Komparasi Metode Automasi dan *Hybrid* pada Pengujian Aplikasi *Mobile Webrtc* Menggunakan Appium

Gabriela Anggerita Jasmin[1], Divi Galih Prasetyo Putri[1,*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada
anggeritaj@mail.ugm.ac.id
[*]Korespondensi: divi.galih@ugm.ac.id;

Abstract – Information technology has transformed communication and interaction in society. It allows people to connect virtually, regardless of distance. Virtual consultations are common, facilitated by applications with video call features using WebRTC technology. WebRTC enables real-time audio, visual, and data communication. Testing is crucial to minimize bugs, but WebRTC's complex workflow makes it challenging. Software testing can be manual or automated. Manual testing involves using two devices to test WebRTC directly. Appium, an open-source tool, is used for automated testing. Appium supports multiple platforms and programming languages, without needing access to source code for device functionalities like the camera and microphone. It's not tied to any specific testing framework. The testing process covers the entire application system, ensuring optimal functionality and addressing all issues. The testing method choice depends on requirements. Testing results favor the automated method in terms of time efficiency, requiring less time. However, the hybrid method surpasses it in test coverage and effectiveness, covering more test cases.

Keywords – End-to-End, Appium, Testing, WebRTC, Automasi, Hybrid

Intisari – Teknologi informasi telah mengubah komunikasi dan interaksi dalam masyarakat. Ini memungkinkan orang untuk terhubung secara virtual, tanpa memandang jarak. Konsultasi virtual menjadi umum, difasilitasi oleh aplikasi dengan fitur panggilan video menggunakan teknologi WebRTC. WebRTC memungkinkan komunikasi audio, visual, dan data secara real-time. Pengujian sangat penting untuk meminimalkan bug, tetapi alur kerja yang kompleks dari WebRTC membuatnya menantang. Pengujian perangkat lunak dapat dilakukan secara manual atau otomatis. Pengujian manual melibatkan penggunaan dua perangkat untuk menguji WebRTC secara langsung. Appium, sebuah alat sumber terbuka, digunakan untuk pengujian otomatis. Appium mendukung berbagai platform dan bahasa pemrograman, tanpa perlu akses ke kode sumber untuk fungsi perangkat seperti kamera dan mikrofon. Appium tidak terikat pada kerangka pengujian tertentu. Proses pengujian meliputi seluruh sistem aplikasi, memastikan fungsi optimal dan menangani semua masalah. Pemilihan metode pengujian tergantung pada kebutuhan. Hasil pengujian lebih mendukung metode otomatis dalam hal efisiensi waktu, membutuhkan waktu yang lebih sedikit. Namun, metode hibrida melampaui metode tersebut dalam cakupan pengujian dan efektivitas, mencakup lebih banyak kasus pengujian.

Kata kunci – *End-to-End*, Appium, Pengujian, *WebRTC*, Automasi, *Hybrid*

I. PENDAHULUAN

Pengaruh teknologi informasi pada kehidupan manusia sangat besar. Dalam berkomunikasi, teknologi informasi memungkinkan orang untuk terhubung secara virtual tanpa memperdulikan jarak. Dengan adanya konektivitas yang luas, pertukaran informasi menjadi merata dan dapat diakses tanpa terbatas oleh waktu dan lokasi [1]. Namun, teknologi ini juga memberikan dampak negatif seperti penyebaran berita palsu dan tindak kejahatan siber. Di sisi lain, teknologi informasi membawa banyak manfaat, seperti kemudahan dalam mendapatkan informasi dan berkomunikasi. Konsultasi virtual menjadi lebih mudah dengan aplikasi berfitur panggilan video menggunakan teknologi WebRTC. WebRTC juga dapat beroperasi di berbagai peramban web, lintas platform atau sistem operasi, baik pada komputer maupun perangkat *mobile*. Kehadiran teknologi ini juga memberikan banyak keuntungan bagi pengguna [2]. Namun, menerapkan WebRTC dalam aplikasi membutuhkan pengujian yang rumit.

Pengujian perangkat lunak bisa dilakukan secara manual atau otomatis. Dalam pengujian manual, dua perangkat digunakan untuk menguji WebRTC secara langsung. Pengujian otomatis menggunakan alat sumber terbuka bernama Appium, yang mendukung berbagai platform dan bahasa pemrograman. Proses pengujian meliputi seluruh sistem aplikasi, dengan tujuan memastikan fungsionalitas optimal dan menyelesaikan semua masalah. Pilihan metode pengujian tergantung pada kebutuhan.

Oleh karena itu, pada penelitian ini dilakukan eksplorasi metode pengujian pada aplikasi yang mengimplementasikan *WebRTC*. Setiap metode yang diimplementasikan akan dievaluasi menggunakan metriks test coverage, time, dan test case effectiveness.

Pengujian hanya mengacu pada satu alur bisnis tertentu yang menuju penggunaan WebRTC. Pengujian tidak mencakup pengaruh penggunaan koneksi internet. Aplikasi yang diuji merupakan aplikasi berbasis *mobile* dengan sistem operasi Android.

Hasil pengujian diharapkan dapat memberikan informasi kepada pengembang tentang bug yang ditemukan dan menjadi panduan dalam melakukan perbaikan bug pada pengembangan selanjutnya.

II. DASAR TEORI

A. Pengujian Perangkat Lunak

1) Pengujian Antarmuka Pengguna Perangkat Lunak

UI merupakan komponen krusial dalam aplikasi Android karena berfungsi sebagai antarmuka yang langsung dilihat oleh pengguna. Kualitas UI yang baik memudahkan pengguna untuk memahami aktivitas yang terjadi dalam aplikasi. Pengujian UI melibatkan berbagai tahap dan komponen seperti tombol, input teks, dan menu dropdown. Hal ini penting karena fungsionalitas UI berdampak langsung pada kualitas perangkat lunak [3].

Pengujian merupakan tahapan penting dalam pengembangan aplikasi Android yang bertujuan untuk memastikan kualitas perangkat lunak. Hasil pengujian mempengaruhi pengembangan selanjutnya. Strategi pengujian yang tepat dapat membantu dalam mendeteksi lebih banyak bug dan menghemat biaya, sementara penerapan strategi yang kurang tepat dapat mengakibatkan bug terlewat dan biaya yang meningkat [4].

2) Pengujian End-to-end

Pengujian end-to-end adalah pengujian sistem perangkat lunak secara menyeluruh, yang melibatkan semua komponen dan modul untuk memastikan integrasinya. Tujuannya adalah untuk menemukan masalah dan memverifikasi bahwa sistem berfungsi dengan baik dalam skenario penggunaan yang sebenarnya. Proses pengujian ini meliputi perencanaan, pembuatan skenario pengujian, pelaksanaan tes, dan analisis hasilnya. Keuntungannya termasuk pengujian yang komprehensif terhadap alur kerja sistem dan mendeteksi masalah sebelum sistem diimplementasikan. Namun, pengujian ini bisa kompleks dan memerlukan sumber daya yang signifikan.

3) Pengujian Black Box

Pengujian perangkat lunak bertujuan untuk memvalidasi dan memverifikasi bahwa perangkat lunak telah dibangun sesuai dengan kebutuhan yang telah ditetapkan. Dalam proses pengujian, dilakukan pemeriksaan untuk menemukan bug atau kesalahan sedini mungkin, sehingga dapat segera diperbaiki oleh pengembang. Pengujian perangkat lunak penting untuk menjamin kualitas perangkat lunak yang dikembangkan.

Ada dua jenis pengujian yang umum digunakan, yaitu white box testing dan black box testing. Black box testing berfokus pada fungsionalitas perangkat lunak, dengan mendeteksi kesalahan yang berkaitan dengan fungsi, antarmuka, struktur data, performa, inisialisasi, dan terminasi [5].

4) Pengujian Automasi Perangkat Lunak

Pengujian perangkat lunak dapat dilakukan secara manual atau automasi. Pengujian automasi menggunakan tools atau framework untuk mengotomatisasi proses pengujian. Dalam pengujian automasi, skrip atau kode program digunakan untuk menjalankan serangkaian tes secara otomatis, termasuk simulasi interaksi pengguna dengan perangkat lunak. Keuntungan pengujian automasi adalah cakupan pengujian yang luas, kemampuan untuk melakukan pengujian berulang-ulang secara otomatis, dan penghematan waktu. Selain itu, pengujian automasi juga dapat mendeteksi banyak bug atau error, dan dokumentasi pengujian dilakukan dengan baik [3].

B. Alat Pengujian Automasi

1) Appium

Appium adalah tools pengujian automasi open source yang dapat digunakan untuk menguji aplikasi *mobile* native, aplikasi web seluler, dan aplikasi hybrid. Appium mendukung pengujian lintas platform, sehingga skrip yang sama dapat digunakan untuk menguji aplikasi di berbagai sistem operasi *mobile*. Appium menggunakan protokol WebDriver untuk mengendalikan aplikasi *mobile* dan melakukan berbagai aksi seperti mengklik dan memasukkan teks. Appium kompatibel dengan berbagai bahasa pemrograman dan menggunakan alat pengujian UIAutomator untuk Android dan XCUITest untuk iOS. Dengan Appium, tester dapat memeriksa dan mengotomatisasi setiap elemen dalam aplikasi *mobile* sesuai dengan skenario pengujian yang telah disusun [6].

2) UiAutomator2

UiAutomator2 Appium adalah framework pengujian otomatis untuk perangkat Android. Framework ini memungkinkan pengujian aplikasi native, hybrid, dan web *mobile* secara otomatis pada emulator dan perangkat fisik. UiAutomator2 Appium berfungsi berdasarkan protokol W3C WebDriver dengan ekstensi khusus untuk menangani skenario-skenario khusus sistem operasi. Framework ini menggunakan server UiAutomator2 yang mengoperasikan perintah-perintahnya dan memanfaatkan framework UiAutomator yang dikembangkan oleh Google. UiAutomator2 Appium terintegrasi dengan baik dalam versi-versi terbaru Android dan menyediakan alat pengujian otomatis untuk berinteraksi dengan perangkat, termasuk alat seperti UiSelector untuk mengambil elemen-elemen dari aktivitas utama emulator [7].

3) JavaScript

JavaScript adalah bahasa pemrograman ringan dan lintas platform yang digunakan untuk pengembangan halaman web dan aplikasi. Selain di browser, JavaScript juga bisa digunakan di sisi server menggunakan Node.js.

Dalam pengujian perangkat lunak, JavaScript sangat penting. Untuk pengujian aplikasi *mobile* secara otomatis, JavaScript bisa digunakan dengan framework seperti Appium atau WebDriverIO. Dengan JavaScript, pengujian aplikasi *mobile* bisa mencakup tindakan seperti mengklik, memasukkan teks, dan memverifikasi elemen aplikasi. Kombinasi JavaScript dan Appium memberikan kekuatan dalam otomatisasi pengujian aplikasi *mobile*.

Dalam pengujian aplikasi *mobile* dengan JavaScript, penting untuk memilih kerangka kerja yang tepat, memahami struktur aplikasi dan platform yang dituju, serta menerapkan prinsip pengujian yang baik. Terdapat juga perpustakaan JavaScript khusus seperti selendroid, Detox, atau WebdriverIO yang dapat meningkatkan efisiensi dan kemudahan dalam pengujian. Dengan JavaScript, pengujian aplikasi *mobile* dapat dilakukan di berbagai platform seperti iOS dan Android, dengan melakukan berbagai aksi dan verifikasi pada elemen aplikasi.

4) Node.js

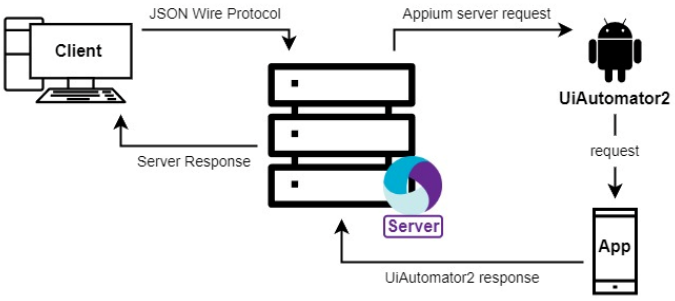
Node.js adalah lingkungan runtime JavaScript yang menggunakan mesin JavaScript V8 dari Google Chrome. Dengan Node.js, pengembang dapat menjalankan kode JavaScript di sisi server untuk membangun aplikasi web yang efisien dan skalabel. Node.js menggunakan model non-blokir I/O yang memungkinkan pengolahan paralel yang responsif terhadap permintaan. Hal ini membuatnya cocok untuk mengembangkan aplikasi jaringan yang membutuhkan penanganan banyak permintaan secara bersamaan. Node.js juga dilengkapi dengan NPM (Node Package Manager) yang menyediakan ribuan paket modul siap pakai untuk memperluas fungsionalitas Node.js.

C. WebSocket

WebSocket adalah protokol komunikasi real-time yang memfasilitasi komunikasi dua arah antara klien dan server. Protokol ini memiliki keunggulan dalam komunikasi langsung, mengurangi beban protokol, dan mendukung komunikasi lintas domain. WebSocket sering digunakan dalam aplikasi web seperti chat, pembaruan data real-time, dan kolaborasi online. Dalam pengujian aplikasi WebSocket, penting untuk menguji fungsionalitas komunikasi real-time, pembaruan data langsung, penanganan kesalahan, dan integrasi dengan komponen lainnya. Pengujian ini melibatkan simulasi pesan, pemantauan respons server, dan pengujian beban untuk memastikan performa yang optimal ketika terdapat banyak koneksi WebSocket.

D. Arsitektur Pengujian

Pada pengujian dengan Appium, skrip pengujian ditulis menggunakan bahasa pemrograman yang didukung oleh klien Appium. Skrip ini berisi perintah dan assersi untuk mengontrol aplikasi, mengambil data, dan memverifikasi hasilnya. Selama pengujian, server Appium menerima perintah dari skrip dan mengirimkannya ke driver UiAutomator2. Driver ini menjalankan perintah pada perangkat Android yang sedang diuji. Respons dari perangkat Android dikirim kembali ke server Appium, yang kemudian meneruskannya ke klien Appium. Dengan arsitektur ini, Appium memungkinkan pengujian otomatis aplikasi Android dengan pengendalian yang lengkap, serta kemampuan untuk menguji fungsionalitas dan memverifikasi perilaku aplikasi di perangkat Android. Arsitektur pengujian diilistrasikan pada Gambar 1.



Gambar 1. Arsitektur Pengujian

E. Test Case

Test case (kasus pengujian) adalah spesifikasi detail yang memberikan petunjuk tentang langkah-langkah, data, dan hasil yang diharapkan dalam pengujian. Test case digunakan untuk menguji fungsionalitas perangkat lunak dengan mengikuti prosedur yang telah ditentukan sebelumnya. Test case yang baik dapat menemukan kesalahan dan cacat yang belum terdeteksi sebelumnya, bukan hanya untuk memverifikasi bahwa program berfungsi dengan benar. Hasil pengujian dengan test case akan dibandingkan dengan hasil yang diharapkan, dan jika ada perbedaan, dilakukan perbaikan pada kode program [9].

F. WebRTC

WebRTC (Web Real-Time Communication) adalah framework open source yang memungkinkan komunikasi real-time seperti panggilan suara, panggilan video, dan pertukaran data langsung dalam aplikasi web dan *mobile*. WebRTC mendukung penggunaan perangkat keras pada perangkat *mobile* seperti kamera dan mikrofon, sehingga memudahkan integrasi fitur panggilan video dan suara langsung. Pengguna dapat melakukan komunikasi peer-to-peer tanpa perlu menginstal plugin atau perangkat tambahan [10]. WebRTC menghubungkan berbagai platform dan perangkat melalui protokol yang disediakan oleh website resmi WebRTC menggunakan JavaScript API.

III. METODOLOGI

A. Spesifikasi Perangkat

Dalam penelitian ini spesifikasi perangkat lunak dan perangkat keras yang digunakan adalah sebagai berikut.

Perangkat Lunak

1) Sistem operasi Android 10
2) Sistem operasi Windows 11 pro 64-bit
3) Visual Studio Code
4) Appium
5) UiAutomator2

Perangkat Keras

1) Laptop Asus Zenbook Flip 14 UX462DA
   - RAM 8 GB
   - Processor AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
   - SSD 256 GB
2) Samsung Galaxy M20
   - RAM 3 GB
   - ROM 32 GB
   - Processor Exynos 7904
   - Android 10
   - Kamera belakang 13MP
   - Kamera depan 8MP
3) Redmi Note 11 Pro
   - RAM 8 GB
   - ROM 128 GB
   - Processor MediaTek Helio G96
   - Android 13
   - Kamera belakang 108MP
   - Kamera depan 16MP

B. Tahapan Penelitian

Pengerjaan penelitian dimulai dengan menganalisis sistem kerja aplikasi dan memahami alur aplikasi (Gambar 2). Selanjutnya, test case dirancang untuk menguji bagian-bagian penting dalam aplikasi. Test case akan direview untuk memastikan kematangannya. Setelah test case siap, dilakukan persiapan kode untuk pengujian otomatis dengan menggunakan Appium. Pengujian otomatis dilakukan dan diikuti oleh pengujian manual. Setelah semua pengujian selesai, dilakukan analisis untuk membandingkan hasil pengujian dengan metrik yang telah ditetapkan, seperti test coverage, time, dan test case effectiveness. Perhitungan metrik pengujian dapat memberikan penilaian terhadap kecacatan yang muncul [8].

1) Test Coverage

Test coverage adalah metrik yang mengevaluasi sejauh mana pengujian telah dilakukan terhadap kode atau sistem yang diuji. Metrik ini dapat mengindikasikan kelengkapan pengujian dan tujuannya adalah untuk menentukan seberapa banyak kode atau fungsionalitas aplikasi yang telah diuji dan sejauh mana pengujian tersebut dilakukan secara efektif [8]. Ada juga konsep scenario coverage yang mengukur sejauh mana skenario atau kasus pengujian telah dijalankan. Hal ini membantu dalam mengevaluasi cakupan pengujian dan memastikan bahwa area yang memadai telah diuji. Dalam menghitung persentase *test coverage*, dapat dilihat pada persamaan (1).

$$TC = \frac{ETC}{TTC} x\ 100\% \qquad (1)$$

TC = Test Coverage
ETC = Executed Test Case
TTC = Total Test Case

2) Time

Test time adalah durasi yang diperlukan untuk menjalankan tes dan mendapatkan hasilnya. Metrik ini digunakan untuk mengevaluasi efisiensi dan efektivitas proses pengujian. Durasi test time dapat berbeda-beda tergantung pada kompleksitas test case, ukuran sistem, konfigurasi perangkat keras dan lunak, serta performa lingkungan pengujian. Penting untuk memantau dan mengelola test time secara efektif untuk memastikan penyelesaian tepat waktu aktivitas pengujian dan mengidentifikasi masalah potensial yang dapat mempengaruhi proses pengujian secara keseluruhan. Manajemen test time yang efisien melibatkan optimisasi proses eksekusi tes, memberikan prioritas pada test case kritis, menggunakan alat otomatisasi, dan menerapkan strategi pengujian paralel atau terdistribusi jika memungkinkan. Dengan mengelola test time dengan baik, proses pengujian dapat dilakukan dengan cepat, meningkatkan produktivitas, dan menghasilkan perangkat lunak berkualitas sesuai dengan jadwal yang ditargetkan. Metrik ini dihitung dari total waktu eksekusi pengujian.

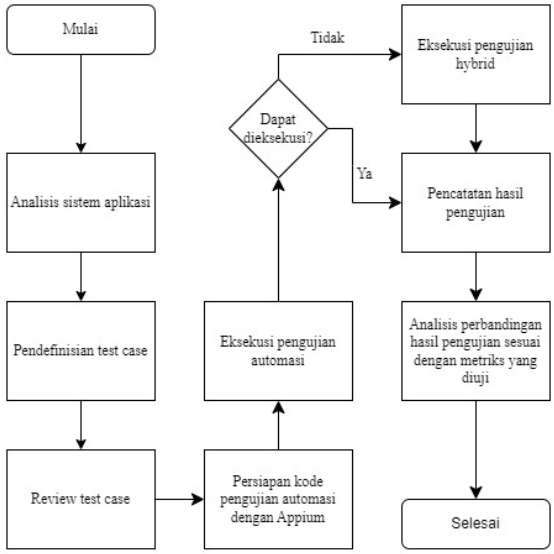
3) Test Effectiveness

Mengukur kekurangan dalam pengujian perangkat lunak adalah suatu disiplin yang penting, tidak peduli apakah pengujian telah diotomatisasi atau tidak [8]. Tujuan utama pengujian perangkat lunak adalah mengidentifikasi bug atau cacat dalam perangkat lunak, baik itu bug yang valid maupun yang tidak valid atau tidak relevan. Jika ditemukan bug yang valid, langkah-langkah perbaikan dan pengujian ulang harus dilakukan, sementara bug yang tidak valid dapat ditutup atau diabaikan.

Efektivitas pengujian perangkat lunak dapat diukur dengan persentase jumlah bug yang ditemukan dibandingkan dengan jumlah total test case yang dijalankan. Rumus untuk menghitung metrik efektivitas test case adalah: jumlah bug yang ditemukan dibagi oleh jumlah total *test case* dikalikan dengan 100% yang dapat dilihat pada persamaan (2).

$$TCE = \frac{DD}{TTC} \text{x } 100\% \qquad (2)$$

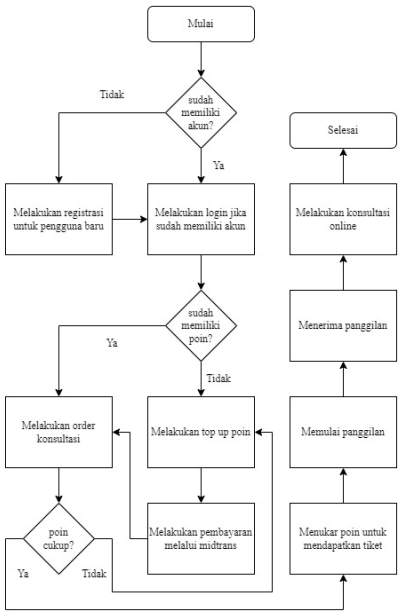
TCE = Test Case Effectiveness
DD = Defects Detected
TTC = Total Test Case



Gambar 2. Tahapan Pengujian

C. Analisis Sistem Aplikasi

Alur bisnis aplikasi Ngampu dimulai dengan registrasi pengguna baru. Pengguna kemudian dapat melakukan login untuk mengakses aplikasi. Selanjutnya, pengguna dapat melakukan top up poin dan melakukan pembayaran melalui midtrans. Setelah itu, pengguna dapat memesan konsultasi melalui menu konsultan. Jika poin cukup, pesanan dapat berhasil dibuat. Jika tidak, pengguna dapat melakukan top up terlebih dahulu. Setelah pesanan konsultasi terbuat, panggilan video sesuai jadwal yang dipilih dapat dimulai. Sesi konsultasi dimulai setelah konsultan menerima panggilan dari pengguna. Analisis sistem aplikasi dapat dilihat pada Gambar 3.



Gambar 3. Alur Bisnis Aplikasi Ngampu

D. Test Case

Pengujian dilaksanakan dengan acuan test case yang telah dibuat seperti yang dapat dilihat pada Tabel 1.

Tabel 1. Test Case

ID	Test Case
TC01	Register valid
TC02	Register dengan Mengosongkan Nama
TC03	Register dengan Mengosongkan Email
TC04	Register dengan Mengosongkan Password
TC05	Register dengan Nama Menggunakan Angka
TC06	Register dengan Email Tanpa @
TC07	Register dengan Email Tanpa . (dot)
TC08	Register dengan Email yang Sudah Terdaftar
TC09	Register dengan Password Tanpa Huruf Kapital
TC10	Register dengan Password Tanpa Huruf Kecil
TC11	Register dengan Password Tanpa Angka
TC12	Register dengan Password Tanpa Simbol
TC13	Register dengan Password Kurang dari Delapan Karakter
TC14	Register dengan Menekan Tombol Daftar Berulang Kali
TC15	Login Valid
TC16	Login dengan Mengosongkan Email
TC17	Login dengan Mengosongkan Password
TC18	Login dengan Email Tanpa @
TC19	Login dengan Email Tanpa . (dot)
TC20	Login dengan Email yang Belum Terdaftar
TC21	Login dengan Password Tanpa Huruf Kapital
TC22	Login dengan Password Tanpa Huruf Kecil
TC23	Login dengan Password Tanpa Angka
TC24	Login dengan Password Tanpa Simbol
TC25	Login dengan Password Kurang dari Delapan Karakter
TC26	Login dengan Password yang Tidak Valid Lebih dari Tiga Kali
TC27	Login dengan Menekan Tombol Login Berulang Kali
TC28	Forgot Password Valid
TC29	Forgot Password dengan Mengosongkan Email
TC30	Forgot Password dengan Email Tanpa @
TC31	Forgot Password dengan Email Tanpa . (dot)
TC32	Forgot Password dengan Email yang Belum Terdaftar
TC33	Top Up dengan Memilih Pilihan Nominal yang Tersedia dan Sudah Login
TC34	Top Up dengan Mengisi Jumlah Nominal dan Sudah Login
TC35	Top Up dengan Mengisi Nominal Nol dan Sudah Login
TC36	Top Up Tanpa Mengisi Nominal dan Sudah Login
TC37	Top Up dengan Memilih Pilihan Nominal yang Tersedia dan Sudah Login namun Unverified
TC38	Top Up dengan Mengisi Jumlah Nominal dan Sudah Login namun Unverified

ID	Test Case
TC39	Top Up dengan Mengisi Nominal Nol dan Sudah Login namun Unverified
TC40	Top Up Tanpa Mengisi Nominal dan Sudah Login namun Unverified
TC41	Top Up dengan Memilih Pilihan Nominal yang Tersedia dan Belum Login
TC42	Top Up dengan Mengisi Jumlah Nominal dan Belum Login
TC43	Top Up dengan Mengisi Nominal Nol dan Belum Login
TC44	Top Up Tanpa Mengisi Nominal dan Belum Login
TC45	Memesan Konsultasi Valid Sudah Login
TC46	Memesan Konsultasi Tanpa Mengisi Judul Konsultasi Sudah Login
TC47	Memesan Konsultasi Tanpa Memilih Waktu Konsultasi Sudah Login
TC48	Memesan Konsultasi dengan Poin Kosong Sudah Login
TC49	Memesan Konsultasi dengan Menekan Tombol Lanjutkan Berulang Kali
TC50	Memesan Konsultasi Valid Belum Login
TC51	Memesan Konsultasi Tanpa Mengisi Judul Konsultasi Belum Login
TC52	Memesan Konsultasi Tanpa Memilih Waktu Konsultasi Belum Login
TC53	Melihat Tiket Konsultasi yang Aktif Sudah Login
TC54	Melihat Tiket Konsultasi yang Aktif Belum Login
TC55	Melakukan Panggilan ke User Lain yang Belum Login
TC56	Melakukan Panggilan ke User Lain yang Sudah Login
TC57	Menerima Panggilan dari User Lain
TC58	Mematikan dan Menyalakan Kamera
TC59	Mematikan dan Menyalakan Mikrofon
TC60	Switch Kamera
TC61	Menekan Tombol Kembali Saat Melakukan Panggilan

IV. HASIL DAN PEMBAHASAN

A. Pengujian Fungsionalitas

Tabel 2 menyajikan data bahwa pada pengujian automasi, terdapat lima puluh test case yang lulus uji, tujuh test case yang mendapatkan peringatan, dua test case yang tidak lulus uji, dan dua test case yang tidak dapat diuji. Sementara itu, pada pengujian hybrid, terdapat lima puluh tiga test case yang lulus uji, enam test case yang mendapatkan peringatan, dan dua test case yang tidak lulus uji.

Tabel 2. Hasil Pengujian

ID	Automated	Hybrid
TC01	Passed	Passed
TC02	Passed	Passed
TC03	Passed	Passed
TC04	Passed	Passed
TC05	Passed	Passed
TC06	Passed	Passed
TC07	Passed	Passed
TC08	Warning	Warning
TC09	Passed	Passed
TC10	Passed	Passed
TC11	Passed	Passed
TC12	Passed	Passed
TC13	Passed	Passed
TC14	Warning	Warning
TC15	Passed	Passed
TC16	Passed	Passed
TC17	Passed	Passed
TC18	Passed	Passed
TC19	Passed	Passed
TC20	Passed	Passed
TC21	Passed	Passed
TC22	Passed	Passed
TC23	Passed	Passed
TC24	Passed	Passed
TC25	Passed	Passed
TC26	Failed	Failed
TC27	Cannot be Tested	Passed
TC28	Passed	Passed
TC29	Warning	Warning
TC30	Passed	Passed
TC31	Passed	Passed
TC32	Passed	Passed
TC33	Passed	Passed
TC34	Passed	Passed
TC35	Passed	Passed
TC36	Passed	Passed
TC37	Passed	Passed
TC38	Passed	Passed
TC39	Passed	Passed
TC40	Passed	Passed
TC41	Passed	Passed
TC42	Passed	Passed
TC43	Passed	Passed
TC44	Passed	Passed
TC45	Passed	Passed
TC46	Warning	Warning
TC47	Warning	Warning
TC48	Passed	Passed
TC49	Cannot be Tested	Passed
TC50	Passed	Passed
TC51	Warning	Warning
TC52	Passed	Passed
TC53	Passed	Passed
TC54	Passed	Passed
TC55	Passed	Passed
TC56	Passed	Passed
TC57	Warning	Passed
TC58	Passed	Passed
TC59	Passed	Passed
TC60	Passed	Passed
TC61	Failed	Failed

B. Perbandingan Metode

1) Test Coverage

Dalam Tabel 3 disajikan data pengujian yang dilakukan, metode automasi berhasil menjalankan lima puluh sembilan test case, dengan lima puluh test case yang berhasil, tujuh test case mendapatkan peringatan, dan dua test case yang gagal. Namun, terdapat dua test case yang tidak dapat dilakukan, yaitu test case menekan tombol login dan lanjutkan pada pemesanan konsultasi secara berulang kali, karena tombol tersebut tidak dapat ditekan lebih dari sekali.

Tabel 3. Perbandingan Metrik Test Coverage

Status	Automated	Hybrid
Passed	50	53
Warning	7	6
Failed	2	2
Cannot be Tested	2	0
TOTAL	61	61

Berdasarkan metriks cakupan pengujian, perhitungan persentase cakupan pengujian pada masing-masing metode adalah sebagai berikut:

- Automated testing : $\frac{59}{61}$ x 100% = 96.72%
- Hybrid testing : $\frac{61}{61}$ x 100% = 100%

Sementara itu, metode hybrid berhasil menjalankan semua enam puluh satu test case, dengan lima puluh tiga test case yang berhasil, enam test case mendapatkan peringatan, dan dua test case yang gagal.

2) Time

Berdasarkan hasil pengujian yang disajikan pada Tabel 4, pengujian dilakukan dalam tiga percobaan. Metode automasi menunjukkan waktu yang lebih singkat dibandingkan metode hybrid. Rata-rata pengujian automasi membutuhkan waktu 1728,737 detik, sedangkan pengujian hybrid membutuhkan waktu 1791,002 detik dengan catatan *test case* yang dapat dieksekusi pada pengujian automasi lebih sedikit dibandingkan dengan pengujian *hybrid*. Selain itu juga seperti yang dipaparkan pada alur pengujian, metode *hybrid* melalui proses yang lebih panjang.

Tabel 4. Perbandingan Metrik Time

	Automated	Hybrid
Time I	1705,508 s	1777,987 s
Time II	1775,812 s	1833,733 s
Time III	1704,891 s	1761,287 s

3) Test Case Effectiveness

Dari hasil pengujian yang disajikan pada Tabel 5, didapatkan hasil bahwa dari empat puluh tujuh *test case* negatif, *automated testing* dapat mendeteksi *defect* sebanyak empat puluh lima. Sedangkan *hybrid testing* dapat mendeteksi sejumlah empat puluh enam *defect*.

Tabel 5. Perbandingan Metrik Test Case Effectiveness

	Automated	Hybrid
Defects Detected	40	41

- *Automated* : $\frac{40}{61}$ x 100% 65.574%
- *Hybrid* : $\frac{41}{61}$ x 100% = 67.21%

Defect yang ditemukan terdiri dari *test case* tipe positif yang berstatus *failed* dan juga *test case* negatif yang berstatus *passed*.

V. SIMPULAN

Berdasarkan jumlah temuan bug pada pengujian pada aplikasi Ngampu, antara pengujian automasi dengan metode *hybrid* tidak jauh berbeda. Metode automasi mendeteksi lima bug *suggestion*, tiga bug *medium*, dan tiga bug *critical* dari total enam puluh satu *test case*. Pada pengujian *hybrid*, dari enam puluh satu *test case* ditemukan sebanyak lima bug *suggestion*, dua bug *medium*, dan satu bug *critical*. Dari hasil tersebut, dapat disimpulkan bahwa aplikasi Ngampu sudah layak untuk rilis dengan beberapa perbaikan

Berdasarkan metriks *test coverage*, metode hybrid jauh lebih baik dibandingkan dengan metode automation karena memiliki cakupan pengujian yang lebih luas yaitu sebesar 100%. Berdasarkan metriks *time*, metode automation jauh lebih cepat dibandingkan dengan metode hybrid dengan rata-rata lama pengujian selama 1728,737 detik. Berdasarkan metriks *test case effectiveness*, metode *hybrid* lebih unggul dari metode automasi persentase keefektifan sebesar 67.21%

REFERENSI

[1] R. Ainanda dan E. V. Haryanto, "Rancang Bangun Aplikasi E-meeting Menggunakan WebRTC (Web Real time Communication) Design and Build E-meeting Applications Using WebRTC (Web Real time Communication)," 2020.

[2] T. Abdulghani, M. Maulana, dan H. Gozali, "Sistem Konsultasi dan Bimbingan Online Berbasis Web Menggunakan Webrtc (Studi Kasus : Fakultas Teknik Universitas Suryakancana)," *Media Jurnal Informatika*, vol. 11, no. 2, 2019, [Daring]. Tersedia pada: http://jurnal.unsur.ac.id/mjinformatika

[3] M. M. Muhtadi, M. Dhandy Friyadi, dan A. Rahmani, "Analisis GUI Testing pada Aplikasi E-Commerce menggunakan Katalon," 2019.

[4] H. Singh, S. K. Jha, D. Gupta, dan A. V. Singh, "GUI Testing Android Application," dalam *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions), ICRITO 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICRITO56286.2022.9965072.

[5] L. Setiyani, "Techno Xplore Jurnal Ilmu Komputer dan Teknologi Informasi PENGUJIAN SISTEM INFORMASI INVENTORY PADA PERUSAHAAN DISTRIBUTOR FARMASI MENGGUNAKAN METODE BLACK BOX TESTING," 2019.

[6] A. R. Rambe, "Pengujian Otomatis dengan Teknik Black Box Menggunakan Appium," 2022.

[7] L. Baird, Z. Shan, dan V. Namboodiri, "Automated Dynamic Detection of Self-Hiding Behavior," dalam *Proceedings - 2019 IEEE 16th International Conference on Mobile Ad Hoc and Smart Systems Workshops, MASSW 2019*, Institute of Electrical and Electronics Engineers Inc., Nov 2019, hlm. 87–91. doi: 10.1109/MASSW.2019.00024.

[8] T. Garrett dan B. Gauf, "Useful Automated Software Testing Metrics," 2011. [Daring]. Tersedia pada: http://www.thefreedictionary.com/metric

[9] A. N. Hasibuan dan T. Dirgahayu, "Pengujian dengan Unit Testing dan Test case pada Proyek Pengembangan Modul Manajemen Pengguna," 2021.

[10] R. Y. Rahmanda, E. Sakti Pramukantoro, dan W. Yahya, "Perancangan dan Implementasi Kelas Virtual FILKOM Universitas Brawijaya dengan Memanfaatkan Teknologi WebRTC (Web Real-Time Communication)," 2018. [Daring]. Tersedia pada: http://j-ptiik.ub.ac.id