Studi Komparasi Teknik Pengujian *End-to-End* pada *E-payment*: Studi Kasus Travelink (Sistem *Web E-ticketing*)

Azza Ulil Afidah[1], Divi Galih Prasetyo Putri[1,*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
azzaulil99@mail.ugm.ac.id
[*]Korespondensi: divi.galih@ugm.ac.id;

Abstract – Financial technology is an important agenda in the financial services industry and the economy in the future. One of the technological innovations in the banking, finance, and trade sectors is electronic payment or commonly called e-payment. The e-payment system is very influential in the ticket purchase transaction process on the web, so it is necessary to ensure the success of transactions in a software through end-to-end testing. Regarding the system development period, currently, software development companies want fast and effective testing times, therefore appropriate test execution methods are needed for end-to-end testing of the system. This study uses a case study of the Travelink e-payment system. The uniqueness of this system is that there are payment methods that may require special testing methods. This study was carried out with a comparison between three testing methods, including manual, automated and hybrid testing (a combination of manual and automated testing). Comparison of the three testing methods is carried out using test metrics, requirement coverage and test execution time. The comparison results with the two of test metrics show that the automated testing method covers 100% of the requirements and is the fastest to use for end-to-end testing of e-payment Credit Card and Virtual Account. The hybrid testing method provides 100% test coverage and is the fastest to use in QRIS e-payment end-to-end testing. It is hoped that this can be a reference for determining the test method that will be used in the e-payment system.

Keywords – End-to-end testing, E-payment, Robot Framework, Automated, Hybrid

Intisari – Teknologi finansial menjadi agenda penting dalam industri jasa layanan keuangan dan perekonomian di masa depan. Salah satu inovasi teknologi di bidang perbankan, keuangan dan perdagangan adalah *electronic payment* atau biasa disebut *epayment*. Sistem *e-payment* sangat berpengaruh dalam proses transaksi pembelian tiket pada *web*, sehingga perlu untuk memastikan keberhasilan transaksi dalam suatu perangkat lunak melalui pengujian *end-to-end*. Berkaitan dengan masa pengembangan sistem, perusahaan pengembang perangkat lunak saat ini menginginkan proses pengujian yang cepat dan efektif, oleh karena itu metode eksekusi pengujian yang tepat diperlukan untuk pengujian *end-to-end* pada sistem. Penelitian ini menggunakan studi kasus sistem *e-payment* Travelink. Keunikan dari sistem ini adalah adanya metode pembayaran yang memerlukan metode pengujian khusus. Penelitian ini dilakukan dengan perbandingan antara ketiga metode pengujian, di antaranya pengujian secara manual, otomatis, dan *hybrid* (gabungan manual dan otomatis). Komparasi ketiga metode pengujian dilakukan dengan menggunakan matriks pengujian *requirement coverage* dan waktu eksekusi pengujian. Hasil perbandingan dengan kedua matriks pengujian menunjukkan bahwa metode pengujian otomatis mencakup 100% *requirement* dan yang paling cepat digunakan untuk pengujian *end-to-end epayment Credit Card* dan *Virtual Account*. Metode pengujian *hybrid* memberikan cakupan pengujian 100% dan paling cepat digunakan pada pengujian *end-to-end e-payment QRIS*. Hal ini diharapkan dapat menjadi referensi untuk menentukan metode pengujian yang akan digunakan pada sistem *e-payment*.

Kata kunci – Pengujian *end-to-end, E-payment,* Robot Framework, Otomatis, *Hybrid*

I. PENDAHULUAN

Teknologi finansial menjadi agenda penting pada industri jasa layanan keuangan dan perekonomian di masa depan. Salah satu inovasi teknologi di bidang perbankan, keuangan dan perdagangan adalah pembayaran elektronik atau *Electronic Payment* [1]. Sistem pembayaran elektronik atau *epayment* digambarkan sebagai proses pembiayaan yang dilakukan secara *online* dalam suatu *e-commerce* [2]. Manfaat sistem *e-payment* bagi individu adalah pelanggan dapat melakukan transaksi kapanpun dan dimanapun. Keunggulan dari segi biaya operasional yang gratis untuk semua bisnis, klien, dan pemasok serta manfaatnya dalam menurunkan aktivitas kriminal dan kasus penipuan, merupakan manfaat yang besar untuk bisnis [3].

Pengaruh besar *web* pada kehidupan keseharian adalah individu merasa terbiasa dengan transaksi *online* di *ecommerce* untuk penjualan dan pembelian produk dan usaha [4]. Berdasarkan beberapa aspek di atas, sistem *e-payment* terbukti sangat berpengaruh dalam proses transaksi pembelian secara *online*, sehingga perlu untuk memastikan keberhasilan proses transaksi dalam suatu *software* melalui proses pengujian.

Teknik pengujian *end-to-end* merupakan pengujian yang dilakukan untuk melakukan verifikasi fungsionalitas terhadap keseluruhan alur sistem dan sering digunakan perusahaan dalam pengembangan aplikasi *web* [5], [6], [7]. Pengujian *end-to-end* adalah teknik pengujian yang memiliki cakupan besar dan membutuhkan waktu yang lama dalam pelaksanaannya. Saat ini perusahaan pengembang perangkat lunak menginginkan masa pengembangan sistem yang lebih singkat [8], sehingga upaya pemilihan teknik eksekusi pengujian *end-to-end* (manual atau automasi) diperlukan agar proses pengujian menjadi lebih singkat.

Adapun sistem *e-payment* yang memiliki berbagai macam metode pembayaran memerlukan metode pengujian yang bervariasi juga. Contohnya *e-payment* pada *e-wallet* seperti QRIS (*Quick Response Code Indonesian Standard*) yang mana dalam proses pengujiannya membutuhkan *device* terpisah untuk melakukan *scan* dan membayar [9], [10], [11], sehingga penelitian ini dilakukan untuk membuktikan bahwa adanya teknik pengujian *end-to-end* secara *hybrid* (gabungan manual dan otomatis) menjadi lebih cepat dan efisien. Metode eksekusi yang memungkinkan untuk pengujian *end-to-end* adalah pengujian manual, otomatis dan gabungan keduanya (*hybrid*). Penelitian ini dilakukan untuk menentukan metode eksekusi yang paling sesuai digunakan untuk pengujian *end-to-end* pada studi kasus sistem *e-payment* Travelink.

II. DASAR TEORI

Beberapa penelitian terdahulu telah membahas tentang perbandingan pengujian manual dan otomatis dengan menggunakan beberapa *framework* atau *automation tools* [12]. Adapun penelitian terdahulu yang membahas terkait *requirement coverage* sebagai matriks pengujian dan juga penelitian yang berkaitan dengan pengujian sistem *e-payment*. Berikut penelitian-penelitian terdahulu dengan topik terkait dengan penelitian ini.

A. Perbandingan Pengujian Manual dan Otomatis

Penelitian terkait perbandingan antara pengujian secara manual dan otomatis pernah dilakukan oleh [8] yang mengembangkan automasi pada pengujian *end-to-end web-based application*. Penelitian ini menggunakan Protractor sebagai *automation tool*. Penelitian ini menyimpulkan bahwa pengujian *end-to-end* secara otomatis membutuhkan pengerjaan yang lama pada iterasi pertama. Namun pada iterasi kedua, lama pengerjaan menjadi lebih pendek karena hanya menjalankan kembali skrip pengujian yang sudah dibuat pada iterasi pertama. Oleh karena itu, jumlah iterasi pelaksanaan pengujian *end-to-end* sangat berpengaruh dalam pemilihan teknik pengujian yang digunakan.

B. Matriks *Requirement Coverage*

Penelitian terdahulu yang menggunakan *requirement coverage* sebagai matriks pengujian pernah dilakukan oleh [13]. Penelitian ini menyediakan analisis perbandingan antara MBT (*Model-based testing*), CT (*Combinatorial Testing*) dan pengujian manual dalam hal MC/DC (*Modified Condition/Decision Coverage*) dan *Requirement Coverage*. Evaluasi efisiensi teknik pengujian dilakukan untuk menentukan perbedaan *test suite* yang dihasilkan oleh masing-masing teknik pengujian. Efisiensi diukur dengan penggunaan waktu eksekusi pengujian dan *requirement coverage*. Penelitian ini menyimpulkan bahwa CT adalah teknik pengujian yang paling efisien jika dibandingkan dengan MBT dan pengujian manual di mana CT ditempuh dalam waktu yang singkat dan mencapai 100% *requirement coverage*.

C. Pengujian Sistem *E-payment*

Penelitian yang dilakukan oleh [14] membahas tentang pengembangan dan pengujian pada *website e-commerce* dengan sistem pembayaran QR *code*. Pengujiannya dilakukan dengan pengujian fungsionalitas dan *system testing*. Adapun pengujian *front-end* dilakukan dengan memesan produk yang kehabisan stok dan verifikasi proses pembayaran berfungsi dan dapat diandalkan tanpa eror atau notifikasi yang menandakan transaksi gagal. Hasil pengujian fungsionalitas adalah pengguna berhasil melakukan pemesanan produk dan menyelesaikan pembayaran dengan sukses.

Berdasarkan dari penelitian-penelitian terdahulu yang serupa dengan penelitian ini, dapat diketahui bahwa perbandingan pengujian terfokus pada metode pengujian manual dan otomatis. Perbedaan penelitian pada studi kasus sistem *e-payment* Travelink ini adalah selain membandingkan metode pengujian manual dan otomatis, penelitian ini juga membandingkan metode pengujian gabungan dari manual dan otomatis (*hybrid*). Perbandingan metode pengujian pada penelitian ini menggunakan dua matriks yaitu *requirement coverage* dan waktu eksekusi pengujian.

III. METODOLOGI

A. Bahan

Data atau informasi utama yang digunakan pada penelitian ini berasal dari sistem *web e-ticketing* Travelink sebagai *System Under Test* (SUT). Sistem Travelink ini diambil dari perusahaan *payment gateway* yaitu PT. Aino Indonesia sebagai studi kasus untuk penelitian ini. Adapun pada studi kasus *website* Travelink yang memiliki sistem *e-payment* terintegrasi dan sistem *e-payment* tersebut yang akan digunakan sebagai bahan utama pada penelitian ini.

Adapun beberapa metode *e-payment* yang tersedia pada Travelink dan digunakan dalam pengujian ini adalah *e-payment credit / debit card, virtual account*, dan QRIS (*Quick Response Code Indonesian Standard*). Ketiga metode tersebut memiliki proses pembayaran yang berbeda, sehingga dipilih sebagai bahan perbandingan teknik pengujian pada penelitian ini.

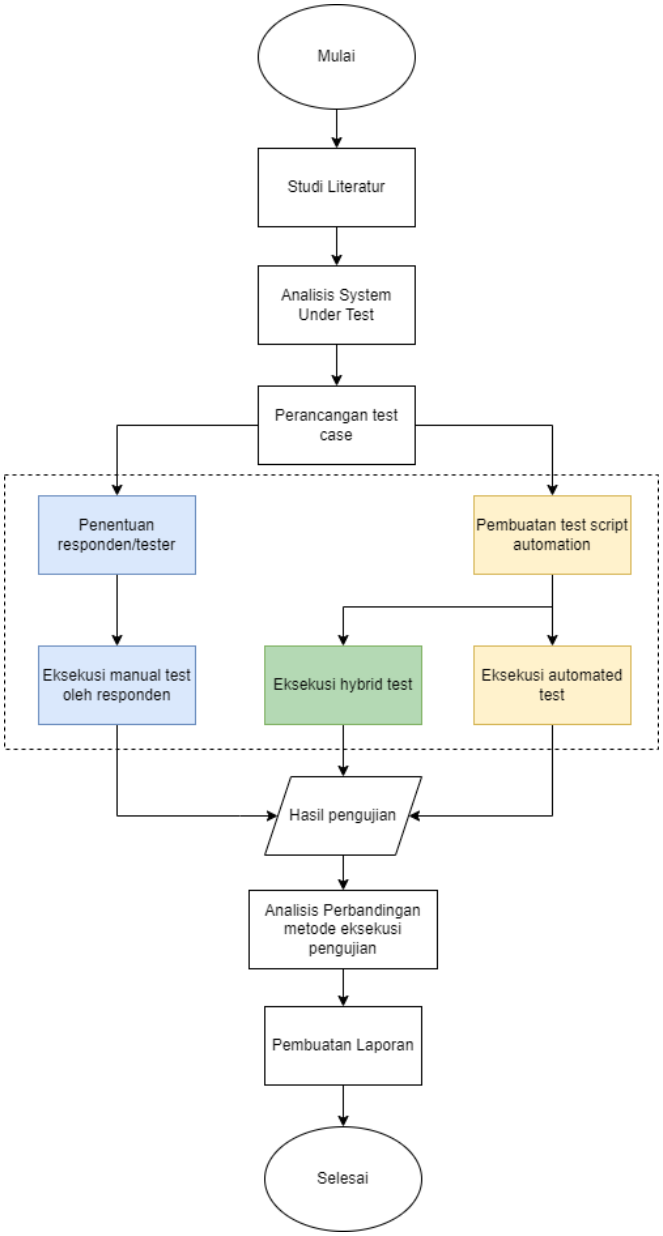
B. Peralatan

Penelitian dilakukan dengan menggunakan perangkat keras dan perangkat lunak. Spesifikasi dan peralatan yang digunakan dalam penelitian ini ditampilkan pada rincian berikut.

1. Notebook Intel® CoreTM i5-7200U, CPU @2.50GHz, RAM 8 GB dengan sistem operasi Windows 10 Pro
2. Chrome browser v118.0 dan chromedriver v118.0
3. Python v3.11.2 dan pip v22.3.1 untuk instalasi Robot Framework
4. Robot Framework v6.0.2
5. Visual Studio Code sebagai *text editor*
6. Mobile app yang digunakan untuk pembayaran *e-payment* QRIS

C. Tahapan Penelitian

Penelitian ini terdiri dari beberapa tahapan yang dapat dilihat pada Gambar 1.

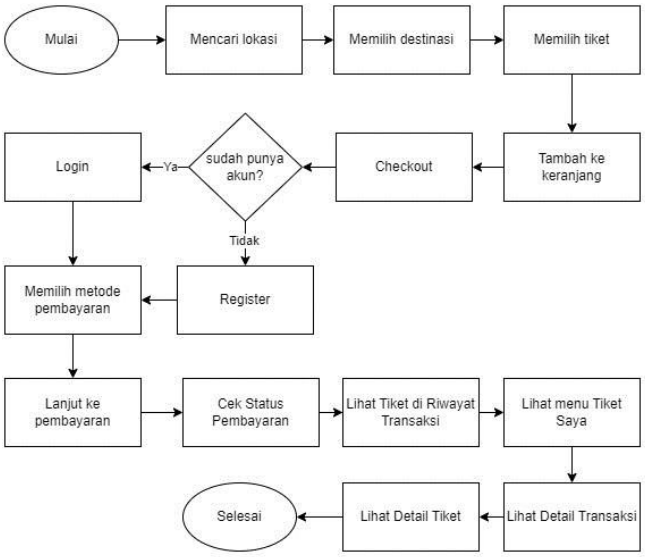


Gambar 1. Alur penelitian

Penelitian dapat dilihat pada Gambar 1 diawali dengan proses analisis *system under test* untuk proses memahami sistem yang diuji terkait alur kerja dan pengenalan fitur-fitur yang akan diuji. Kemudian dilakukan proses studi literatur terhadap penelitian-penelitian terdahulu yang membahas tentang bagaimana membuat pengujian otomatis dan perbandingannya dengan pengujian manual. Proses selanjutnya adalah perancangan *test case* atau skenario pengujian yang akan digunakan sebagai panduan dalam melakukan pengujian serta untuk memenuhi persyaratan kebutuhan dari sistem yang diuji. Setelah dilakukan perancangan *test case*, dilanjutkan proses eksekusi pengujian yang terbagi menjadi tiga metode, yaitu manual, otomatis, dan *hybrid*. Setelah didapatkan *output data* untuk perbandingan dari masing-masing metode pengujian, proses selanjutnya yaitu analisis perbandingan metode eksekusi pengujian dengan menggunakan waktu eksekusi dan *requirement coverage*. Proses yang terakhir adalah pembuatan laporan untuk menyajikan penelitian ini sesuai sistematika penulisan yang telah ditentukan. Alur penelitian terdapat tiga proses yang dibedakan dengan warna biru, hijau dan kuning. Proses dengan warna biru merupakan proses pengujian secara manual. Sedangkan proses dengan warna kuning adalah proses pengujian otomatis dan warna hijau untuk proses pengujian secara *hybrid*.

D. Analisis Sistem *Under Test*

Travelink merupakan sistem berbasis web yang digunakan untuk pembelian tiket masuk wisata secara *online* dan memiliki sistem pembayaran terintegrasi dengan aplikasi. Alur utama dalam sistem ini adalah pembeli dapat memilih tiket dan menambahkan ke keranjang, kemudian *checkout* keranjang dan melakukan pembayaran dengan metode pembayaran yang telah tersedia. Lalu sistem akan *generate*/membuat tiket *online* yang dapat digunakan untuk validasi tiket untuk masuk ke dalam destinasi. Gambaran alur utama pada sistem Travelink dapat dilihat pada Gambar 2.



Gambar 2. *User flow* Travelink

E. Perancangan Pengujian

1. Perancangan *Test Case*

Tahap perancangan *test case* adalah proses perancangan langkah pengujian yang akan digunakan sebagai dasar dari penyusunan skrip pengujian. Perancangan *test case* perlu dilakukan agar langkah-langkah yang diterapkan pada *framework* yang dipakai sudah benar, tidak cacat, dan dapat

diimplementasikan. Tahapan ini berguna untuk memastikan cakupan pengujian lengkap dan sesuai dengan melakukan verifikasi dan validasi terhadap kriteria dari fitur yang akan diuji. Tahapan perancangan *test cast* dapat ditunjukkan pada Tabel 1.

Tabel 1. *Test case*

Test ID	*Test Type*	*Pre-condition*	*Test Case*	*Test Step*	*Expected Result*
REG_01	Positif	User berada di halaman Registrasi	Registrasi dengan data valid	1. Isi *form register* dengan data valid 2. Klik tombol Lanjutkan	*Register* berhasil
REG_02	Negatif	User berada di halaman Registrasi	Registrasi dengan data kosong	1. Klik kolom email 2. Biarkan kosong 3. Klik tombol Lanjutkan	Menampilkan pesan *error* pada *text field* bahwa kolom ini harus diisi
REG_03	Negatif	User berada di halaman Registrasi	Registrasi dengan email terdaftar	1. Isi *form register* dengan email yang sudah didaftarkan 2. Klik tombol Lanjutkan	*Register* gagal dan muncul pesan *error* pada *text field* email bahwa email sudah digunakan
REG_04	Negatif	User berada di halaman Registrasi	Format email tidak valid	1. Isi kolom email dengan format tidak valid	Menampilkan pesan *error* pada *text field* bahwa email ini tidak valid
REG_05	Negatif	User berada di halaman Registrasi	Nomor hp diisi selain angka	1. Isi kolom nomor hp dengan huruf/simbol	Menampilkan pesan *error* pada *text field* bahwa kolom ini hanya bisa diisi dengan angka

Perancangan *test case* mempertimbangkan data *input* dan *output* dari setiap fitur yang menyebabkan *test case* dibagi menjadi dua jenis yaitu *test case* positif dan negatif. *Test case* positif dirancang untuk memeriksa apakah aplikasi melakukan apa yang diharapkan dengan memberikan data input yang valid. Sedangkan *test case* negatif dibuat untuk memeriksa apakah aplikasi berperilaku seperti yang diharapkan dengan *input* negatif dan memvalidasi aplikasi terhadap data yang tidak valid. *Test case* negatif bertujuan agar aplikasi tidak melakukan apa yang seharusnya tidak dilakukan dan untuk memeriksa apakah kesalahan ditampilkan pada kondisi yang seharusnya. Hal ini diperlukan untuk teknik pengujian *end-to-end* dalam menguji fungsionalitas sistem secara keseluruhan. Salah satu *test case* yang telah dibuat dapat dilihat pada Tabel 1.

2. Pengaturan Robot Framework

Tahap ini mencakup aktivitas persiapan *environment* yang akan digunakan untuk pengujian, pengidentifikasian elemen halaman sistem, pembuatan *keywords* serta pembenahan pada setiap *error* yang mungkin muncul ketika proses pembuatan. *Environment* yang digunakan mencakup *framework* pengujian, *library–library* yang terkait dengan *framework* pengujian yang dipilih, serta perangkat lunak lain seperti *extension browser* untuk pengambilan elemen halaman sistem.

Instalasi Robot framework dapat menggunakan pip, program untuk manajemen paket di Python. Instalasi Python pada komputer biasanya menginstal pip secara otomatis dan dokumentasi lebih lengkapnya terdapat pada situs resmi Robot framework. Setelah semua terpasang, maka selanjutnya adalah pembuatan *file* baru dengan ekstensi .robot dalam *folder project* Robot. Selanjutnya proses *coding* menggunakan Selenium Library dan Selenium *webdriver* untuk menjalankan langkah pengujian pada elemen halaman.

Perihal pengaturan *Robot Framework* dijelaskan pada beberapa poin penting di bawah ini:

a. *File Directory*

Project Robot framework disimpan dalam satu *folder* yang berisi file-file .robot dengan berbagai jenis skrip pengujian di dalamnya. Semua *folder* dibuat dengan nama *folder* sesuai dengan yang diinginkan.

b. Pengumpulan *Variables*

Variables pada Robot framework ini didefinisikan dengan bagian *** Variables *** yang tersedia di semua *test cases* dan *keywords* dalam *file* yang sama. Pengumpulan *variable* yang menjadi satu *file* berguna untuk penulisan yang berulang dan mudah dipahami.

c. Membuat *Keywords*

Penggunaan *keywords* yang berasal dari *Selenium Library* untuk membuat skrip detail Langkah pengujian dalam setiap *test case*. *Robot framework* memiliki *default keywords* berupa variasi interaksi pengguna pada suatu *website* seperti halnya *click, hover, hold, drag,* dan *scroll*. Pada proyek ini, *keywords* dikumpulkan menjadi beberapa *file* .resource sesuai dengan fitur yang diuji agar pembacaan dokumentasi yang lebih mudah.

d. Menjalankan Skrip Pengujian

Pembuatan skrip pengujian perlu adanya validasi untuk memastikan bahwa interaksi yang dilakukan berhasil. Pengecekan elemen yang dihasilkan oleh *robot framework*

apakah sudah sesuai dengan yang seharusnya muncul atau tidak. Hal ini dilakukan dengan menjalankan skrip pengujian yang telah dibuat. *Tools* yang digunakan untuk menjalankan skrip pengujian sama seperti yang digunakan untuk pembuatan skrip pengujian yaitu *Visual Studio Code* dengan fitur *Terminal*. Cara menjalankan atau *running test script* dapat dilihat pada Gambar 3.

Gambar 3. *Command* untuk menjalankan skrip

e. Mengakses *Test Report* dan *Test Log*

Test report dihasilkan dalam bentuk *file HTML* dan berisi tentang laporan pengujian yang telah dilakukan pada setiap langkah pengujian, skenario dan keseluruhan pengujian. Informasi pada *test report* terdapat data seperti *outcome status* dan lama eksekusi pengujian. Selain *test report*, terdapat *test log* yang berisi laporan pengujian secara detail seperti gambar tangkapan layar (*screenshot)* apabila terdapat skrip pengujian yang gagal.

3. Penentuan Responden (*Manual Tester*)

Beberapa responden digunakan sebagai penguji dalam pengujian manual untuk mendapatkan lama eksekusi yang dibutuhkan secara *general*. Responden diambil menggunakan teknik *non probability Convenience sampling*. Teknik tersebut menggunakan populasi target yang memenuhi kriteria tertentu seperti kedekatan geografis, kemudahan untuk komunikasi, ketersediaan pada waktu tertentu, atau kesediaan untuk berpartisipasi dalam penelitian. Hal ini yang menjadikan beberapa responden yang terlibat hanya anggota tim Travelink pada PT. Aino Indonesia dengan tujuan menjaga hal yang *confidential* pada *project* ini.

Responden berjumlah tiga orang dan masing-masing menguji sistem secara manual dengan tiga *end-to-end test case* yang sama yaitu *test case e-payment Credit Card, Virtual Account* dan QRIS. Responden dijelaskan beberapa hal sebelum dilakukan eksekusi pengujian, terkait tata cara pelaksanaan pengujian *end-to-end*. Tujuannya adalah untuk menyetarakan kemampuan responden sebagai penguji dan dapat memahami cara melakukan pengujian *end-to-end* secara manual.

IV. HASIL DAN PEMBAHASAN

A. Implementasi *Test Case* Menggunakan Robot Framework

Test case yang telah dirancang kemudian diimplementasikan pada *Robot Framework* dengan penulisan skrip pengujian dengan menggunakan SeleniumLibrary. Implementasi *test case* ini terbagi menjadi eksekusi pengujian *end-to-end* secara otomatis dan *hybrid*, sehingga akan ada 2 *test suite* yaitu *automated testing* dan *hybrid testing*. Salah satu skrip pengujian pada *automated testing* yaitu *test case* pembayaran sukses dengan metode *Virtual Account* (VA) dapat dilihat pada Gambar 4.

```
Pay with VA
    Copy VA number
    Open ATM Simulator in new tab
    Select bank code    013    # VA Permata
    Input VA number
    Execute VA payment
    Take a screenshot    success_VApayment
    Switch Window    title=payment_front

Copy VA number
    Wait Until Element Is Visible
    ...    xpath=//*[@id="card-summary-wrap"]/span[2]/span/div[2]/img    timeout=10s
    Click Element    xpath=//*[@id="card-summary-wrap"]/span[2]/span/div[2]/img
    ...    # copy VA number

Open ATM Simulator in new tab
    Execute Javascript    window.open('')    # open new tab
    Get Window Titles    # get all tab titles
    Switch Window    title=undefined    # switch to new tab
    Go To    ${atm_simulator_url}
    Wait Until Page Contains    Virtual Account Simulator

Select bank code    [Arguments]    ${bank_code}
    Click Element    bankcode
    Click Element    xpath=//option[@value='${bank_code}']

Input VA number
    Click Element    //input[@id='rekening']
    Press Keys    //input[@id='rekening']    CTRL+V    # paste VA number

Execute VA payment
    Click Button    inq-btn
    Wait Until Element Is Visible    //button[@id='pay']
    Click Button    //button[@id='pay']
    Wait Until Element Is Visible    //span[@id='pay-status']
```

Gambar 4. Skrip pengujian pembayaran dengan *virtual account*

Test case ini dilakukan saat berada di halaman pembayaran. Langkah pengujian diawali dengan menyalin nomor *virtual account*. Penyalinan *virtual account* dilakukan dengan klik tombol *copy*. Selanjutnya, ATM *Simulator* dibuka pada *tab* baru dan diverifikasi elemen halamannya. Kode bank dipilih sesuai metode pembayaran yang dipilih sebelumnya dan tempel *virtual account* pada *text field* nomor rekening. Tombol *inquire* diklik dan tunggu hingga tombol Bayar muncul. Saat tombol Bayar muncul, kemudian diklik dan ditunggu status pembayaran muncul. Terakhir, sistem diarahkan untuk kembali ke halaman pembayaran. Skrip pengujian pembayaran dengan QRIS dapat dilihat pada Gambar 5.

```
Pay with QRIS
    Wait Until Element Is Visible
    ...    xpath=//span[contains(text(),'Status Pembayaran')]    timeout=10s
    Page Should Contain    Kode QR
    Page Should Contain Element    //*[@id="card-summary-wrap"]/div[2]/p/canvas
    ...    # qris image
```

Gambar 5. Skrip pengujian pembayaran dengan QRIS

Skrip pengujian pada Gambar 5 ini hanya dilakukan untuk verifikasi halaman pembayaran. Hal ini dikarenakan pembayaran QRIS tidak dapat diuji secara otomatis, sehingga pada skrip pengujian ini akan dilakukan pengujian secara manual yang termasuk dalam proses *hybrid testing*. Langkah pengujian pada *test case* ini diawali dengan verifikasi *element text* Status Pembayaran yang mengacu pada halaman pembayaran QRIS sudah muncul atau belum. Halaman pembayaran tersebut juga diverifikasi apakah terdapat Kode QR atau tidak. Hasil pengujian dari beberapa *test case* ditunjukkan pada Tabel 2.

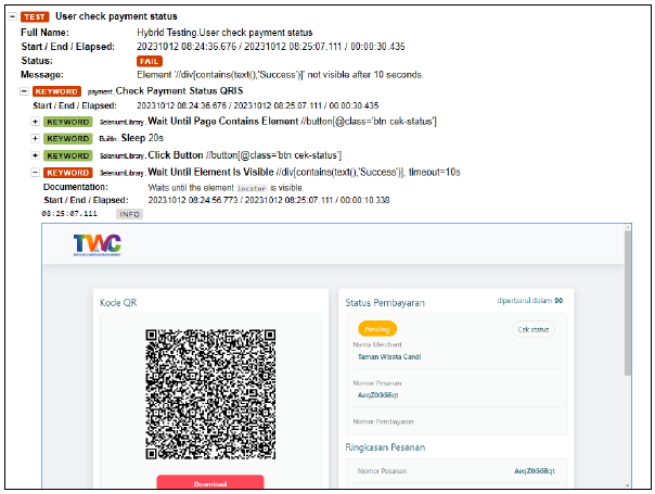
Tabel 2. Hasil Pengujian

Test case	*Manual*		*Automated*		*Hybrid*	
	Status	*Defects*	*Status*	*Defects*	*Status*	*Defects*
Pembayaran dengan QRIS	*PASS*	-	*CAN'T BE TESTED*	*Scan QRIS can't be automated*	*PASS*	-
Periksa status pembayaran setelah pembayaran sukses	*PASS*	-	*FAIL*	*The status does not change to success because there is no QRIS payment*	*PASS*	-
Lihat Tiket yang berhasil dibeli	*PASS*	-	*FAIL*	*No tickets were purchased*	*PASS*	-

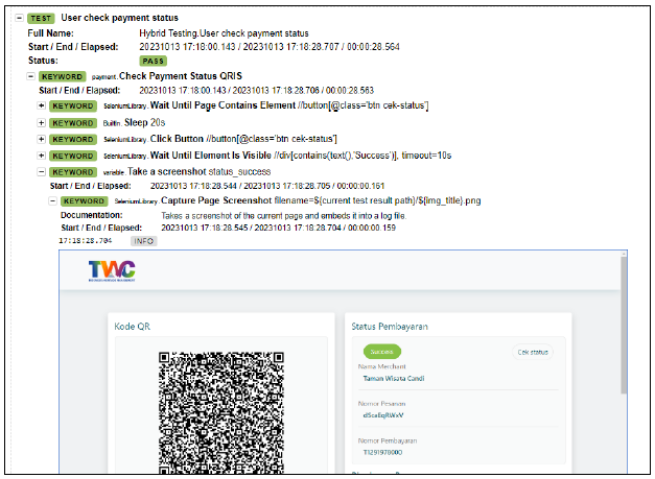
B. Hasil Pengujian

Pengujian *end-to-end* pada sistem *e-payment* Travelink dilakukan pada 57 *test cases* dengan 3 metode yaitu pengujian manual, pengujian otomatis dan pengujian *hybrid*. Tabel 2 terdiri dari data *Test ID* yang diuji, kolom Status dan *Defects* pada setiap metode eksekusi. Kolom Status memiliki 3 jenis status untuk menandai dari hasil pengujian. Status *PASS* dengan warna hijau untuk *test case* yang lolos uji, sedangkan status *FAIL* yang berwarna merah untuk *test case* yang gagal uji. Adapun status *CAN'T BE TESTED* dengan warna abu-abu yang artinya *test case* tidak dapat dieksekusi dan gagal. Kolom *Defects* digunakan untuk menuliskan keterangan penyebab *test case* tidak lolos uji.

Tabel 2 merupakan hasil pengujian dari beberapa *test case* yang menunjukkan hasil yang signifikan. Berdasarkan tabel tersebut, dapat dilihat bahwa pada pengujian secara manual dan *hybrid*, semua *test case* berhasil dieksekusi dan lolos uji. Hasil pengujian secara *automated* terdapat 54 *test cases PASS* atau lolos uji, dua *test cases FAIL* atau gagal uji dan satu *test case* tidak dapat dieksekusi. Satu *test case* yang tidak dapat dieksekusi terjadi pada *test case* dengan id PYM_04 yaitu pengujian pembayaran QRIS. Pengujian pada pembayaran QRIS membutuhkan sistem terpisah yaitu *mobile app* untuk melakukan pembayaran, sehingga tidak bisa diuji secara otomatis dan berujung gagal uji. Kemudian *test case* dengan id PYM_05 yang menguji status pembayaran setelah pembayaran, tidak lolos uji karena saling berkaitan dengan *test id* PYM_04. Status pembayaran tidak berubah menjadi sukses karena tidak ada proses pembayaran. *Test id* PYM_06 juga berkaitan dengan *test case* dengan id PYM_05, pengujian terhadap tiket yang berhasil dibeli, gagal uji karena tidak ada tiket yang dibeli. Hasil pengujian *test case* yang gagal uji dapat dilihat pada Gambar 6, sedangkan untuk hasil pengujian *test case* yang lolos uji dapat dilihat Gambar 7.



Gambar 6. *Failed test report*



Gambar 7. *Success test report*

Tabel 3. *Requirement Traceability Matrix*

No.	*Requirements*	*Test ID*	*Manual Testing Result*	*Automated Testing Result*	*Hybrid Testing Result*
1.	*Payment Page* – Pengguna dapat melihat tata cara pembayaran sesuai metode yang dipilih. Sistem akan mengarahkan pengguna untuk mengisi akun kartu kredit jika memilih metode tersebut. Selain kartu kredit, pengguna melakukan pembayaran di luar sistem.	PYM_02 (VA) PYM_03 (CC) PYM_04 (QRIS)	*Covered*	*Not Covered*	*Covered*
2.	Cek Status Pembayaran – Pengguna dapat melihat dan memperbarui status pembayaran. Pengguna dapat melihat tiket yang berhasil dibeli.	PYM_05 PYM_06 HST_17 HST_18 HST_19	*Covered*	*Not Covered*	*Covered*

C. Analisis Perbandingan Metode Eksekusi Pengujian

1. *Requirement Coverage*

Pengujian yang telah dilakukan menghasilkan data yang dapat diolah menjadi bahan perbandingan metode eksekusi pengujian. Perbandingan pertama menggunakan data *Requirement Coverage* yang berasal dari *Requirement Traceability Matrix* atau RTM. Tabel 3 menampilkan beberapa *requirements* dari total 15 *requirements* pada sistem Travelink yang telah dilakukan pengujian *end-to-end* secara manual, otomatis dan *hybrid*. RTM yang dibuat pada penelitian ini terdiri dari data *requirements system*, *test case* yang mencakup *requirements system*, dan status *requirement* dari hasil eksekusi tiap metode pengujian. *Requirement* yang dituliskan merupakan dokumen *user acceptance criteria* atau UAC. UAC berasal dari kebutuhan sistem secara bisnis yang digunakan sebagai acuan dalam pengembangan dan pengujian sistem. Status *requirement* terdiri dari *Covered* dan *Not Covered*. Status *Covered* diberikan pada *requirement* yang berhasil dalam pengujian *test case* yang mencakup *requirement* tersebut. Sedangkan t*est cases* yang terdapat kegagalan atau *bug* setelah pengujian, maka *requirement* terkait diberi status *Not covered*.

Berdasarkan Tabel 3 diperoleh informasi bahwa terdapat 2 *requirements* tidak ter-*cover* pada pengujian otomatis. Dua *requirements* yang tidak ter-*cover* adalah pembayaran QRIS dan cek status pembayaran QRIS. Hal ini dikarenakan *test case* pembayaran QRIS tidak dapat diuji secara otomatis. *Test case* cek status pembayaran mengalami gagal uji karena tidak ada pembayaran yang berhasil, sehingga pengujian otomatis tidak dapat mencakup 2 *requirements* tersebut. Pengujian *end-to-end* secara manual dan *hybrid* semua *requirement* tercakup oleh *test case* yang dieksekusi, baik itu *test case Credit Card, Virtual Account*, maupun QRIS. *Test case Credit Card* dan *Virtual Account* yang dilakukan dengan pengujian otomatis mampu mencakup seluruh *requirement*. Data dari masing-masing metode pengujian tersebut kemudian dihitung menggunakan Persamaan (1) seperti berikut.

$$\text{Requirement Coverage (RC)}\% = \frac{NRC}{TNR} \times 100\% \quad (1)$$

a. RC *Manual testing:* $\frac{15}{15}$ x 100% = 100%

b. RC Automated testing: $\frac{13}{15}$ x 100% = 87%

c. RC *Hybrid testing*: $\frac{15}{15}$ x 100% = 100%

4. Waktu Eksekusi

Waktu eksekusi dihitung dengan Persamaan (2), dan waktu eksekusi pengujian manual ditampilkan pada Tabel 4.

$$\text{Rata waktu eksekusi manual test} = \frac{total\ waktu\ eksekusi\ manual\ test}{total\ responden} \quad (2)$$

Tabel 4. Waktu eksekusi pengujian manual (detik)

	End-to-end test case		
	TC 1 (CC)	**TC 2 (VA)**	**TC 3 (QRIS)**
Responden 1	1.075	817	925
Responden 2	661	727	802
Responden 3	850	906	681
Rata-rata	862	816,677	802,677

Perbandingan metode eksekusi pengujian pada penelitian ini juga menggunakan waktu eksekusi pengujian. Tabel 4

menampilkan data rata-rata waktu eksekusi pengujian manual dengan menggunakan Persamaan (2). Metode eksekusi pengujian manual dilakukan oleh responden berjumlah tiga orang. Masing-masing responden menguji tiga *end-to-end test case* (CC, VA, dan QRIS). Berdasarkan data dari tabel tersebut, waktu eksekusi setiap *end-to-end test case* dijumlah dan dibagi dengan total responden, sehingga didapatkan rata-rata waktu eksekusi. Rata-rata waktu eksekusi pengujian manual kemudian digunakan untuk bahan perbandingan dengan metode eksekusi yang lain.

Berdasarkan Tabel 5, pengujian *end-to-end test case* dengan pembayaran *Credit Card* dan *Virtual Account* paling cepat jika dilakukan dengan metode pengujian otomatis. Perbedaan waktu eksekusi antara otomatis dan *hybrid* tidak jauh berbeda dibandingkan dengan manual. Pengujian pada *end-to-end test case* dengan pembayaran QRIS paling cepat jika dilakukan dengan metode pengujian otomatis. Hal ini disebabkan oleh jumlah *test case* yang dapat dieksekusi lebih sedikit dan terdapat *test case* yang gagal, sehingga pengujian secara otomatis berjalan lebih cepat namun tidak lolos uji. Semua *test case* QRIS lolos uji pada metode *hybrid* dan manual. Metode *hybrid* lebih cepat 599,626 detik dibandingkan dengan manual.

5. Analisis Berdasarkan Matriks *Requirement Coverage* dan Waktu Eksekusi

Berdasarkan dari perbandingan dengan matriks pengujian *requirement coverage*, metode pengujian manual dan *hybrid* memberikan cakupan pengujian 100% pada semua *end-to-end test case*, sedangkan pengujian otomatis pada *test case* QRIS menghasilkan cakupan pengujian 87%. Adapun dua *requirement* yang tidak ter-*cover* terjadi pada pembayaran QRIS dan cek status pembayaran QRIS. Hal ini menunjukkan bahwa pengujian *end-to-end test case Credit Card, Virtual Account,* dan QRIS yang dilakukan secara manual dan *hybrid* mampu mencakup semua *requirements*. Pengujian otomatis dapat mencakup semua *requirements* dengan kasus pengujian *end-to-end Credit Card* dan *Virtual Account*.

Berdasarkan dari perbandingan dengan matriks pengujian waktu eksekusi pengujian, pengujian *end-to-end test case* pembayaran *Credit Card* dan *Virtual Account* paling cepat dilakukan dengan metode pengujian secara otomatis. Pengujian secara otomatis juga terlihat paling cepat pada *endto-end test case* pembayaran QRIS, namun hal itu disebabkan oleh beberapa *test case* gagal uji. Berdasarkan perbandingan antara manual dan *hybrid*, metode pengujian *hybrid* lebih cepat dan efektif dilakukan untuk *end-to-end test case* pembayaran QRIS.

Analisis dari perbandingan di atas hanya menggunakan waktu saat eksekusi pengujian saja, sedangkan pada pengujian otomatis dan *hybrid* memerlukan waktu yang lama untuk pembuatan kode program. Pembuatan kode program yang dilakukan juga membutuhkan wawasan terkait *automation tools* yang digunakan. Hal ini menunjukkan bahwa eksekusi pengujian otomatis dan *hybrid* dapat berjalan dalam waktu yang singkat, namun membutuhkan waktu yang relatif lama saat pembuatan kode program. Hasil perbandingan dari kedua matriks pengujian dapat dilihat pada Tabel 5.

Tabel 5. Perbandingan berdasarkan matriks *Requirement Coverage* dan Waktu Eksekusi

Metode dan matriks pengujian		***E2E Test case***		
		TC 1 (CC)	**TC 2 (VA)**	**TC 3 (QRIS)**
Manual	*Requirement Coverage*	100%	100%	100%
	Lama Eksekusi (detik)	1.075	817	925
Otomatis	*Requirement Coverage*	100%	100%	87%
	Lama Eksekusi (detik)	661	727	802
Hybrid	*Requirement Coverage*	100%	100%	100%
	Lama Eksekusi (detik)	850	906	854

V. SIMPULAN

Berdasarkan sejumlah *defects* yang ditemukan pada pengujian *end-to-end* QRIS yang dilakukan secara otomatis menggunakan Robot Framework, menghasilkan persentase *test coverage*, yaitu *Requirement Coverage* sebesar 87%. Selain pengujian tersebut, metode pengujian manual dan hybrid menghasilkan *Requirement Coverage* sebesar 100%.

Berdasarkan analisis perbandingan dari waktu eksekusi pengujian dihasilkan bahwa metode pengujian otomatis adalah metode yang paling cepat untuk semua *test case*. Namun, metode pengujian otomatis pada *test case end-to-end* QRIS tidak mencakup 100% *test coverage*, maka dapat disimpulkan bahwa metode pengujian hybrid adalah metode yang paling cepat dan sesuai untuk *test case end-to-end* QRIS. Sedangkan metode pengujian otomatis adalah metode yang paling cepat dan sesuai untuk *test case end-to-end Credit Card* dan *Virtual Account*.

REFERENSI

[1] T. P. Nagarhalli, V. Vaze, and N. K. Rana, "A Review of Current Trends in the Development of Chatbot Systems," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India: IEEE, Mar. 2020, pp. 706–710. doi: 10.1109/ICACCS48705.2020.9074420.

[2] K. Kaur and A. Pathak, "E-Payment System on E-Commerce in India," 2015. [Online]. Available: www.ijera.com

[3] M. Y. M. Al-Sabaawi, A. A. Alshaher, and M. A. Alsalem, "User trends of electronic payment systems adoption in developing countries: an empirical analysis," *Journal of Science and Technology Policy Management*, vol. 14, no. 2, pp. 246–270, Mar. 2023, doi: 10.1108/JSTPM-11-2020-0162.

[4] M. Masihuddin, B. Ul, I. Khan, M. Mueen, U. I. Mattoo, and R. F. Olanrewaju, "A Survey on E-Payment Systems: Elements, Adoption, Architecture, Challenges and Security Concepts," *Indian J Sci Technol*, vol. 10, no. 20, 2017, doi: 10.17485/ijst/2017/v10i20/113930.

[5] D. Manova, D. Petrova-Antonova, and S. Ilieva, *TASSA Methodolgy: End-to-End Testing of Web Service Compositions*. 2018.

[6] M. Leotta, D. Clerissi, F. Ricca, and P. Tonella, "Approaches and Tools for Automated End-to-End Web Testing," in *Advances in Computers*, vol. 101, Elsevier, 2016, pp. 193–237. doi: 10.1016/bs.adcom.2015.11.007.

[7] S. Di Meglio and L. L. L. Starace, "Towards Predicting Fragility in End-to-End Web Tests," in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, Salerno Italy: ACM, Jun. 2024, pp. 387–392. doi: 10.1145/3661167.3661179.

[8] J. Lian Min, A. Istiqomah, A. Rahmani, P. Negeri Bandung, and P. P. Tester Padepokan Tujuh Sembilan-Bandung, "EVALUASI PENGGUNAAN MANUAL DAN AUTOMATED SOFTWARE TESTING PADA PELAKSANAAN END-TO-END TESTING," *Jurnal Teknologi Terapan)* |, vol. 6, no. 1, 2020.

[9] A. Gunawan, A. F. Fatikasari, and S. A. Putri, "The Effect of Using Cashless (QRIS) on Daily Payment Transactions Using the Technology Acceptance Model," *Procedia Comput. Sci.*, vol. 227, pp. 548–556, 2023, doi: 10.1016/j.procs.2023.10.557.

[10] A. Rachman, N. Julianti, and S. Arkoyah, "Challenges and Opportunities for QRIS Implementation as a Digital Payment System in Indonesia," *EkBis J. Ekon. Dan Bisnis*, vol. 8, no. 1, pp. 1–13, Jun. 2024, doi: 10.14421/EkBis.2024.8.1.2134.

[11] M. Susanti and H. Kresnha Reza, "Added Value and Ease of Using Quick Responses Qris Indonesian Standard (QRIS)," *Int. J. Sci. Technol. Manag.*, vol. 3, no. 3, pp. 715–723, May 2022, doi: 10.46729/ijstm.v3i3.518.

[12] G. A. Jasmin and D. G. P. Putri, "Komparasi Metode Automasi dan Hybrid pada Pengujian Aplikasi Mobile WebRTC Menggunakan Appium," *J. Internet Softw. Eng.*, vol. 5, no. 1, pp. 29–36, May 2024, doi: 10.22146/jise.v5i1.9034.

[13] M. N. Zafar, W. Afzal, and E. Enoiu, "Evaluating System-Level Test Generation for Industrial Software: A Comparison between Manual, Combinatorial and Model-Based Testing," in *Proceedings - 3rd ACM/IEEE International Conference on Automation of Software Test, AST 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 148–159. doi: 10.1145/3524481.3527235.

[14] J. Islam, "Undergraduate Thesis Subject: Smart E-commerce Website with Digital payment," 2023, doi: 10.13140/RG.2.2.14218.24004.