Jurnal Teknologi Sistem Informasi dan Aplikasi **ISSN: 2654-3788**
Penerbit: Program Studi Teknik Informatika Universitas Pamulang **e-ISSN: 2654-4229**
Vol. 6, No. 3, Juli 2023 (299-310) **DOI: 10.32493/jtsi.v6i3.30328**

Otomatisasi Pengujian Perangkat Lunak Menggunakan Telenium pada Aplikasi Berbasis *Kivy Framework*

Alim Budhi Utomo [1], Suprihadi [2]

[1]Teknik Informatika, Universitas Kristen Satya Wacana, Jl. Dr. O. Notohamidjojo No.1, Salatiga, Jawa Tengah, Indonesia, 50715
e-mail: [1] 672019213@student.uksw.edu, [2] suprihadi@uksw.edu

Submitted Date: May 23rd, 2023 Reviewed Date: May 31st, 2023
Revised Date: June 02nd, 2023 Accepted Date: June 03rd, 2023

Abstract

PT XYZ is a retail company that uses POS (Point of Sales) application in each of its stores to record transactions, automatic calculations, and print receipts. PT XYZ has millions of transactions every day, therefore system testing is needed to ensure that the POS application is running properly. When testing is done manually, there are often various problems such as inaccurate results due to limited human resources to a lot of time consuming because it is done repeatedly. To overcome these problems, test automation is made using Telenium which is integrated with Jenkins. This research was conducted based on the black box testing method. Input in the system in the form of a test scenario is then executed and the results will be sent via email. During testing, if an error occurs, the location of the error will be displayed in detail and sent via email by Jenkins. The results show that the application of Telenium in testing Kivy-based applications facilitates the testing process and accelerates the discovery of errors in the system so as to improve the quality of applications with a more efficient approach.

Keywords: Automation Testing; Telenium; Jenkins; Point of Sales; Black Box Testing

Abstrak

PT XYZ merupakan perusahaan ritel yang menggunakan aplikasi POS (*Point of Sales*) di setiap gerainya untuk mencatat transaksi, perhitungan otomatis, hingga mencetak nota. PT XYZ memiliki jutaan transaksi setiap harinya, maka dari itu pengujian sistem diperlukan untuk menjamin bahwa aplikasi POS berjalan dengan baik. Ketika dilakukan pengujian secara manual kerap terjadi berbagai masalah seperti hasil yang tidak akurat karena keterbatasan sumber daya manusia hingga memakan waktu yang banyak karena dilakukan secara berulang. Untuk mengatasi masalah tersebut maka dibuatlah otomatisasi pengujian menggunakan Telenium yang diintegrasikan dengan Jenkins. Penelitian ini dilakukan berdasarkan metode *black box testing*. Masukan dalam sistem berupa skenario pengujian kemudian dieksekusi dan hasilnya akan dikirimkan melalui email. Pada saat pengujian, jika terjadi kesalahan maka letak kesalahan tersebut akan ditampilkan secara rinci dan dikirimkan melalui email oleh Jenkins. Hasil penelitian menunjukkan bahwa penerapan Telenium dalam pengujian aplikasi berbasis Kivy memudahkan proses pengujian dan mempercepat penemuan kesalahan dalam sistem sehingga dapat meningkatkan kualitas aplikasi dengan pendekatan yang lebih efisien.

Keywords: Pengujian Otomatis; Telenium; Jenkins; Point of Sales; Pengujian Black Box

1 Pendahuluan

PT XYZ adalah satu dari sekian banyak perusahaan ritel yang menggunakan aplikasi POS (*Point of Sales*) di setiap gerainya untuk mencatat data transaksi, penghitungan otomatis, dan mencetak nota untuk pelanggan. POS adalah suatu sistem dalam dunia bisnis yang menggunakan perangkat lunak dan perangkat keras yang terhubung satu sama lain untuk memudahkan proses transaksi (Pratama & Somya, 2021).

Sebagai salah satu perusahaan ritel yang memiliki ribuan gerai, membuat PT XYZ memiliki jutaan transaksi per harinya. Untuk memastikan bahwa aplikasi POS dapat berjalan sesuai dengan fungsionalitas yang diinginkan maka dibutuhkan pengujian sistem. Pengujian sistem bertujuan untuk melihat tingkat kesalahan pada suatu perangkat lunak (Jaya, 2018).

Dalam melakukan pengujian sistem, setidaknya dibutuhkan 30% hingga 35% dari total upaya dalam sebuah proyek (Mustofa & Fajar, 2018). Maka dari itu pemilihan strategi dan teknik pengujian menjadi sangat penting untuk menjamin kualitas suatu perangkat lunak (Utomo et al., 2018). Pengujian manual adalah salah satu jenis pengujian perangkat lunak yang banyak digunakan, termasuk di PT XYZ.

Ketika pengujian dilakukan secara manual, seringkali menemui hasil yang tidak akurat. Hal tersebut bisa disebabkan karena jumlah sumber daya manusia yang terbatas atau bisa juga dikarenakan kesalahan dari penguji (Thooriqoh et al., 2021). Selain itu pengujian manual memakan sumber daya dan waktu yang banyak karena sifatnya yang berulang untuk menjamin bahwa sebuah sistem bebas dari kesalahan ketika terdapat fitur baru dalam pengembangan yang berkelanjutan (Panjaitan & Mantra, 2020)**.**

Pengujian otomatis merupakan pengujian yang dijalankan otomatis dengan memvalidasi hasil yang diharapkan dengan hasil aktual yang diperoleh. Pengujian ini bergantung pada skrip pengujian yang sudah dibuat. Keuntungan yang didapatkan ketika melakukan pengujian otomatis yaitu dapat dilakukan berkali-kali, umpan balik yang lebih cepat, dan pastinya dijalankan secara otomatis (Wicaksono & Rani, 2022).

PT XYZ memiliki transaksi yang sangat padat terutama pada tiap aplikasi POS di setiap gerainya. Karena fitur di dalam aplikasi tersebut sangat kompleks, maka semakin tinggi pula kemungkinan terjadinya kesalahan dalam sistem. Aplikasi POS yang digunakan oleh PT XYZ dibangun dengan menggunakan bahasa pemrograman Python dan *framework* Kivy. Kivy merupakan salah satu *library* Python yang bersifat *open source* untuk membangun aplikasi berbasis *desktop* atau *mobile* (Bhoyarkar et al., 2019).

Aplikasi POS di PT XYZ masih dalam proses pengembangan fitur yang terkadang bisa merusak fitur yang lain. Oleh karena itu, perlu dilakukan pengujian regresi untuk memastikan tidak ada kesalahan akibat dari pengembangan tersebut (Maspupah & Bakhrun, 2021). Telenium merupakan sebuah *framework* yang dapat digunakan untuk pengujian jarak jauh pada aplikasi berbasis Kivy (Virbel, 2021)**.**

Berdasarkan permasalahan yang telah diuraikan, dalam penelitian ini akan dirancang suatu sistem otomatisasi pengujian perangkat lunak yang diimplementasikan pada aplikasi POS. Tidak hanya itu, dalam penelitian ini juga terdapat integrasi antara Telenium dengan Jenkins. Dengan Jenkins pengujian akan berjalan di belakang layar.

2 Penelitian Terdahulu

Kosasih menyatakan bahwa perlu adanya analisis untuk membandingkan efektivitas pengujian aplikasi secara otomatis dengan pengujian secara manual (Kosasih & Cahyono, 2020). Penelitian ini menggunakan Katalon Studio dengan menerapkan metode *black box testing*. Hasil analisis perbandingan dari penelitian ini menunjukkan bahwa pemilihan metode pengujian tergantung dari kebutuhannya, apabila dibutuhkan pengujian berulang yang menguji banyak *platform* dengan banyak data, maka dibutuhkan pengujian otomatis.

Setiawan menyatakan bahwa pengujian secara manual rawan akan kesalahan serta memakan waktu yang lama. Selain itu pengujian manual juga dinilai tidak efektif sehingga menyebabkan pengujian yang repetitif. Pelaporan kesalahan dalam sistem yang dilakukan secara manual akan menyebabkan keterlambatan (Setiawan et al., 2019). Penelitian ini menggunakan Selenium dalam pengujian otomatisnya, sedangkan untuk pelaporan hasil pengujiannya menggunakan Jenkins. Hasil dari penelitian ini yaitu sebuah sistem pengujian otomatis yang terintegrasi Jenkins. Pengujian otomatis dapat mengurangi kesalahan manusia ketika melakukan pengujian secara manual dan meningkatkan efektivitas serta efisiensi pengujian aplikasi**.**

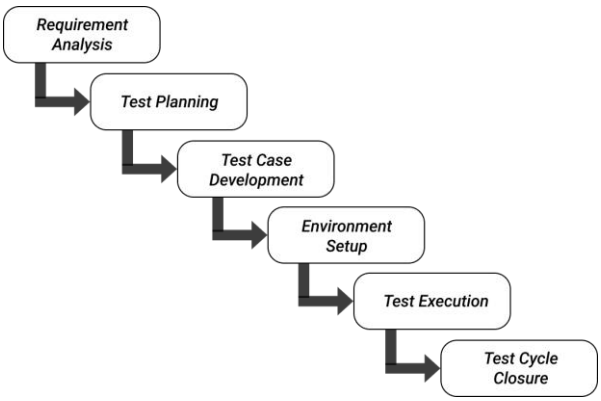
Mustika menyatakan bahwa pengujian otomatis merupakan pengujian dengan menggunakan alat dan skrip yang dapat menjalankan skenario pengujian secara berulang-ulang, cepat, akurat, serta menghemat waktu dan biaya jika dibandingkan dengan pengujian manual (Mustika & Novrina, 2018). Pada penelitian ini Selenium digunakan untuk melakukan pengujian

otomatis pada aplikasi berbasis web. Hasil dari penelitian ini yaitu pengujian otomatis yang dapat menghasilkan status sukses atau gagal untuk setiap kasus pengujian yang telah dibuat serta panjang durasi pengujiannya.

3 Metode Penelitian

Penelitian ini menggunakan metodologi yang mengadopsi siklus hidup pengujian perangkat lunak atau *Software Testing Life Cycle* (STLC) dan menyesuaikan dengan kebutuhan perusahaan. Pengujian aplikasi dilakukan secara otomatis menggunakan Telenium di mana aplikasi yang diuji berbasis Kivy. Penelitian ini dilakukan berdasarkan metode *black box testing* di mana pengujian dilakukan dengan mengamati masukan dan keluaran dari perangkat lunak (Fahrezi et al., 2022).

Gambar 1 merupakan ilustrasi tahapan penelitian yang akan dilakukan dalam menyelesaikan penelitian ini.



Gambar 1. Metode Penelitian

Tahapan penelitian mengenai perancangan otomatisasi pengujian perangkat lunak pada aplikasi berbasis Kivy seperti yang ditunjukkan pada Gambar 1 adalah sebagai berikut (Arfan & Hendrik, 2022):

1) *Requirement Analysis*, pada tahap ini dilakukan interaksi dengan klien, *business analyst*, dan pengembang aplikasi untuk memahami permasalahan yang terjadi serta kebutuhan fungsional dari aplikasi yang akan diuji. Kebutuhan fungsional merupakan jenis kebutuhan yang mencakup seluruh proses yang dapat dilakukan. Karena aplikasi POS di PT XYZ masih dalam proses pengembangan fitur, setiap kali ada fitur baru, fitur lama harus diuji kembali sehingga terjadi pengujian berulang. Oleh karena itu, solusi terbaik yaitu dengan membuat sistem pengujian otomatis. Seorang penguji dapat mengulangi pekerjaan di setiap siklus pengujian hanya dengan memutar kembali skenario yang pernah dibuat (Yutia & Satrinia, 2021).
2) *Test Planning*, pada tahap ini akan dibuat strategi dan perencanaan berdasarkan *requirement analysis*. Hasil dari tahap ini adalah penentuan IDE Pycharm dan Telenium dalam pembuatan sistem pengujian otomatis. Telenium dapat bekerja pada aplikasi berbasis Kivy sama halnya dengan aplikasi POS PT XYZ yang juga dibangun menggunakan Kivy. Jenkins akan dimanfaatkan untuk mendapatkan laporan hasil pengujian. Pada tahap ini juga akan dilakukan identifikasi komponen sistem yang akan diuji.
3) *Test Case Development*, pada tahap ini seluruh skenario pengujian yang telah ditentukan pada tahap sebelumnya baik skenario positif maupun skenario negatif akan dikonversi menjadi skrip otomatisasi. Skrip tersebut akan ditulis dalam bahasa pemrograman Python dengan memanfaatkan Telenium. Hasil dari tahap ini yaitu sebuah skrip otomatisasi yang siap digunakan dan diintegrasikan dengan Jenkins.
4) *Environment Setup*, pada tahap ini akan dipastikan bahwa lingkungan pengujian baik perangkat lunak maupun perangkat keras berjalan sesuai dengan rencana. Beberapa hal yang perlu diperhatikan dari segi perangkat lunak yaitu IDE Pycharm, Python, Telenium, dan Jenkins. Beberapa perangkat lunak tersebut harus dipastikan dapat berjalan sebagaimana mestinya. Di samping itu, juga perlu dipersiapkan sebuah perangkat yang mampu merancang serta menjalankan pengujian otomatis dengan baik.
5) *Test Execution*, setelah semua skenario pengujian diubah ke dalam skrip otomatisasi dan memastikan bahwa setiap *environment* berjalan baik, langkah selanjutnya yaitu melakukan eksekusi pengujian otomatis dengan memanfaatkan Jenkins. Jenkins akan membantu menjalankan pengujian otomatis berjalan di belakang layar. Hasil yang diperoleh dari kegiatan ini adalah laporan hasil pengujian dan temuan kesalahan dalam sistem. Terdapat berbagai macam fitur yang

disediakan oleh Jenkins, namun pada penelitian ini, Jenkins Pipeline dipilih untuk menjalankan proses pengujian. Pipeline pada Jenkins merupakan rangkaian fitur yang mendukung pengembangan dan proses integrasi aplikasi dengan konsep *continuous delivery* (Putra, 2018).

6) *Test Cycle Closure*, tahap ini merupakan tahap terakhir yang dilakukan pada penelitian ini. Di tahap ini dilakukan analisis serta evaluasi terhadap pengujian yang telah dilakukan, apakah ditemukan kesalahan pada aplikasi POS selama dijalankannya proses pengujian. Hasil dari tahap ini berupa kesimpulan yang menyatakan apakah seluruh fungsionalitas dari aplikasi POS sudah berjalan dengan baik berdasarkan seluruh skenario yang sudah dibuat pada tahap sebelumnya. Di samping itu, perlu juga dilakukan uji kelayakan sistem otomatisasi pengujian perangkat lunak yang telah dibangun dengan membuat kuesioner di mana kuesioner tersebut memuat pertanyaan-pertanyaan yang mengacu pada standard ISO 9126. Dari tahap ini juga akan diperoleh kesimpulan yang menyatakan apakah sistem otomatisasi yang dibangun sudah layak untuk digunakan atau bahkan sebaliknya.

4 Hasil dan Pembahasan

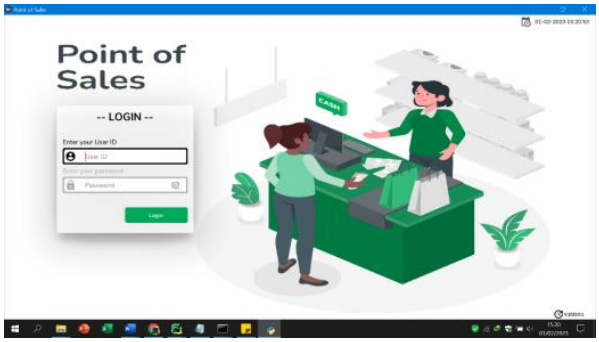
Di dalam bab ini penjelasan akan dibagi ke dalam enam subbab berdasarkan tahapan penelitian yang telah ditentukan sebelumnya meliputi tahap *requirement analysis*, *test planning*, *test case development*, *test environment*, *test execution*, dan tahap terakhir yakni *test cycle closure.*

4.1 Requirement Analysis

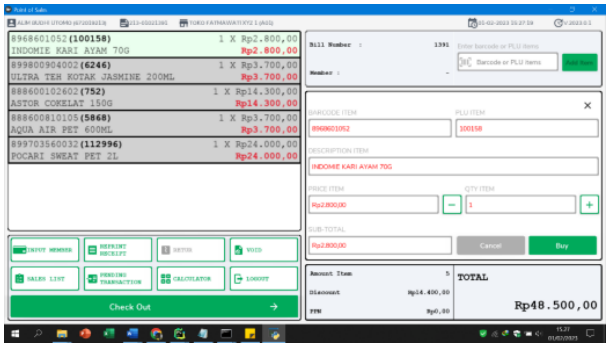
Aplikasi POS di PT XYZ merupakan sebuah aplikasi yang dikhususkan bagi seorang kasir toko untuk memudahkan dalam melakukan transaksi ketika melayani pembeli. Berikut merupakan kebutuhan fungsional dari aplikasi POS:

a. Sistem dapat melakukan pengolahan data transaksi seperti melihat barang yang dijual, memasukkan barang yang akan dibeli, dan menghapus barang yang tidak dibeli.
b. Sistem dapat mengelola data pelanggan seperti memasukkan nomor *member*.
c. Sistem mampu melakukan penghitungan otomatis selama transaksi berlangsung untuk mengetahui total biaya yang harus dibayarkan oleh pelanggan.
d. Sistem mampu menyelesaikan proses pembayaran dan mencetak nota pembelian.
e. Sistem dapat menunda proses transaksi yang sedang berlangsung dan memulai transaksi yang baru.

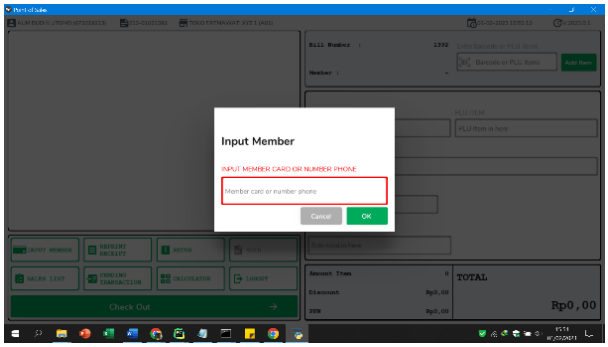
Aplikasi POS di PT XYZ memiliki empat fitur utama meliputi halaman masuk aplikasi, halaman pembelian, halaman *input member*, dan halaman pembayaran. Adapun tampilan dari empat fitur utama aplikasi POS dapat dilihat pada Gambar 2, Gambar 3, dan Gambar 4, dan Gambar 5.



Gambar 2. Halaman Masuk Aplikasi POS

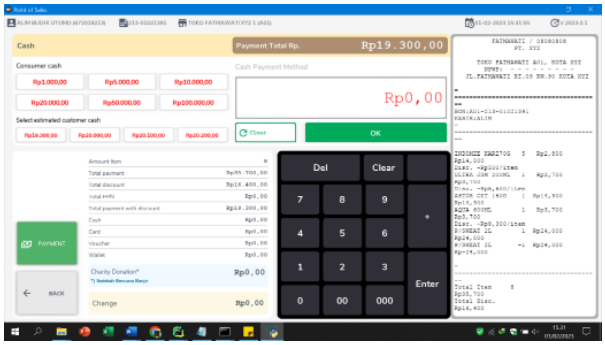


Gambar 3. Halaman Pembelian POS



Gambar 4. Halaman *Input Member* POS

Jurnal Teknologi Sistem Informasi dan Aplikasi **ISSN: 2654-3788**
Penerbit: Program Studi Teknik Informatika Universitas Pamulang **e-ISSN: 2654-4229**
Vol. 6, No. 3, Juli 2023 (299-310) **DOI: 10.32493/jtsi.v6i3.30328**



Gambar 5. Halaman Pembayaran POS

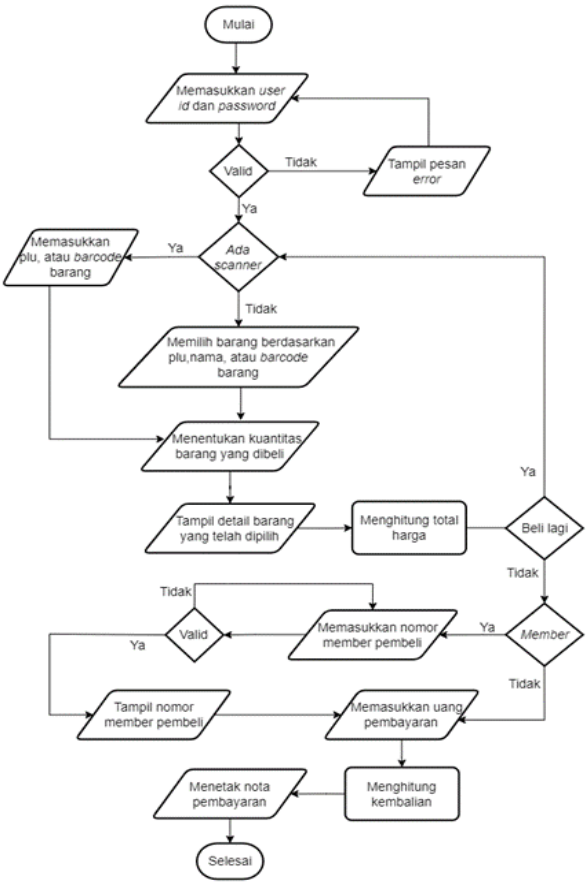
Alur dari aplikasi POS yang akan dijadikan acuan dalam pembuatan skenario pengujian menggunakan Telenium berdasarkan kebutuhan fungsionalnya dapat dilihat pada Gambar 6.

4.2 Test Planning

Untuk mengoptimalkan waktu ketika melakukan pengujian, akan dibuat sebuah sistem otomatisasi pengujian perangkat lunak menggunakan Telenium yang diintegrasikan dengan Jenkins. Perencanaan biaya pada tahap ini tidak akan menjadi fokus dalam pembahasan dikarenakan semua alat yang digunakan bersifat *open source* sehingga dapat digunakan secara gratis, dimodifikasi, dan didistribusikan oleh siapa saja, termasuk untuk kepentingan penelitian.

Setelah memahami fungsionalitas dan alur kerja dari aplikasi POS selanjutnya skenario pengujian dapat dibuat seperti yang tertera pada Tabel 1. Skenario pengujian meliputi aktivitas masuk ke aplikasi, melakukan transaksi hingga melakukan pembayaran serta mencetak nota.

Skenario pengujian terdiri dari 16 skenario positif dan 11 skenario negatif. Skenario nomor 1 – 4 berada pada halaman masuk aplikasi, 5 – 22 dan 27 pada halaman pembelian, serta nomor 23 – 26 pada halaman pembayaran.



Gambar 6. *Flowchart* aplikasi POS

Tabel 1. Skenario Pengujian

No	Fungsionalitas	Skenario Pengujian	Hasil yang Diharapkan
1	Masuk Aplikasi.	Memasukkan id pengguna yang sudah kadaluwarsa. (Skenario Negatif)	Sistem menampilkan pesan kesalahan yang mengatakan bahwa id pengguna kadaluwarsa.
2	Masuk Aplikasi.	Mencoba masuk tanpa memasukkan id pengguna atau kata sandi. (Skenario Negatif)	Sistem menampilkan pesan kesalahan yang mengatakan bahwa id pengguna dan kata sandi tidak boleh kosong.
3	Masuk Aplikasi.	Memasukkan id pengguna atau kata sandi secara asal dan tidak terdaftar pada sistem. (Skenario Negatif)	Sistem menampilkan pesan kesalahan yang mengatakan bahwa id pengguna dan kata sandi tidak ditemukan atau salah.
4	Masuk Aplikasi.	Memasukkan id pengguna dan kata sandi yang sesuai dan terdaftar pada sistem. (Skenario Positif)	Berhasil masuk dan sistem menampilkan halaman pembelian yang masih kosong.

No	Fungsionalitas	Skenario Pengujian	Hasil yang Diharapkan
5	Menunda Transaksi.	Mencoba menunda transaksi ketika tidak ada transaksi yang sedang berlangsung. (Skenario Negatif)	Sistem menampilkan pesan kesalahan yang mengatakan bahwa kasir harus mengisi keranjang sebelum melakukan penundaan transaksi.
6	*Checkout.*	Mencoba melakukan *checkout* ketika tidak ada satu pun barang di keranjang. (Skenario Negatif)	Sistem menampilkan pesan kesalahan yang mengatakan bahwa keranjang kosong dan kasir harus mengisinya minimal dengan satu barang.
7	Daftar Barang yang Dijual.	Mencari barang berdasarkan nama barang di kolom pencarian pada daftar barang. (Skenario Positif)	Sistem berhasil menampilkan barang yang dicari.
8	Daftar Barang yang Dijual.	Mencari barang berdasarkan PLU atau *Price Look-Up* barang di kolom pencarian pada daftar barang. (Skenario Positif)	Sistem berhasil menampilkan barang yang dicari.
9	Daftar Barang yang Dijual.	Mencari barang berdasarkan kode batang barang di kolom pencarian pada daftar barang. (Skenario Positif)	Sistem berhasil menampilkan barang yang dicari.
10	Kolom Kode Batang dan PLU.	Mencari barang tanpa memasukkan kode batang atau PLU pada kolom yang tersedia. (Skenario Negatif)	Sistem menampilkan pesan kesalahan yang mengatakan bahwa kolom tidak boleh kosong.
11	Kolom Kode Batang dan PLU.	Memasukkan kode batang atau PLU secara asal yang tidak terdaftar pada sistem. (Skenario Negatif)	Sistem menampilkan pesan kesalahan yang mengatakan bahwa barang tidak ditemukan.
12	Kolom Kode Batang dan PLU.	Memasukkan PLU yang sesuai dan terdaftar pada sistem. (Skenario Positif)	Sistem berhasil menampilkan barang yang dicari.
13	Kolom Kode Batang dan PLU.	Memasukkan kode batang yang sesuai dan terdaftar pada sistem. (Skenario Positif)	Sistem berhasil menampilkan barang yang dicari.
14	Perubahan jumlah barang.	Mengubah jumlah barang dari satu barang menjadi nol barang. (Skenario Negatif)	Sistem menampilkan pesan kesalahan yang mengatakan bahwa batas minimal pembelian yaitu satu.
15	Perubahan jumlah barang.	Mengubah jumlah barang dengan mengisi kolom yang tersedia. (Skenario Positif)	Sistem berhasil menampilkan perubahan jumlah barang yang diinginkan.
16	Perubahan jumlah barang.	Mengubah jumlah barang dengan menekan tombol tambah dan kurang yang tersedia. (Skenario Positif)	Sistem berhasil menampilkan perubahan jumlah barang yang diinginkan.
17	Pembatalan pemilihan barang.	Membatalkan barang yang dipilih sebelum dimasukkan ke keranjang. (Skenario Positif)	Sistem berhasil menghilangkan detail barang yang telah dibatalkan.
18	Kalkulator.	Mengoperasikan kalkulator dan membandingkan hasilnya dengan total harga yang tertera. (Skenario Positif)	Sistem berhasil membuka kalkulator dan hasil hitungannya sesuai dengan total harga yang tertera.
19	Memasukkan nomor *member*.	Memasukkan nomor *member* pada kolom yang tersedia. (Skenario Positif)	Sistem berhasil menampilkan nomor *member* sesuai dengan yang telah dimasukkan.
20	Hapus barang.	Memasukkan kata sandi yang salah ketika melakukan otorisasi hapus barang. (Skenario Negatif)	Sistem menampilkan pesan kesalahan yang mengatakan bahwa kata sandi tidak valid.
21	Hapus barang.	Memasukkan kata sandi yang benar ketika melakukan otorisasi hapus barang. (Skenario Positif)	Sistem menampilkan perubahan warna menjadi warna merah pada keterangan barang yang dimaksud.

No	Fungsionalitas	Skenario Pengujian	Hasil yang Diharapkan
22	Menunda Transaksi.	Menunda transaksi dengan menekan tombol *pending*. (Skenario Positif)	Sistem menghilangkan sementara isi keranjang dan tombol *pending* berubah menjadi *unpending*.
23	Hapus Kolom Pembayaran Tunai.	Memasukkan nominal tunai kemudian menghapusnya dengan menekan tombol *clear*. (Skenario Positif)	Sistem menampilkan kolom pembayaran tunai dengan keterangan Rp 0,00.
24	Pembayaran.	Melakukan pembayaran tanpa memasukkan nominal uang tunai. (Skenario Negatif)	Sistem menampilkan pesan kesalahan yang mengatakan bahwa tidak ada jumlah uang yang dimasukkan.
25	Pembayaran.	Melakukan pembayaran dengan jumlah uang yang kurang dari total belanja. (Skenario Negatif)	Sistem menampilkan pesan kesalahan yang mengatakan bahwa jumlah uang kurang dari total belanja.
26	Pembayaran.	Melakukan pembayaran dengan jumlah uang yang lebih dari total belanja. (Skenario Positif)	Sistem menampilkan total kembalian kemudian berhasil mencetak nota.
27	Keluar Aplikasi.	Keluar dari aplikasi dengan menekan tombol *logout.* (Skenario Positif)	Sistem kembali ke halaman masuk aplikasi.

4.3 Test Case Development

Pada tahap ini, seluruh skenario yang telah dibuat pada tahap sebelumnya yang juga dapat dilihat pada pada Tabel 1 akan diubah ke dalam skrip otomatisasi yang ditulis menggunakan bahasa pemrograman Python. Hasil dari tahap ini yakni 27 berkas .json sesuai dengan jumlah skenario yang sudah ditentukan pada tahap sebelumnya. Di setiap berkas tersebut berisi langkah-langkah dalam menekan, mengisi masukan, ataupun memvalidasi objek pada fitur aplikasi POS sesuai dengan fungsionalitas dari aplikasi tersebut.

Kode Program 1. Fungsi untuk Mengekspor Skenario

```
def export_scenario(self):
    with open(self.scenario + '.scenario',
mode = 'w', encoding='utf-8') as
scenario_file:
        json.dump(self.actions,
scenario_file)
```

Kode Program 1 merupakan fungsi yang digunakan untuk membuat berkas .json secara otomatis. Fungsi tersebut mengubah perintah-perintah Telenium menjadi format tertentu kemudian akan dimasukkan ke dalam berkas berekstensi .scenario. Semua skenario akan diekspor ke dalam bentuk json. Contoh skenario yang merupakan hasil dari pemanfaatan fungsi tersebut ditunjukkan pada Kode Program 2.

Kode Program 2. Contoh Skenario Positif

```
[
  {
    "command": "SET_ATTR",
    "selector":
"//BorderIconLeftTextField[0][@hint_text=\"
Barcode or PLU items\"]",
    "field": "text_input_text",
    "value": "5868",
    "delay": 2
  },
  {
    "command": "WAIT_CLICK",
    "selector":
"//PrimaryButton[0][@text=\"Add Item\"]",
    "delay": 1
  },
  {
    "command": "ASSERT_EQUAL",
    "selector": "//BorderTextField[1]",
    "field": "text",
    "value": "5868",
    "delay": 3
  },
  {
    "command": "WAIT_CLICK",
    "selector": "//PrimaryButton[1]",
    "delay": 1
  },
  {
    "command": "ASSERT_EQUAL",
    "selector":
"//ShoppingTable[3]/Label[2]",
    "field": "text",
    "value": "AQUA AIR PET 600ML",
    "delay": 4
  }
]
```

Kode Program 2 merupakan salah satu contoh skenario ketika menambahkan suatu barang dengan memasukkan PLU barang pada kolom yang

tersedia. Skenario tersebut juga merupakan hasil konversi dari skenario pengujian yang ada pada Tabel 1 tepatnya pada skenario nomor delapan. Dalam satu skenario tersebut terdapat beberapa perintah seperti menekan tombol, mengisi kolom pencarian, dan memvalidasi objek untuk memastikan apakah fitur aplikasi berfungsi dengan benar. Untuk menjalankan setiap perintah terdapat waktu penundaan untuk memastikan bahwa perintah sebelumnya sudah selesai dijalankan.

Kode Program 3. Fungsi untuk Memvalidasi Objek

```
def assert_equal(self, object_type, value,
field='text'):
    self.cli.select_and_store('it',
object_type)
    result =
self.cli.evaluate(f'it.{field}=="{value}"')
    print(result)
    if not result:
        raise Exception('Assertion Error -
Value Not Match')

self.actions.append(TeleniumCommand.create_
assert_equal(object_type, field, value))
```

Kode Program 3 merupakan sebuah fungsi yang digunakan untuk memeriksa kesesuaian antara hasil yang diharapkan dengan hasil aktual yang didapat. Sistem akan memeriksa apakah objek yang diharapkan muncul di tampilan aplikasi ketika pengujian dijalankan atau malah sebaliknya. Jika hasil yang diharapkan tidak sesuai dengan hasil aktual maka eksekusi pengujian akan dihentikan dan pada keluaran konsol muncul kalimat "*Assertion Error - Value Not Match*".

Kode Program 4. Fungsi untuk Membuat Tangkapan Layar

```
def screenshot(self):
    splited = self.filename.split("/")
    filename = splited[len(splited)-1]
    sc_file = os.getenv("WORKSPACE") + '\\'
+ filename + '.png'
    self.cli.screenshot(sc_file)
```

Kode Program 4 merupakan sebuah fungsi yang digunakan untuk mengambil tangkapan layar pada tampilan aplikasi POS. Sistem akan menjalankan fungsi ini apabila ditemukan kesalahan ketika menjalankan pengujian. Tangkapan layar tersebut akan disimpan di direktori WORKSPACE dan berekstensi .png. Selain itu tangkapan layar juga akan secara otomatis disimpan dengan nama yang sama dengan nama skenario yang sedang dijalankan pada saat itu. Hal tersebut bertujuan agar lebih mudah dalam menemukan di mana letak kesalahan pada aplikasi POS.

Kode Program 5. Berkas Runner.py

```
path = "scenario/full scenario/"
files = os.listdir(path)
login = []
for f in files:
    login.append(Runner(f"{path}{f}"))
try:
    for i in login:
        i.run()
        print("--------------------")
        print( i.filename.split("/")[-1])
        print("BERHASIL DIJALANKAN")

except Exception as exc:
    print(traceback.format_exc())
    raise exc
```

Kode Program 5 merupakan potongan kode berkas runner.py. Berkas ini yang nantinya akan dijalankan untuk memulai sebuah pengujian otomatis. Seperti yang dapat dilihat pada kode program tersebut skenario yang ada pada folder dimasukkan ke dalam *list* dengan memanfaatkan perintah *append* kemudian sistem akan melakukan perulangan sesuai dengan panjang *list* tersebut. Setelah berhasil menjalankan satu skenario maka sistem akan menampilkan bahwa skenario tersebut berhasil dijalankan begitu juga seterusnya.

4.4 Environment Setup

Dalam melakukan pengujian otomatis terdapat beberapa kebutuhan yang harus dipenuhi. Kebutuhan tersebut di antaranya yaitu melakukan proses instalasi Python dan Telenium untuk merancang skenario pengujian berdasarkan fungsionalitas dari aplikasi yang diuji. Dalam merancang sistem otomatisasi pengujian digunakan IDE Pycharm.

Sementara itu, untuk menjalankan serta menerima laporan hasil pengujian perlu dilakukan proses instalasi Jenkins. Karena Jenkins berjalan pada lingkungan Java, maka Java juga harus terpasang pada komputer yang akan digunakan untuk melakukan pengujian otomatis. Adapun spesifikasi perangkat yang digunakan dapat dilihat pada Tabel 2.

Tabel 2. Spesifikasi Perangkat

Nama Perangkat	Laptop Acer Swift SF313-51
Sistem Operasi	Windows 10
Tipe Sistem	64-bit *operating system*, x64-*based processor*
Processor	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
RAM	8,00 GB

4.5 Test Execution

Tahap awal dalam menjalankan pengujian otomatis yaitu dengan membuat proyek Jenkins Pipeline kemudian melakukan beberapa konfigurasi di dalamnya.

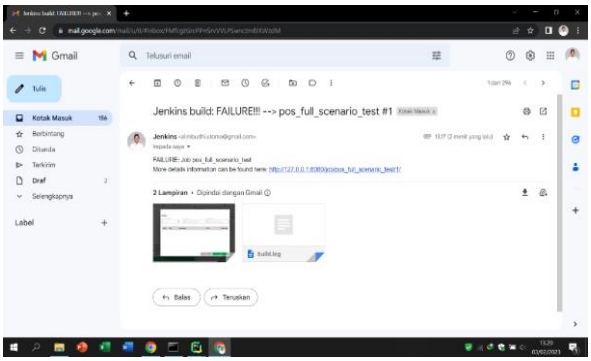
Kode Program 6. Konfigurasi Notifikasi Email

```
failure {
                        emailext to:
"alim.abcde@gmail.com",
                        subject: "Jenkins
build: ${currentBuild.currentResult}!!! -->
${env.JOB_NAME} #${env.BUILD_NUMBER}",
                        body:
"${currentBuild.currentResult}: Job
${env.JOB_NAME}\nMore details information
can be found here: ${env.BUILD_URL}",
                        attachLog: true,
                        attachmentsPattern:
"*.png"
                    }
```

Kode Program 6 merupakan potongan kode konfigurasi Jenkins Pipeline yang berisi sebuah perintah di mana ketika pengujian dilakukan dan menemui kesalahan maka Jenkins akan mengirimkan pesan ke email. Di dalam pesan email tersebut melampirkan dua hal yaitu log proses pengujian dan hasil tangkapan layar di mana kesalahan ditemukan.

Gambar 7. Hasil Pengujian Otomatis

Gambar 7 merupakan hasil dari status pengujian aplikasi POS secara otomatis. Berdasarkan gambar tersebut menunjukkan bahwa pengujian aplikasi POS yang telah dilakukan mendeteksi kesalahan pada *Logout – POS Application stage* di mana di dalam tahap tersebut terdapat sebuah skenario yaitu skenario ketika keluar aplikasi. Hal tersebut ditunjukkan dengan tahapan yang berwarna merah. Dalam Jenkins Pipeline warna hijau menandakan pengujian berhasil dijalankan, sementara itu warna merah menandakan ditemukan kesalahan. Tidak hanya itu, pada bagian ini juga dapat dilihat waktu yang dibutuhkan untuk menjalankan setiap tahapan pengujian.



Gambar 8. Notifikasi Email Hasil Pengujian

Gambar 8 merupakan sebuah pesan email yang dikirimkan secara otomatis kepada alamat email yang sudah dimasukkan ketika mengatur konfigurasi pada tahap sebelumnya. Dengan begitu, untuk mengetahui status pengujian yang dilakukan dapat dilihat dari subjek email yang masuk. Di dalam email tersebut terdapat dua lampiran yaitu tangkapan layar aplikasi dan log ketika pengujian dijalankan. Selain itu untuk melihat informasi yang lebih lengkap dari status pengujian yang dilakukan di badan email terdapat sebuah tautan yang apabila ditekan akan secara otomatis menuju ke halaman Jenkins.

4.6 Test Cycle Closure

Setelah melakukan serangkaian pengujian mulai dari tahap *requirement analysis* hingga *test execution*, diperoleh kesimpulan berdasarkan tahap *test execution* dari 27 skenario yang mewakili empat fitur utama aplikasi POS terdapat satu fitur yang tidak memenuhi atau tidak sesuai dengan fungsionalitasnya. Hal tersebut dapat dilihat pada Tabel 3.

Tabel 3. Rangkuman Hasil Pengujian

No	Status Pengujian	Waktu (detik)
1	Valid	156
2	Valid	
3	Valid	
4	Valid	
5	Valid	469
6	Valid	
7	Valid	
8	Valid	
9	Valid	
10	Valid	
11	Valid	
12	Valid	
13	Valid	
14	Valid	
15	Valid	
16	Valid	
17	Valid	
18	Valid	
19	Valid	
20	Valid	
21	Valid	
22	Valid	
23	Valid	186
24	Valid	
25	Valid	
26	Valid	
27	Gagal	22

Tabel 3 merupakan rangkuman dari hasil pengujian dari seluruh skenario yang sudah disiapkan pada tahap sebelumnya. Dari 27 skenario terdapat satu skenario yang gagal dijalankan. Kesalahan dalam sistem ditemukan pada skenario nomor 27 yakni fitur keluar aplikasi. Untuk menjalankan seluruh skenario dibutuhkan waktu selama 833 detik.

Setelah berhasil membuat dan menjalankan sistem otomatisasi pengujian perangkat lunak, tahap terakhir yaitu dilakukan verifikasi tingkat kualitas sistem yang telah dirancang dengan mengacu pada beberapa karakteristik kualitas perangkat lunak standar ISO 9126, yaitu *functionality*, *usability*, *efficiency*, *maintainability*, dan *portability*. Model kualitas ISO 9126 dianggap sebagai model kualitas yang komprehensif dan memiliki analisis yang lebih baik dibandingkan dengan model kualitas lainnya. Selain itu, standar ini mudah untuk diadaptasi dan dapat diterapkan pada berbagai jenis sistem (Banjarnahor et al., 2018).

Penelitian ini menggunakan instrumen berupa kuesioner yang melibatkan 20 responden. Dalam penelitian ini skala pengukuran yang digunakan adalah Skala Likert. Setelah mengumpulkan hasil kuesioner, langkah berikutnya adalah memberikan skor pada setiap jawaban yang diberikan seperti yang dapat dilihat pada Tabel 4 (Onsent & Susetyo, 2022).

Tabel 4. Skor Jawaban Pada Kuesioner

Jawaban	Skor
Sangat Setuju	5
Setuju	4
Netral	3
Tidak Setuju	2
Sangat Tidak Setuju	1

Tabel 4 menggambarkan pemberian skor dari pilihan jawaban yang ada pada kuesioner. Adapun butir pilihan jawaban responden sejumlah lima pilihan jawaban, di antaranya Sangat Setuju, Setuju, Netral, Tidak Setuju, dan Sangat Tidak Setuju. Untuk menilai sejauh mana responden setuju atau tidak setuju terhadap sistem sistem otomatisasi pengujian perangkat lunak yang dibuat, penelitian ini menggunakan skala yang dapat dilihat pada Tabel 5.

Tabel 5. Skala Pengukuran Likert

Indeks *Range*	Kriteria
80% -100%	Sangat Setuju
60% -79,99%	Setuju
40% -59,99%	Kurang Setuju
20% -39,99%	Tidak Setuju
0% -19,99%	Sangat Tidak Setuju

Berikut ini merupakan hasil keseluruhan dari pengujian ISO 9126 seperti yang dapat dilihat pada Tabel 6.

Tabel 6. Rangkuman Analisis ISO 9126

Aspek	Skor Aktual	Skor Ideal	% Skor Aktual	Kriteria
Functionality	424	500	84,80%	Sangat Setuju
Usability	336	400	84,00%	Sangat Setuju
Efficiency	175	200	87,50%	Sangat Setuju
Maintainability	267	300	89,00%	Sangat Setuju
Portability	161	200	80,50%	Sangat Setuju
Total	1363	1600	85,19%	Sangat Setuju

Hasil pengukuran ISO 9126 yang dilakukan menunjukkan bahwa secara keseluruhan kelayakan sistem yang dihasilkan mencapai 85,19% dan dapat dikategorikan "Sangat Setuju" dengan merujuk pada rentang nilai yang tercantum pada Tabel 5. Oleh karena itu, sistem otomatisasi pengujian perangkat lunak ini memenuhi syarat untuk diterapkan pada aplikasi berbasis Kivy di PT XYZ.

5 Kesimpulan

Penerapan pengujian otomatis pada aplikasi POS PT XYZ berhasil dilakukan berdasarkan skenario yang telah disusun. Jika terjadi kesalahan selama pengujian, informasi detail mengenai letak kesalahan akan ditampilkan melalui email yang dikirim oleh Jenkins berupa log pengujian dan tangkapan layar. Penerapan Telenium dalam pengujian aplikasi berbasis Kivy memudahkan penguji dalam melakukan proses pengujian dan mempercepat identifikasi kesalahan dalam sistem sehingga dapat meningkatkan kualitas aplikasi secara efisien. Hasil kuesioner memperoleh persentase sebesar 85,19% dan digolongkan dalam kriteria “Sangat Setuju” terhadap sistem otomatisasi pengujian yang dibuat.

Referensi

Arfan, A., & Hendrik. (2022). Penerapan STLC dalam Pengujian Automation Aplikasi Mobile (Studi kasus : LMS Amikom Center). *AUTOMATA*, *3*(2), 1–6.

Banjarnahor, D., Darwiyanto, E., & Suwawi, D. D. J. (2018). *Analisis Kualitas Sistem Presensi Pada I-Gracias Universitas*. *5*(3), 7428–7440. https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/7068/6962

Bhoyarkar, A., Solanki, A., & Balbudhe, A. (2019). Application Development using Kivy Framework. *Ijarcce*, *8*(2), 53–58. https://doi.org/10.17148/ijarcce.2019.8209

Fahrezi, A., Salam, F. N., Ibrahim, G. M., Rahman, R., & Saifudin, A. (2022). Pengujian Black Box Testing pada Aplikasi Inventori Barang Berbasis Web di PT . AINO Indonesia. *Jurnal Ilmu Komputer Dan Pendidikan*, *1*(1), 1–5.

Jaya, T. S. (2018). Pengujian Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis. *Jurnal Informatika Pengembangan IT (JPIT)*, *3*(2), 45–46. http://www.ejournal.poltektegal.ac.id/index.php/informatika/article/view/647/640

Kosasih, Y., & Cahyono, A. B. (2020). Perancangan Sistem Dalam Pengujian Aplikasi The Point Of Sale (Studi Kasus TPOS PT. JAVASIGNA INTERMEDIA). *Teknik Informatika*, *3*(2), 24–30.

Maspupah, A., & Bakhrun, A. (2021). Perbandingan Kemampuan Regression Testing Tool Pada Regression Test Selection: Starts Dan Ekstazi. *JTT (Jurnal Teknologi Terapan)*, *7*(1), 59–67. https://doi.org/10.31884/jtt.v7i1.319

Mustika, N. R., & Novrina. (2018). Automated Black Box Testing using Selenium Python. *International Journal of Computer Science and Software Engineering (IJCSSE)*, *7*(9), 201–204. www.IJCSSE.org

Mustofa, K., & Fajar, S. P. (2018). Selenium-Based Multithreading Functional Testing. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *12*(1), 63–72. https://doi.org/10.22146/ijccs.28121

Onsent, G. S., & Susetyo, Y. A. (2022). Rancang Bangun Sistem Sinkronisasi Data Menggunakan Google Cloud Pub/Sub Dan Flask Di Pt Xyz. *Jurnal Mnemonic*, *5*(2), 86–92. https://doi.org/10.36040/mnemonic.v5i2.4645

Panjaitan, M. M., & Mantra, I. (2020). Pembangunan Framework Web Automation Testing Menggunakan Serenity Bdd Pada Studi Kasus Aplikasi. *Seminar Nasional Mahasiswa Ilmu Komputer Dan Aplikasinya (SENAMIKA)*, 25–33.

Pratama, R. Y., & Somya, R. (2021). Perancangan Aplikasi Point Of Sales (POS) Berbasis Android (Studi Kasus: Warkop Vape Salatiga). *JATISI (Jurnal Teknik Informatika Dan Sistem Informasi)*, *8*(4), 1923–1938. https://doi.org/10.35957/jatisi.v8i4.1218

Putra, R. A. (2018). Analisa Implementasi Arsitektur Microservoces Berbasis Kontainer Pada Komunitas Pengembang Perangkat Lunak Sumber Terbuka (OpenDayLight DevOps Community). *Jurnal Sistem Infomasi Teknologi Informasi Dan Komputer (Just It) Universitas Bina Nusantara Magister Manajemen Sistem Informasi Jakarta*, *9*(2), 150–162.

Setiawan, F. A., Putra, S. D., & Sahlinal, D. (2019). Pengujian Proyek Website Otomatisasi Dengan Pendekatan Integrasi Antara Selenium Dan Testng. *Repository Polinela*, 1–14. http://repository.polinela.ac.id/518/1/Artikel fix Fransiskus Andika.pdf

Thooriqoh, H. A., Annisa, T. N., & Yuhana, U. L. (2021). Selenium Framework for Web Automation Testing: A Systematic Literature Review. *JUTI: Jurnal Ilmiah Teknologi Informasi*, *19*(2), 65–76.

Utomo, D. W., Kurniawan, D., & Astuti, Y. P. (2018). Teknik Pengujian Perangkat Lunak Dalam Evaluasi Sistem Layanan Mandiri Pemantauan Haji Pada Kementerian Agama Provinsi Jawa Tengah. *Simetris: Jurnal Teknik Mesin, Elektro Dan Ilmu Komputer*, *9*(2), 731–746. https://doi.org/10.24176/simet.v9i2.2289

Virbel, M. (2021). *Telenium*. Pypi. https://pypi.org/project/telenium/

Wicaksono, F. D., & Rani, S. (2022). Rancang Bangun Automation Test Journey (Studi Kasus : Marketplace PT . Tokopedia). *AUTOMATA*, *3*(2), 1–8.

Yutia, S. N., & Satrinia, D. (2021). Automated Functional Testing pada API menggunakan Keyword Driven Framework. *Journal of Informatics and Communications Technology (JICT)*, *1089*, 1–14.