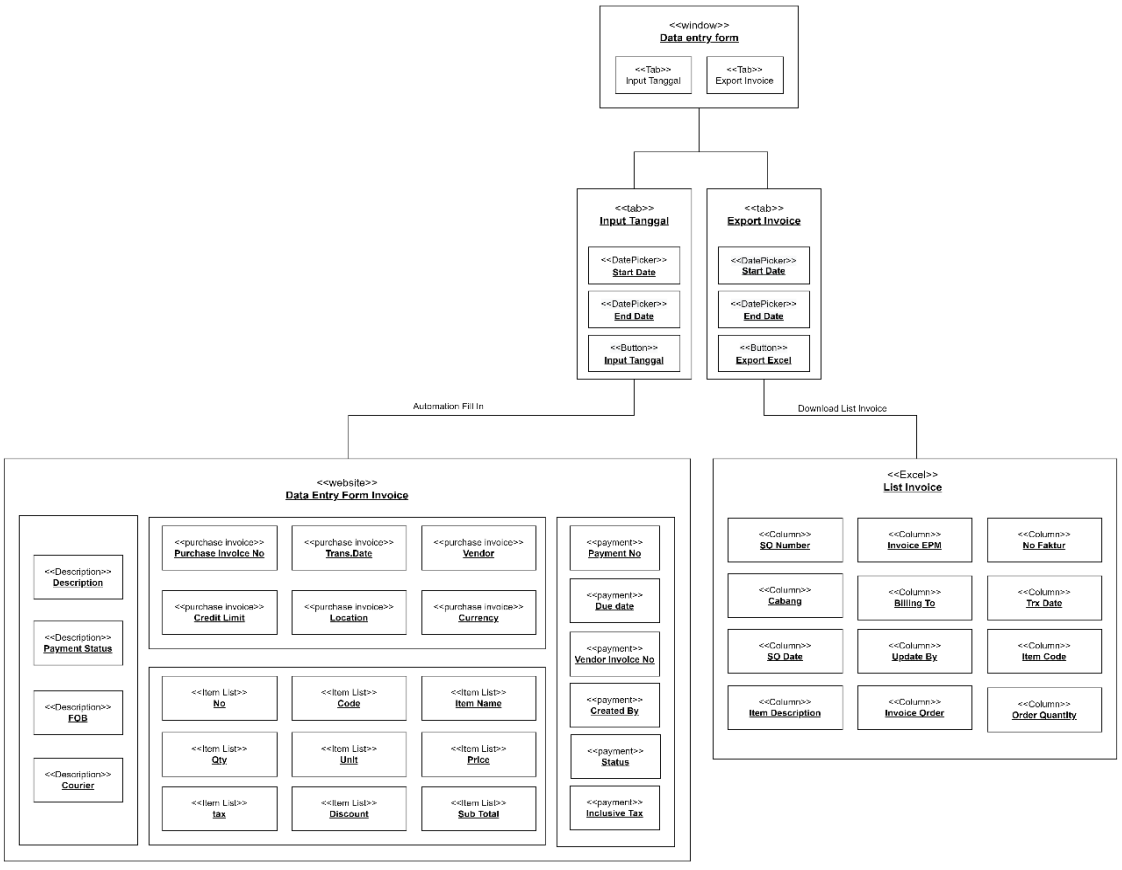
Dilanjutkan dengan Perancangan basis data merupakan tahap di mana perencanaan struktur dan tata letak penyimpanan data yang akan digunakan dalam aplikasi RPA. Mencakup pemodelan entitas dan atribut, perancangan *database*, serta hubungan antar tabel. Perancangan basis data mengacu pada konsep dasar dan hubungan yang telah diidentifikasi dalam sistem. Tahap perancangan basis data dilakukan dalam bentuk basis data logis, yang berbeda dengan perancangan basis data konseptual yang lebih berfokus pada gambaran umum. Dalam perancangan basis data logis, perhatian diberikan pada detail-detail *database* yang lebih mendalam [10]. Perancangan ini menggunakan desain konseptual *database* didasarkan pada konsep mendasar dan relasi yang teridentifikasi dalam sistem. Model konseptual data *warehouse* atau dimensional model merupakan suatu desain logic yang merepresentasikan data dalam

bentuk standar, dan mendukung dilakukannya akses terhadap data dengan cepat [11]. Rancangan basis data konseptual dapat ditunjukkan pada **Gambar 3**.



Gambar 3. *Conceptual Database Design*

Kemudian dari segi *user interface* pengembangan menggunakan *Windows Navigation Diagram* digunakan untuk merancang, memodelkan, dan menggambarkan urutan dialog antara perangkat lunak dan pengguna dengan memperhatikan perilaku yang diharapkan dari pengguna. *Diagram* ini merupakan model konseptual dengan tingkat abstraksi tinggi untuk mewakili kebutuhan pengguna dan persyaratan fungsional untuk menggambarkan bagaimana antarmuka pengguna akan terstruktur, termasuk tata letak halaman, tautan antar halaman, aliran navigasi, dan elemen-elemennya. Berikut merupakan gambaran *Windows Navigation Diagram* aplikasi RPA yang dapat dilihat pada **Gambar 4**.



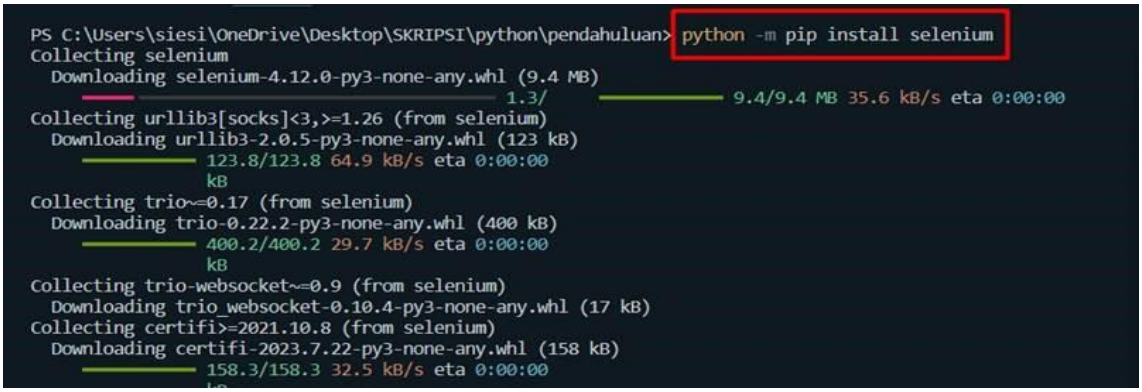
Gambar 4. *Windows Navigation Diagram*

4) Tahapan *Construction*

Selanjutnya pada tahap keempat, pengkodean, di mana gambaran kebutuhan diterjemahkan menjadi bahasa yang dapat dipahami oleh komputer. Pada tahap ini, kebutuhan pelanggan diubah menjadi kode yang nyata untuk proyek pengembangan perangkat lunak [8]. Dalam proses ini, data dikumpulkan dari analisis dan hasil perancangan yang kemudian digunakan sebagai dasar untuk merancang alur sistem aplikasi RPA sesuai dengan kebutuhan pengguna. Hasilnya juga menjadi panduan dalam tahap perancangan aplikasi. Diwujudkan pengembangan aplikasi ini yaitu dengan menggunakan bahasa pemrograman Python, dimulai dari antarmuka yang dirancang dengan menggunakan GUI Tkinter dan diintegrasikan dengan Selenium *Webdriver* untuk mengotomatisasi penginputan data *invoice*. *Database* PostgreSQL dipilih sebagai basis data untuk mengambil seluruh data *invoice* yang diperlukan. Proses pengembangan dimulai dengan instalasi Python dan pengaturan perangkat lunak yang diperlukan. Bahasa pemrograman yang akan menjadi inti dari aplikasi ini, berikut merupakan langkah-langkah penginstalan python [12]:

1. mengunduh *installer* Python untuk *Windows* dari situs python.org.
2. Pilih versi Python 3 terbaru yang tersedia.
3. Pastikan untuk memilih versi yang sesuai dengan arsitektur sistem (x86 untuk *Windows* 32-bit, x86-64 untuk *Windows* 64-bit).
4. Setelah mengunduh, jalankan *installer* Python untuk menginstalnya di komputer.

kemudian pada langkah berikutnya yaitu membangun sistem RPA dibutuhkan *tools* yang akan digunakan untuk melakukan otomatisasi interaksi aplikasi Web, yaitu Selenium *Webdriver*. Selenium webdriver berkomunikasi langsung dengan *browser*, sehingga selenium webdriver lebih cepat dari selenium RC [13]. Selenium webdriver mendukung beberapa *browser website* Selenium *Webdriver* akan memungkinkan akses ke situs *website* dan pelaksanaan penginputan otomatis berdasarkan tanggal *invoice* yang dimasukkan oleh pengguna. Untuk *command* penginstalan selenium dapat ditujukkan pada **Gambar 5**.



Gambar 5. *Install* Selenium *Webdriver*

Selain Selenium, digunakan juga *library* "*Time*" dan "OS" yang sudah tersedia dalam instalasi Python. *Library* "*Time*" digunakan untuk mengatur waktu jeda dalam operasi sistem, sementara *library* "OS" berfungsi untuk menentukan lokasi *path ChromeDriver*. Kode awal untuk setup selenium webdriver ditunjukkan pada **Gambar 6**.

Gambar 6. Setup Selenium *chromedriver*

Setelah mengaktifkan library yang diperlukan, langkah selanjutnya adalah menyiapkan penggunaan *browser*. Dalam hal ini, *ChromeDriver* yang telah disiapkan sebelumnya (https://chromedriver.chromium.org/) akan dipersiapkan dengan mengatur direktori dan preferensi yang diperlukan untuk penggunaan *browser*.

5) Tahapan *Deployment*

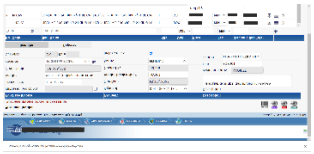
Kemudian memasuki tahap yang terakhir, *Deployment* adalah tahap implementasi perangkat lunak kepada *user*. Proses ini melibatkan kontrol yang berkelanjutan, pembaruan, evaluasi, serta pengembangan perangkat lunak berdasarkan masukan dari pelanggan agar sistem dapat beroperasi sesuai dengan kebutuhan pengguna [14]. Tahap ini dapat dianggap sebagai tahap *final* setelah alur sistem direncanakan dengan matang. Setelah proses komunikasi, perencanaan, pemodelan, dan konstruksi selesai, alur sistem yang telah *final* akan digunakan oleh para pengembang perangkat lunak. Selanjutnya, perancangan akan terus diperbarui untuk mendukung pengembangan perangkat lunak secara berkala.

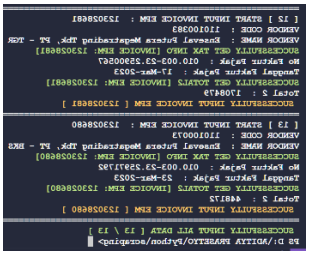
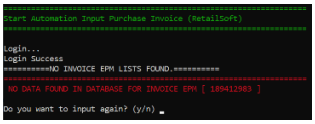
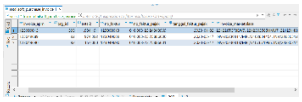
4. Hasil dan Pembahasan

4.1 Pengujian Beta

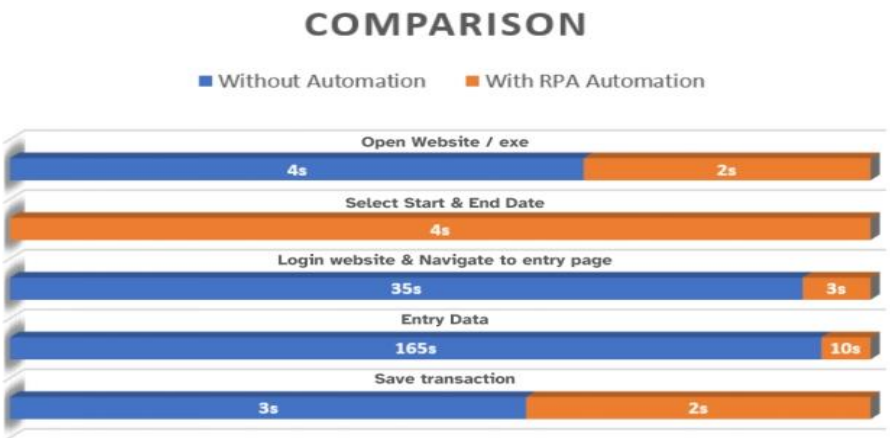
Pada tahap pengujian integrasi dan sistem, dilakukan pengujian komprehensif terhadap aplikasi yang telah dikembangkan. Langkah ini bertujuan untuk memverifikasi bahwa aplikasi berfungsi dengan baik dan sesuai dengan tujuan yang telah ditentukan. Pengujian akan menggunakan metode *Black Box Testing*, di mana aplikasi diuji tanpa memperhatikan struktur internalnya. *Black Box Testing* adalah untuk mencari fungsi yang salah atau hilang, kesalahan antar muka, kesalahan dalam struktur data atau akses *database* eksternal, kesalahan kinerja, inisialisasi dan kesalahan terminasi, validasi fungsional, kesensitifan sistem terhadap nilai *input* tertentu, dan batasan suatu data [15]. Seluruh proses pengujian melibatkan pengguna aplikasi dari awal hingga akhir untuk memastikan kelancaran dan konsistensi aplikasi. Untuk melihat *scenario* pengujian dapat dilihat pada **Tabel 2**.

Tabel 2. Scenario *Testing*

No.	***Scenario Testing***	***Sections***	***Expectation Result***	***Result***
1.	*User* mengisi *field* tanggal *start date*: 17-Mar-2023 *end date* : 17-Mar-2023 *User* memilih "*No*" untuk menjalankan sistem dalam *mode headless* *User* klik tombol *input invoice*	*Main Page*	- *Invoice* akan diinput berdasarkan tanggal mulai dan tanggal selesai. - Sistem akan menjalankan proses penginputan ini di latar belakang dan akan menampilkan status proses pada *command prompt*.	Uji Berhasil 

2.	*User* mengisi *field* tanggal *start date*: 22-Feb-2023 *end date* : 23-Feb-2023 *User* memilih "*Yes*" untuk menjalankan sistem dalam *mode headless* *User* klik tombol *input invoice*		- *Invoice* akan diinput berdasarkan tanggal mulai dan tanggal selesai. - Sistem akan menjalankan proses penginputan ini di latar belakang dan akan menampilkan status proses pada *command prompt*.	Uji Berhasil 
3.	*User* mengisi *field* tanggal *Start date* lebih besar dari *End date*		- *Invoice* tidak bisa diproses - *Return message "No Data Found"*	Uji Berhasil 
4.	*Record Log Input* pada *Database*	*Database*	- *Log invoice* akan tercatat ke dalam *database* ketika berhasil menginput *invoice*	Uji Berhasil 
5.	*User* menginput tanggal ketika *weekend / diluar jam operasional*	*Scheduler*	- Sistem *standby* sampai *populate* data berikutnya tersedia (senin-jumat 10.05 AM)	Uji Berhasil 

4.2 Pengujian Efisiensi Waktu



Gambar 7. Perbandingan Kecepatan antara RPA dan Manusia

Dalam pengujian ini, perbandingan dilakukan antara durasi proses yang dilakukan secara manual oleh tenaga manusia dengan yang dikerjakan oleh sistem RPA. Berikut adalah hasil perbandingan yang tercatat dan divisualisasikan pada grafik **Gambar 7.**

Dari analisis tabel, terlihat bahwa penggunaan sistem yang telah diimplementasikan memberikan peningkatan signifikan dalam proses *invoice processing* dibandingkan dengan pendekatan manual. RPA telah membuktikan kecepatannya dalam menyelesaikan tugas ini, menghasilkan efisiensi waktu yang substansial, dan secara efektif memangkas durasi pekerjaan yang dibutuhkan

4.3. Pengujian Akurasi

Pada sub-bab ini, fokus diberikan pada pengujian yang dilakukan untuk mengukur tingkat akurasi dalam implementasi RPA (*Robotic Process Automation*). Pengujian akurasi adalah aspek penting dalam mengevaluasi kinerja sistem RPA, Pengujian ini bertujuan untuk memberikan gambaran yang jelas mengenai sejauh mana RPA dapat memberikan hasil yang konsisten dan akurat dalam menjalankan tugas yang sebelumnya dilakukan secara manual. berikut ini hasil tes penginputan yang dilakukan selama 7 hari yang dapat dilihat dari **Tabel 3**.

Tabel 3. Akurasi Pengujian Sistem

Tanggal	Jumlah Data	Berhasil	Gagal	Akurasi
17/10/2023	118	118	0	100%
18/10/2023	176	169	7	96.02%
19/10/2023	96	95	1	98.95%
9/11/2023	198	196	2	98.98%
10/11/2023	212	211	1	99.52%

Rata-rata hasil persentase keberhasilan ini dapat dihitung menggunakan rumus berikut:

$$Rata - rata\ Akurasi = \frac{\text{Total Akurasi}}{\text{Jumlah Pengujian}}$$

Hasil Perhitungan rata-rata akurasi $\frac{100\% + 96.02\% + 98.95\% + 98.98\% + 99.52\%}{5 \text{ pengujian}}$ = 98.69%

Dalam hasil pembahasan mengenai akurasi sistem RPA penginputan invoice, evaluasi dilakukan dengan menghitung total kegagalan dan keberhasilan selama 5 hari uji coba. Hasil penelitian menunjukkan bahwa rata-rata persentase akurasi mencapai 98.69%. pencatatan invoice yang gagal akan tetap dicatat pada saat pencatatan berikutnya. Evaluasi akurasi ini memberikan gambaran positif terkait kinerja sistem selama periode pengujian, menunjukkan tingkat keberhasilan yang tinggi dalam tugas penginputan invoice.

Adapula penelitian tentang akurasi sistem RPA penginputan invoice telah banyak dilakukan oleh peneliti terdahulu. Beberapa metode yang umum digunakan untuk mengetes akurasi sistem RPA ini adalah perbandingan dengan data manual, menggunakan data uji, dan menggunakan metode statistik.

Peneliti Quaum, et al. menggunakan metode perbandingan dengan data manual untuk mengetes akurasi sistem RPA. Data yang diinput oleh sistem RPA dibandingkan dengan data yang diinput secara manual oleh 10 orang operator. Hasil penelitian menunjukkan bahwa akurasi sistem RPA mencapai 96,7%.

Peneliti Rizkiyani, et al. menggunakan metode perbandingan dengan data manual untuk mengetes akurasi sistem RPA. Data yang diinput oleh sistem RPA dibandingkan dengan data yang diinput secara manual oleh 3 orang operator. Hasil penelitian menunjukkan bahwa akurasi sistem RPA mencapai 97,5%

Peneliti abadi, et al. menggunakan metode perbandingan dengan data manual untuk mengetes akurasi sistem RPA. Data yang diinput oleh sistem RPA dibandingkan dengan data yang diinput secara manual oleh 2 orang operator. Hasil penelitian menunjukkan bahwa akurasi sistem RPA mencapai 98,8%.

Secara umum, dapat disimpulkan bahwa hasil penelitian ini konsisten dengan hasil penelitian terdahulu. Akurasi sistem RPA penginputan invoice umumnya berada pada kisaran 90-99%. Hal ini menunjukkan bahwa sistem RPA penginputan invoice yang diuji memiliki kinerja yang baik. Perbedaan hasil akurasi antara penelitian saya dan peneliti terdahulu dapat disebabkan oleh beberapa faktor, seperti jumlah data yang digunakan, jenis kesalahan yang dimasukkan ke dalam data uji, dan metode yang digunakan untuk mengetes akurasi sistem RPA.

5. Simpulan

Berdasarkan hasil penelitian ini, penerapan sistem RPA berbasis Python menggunakan Selenium *Webdriver* untuk penginputan *purchase invoice* secara otomatis di PT Enseval Putera Megatrading terbukti dapat meningkatkan produktivitas karyawan, pengiriman *invoice* tepat waktu, serta mendukung kelancaran operasional dan hubungan bisnis dengan *partner fulfillment* Perusahaan. Aplikasi RPA ini dapat berjalan dengan baik dan berhasil sesuai ekspektasi. Hasil perbandingan antara penginputan manual oleh tenaga manusia dengan yang dikerjakan oleh sistem RPA menunjukkan penghematan waktu yang jauh lebih baik, yaitu hingga 186 detik dalam satu *invoice*. Akurasi keberhasilan penginputan *invoice* yang diinput selama 5 hari mencapai 98,69%.

Dengan demikian, penerapan sistem RPA ini dapat menjadi solusi yang tepat untuk mengatasi permasalahan penginputan *purchase invoice* secara manual yang rentan terhadap kesalahan dan keterlambatan serta mengurangi peningkatan jam kerja karyawan.

Daftar Referensi

[1] PT Enseval Putera Megatrading, “Sekilas Enseval,”PT Enseval Putera Megatrading,Tbk. 2020. [Online]. Available: https://www.enseval.com. [Diakses 10 November 2023].

[2] M. D. Quaum, A. Jain, V. Solanki, D. Gokulapati dan A. Preetham, “Method of *Robotic Process Automation* in *Invoice* Processing and Mailing,” *International Journal of Research in Engineering, Science and Management,* vol. 5, no. 4, pp. 100-103, 2022.

[3] R. P. Avrianto, M. I. Faried, E. Dazki dan R. E. Indrajit, “Robotic Process Automation For Quality Control Assessment Using Selenium,” *Jurnal Teknik Informatika (JUTIF),* vol. 3, no. 5, pp. 1301-1312, 2022.

[4] D. R. Rizkiyani, K. Sujatmoko dan F. Akhyar, “Implementasi Virtual Costumer Service Dengan Robotic Process Automation (RPA) Dan Kecerdasan Buatan,” *e-Proceeding of Engineering,* vol. 8, no. 6, pp. 11582-11587, 2021.

[5] G. I. Abadi, K. Sujatmoko dan Rustam, “Perancangan Robot Untuk Ekstraksi Data *Invoice* Menggunakan *Robotic Process Automation* (RPA) Yang Terintegrasi Dengan *Machine Learning Document Understanding*,” *e-Proceeding of Engineering,* vol. 8, no. 6, pp. 3688-3702, 2022.

[6] S. Sagar, S. Sania, P. Atharva dan P. Manoj, “*Invoice* Processing Using *Robotic Process Automation*,” *International Journal of Scientific Research in Computer Science, Engineering and Information Technology,* vol. 6, no. 2, pp. 216-223, 2020.

[7] T. Pricillia dan Zulfachmi, “Survey Paper: Perbandingan Metode Pengembangan Perangkat Lunak (*Waterfall*, Prototype, RAD),” *Bangkit Indonesia,* vol. X, no. 1, pp. 6-9, 2021.

[8] F. N. R. Saputri, W. Harianto dan D. Aditya, “Analisis Dan Perancangan Aplikasi Ibu Siaga Dengan Pendekatan Metode Waterfall,” *KURAWAL Jurnal Teknologi, Informasi dan Industri,* vol. 4, no. 1, pp. 43-47, 2021.

[9] A. Asyhadi, “Analisis dan Perancangan Sistem Informasi Purchase Order Percetakan Menggunakan Whatsapp Gateway pada PT MIG,” *Processor: Jurnal Ilmiah Sistem Informasi, Teknologi Informasi dan Sistem Komputer,* vol. 16, no. 1, pp. 20-30, 2021.

[10] L. Davidson, Pro SQL Server Relational *Database* Design and Implementation, United States: Apress, 2021.

[11] Y. Indarta, D. Irfan, M. Muksir, W. Simatupang dan F. Ranuharja, “Analisis dan Perancangan *Database* Menggunakan Model Konseptual Data Warehouse Sistem Manajemen Transaksi Toko Online Haransaf,” *Edukatif : Jurnal Ilmu Pendidikan,* vol. 3, no. 6, pp. 4448-4455, 2021.

[12] A. R. d. Paz, Tkinter GUI Application *Development* Cookbook, Birmingham, UK: Packt Publishing Ltd, 2018.

[13] S. Nyamathulla, D. P. Ratnababu, N. S. Shaik dan B. L. N, "A Review on Selenium Web Driver with Python," *Annals of the Romanian Society for Cell Biology,* vol. 25, no. 4, pp. 1670-16768, 2021.

[14] N. A. Widiastuti dan T. Tamrin, "Penerapan Aplikasi Mobile Location Based Service Untuk Persebaran Usaha Mikro Kecil Menengah Dikabupaten Jepara," *SIMETRIS,* vol. 11, no. 1, pp. 271-277, 2020.

[15] R. Parlika, T. A. Nisaa, S. M. Ningrum dan B. A. Haque, "Studi Literatur Kekurangan dan Kelebihan Pengujian *Black Box*," *Teknomatika,* vol. 10, no. 2, pp. 131-130, 2020.