Evaluasi Perangkat Lunak Aplikasi Cards Edu Melalui Penerapan Software Testing Life Cycle (STLC)

[1st]Aorinka Anendya Chazanah
S1 Rekayasa Perangkat Lunak
Universitas Telkom Purwokerto
Purwokerto, Indonesia
aorinkanendya@student.telkomuniversity.ac.id

[2nd]Arif Amrulloh
S1 Rekayasa Perangkat Lunak
Universitas Telkom Purwokerto
Purwokerto, Indonesia
arifta@telkomuniversity.ac.id

[3rd]Rifki Adhitama
S1 Rekayasa Perangkat Lunak
Universitas Telkom Purwokerto
Purwokerto, Indonesia
rifkiadhitama@telkomuniversity.ac.id

***Abstrak* — Perkembangan aplikasi pendidikan berbasis *mobile* menuntut kualitas perangkat lunak yang andal agar dapat mendukung kegiatan belajar-mengajar secara optimal. *Cards* Edu merupakan aplikasi manajemen sekolah berbasis *mobile* yang digunakan oleh guru dan murid, namun masih ditemukan berbagai *bug* dan kendala fungsional yang mengganggu pengalaman pengguna. Penelitian ini bertujuan untuk mengevaluasi kualitas fungsional aplikasi *Cards* Edu serta membandingkan efektivitas dan efisiensi pengujian manual dan otomatis. Metode yang digunakan meliputi pengujian fungsional terhadap fitur utama yang diprioritaskan berdasarkan kebutuhan pengguna dengan prinsip Pareto. Berdasarkan pengujian terhadap 14 fitur utama yang menghasilkan 128 skenario pengujian, diperoleh hasil bahwa pada pengujian manual terdapat 120 skenario berhasil dan 8 gagal, sedangkan pada pengujian otomatis terdapat 115 berhasil dan 14 gagal. Pengujian otomatis lebih efektif dalam menemukan kesalahan dengan nilai 10,94% dibandingkan 6,25% pada pengujian manual, serta lebih cepat dengan waktu 56 menit 47 detik dibandingkan 63 menit. Permasalahan yang ditemukan meliputi validasi *input*, konsistensi pesan, dan stabilitas aplikasi. Hasil ini menunjukkan bahwa dalam kerangka *Software Testing Life Cycle* (STLC), pengujian otomatis lebih efektif dalam mendeteksi kesalahan, namun belum sepenuhnya menggantikan pengujian manual karena keterbatasan pada fitur yang melibatkan interaksi perangkat keras.**

***Kata kunci* — *software testing life cycle*, pengujian manual, pengujian otomatis, katalon studio, kualitas perangkat lunak**

I. PENDAHULUAN

Kemajuan teknologi informasi dan komunikasi telah mendorong transformasi di bidang pendidikan. Pemanfaatan teknologi memungkinkan proses pembelajaran beralih dari metode konvensional menuju pendekatan yang lebih modern, fleksibel, dan berbasis digital [1],[2]. Inovasi *seperti e-learning*, kelas virtual, dan multimedia pembelajaran turut mendukung terciptanya ekosistem pendidikan yang adaptif [3]. Lebih lanjut, aplikasi yang dirancang untuk keperluan pendidikan efektif dalam mempercepat sebaran materi pembelajaran dan mendorong partisipasi aktif siswa [4]. Tren *mobile learning* (M-*learning*) juga menunjukkan peningkatan signifikan, dengan ponsel pintar berfungsi sebagai kanal pembelajaran sentral yang terbukti berdampak positif pada peningkatan performa belajar dan antusiasme siswa [5].

Sejalan dengan perkembangan M-*learning* di Indonesia, aplikasi *Cards* Edu diluncurkan pada 2024 sebagai bagian dari *Cards* - Sistem Manajemen Sekolah & Pesantren oleh PT. Cazh Teknologi Inovasi. Aplikasi ini berfungsi mendigitalisasi pengelolaan akademik dan memperlancar komunikasi guru-siswa melalui fitur seperti absensi dan rekapitulasi presensi. Meskipun memberikan berbagai kemudahan bagi pengguna, kualitas perangkat lunak tetap menjadi aspek yang perlu diperhatikan. Pengujian yang menyeluruh diperlukan untuk memastikan sistem dapat berfungsi sesuai kebutuhan, meminimalkan kesalahan, serta mampu mendukung aktivitas pengguna secara optimal [6].

Pra-survei yang dilakukan menunjukkan bahwa mayoritas pengguna *Cards* Edu (58,94%) pernah mengalami *bug*, dengan dominasi pada pengguna Android (92,63%). Dampak *bug* terhadap pengalaman pengguna cukup signifikan, mengganggu aktivitas dan memerlukan upaya perbaikan atau *restart* aplikasi. Temuan ini menegaskan perlunya evaluasi kualitas perangkat lunak yang lebih mendalam [7].

Dalam konteks pengujian perangkat lunak, metode *black box* testing berfokus pada fungsionalitas sistem dari perspektif pengguna tanpa menganalisis kode internal [8]. Untuk memastikan proses pengujian yang sistematis dan terukur, diterapkan kerangka kerja *Software Testing Life Cycle* (STLC) [9]. Alat bantu seperti Katalon Studio dan *plugin* manajemen pengujian seperti *Zephyr* dapat memfasilitasi pengujian otomatis dan manual [10].

Oleh karena itu, penelitian ini bertujuan untuk mengevaluasi kualitas perangkat lunak aplikasi *Cards* Edu melalui penerapan STLC, membandingkan efektivitas pengujian manual menggunakan *Zephyr* dan pengujian otomatis menggunakan Katalon Studio, serta menyajikan hasil pengujian sebagai acuan bagi pengembang untuk peningkatan kualitas fungsionalitas aplikasi.

II. KAJIAN TEORI

A. Evaluasi Perangkat Lunak

Evaluasi perangkat lunak bertujuan untuk mengukur tingkat kesesuaian suatu aplikasi atau sistem dengan sasaran dan kebutuhan penggunanya. Proses ini melibatkan peninjauan aspek fungsional dan non-fungsional, termasuk reliabilitas, kemudahan navigasi, efisiensi, dan kepuasan pengguna. Tujuannya adalah memastikan sistem tidak hanya berfungsi secara teknis, tetapi juga memberikan pengalaman pengguna yang superior dan meningkatkan produktivitas [11].

B. *Mobile Learning*

M-learning atau *mobile learning* merupakan bentuk pengembangan dari *e-learning* yang menghadirkan proses pembelajaran yang tidak lagi terpusat pada satu tempat atau satu sumber. Pembelajaran menjadi lebih fleksibel karena memanfaatkan perangkat teknologi informasi yang bersifat portabel dan mobile, seperti pada, tablet PC, dan *smartphone*. Dengan demikian, *m-learning* memungkinkan kegiatan belajar mengajar berlangsung kapan saja dan di mana saja tanpa dibatasi oleh ruang dan waktu. [12]. Model pembelajaran ini memungkinkan siswa untuk mengulangi materi, mengerjakan tugas, serta berinteraksi dengan konten pembelajaran secara mandiri tanpa harus terikat pada jadwal tatap muka [13].

C. ISO/IEC 25010:2011

Model ISO/IEC 25010:2011 merupakan standar internasional yang menyediakan kerangka kerja untuk menilai kualitas suatu produk perangkat lunak dan sistem informasi. Standar ISO/IEC 25010:2011 adalah model kualitas produk perangkat lunak dalam kerangka kerja SQuaRE *(Software Quality Requirements and Evaluation)*. Standar ini menjelaskan kerangka kualitas produk yang diterapkan pada produk ICT *(Information and Communication Technology)*, termasuk perangkat lunak, firmware, perangkat keras, data, serta infrastruktur komunikasi [14].

D. *Simple Random Sampling*

Simple random sampling merupakan teknik pengambilan sampel dari populasi yang dilakukan secara acak tanpa memperhitungkan adanya tingkatan atau strata tertentu dalam populasi. Metode ini disebut sederhana karena setiap anggota populasi memiliki peluang yang sama untuk dipilih, tanpa memperhatikan perbedaan karakteristik antar anggota populasi. [15].

E. Prinsip Pareto

Prinsip Pareto, yang digagas Vilfredo Pareto dari observasi kekayaan di Italia abad ke-19, menyatakan bahwa segelintir orang menguasai sebagian besar aset. Konsep ini dirangkum sebagai aturan 80/20, mengindikasikan bahwa minoritas penyebab berkontribusi pada mayoritas hasil. Penerapan prinsip ini meluas di ekonomi, manajemen, dan analisis data [32]. Dalam pengujian perangkat lunak, ia membantu identifikasi komponen berisiko tinggi dan *defect* dominan untuk diprioritaskan dalam evaluasi [16].

F. Analisis Deskriptif

Analisis deskriptif merupakan metode analisis untuk mendeskripsikan atau menggambarkan data yang telah dihimpun sesuai dengan kondisi sebenarnya melalui penyajian dalam bentuk tabel, grafik, persentase, atau narasi yang menjelaskan kondisi aktual responden. Dalam penelitian, analisis deskriptif digunakan untuk menampilkan hasil survei secara jelas dan sistematis, sehingga dapat memahami pola serta karakteristik data berdasarkan jawaban responden [15].

G. Pengujian Perangkat Lunak

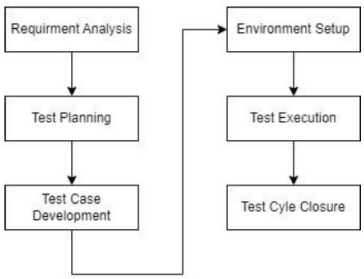
Pengujian perangkat lunak merupakan suatu proses untuk memastikan bahwa program yang dikembangkan telah berfungsi dengan baik dan sesuai dengan kebutuhan pengguna. Proses ini dilakukan dengan mengeksekusi perangkat lunak menggunakan data uji guna mengidentifikasi kesalahan, bug, atau permasalahan lain yang berpotensi muncul. Selain itu, pengujian menjadi langkah penting untuk meminimalkan potensi kerugian akibat kesalahan sistem dan menjaga kepercayaan pengguna terhadap keakuratan data serta stabilitas sistem [17].

H. *Black Box Testing*

Pengujian perangkat lunak *black box* berfokus pada spesifikasi fungsional sistem tanpa menguji kode sumber atau struktur internalnya. Pendekatan ini mengamati perilaku eksternal sistem, seperti fungsi, antarmuka, dan alur proses, tanpa mempermasalahkan implementasi kode. Pengujiannya dilakukan dengan memberikan *input* sesuai kebutuhan fungsional dan mengevaluasi *outputnya*. Dengan demikian, inti dari *black box* testing adalah pengujian *input-output* melalui antarmuka tanpa memerlukan pemahaman mendalam tentang detail internal perangkat lunak [17].

I. *Software Testing Life Cycle* (STLC)

Software Testing Life Cycle (STLC) merupakan suatu kerangka kerja sistematis yang digunakan untuk memastikan bahwa perangkat lunak telah memenuhi kebutuhan pengguna serta terbebas dari kesalahan sebelum dirilis. STLC terdiri dari serangkaian tahapan pengujian yang saling berkaitan, dimulai sejak kebutuhan sistem didefinisikan hingga seluruh aktivitas pengujian selesai dan dievaluasi. Penerapan STLC bertujuan untuk meningkatkan kualitas, keandalan, dan kesesuaian perangkat lunak dengan spesifikasi yang telah ditetapkan melalui proses pengujian yang terstruktur dan terdokumentasi [18].



GAMBAR 1
(Tahap *Software Testing Life Cycle)*

J. Skenario Uji

Dokumen skenario uji menyajikan langkah-langkah terstruktur yang dirancang penguji untuk memverifikasi

kesesuaian sistem dengan kebutuhan pengguna, kepatuhan terhadap standar, dan fungsionalitas normal. Pendekatan pengujian ini fokus pada pengujian menyeluruh setiap alur bisnis terdefinisi dari awal hingga akhir, mencakup kasus positif dan negatif untuk evaluasi komprehensif [19]. Umumnya, sebuah kasus uji meliputi identifikasi unik, deskripsi skenario, ekspektasi hasil, pencatatan hasil aktual, serta penarikan kesimpulan atau penentuan status [20].

K. Pengujian Manual

Pengujian manual melibatkan interaksi langsung manusia untuk mengevaluasi perangkat lunak, tanpa keterlibatan otomatisasi. Proses ini mengharuskan penguji menjalankan aplikasi secara manual, memverifikasi fungsionalitas, memasukkan data uji, dan mencatat hasilnya, berdasarkan prosedur yang telah ditetapkan. Kelebihan utamanya adalah adaptabilitas terhadap berbagai skenario pengujian. Metode ini juga memungkinkan pendeteksian cacat yang sulit diidentifikasi oleh alat otomatis, berkat intuisi dan pemahaman domain penguji [21].

L. Jira

Jira adalah platform manajemen proyek yang umum digunakan dalam pengembangan perangkat lunak untuk mengatur alur kerja secara sistematis. Fiturnya mencakup pengelolaan *backlog*, pelacakan progres tugas, manajemen laporan *bug*, dan fasilitasi kolaborasi tim melalui *sprint planning* serta *scrum boards* [22]. Selain itu, Jira juga berfungsi sebagai alat *issue* dan *bug tracking* yang fleksibel, terintegrasi dengan berbagai *developer tools*, serta memungkinkan pelaporan dan pemetaan alur kerja yang lebih terstruktur [23].

M. Zephyr

Zephyr adalah salah satu *plugin* manajemen pengujian yang populer untuk Jira, yang menyediakan cara terstruktur untuk mengelola *test case* dan siklus pengujian. *Plugin* ini terintegrasi langsung dengan Jira dan menambahkan fungsionalitas manajemen pengujian. Dengan Zephyr, penguji dapat membuat, mengorganisasi, dan mengeksekusi *test case* di Jira. Fitur-fitur utama Zephyr mencakup pengelompokan *test case* dalam siklus *(test cycles),* pelaksanaan pengujian langsung di Jira, keterkaitan *test case* dengan *bug*, serta pembuatan laporan pengujian [24].

N. Pengujian Otomatis

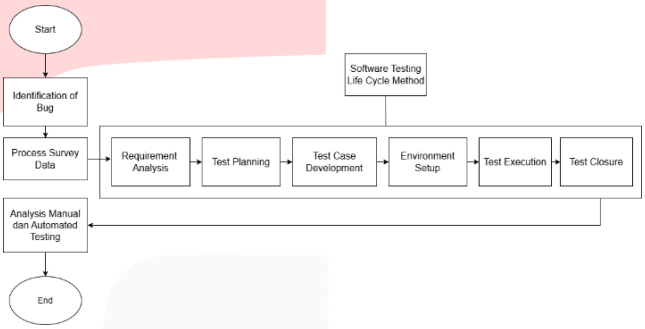
Pengujian otomatis adalah metode pengujian perangkat lunak yang memanfaatkan skrip dan alat terprogram untuk mengotomatiskan proses pengujian, termasuk pengujian fungsional dan performa. Alat ini merekam interaksi, menghasilkan data uji, dan memvalidasi keluaran tanpa campur tangan manual. Keuntungannya adalah efisiensi dan konsistensi yang meningkat karena kecepatan eksekusi yang lebih tinggi dan stabilitas hasil dibandingkan pengujian manual [20]. Automasi pengujian dapat mempercepat siklus hingga sekitar 36% dan mengurangi ketidakpastian, yang meningkatkan efisiensi proses serta keandalan perangkat lunak secara keseluruhan [25].

O. Katalon Studio

Katalon Studio, perangkat lunak otomatisasi pengujian yang dirilis Katalon Inc. pada September 2016, dibangun berdasarkan kerangka kerja *open source* seperti Selenium dan Appium. Perangkat ini memfasilitasi pengujian beragam aplikasi (web, seluler, API, desktop) menggunakan bahasa Groovy untuk skrip uji [26]. Penelitian [27] mengonfirmasi keandalannya berkat antarmuka intuitif dan fitur otomatisasi yang efisien untuk skenario kompleks, memungkinkan penyesuaian teknis yang fleksibel. Studi tersebut juga menunjukkan eksekusi pengujian yang konsisten, bebas *error*, dan waktu eksekusi lebih cepat dibanding Selenium IDE, menjadikannya solusi yang lebih efisien.

III. METODE

Penelitian ini mengadopsi pendekatan pengujian sistem kerangka kerja STLC secara terstruktur. Alur penelitian dirancang melalui enam tahapan utama yang mencakup, *Requirement Analysis*, *Test Planning*, *Test Case Development*, *Environment Setup*, *Test Execution*, dan *Test Cycle Closure*.



GAMBAR 2
(Tahapan Penelitian)

Parameter pengujian dihitung menggunakan formulasi berikut:

A. Ukuran Sampel Responden (Rumus Slovin)

$$n = \frac{N}{1 + N(d)^2}$$

dimana:
n = jumlah sampel
N = jumlah populasi
$(d)^2$ = presisi atau tingkat toleransi kesalahan

B. Perhitungan Akumulasi Frekuensi Pareto

$$Fk = F_{k-1} + Fi$$

dimana:
Fk = frekuensi kumulatif pada data ke-k
F_{k-1} = frekuensi kumulatif sebelumnya
Fi = frekuensi *bug* pada fitur ke-i

C. Persentase Akumulasi Pareto

$$Pk = \frac{F_k}{N}\ x\ 100\% \qquad (2.3)$$

dimana:
Pk = persentase kumulatif
F_k = frekuensi kumulatif

N = total seluruh frekuensi bug

D. *Test Case Coverage*

$$Test\ Coverage = \frac{Number\ of\ test\ cases\ executed}{Total\ number\ of\ test\ cases}\ x\ 100\%$$

Metrik ini mengukur kelengkapan eksekusi skenario pengujian fungsional.

E. *Test Execution Time*

$$Time\ Efficiency = \frac{Waktu\ pengujian\ manual}{Waktu\ pengujian\ otomatis}$$

F. *Test Case Effectiveness*

$$Test\ Case\ Effectiveness = \frac{Number\ of\ detected\ defects}{Total\ of\ running\ test\ cases}\ x\ 100\%$$

IV. HASIL DAN PEMBAHASAN

A. Analisis Kebutuhan Pengujian Berdasarkan Prinsip Pareto

Tahap awal STLC diimplementasikan dengan memetakan sebaran keluhan fungsional dari data primer kuesioner. Pra-survei dilakukan untuk menggali kondisi fungsionalitas dan keandalan sistem dari sisi pengguna.

TABEL 1
(Instrumen Pertanyaan Kuesioner)

No	Pertanyaan	Responden
1.	Apakah Anda pernah menemui *error* atau *bug* saat menggunakan aplikasi?	Murid & Guru
2.	Saat *bug* muncul, bagaimana kondisi aplikasi?	Murid & Guru
3.	Apakah *bug* tersebut mengganggu aktivitas Anda?	Murid & Guru
4.	Pada fitur apa saja Anda pernah menemukan *bug*?	Murid & Guru
5.	Setelah *bug* terjadi, apakah Anda bisa melanjutkan aktivitas?	Murid & Guru
6.	Apa *device* yang Anda pakai untuk menggunakan aplikasi?	Murid & Guru

Diperoleh data dengan total 1.888 dari pengguna, kemudian 95 responden diambil sebagai sampel menggunakan rumus Slovin dan tingkat kesalahan 10%, lalu diseleksi melalui *simple random* sampling untuk memastikan keterwakilan. Analisis persentase data kuesioner mengungkap bahwa 58,94% responden pernah menghadapi *bug*, didominasi pengguna Android (92,63%). Jenis *bug* yang dilaporkan meliputi aplikasi fungsional namun bermasalah (58,33%), gangguan pada fitur lain (8,33%), dan kerusakan parah seperti *crash* (33,33%). Gangguan *bug* terhadap aktivitas pengguna cukup terasa, terlihat dari 71,43% yang mengalami hambatan dalam penggunaan, 66,67% yang memerlukan percobaan ulang atau jeda pemulihan, dan 16,67% yang terhenti aktivitasnya hingga aplikasi dibuka kembali.

Kemudian melalui pengolahan data menggunakan analisis deskriptif, frekuensi kegagalan fungsi pada setiap fitur diurutkan dari nilai terbesar ke terkecil guna menyusun frekuensi distribusi Pareto.

TABEL 2
(Distribusi Frekuensi *Bug)*

No	Fitur	Frekuensi	Frekuensi Kumulatif	Persentase	Persentase Frekuensi Kumaltif
1.	Registrasi	20	20	16.81%	16.81%
2.	*Login* dengan Google	15	35	12.61%	29.41%
3.	*Login* dengan Nomor Kartu	13	48	10.92%	40.34%
4.	Isi Kartu	10	58	8.40%	48.74%
5.	Scan QR	6	64	5.04%	53.78%
6.	*Login* dengan Apple ID	5	69	4.20%	57.98%
7.	Reset PIN	5	74	4.20%	62.18%
8.	Kesiswaan (Perizinan)	5	79	4.20%	66.39%
9.	Tugas	4	83	3.36%	69.75%
10.	Rekap Keuangan	3	86	2.46%	70.49%
11.	Rekap Presensi	3	89	2.46%	72.95%
12.	Jadwal Kegiatan	3	92	2.46%	75.41%
13.	*List* Notifikasi	2	94	1.64%	77.05%

No	Fitur	Frekuensi	Frekuensi Kumulatif	Persentase	Persentase Frekuensi Kumaltif
14.	Pengumuman	2	96	1.64%	78.69%
15.	Riwayat	2	98	1.64%	80.33%

Catatan: Fitur *Login* dengan Apple ID dieksklusi dari cakupan pengujian karena pengujian dikhususkan pada perangkat bersistem operasi Android.

Berdasarkan aturan Pareto, batas akumulasi persentase kesalahan fungsional sebesar 80% menetapkan 14 fitur tersebut sebagai ruang lingkup fungsional prioritas yang harus diuji secara menyeluruh . Analisis kebutuhan pengujian (*requirement analysis*) dilakukan dengan memetakan kebutuhan spesifik untuk peran Murid dan Guru pada masing-masing fitur prioritas berdasarkan spesifikasi teknis dalam dokumen PRD.

TABEL 3
(Requirement Analysis)

No	Fitur	Kebutuhan Pengujian	
		Murid	Guru
1.	Registrasi	Pendaftaran via Google; Penanganan nomor ponsel kosong/terdaftar; Validasi kartu (<16 digit); Validasi PIN (<6 digit)	Pendaftaran via Google; Penanganan nomor ponsel kosong/terdaftar; Validasi kartu (<16 digit); Validasi PIN (<6 digit)
2.	*Login* Google	*Autentikasi* akun Murid via SSO Google	*Autentikasi* akun Guru; Pembatalan *login* Google
3.	*Login* Kartu	Masuk menggunakan nomor kartu, Validasi pesan kesalahan saat entri PIN salah	Masuk menggunakan nomor kartu, Validasi pesan kesalahan saat entri PIN salah
4.	Isi Kartu	Pengisian saldo via VA, QRIS, Transfer Bank, pengisian < 15.000	P Pengisian saldo via VA, QRIS, Transfer Bank, pengisian < 15.000
5.	Scan QR	Pemindaian presensi harian masuk/keluar, Penanganan ganda, Pemindaian di luar jadwal	Pemindaian presensi harian masuk/keluar Penanganan ganda, Pemindaian di luar jadwal
6.	Reset PIN	Pemulihan kode via OTP Email/WhatsApp, Pembatalan proses	Pemulihan kode via OTP Email/WhatsApp, Pembatalan proses
7.	Perizinan	Melihat daftar izin, Filter bulanan, Detail izin, Pembatalan pengajuan, Ubah jadwal	Input izin baru dengan/tanpa data wajib, Perubahan jadwal, Manajemen izin murid
8.	Tugas	Pemantauan tugas aktif/selesai, Filter mapel, Pengiriman/pengeditan berkas, Melihat nilai	Pembuatan tugas baru, Pengeditan tugas, Penghapusan tugas, Penilaian pekerjaan murid
9.	Rekap Keuangan	Menampilkan grafik finansial, Filter bulanan, Detail mutasi masuk/keluar	Menampilkan grafik finansial, Filter bulanan, Detail mutasi masuk/keluar
10.	Rekap Presensi	Riwayat kehadiran, Filter bulanan, Presensi pelajaran dan kegiatan beserta filter temporal	Riwayat kehadiran, Filter bulanan, Presensi mengajar guru dan presensi kegiatan
11.	Jadwal Kegiatan	Akses kalender jadwal kegiatan, Menampilkan detail rincian kegiatan	Akses kalender jadwal kegiatan, Menampilkan detail rincian kegiatan
12.	*List* Notifikasi	Akses riwayat notifikasi, Tandai telah dibaca, Penghapusan seluruh notifikasi	Akses riwayat notifikasi, Tandai telah dibaca, Penghapusan seluruh notifikasi
13.	Pengumuman	Daftar pengumuman, Detail informasi, Pencarian judul pengumuman	Daftar pengumuman, Detail informasi, Pencarian judul pengumuman
14.	Riwayat	Pelacakan kronologis transaksi, Detail mutasi, Filter riwayat bulanan	Pelacakan kronologis transaksi, Detail mutasi, Filter riwayat bulanan

Tahap perancangan kasus uji menghasilkan total 128 skenario pengujian fungsional yang mencakup alur penggunaan normal (*positive flow*) maupun skenario pengecualian (*negative flow*).

TABEL 4
(Test Case)

No.	Name	Status	*Priority*	*Type*	Step	*Expected Result*
1.	Registrasi Akun Murid via SSO Google.	Pass	Normal	Positive	Buka app – Klik SSO Google – Pilih akun – Isi *form* – Klik lanjutkan	Berhasil registrasi, menampilkan halaman selesai pendaftaran kemudian diarahkan ke halaman utama
2.	Registrasi Murid via Google dengan Nomor Ponsel Terdaftar	Pass	Normal	Negative	Lakukan step 1 – Masukkan nomor sudah terdaftar – Klik lanjutkan	Berhasil terdeteksi dan menampilkan kesalahan kalau nomor sudah di gunakan
3.	...	...	...	...	...	...
128.	Guru Memfilter Riwayat Transaksi Bulanan	Pass	Normal	Positive	Bukan riwayat transaksi – Klik penyaring filter – pilih rentang waktu tertentu – Klik terapkan	Berhasil menyaring dan menyajikan daftar sesuai dengan rentang waktu yang di pilih

B. Lingkungan Pengujian dan Eksekusi Pengujian

Persiapan lingkungan pengujian (*environment setup*) dilakukan untuk memastikan keselarasan operasional antara

pengetesan manual dan otomatis . Pengujian fisik menggunakan real *device smartphone* OPPO Reno 15F yang dihubungkan melalui koneksi USB ke laptop pengetes . Aplikasi Vysor digunakan untuk mengendalikan antarmuka ponsel langsung dari laptop tanpa perlu melakukan manipulasi fisik berulang .

Untuk pengujian manual, 128 skenario pengujian diimpor secara massal dari format Excel (CSV) ke platform Jira yang telah ditambahkan ekstensi Zephyr . Kasus uji dikelompokkan ke dalam direktori berdasarkan fitur dan peran pengguna . Pengetes mengeksekusi setiap langkah pengujian secara berurutan, mendokumentasikan hasil aktual, memperbarui status skenario (*passed* atau *failed*), serta melampirkan bukti tangkapan layar langsung di dalam modul *Test Cycles* Zephyr .

Untuk pengujian otomatis, lingkungan pengembangan dikonfigurasi melalui integrasi Katalon Studio v10 sebagai mesin utama penulisan skrip pengujian berbasis Groovy , Node.js sebagai lingkungan eksekusi server Appium , dan Android SDK *Platform Tools* untuk memfungsikan perintah ADB (*Android Debug Bridge*) agar laptop dapat mendeteksi koneksi fisik ponsel pintar . Eksekusi otomatisasi dilakukan dengan merekam elemen antarmuka (*object spying*) aplikasi *Cards* Edu ke dalam *Object Repository* Katalon Studio, kemudian skrip pengetesan disusun dalam bentuk *Test Case* Groovy dan dijalankan secara berurutan .

TABEL 5
(Pengujian Fitur Registrasi – Murid (Manual))

No	*Name Test Case*	Status	Keterangan
1.	Registrasi Akun Murid via SSO Google	Pass	Berhasil menampilkan halaman pendaftaran selesai
2.	Registrasi Akun Murid via Google dengan Nomor Ponsel Terdaftar	Pass	Berhasil menampilkan pesan *error* nomor ponsel sudah digunakan
3.	Registrasi Akun Murid via Google tanpa Nomor Ponsel	Pass	Berhasil menampilkan pesan *error* nomor ponsel wajib diisi
4.	Registrasi Akun Murid via Google tanpa Nomor Kartu	Fail	Gagal menampilkan pesan *error* nomor kartu wajib diisi
5.	Registrasi Akun Murid via Google dengan Nomor Kartu < 16 Digit	Pass	Berhasil menampilkan pesan nomor kartu harus 16 digit
6.	Registrasi Akun Murid via Google dengan Nomor Kartu Terdaftar	Fail	Gagal menampilkan pesan *error* dalam Bahasa Indonesia
7.	Registrasi Akun Murid via Google PIN kurang dari 6 digit	Pass	Berhasil menampilkan pesan *error* PIN harus 6 digit
8.	Registrasi Akun Murid via Google dengan PIN Tidak Sesuai	Fail	Gagal menampilkan pesan *error* dalam Bahasa Indonesia

TABEL 6
(Pengujian Fitur Registrasi – Guru (Manual))

No	*Name Test Case*	Status	Keterangan
1.	Registrasi Akun Guru via SSO Google	Pass	Berhasil menampilkan halaman pendaftaran selesai
2.	Registrasi Akun Guru via Google dengan Nomor Ponsel Terdaftar	Pass	Berhasil menampilkan pesan *error* nomor ponsel sudah digunakan
3.	Registrasi Akun Guru via Google tanpa Nomor Ponsel	Pass	Berhasil menampilkan pesan *error* nomor ponsel wajib diisi
4.	Registrasi Akun Guru via Google tanpa Nomor Kartu	Fail	Gagal menampilkan pesan *error* nomor kartu wajib diisi
5.	Registrasi Akun Guru via Google dengan Nomor Kartu < 16 Digit	Pass	Berhasil menampilkan pesan nomor kartu harus 16 digit .
6.	Registrasi Akun Guru via Google dengan Nomor Kartu Terdaftar	Fail	Gagal menampilkan pesan *error* dalam Bahasa Indonesia
7.	Registrasi Akun Guru via Google PIN kurang dari 6 digit	Pass	Berhasil menampilkan pesan *error* PIN harus 6 digit
8.	Registrasi Akun Guru via Google dengan PIN Tidak Sesuai	Fail	Gagal menampilkan pesan *error* dalam Bahasa Indonesia

C. Analisis Perbandingan Metrik Pengujian

Berdasarkan hasil pengujian manual dan otomatis yang telah dilakukan terhadap 128 *test case*, diperoleh beberapa metrik yang digunakan untuk membandingkan efektivitas kedua metode pengujian. Perbandingan tersebut disajikan pada Tabel 7.

TABEL 7
(Perbandingan Pengujian Manual dan Otomatis)

Parameter Evaluasi	Manual	Otomatis
Jumlah *Test Case*	128	128
Jumlah *Test Case* berhasil dieksekusi	128	128
Test Coverage	100%	100%
Test case berhasil	120	115
Test case gagal	8	14
Jumlah ditemukan bug	8	14
Test Case Effectiveness	6,25%	10,94%
Execution Time	63 menit 00 detik	56 menit 47 detik
Efisiensi Waktu	1,00x (Acuan)	1,11x (Lebih Cepat)

Data hasil pengujian membuktikan bahwa secara umum pengujian otomatis menggunakan Katalon Studio lebih efisien dan efektif dibandingkan pengujian manual . Dari dimensi efisiensi temporal, pengujian otomatis mampu menyelesaikan seluruh eksekusi kasus uji dalam durasi 56 menit 47 detik, atau sekitar 1,11 kali lebih cepat dibandingkan pengujian manual yang membutuhkan waktu 63 menit . Dari dimensi efektivitas deteksi kesalahan, metode otomatis mencatatkan persentase pencapaian sebesar 10,94% dengan mendeteksi 14 skenario gagal, sedangkan metode manual hanya berhasil mendeteksi 8 skenario gagal dengan tingkat efektivitas 6,25% . Perbedaan hasil ini memerlukan analisis mendalam terkait faktor kausalitas yang mempengaruhinya .

D. Analisis Penemuan Bug

Bug yang ditemukan selama pengujian didominasi oleh ketidaksesuaian pesan kesalahan dan kendala pada proses integrasi layanan pihak ketiga. Perbedaan jumlah bug yang teridentifikasi antara pengujian manual dan pengujian otomatis tidak secara langsung menunjukkan bahwa salah satu metode lebih unggul. Perbedaan tersebut dipengaruhi oleh karakteristik masing-masing metode dalam mendeteksi dan mengevaluasi kesalahan pada sistem.

Pada fitur Registrasi dan Login melalui Nomor Kartu, ditemukan ketidaksesuaian bahasa pada pesan kesalahan yang ditampilkan ketika pengguna memasukkan kredensial yang tidak valid. Sistem menampilkan pesan dalam Bahasa Inggris, seperti *"Incorrect card PIN"*, sementara standar yang digunakan mengharuskan penggunaan Bahasa Indonesia. Pada pengujian manual, kondisi ini tidak dikategorikan sebagai bug karena penguji berfokus pada fungsi utama sistem, yaitu menolak akses dengan kredensial yang salah. Sebaliknya, pengujian otomatis mendeteksi kondisi tersebut sebagai kegagalan karena proses verifikasi dilakukan berdasarkan kesesuaian hasil aktual dengan hasil yang diharapkan. Temuan ini menunjukkan adanya inkonsistensi dalam pengelolaan pesan sistem dan pelokalan bahasa pada aplikasi.

Pada fitur Isi Kartu, ditemukan perbedaan hasil antara pengujian manual dan pengujian otomatis. Seluruh skenario pada pengujian manual berhasil dijalankan, sedangkan beberapa skenario pada pengujian otomatis mengalami kegagalan. Hasil analisis menunjukkan bahwa kondisi tersebut berkaitan dengan integrasi layanan pembayaran pihak ketiga yang menggunakan antarmuka dinamis. Perubahan elemen antarmuka serta waktu muat halaman yang bervariasi menyebabkan alat otomatisasi mengalami kesulitan dalam mengidentifikasi elemen yang diperlukan selama proses pengujian. Akibatnya, beberapa skenario dihentikan sebelum transaksi selesai diproses. Sementara itu, penguji manual dapat menyesuaikan tindakan berdasarkan kondisi antarmuka yang muncul sehingga proses transaksi tetap dapat diselesaikan.

Secara keseluruhan, temuan bug menunjukkan bahwa sistem masih perlu ditingkatkan pada aspek konsistensi antarmuka dan stabilitas integrasi layanan eksternal. Selain itu, hasil pengujian menunjukkan bahwa pengujian otomatis lebih efektif dalam mendeteksi ketidaksesuaian yang bersifat detail dan terukur, sedangkan pengujian manual lebih fleksibel dalam menghadapi perubahan kondisi yang terjadi selama proses pengujian.

E. Rekomendasi Hasil Perbaikan

Berdasarkan hasil pengujian dan analisis yang telah dilakukan, terdapat beberapa rekomendasi yang dapat diberikan kepada pengembang sebagai upaya peningkatan kualitas fungsionalitas sistem, yaitu:

1. Perbaikan mekanisme validasi *input*: Sistem perlu memastikan bahwa setiap *input* pengguna tervalidasi dengan baik serta menampilkan pesan yang jelas dan konsisten agar pengguna dapat memahami kesalahan yang terjadi
2. Perbaikan konsistensi bahasa pada sistem: Pesan *error* dan notifikasi perlu disesuaikan dengan bahasa yang digunakan dalam aplikasi agar lebih konsisten dan mudah dipahami oleh pengguna.
3. Peningkatan penanganan *error (error handling):* Perlu dilakukan perbaikan pada mekanisme penanganan *error* untuk mencegah terjadinya *crash* atau *force close,* sehingga aplikasi dapat berjalan lebih stabil.
4. Peningkatan stabilitas fitur yang bermasalah: Fitur yang masih mengalami kegagalan, seperti isi kartu, perlu diperbaiki pada aspek logika dan pengelolaan *resource* agar dapat berjalan dengan optimal.

V. KESIMPULAN

Berdasarkan hasil pengujian fungsional pada aplikasi mobile pendidikan Cards Edu menggunakan kerangka kerja *Software Testing Life Cycle* (STLC), dapat disimpulkan bahwa pengujian manual maupun pengujian otomatis berhasil mengeksekusi seluruh 128 kasus uji yang dirancang dengan tingkat cakupan pengujian *(test coverage)* sebesar 100%.

Pada pengujian manual menggunakan Jira Zephyr, sebanyak 120 skenario dinyatakan berhasil *(passed)* dan 8 skenario dinyatakan gagal *(failed)* dengan waktu eksekusi selama 63 menit. Sementara itu, pengujian otomatis menggunakan Katalon Studio yang terintegrasi dengan Appium dan Node.js menghasilkan 114 skenario berhasil dan 14 skenario gagal dengan waktu eksekusi 56 menit 47 detik.

Hasil pengujian menunjukkan bahwa pengujian otomatis memiliki efektivitas penemuan defect yang lebih tinggi, yaitu sebesar 10,94%, dibandingkan pengujian manual sebesar 6,25%. Selain itu, pengujian otomatis juga menunjukkan efisiensi waktu yang lebih baik dengan rasio kecepatan sekitar 1,11 kali dibandingkan pengujian manual. Perbedaan jumlah *defect* yang ditemukan menunjukkan bahwa pengujian otomatis mampu mendeteksi ketidaksesuaian sistem secara lebih konsisten, terutama pada aspek validasi dan kesesuaian keluaran sistem dengan hasil yang diharapkan.

Berdasarkan rekapitulasi hasil pengujian, *defect* yang ditemukan didominasi oleh ketidaksesuaian pesan kesalahan, inkonsistensi bahasa pada sistem, serta kendala pada fitur yang terintegrasi dengan layanan pihak ketiga. Meskipun demikian, pengujian manual tetap memiliki peran penting karena lebih fleksibel dalam menghadapi kondisi antarmuka yang dinamis dan skenario yang memerlukan penyesuaian selama proses pengujian. Oleh karena itu, kombinasi pengujian manual dan otomatis dapat menjadi pendekatan

yang efektif untuk meningkatkan kualitas dan keandalan aplikasi mobile.

REFERENSI

[1] R. Syerlita dan I. Siagian, "Dampak perkembangan revolusi industri 4.0 terhadap pendidikan di era globalisasi saat ini," *Journal on Education*, vol. 7, no. 1, pp. 3507–3515, 2024.

[2] D. A. Rabbani dan F. U. Najicha, "Pengaruh perkembangan teknologi terhadap kehidupan dan interaksi sosial masyarakat Indonesia," Nov. 2023. [Online]. *Available*: https://www.researchgate.net

[3] A. P. Sari dan Munir, "Pemanfaatan teknologi digital dalam inovasi pembelajaran untuk meningkatkan efektivitas kegiatan di kelas," Teknologi Transformasi Digital (*Digitech*), vol. 4, no. 2, Sep. 2024, doi: 10.47709/digitech.v4i2.5127.

[4] C. J. R. Masinambowal, J. S. J. Lengkong, dan V. N. J. Rotty, "Inovasi digital dalam manajemen sekolah: Meningkatkan kinerja pendidikan di era teknologi," *Academy of Education Journal*, vol. 16, no. 1, pp. 8–17, Jan. 2025.

[5] A. A. Ardiansyah dan Nana, "Peran *mobile learning* sebagai inovasi dalam meningkatkan hasil belajar siswa pada pembelajaran di sekolah," Indonesian *Journal of Educational Research and Review*, vol. 3, no. 1, Apr. 2020.

[6] R. Abdillah, R. Hermawan, W. Hermawansyah, I. Adkha, dan H. Arifin, "Pengujian perangkat lunak sistem informasi inventori pada usaha jasa pengiriman paket," Polygon: Jurnal Ilmu Komputer dan Ilmu Pengetahuan Alam, vol. 2, no. 4, pp. 166–175, Jul. 2024, doi: 10.62383/polygon.v2i4.199.

[7] A. Tkalich, E. Klotins, T. Sporsem, V. Stray, N. B. Moe, dan A. Barbala, "*User feedback in continuous software engineering: Revealing the state-of-practice," Empirical Software Engineering*, vol. 30, art. no. 79, Mar. 2025, doi: 10.1007/s10664-024-10557-2.

[8] A. R. Rambe dan H. Prihantoro, "Pengujian otomatis aplikasi *mobile* dengan teknik *black-box* menggunakan Appium," *Prosiding Automata*, Aug. 21, 2022. [Online]. Available: journal.uii.ac.id

[9] A. Arfan and H. Hendrik, "Penerapan STLC dalam pengujian automation aplikasi mobile (studi kasus: LMS Amikom Center PT. GIT Solution)," Prosiding Automata, Aug. 21, 2022. [Online]. *Available*: journal.uii.ac.id

[10] Mutiara, N. Qomariyah, dan D. K. Widyawati, "Pengujian Otomatis Sistem Informasi Indo Towing dengan Katalon Studio," ROUTERS: Jurnal Sistem dan Teknologi Informasi, vol. 2, no. 1, pp. 35–41, Feb. 2024.

[11] D. Gobov and O. Zuieva, "*Software Quality Attributes in Requirements Engineering," International Journal of Information Technology and Computer Science* (IJITCS), vol. 17, no. 4, pp. 38–48, Aug. 2025

[12] G. Santoso, Pemanfaatan *Mobile Learning* untuk Meningkatkan Sistem Pembelajaran bagi Masyarakat. Indonesia: AKPRIND *Press*, 2018, ISBN: 978-602-7619-51-7.

[13] T. S. Fauzan, "*Mobile Learning* Berbasis Android sebagai Media Pembelajaran PAI di MAN 2 Yogyakarta," Jurnal Tarbiyah Almuslim, vol. 2, no. 2, pp. 121-136, Des. 2024.

[14] U. L. Yuhana, Kualitas Perangkat Lunak: Pendekatan Teoretis dan Praktis. Yogyakarta, Indonesia: Penerbit ANDI, 2025.

[15] M. Darwin, M. R. Mamondol, S. A. Sormin, Y. Nurhayati, H. Tambunan, D. Sylvia, I. M. D. M. Adnyana, B. Prasetiyo, P. Vianitati, dan A. A. Gebang, Metode Penelitian Pendekatan Kuantitatif, T. S. Tambunan, Ed. Bandung, Indonesia: CV. Media Sains Indonesia, Jun. 2021.

[16] A. Jauhari, D. R. Anamisa, dan F. A. Mufarroha, Rekayasa Perangkat Lunak. Malang, Indonesia: Media Nusa *Creative*, 2019.

[17] W. S. A. Banjarnahor *and* J. Sinuraya, Buku Ajar: Praktik Pengujian Perangkat Lunak. Indonesia: FAASLIB Serambi Media, 2025.

[18] S. Mishra, *Navigating Software Engineering: Concepts*, *Practices, and Principles*. India: *DeepMisti Publication*, 2023.

[19] M. A. Saddam, Perancangan dan Pengembangan Sistem Informasi Berbasis *Scrum*. Yogyakarta, Indonesia: *Deepublish* Digital, 2024.

[20] M. Sunandi, D. Zulfangga, D. Ardiansah, M. R. Shiddiq, S. Rahmah, and S. A. Atmaja, "Penerapan Black Box Testing Pada *Website Cobalt.Tools* untuk Pengujian Fungsionalitas Pengunduhan Media," PCTIF, vol. 3, no. 1, pp. 108–114, Jun. 2025.

[21] M. Juansen, R. G. Alam, and Prihandoko, Pengujian dan Implementasi Sistem Informasi. Yogyakarta, Indonesia: Deepublish Digital, 2024.

[22] M. A. Hidayatullah, K. A. N. Mangkunegara Putra, dan M. 'Ainul Yaqin, "Evaluasi Alat Pendukung Pada Manajemen Proyek Perangkat Lunak: Untuk Pengembangan Perangkat Lunak", AnoaTIK, vol. 2, no. 1, hlm. 60–70, Jun 2024.

[23] Y. A. Ramadhini, Improvisasi *Task* pada *Software* Manajemen Proyek Taiga di PT Javan Cipta Solusi, Skripsi, Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia, 2022.

[24] A. R. R. N. Chandrika, "*Test Case Management in Agile Software Development Using Jira*," *International Journal for Multidisciplinary Research* (IJFMR), vol. 4, no. 4, Jul.–Aug. 2024.

[25] A. Bauer, J. Frattini, and E. Alégroth, "*Augmented* Testing To *Support Manual* GUI-*Based Regression* Testing: An *Empirical* Study," *Empirical Software Engineering*, vol. 29, no. 6, Art. 140, 2024.

[26] D. S. Prasetyo *and* W. Silfianti, "Analisis Perbandingan Pengujian Manual dan *Automation Testing* pada *Website* E-*Commerce*," Jurnal Ilmiah Teknologi Informasi (JUIT), vol. 2, no. 2, pp. 127–131, Mei 2023.

[27] M. A. L. Bhanuwa and M. Jannah, "Analisis perbandingan *automation* testing Katalon Studio dan Selenium IDE pada *website e-commerce* menggunakan metode *end-to-end* (studi kasus: Enterkomputer.com)," Positif: Jurnal Sistem dan Teknologi Informasi, 202