DIGITALISASI PROSES PENGUJIAN APLIKASI BERBASIS ANALISIS *LOG ERROR* DENGAN METODE *REGRESSION TESTING* DI PT DATACARAKA SOLUSINDO

Rustama Banjarnahor, Yogi Bachtiar, Achmad Sarwandianto
Teknik Informatika, Universitas Indraprasta PGRI
Jl. Raya Tengah No.80, RT.6/RW.1, Jakarta Timur
rustama.banjarnahor@gmail.com

ABSTRAK

Digitalisasi dalam pengujian aplikasi menjadi kebutuhan perusahaan perangkat lunak untuk meningkatkan kulaitas aplikasi yang dikembangkan. PT Datacaraka Solusindo masih menggunakan metode manual dalam dokumentasi pengujian, yang menyebabkan ketidakefisienan dalam proses perbaikan, kesulitan kolaborasi antar tim penguji dan programmer, serta kurang optimalnya pemantauan oleh project manager. Penelitian ini bertujuan untuk mengembangkan sistem digitalisasi proses pengujian aplikasi berbasis analisis *log error* dengan metode *regression testing.* Sistem ini dirancang untuk mempercepat proses perbaikan dari *log error*, meningkatkan efektivitas komunikasi antar tim, serta pemantauan yang lebih baik bagi project manager. Metode yang digunakan dalam penelitian ini adalah *Regression Testing*, yang berfungsi untuk menguji ulang perbaikan *log error* guna memastikan bahwa perubahan yang dilakukan tidak mempengaruhi fungsi lain dalam aplikasi. Implementasi sistem ini dilakukan dengan pendekatan berbasis analisis *log error* yang memungkinkan deteksi kesalahan lebih cepat dan akurat. Hasil penelitian ini menunjukan bahwa digitalisasi pengujian yang dikembangkan dapat meningkatkan efisiensi dalam dokumentasi, mempercepat proses perbaikan *log error*, serta memudahkan project manager dalam memantau perkembangan pengujian aplikasi secara *real-time*. Dengan sistem ini, PT. Datacaraka Solusindo dapat meningkatkan kualitas aplikasi yang dikembangkan serta mempercepat kolaborasi antara tim penguji dan programmer.

Kata kunci: Digitalisasi, Pengujian Aplikasi, *Log Error, Regression Testing.*

1. PENDAHULUAN

Di era digital, perusahaan seperti PT Datacaraka Solusindo menghadapi tantangan dalam pencatatan dan pelacakan log kesalahan selama proses pengembangan aplikasi perangkat lunak. Tahap pengujian sangat penting untuk memastikan bahwa aplikasi yang dikembangkan berjalan sesuai spesifikasi dan bebas dari kesalahan yang dapat mengganggu pengguna akhir. Saat ini, dokumentasi proses pengujian dan log kesalahan dilakukan secara manual menggunakan Microsoft Word dan Excel, yang menyebabkan inefisiensi dan menghambat kolaborasi antar tim. Kemampuan perusahaan dalam mengidentifikasi dan memperbaiki kesalahan dalam aplikasi yang diuji, sehingga meningkatkan kualitasnya. Proyek ini mencakup pencatatan pengujian aplikasi secara komprehensif, termasuk pengujian regresi, yang memastikan bahwa perubahan yang dilakukan oleh programmer terhadap kesalahan yang diperbaiki berfungsi dengan baik dan tidak menyebabkan kesalahan baru pada fitur aplikasi yang ada.

Masalah yang teridentifikasi meliputi dokumentasi pengujian aplikasi yang tidak memadai, kolaborasi yang tidak efisien antara tim pengujian dan pemrograman, penentuan waktu pengujian regresi secara manual, dan kesulitan memantau dan mengevaluasi proses pengujian aplikasi. Untuk membatasi cakupan permasalahan, penulis menerapkan batasan permasalahan, yaitu meliputi penelitian yang dikembangkan untuk merekam atau mendokumentasikan skenario pengujian, log kesalahan, dan pengujian regresi pada aplikasi, log kesalahan prioritas yang dilakukan dengan empat nilai inti, dan sistem yang dibangun menggunakan bahasa pemrograman PHP dan database mySQL.

Sistem dapat secara otomatis menentukan batas waktu pengujian regresi log kesalahan sesuai prioritas, sehingga kualitas aplikasi tetap konsisten dengan standar yang ditetapkan. Penelitian ini bertujuan untuk mendigitalkan proses pengujian aplikasi berdasarkan analisis log kesalahan dengan merancang sistem untuk dokumentasi, perekaman skenario pengujian, log kesalahan, dan pengujian regresi. Sistem harus dapat diakses secara *real-time* oleh tim programmer, mengoptimalkan waktu pengujian regresi dari perbaikan log kesalahan, dan meningkatkan efektivitas pemantauan untuk pemantauan pengembangan aplikasi secara *real-time*.

2. TINJAUAN PUSTAKA

2.1. Analisis *Log Error*

Analisis *log error* adalah proses untuk memeriksa, mengidentifikasi, dan menganalisis pesan kesalahan (*error messages*) yang tercatat dalam log file aplikasi atau sistem. Tujuan utama dari analisis ini adalah untuk menemukan pola atau tren dari kesalahan yang terjadi, yang dapat memberikan wawasan mengenai penyebab masalah dalam sistem dan memberikan petunjuk untuk perbaikan. *Log error* biasanya berisi informasi tentang jenis kesalahan, waktu kejadian, serta konteks di mana kesalahan terjadi, yang membantu pengembang dalam melakukan debugging dan memperbaiki perangkat

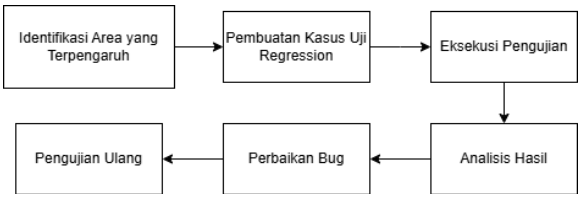
lunak [1]. Analisis *log error* tidak hanya membantu memperbaiki masalah yang terjadi secara langsung, tetapi juga memberikan umpan balik untuk perbaikan berkelanjutan terhadap ketahanan sistem. Hal ini menegaskan pentingnya penggunaan *log error* untuk memperbaiki kesalahan teknis serta meningkatkan ketahanan sistem secara [2].

2.2. Regression Testing

Regression testing merupakan salah satu jenis pada pengujian *black box* yang bertujuan untuk memverifikasi software yang sebelumnya telah dilakukan perubahan atau pengembangan guna meminimalisir kemungkinan kesalahan yang terjadi pada software yang telah dimodifikasi [3].

Regression testing dapat menguji fungsi program secara fungsional dan nonfungsional. Biasanya, regression testing menggunakan kasus uji yang ditulis pada tahap awal pengembangan suatu pengujian [4]. Hal tersebut untuk memastikan bahwa perubahan pada versi baru program tidak merusak fungsi yang sudah ada [5]. *Regression testing* adalah jenis pengujian yang bertujuan untuk memverifikasi perubahan yang dibuat terhadap aplikasi atau perangkat lunak (*bug fixes*, *code merges*, migrasi sistem, database, web server atau aplikasi server) untuk mengkonfirmasi bahwa fungsi program secara fungsional yang sudah ada masih dapat berfungsi [6].

Dengan adanya *regression testing* untuk memastikan bahwa perubahan pada tidak menimbulkan defect baru [7]. Alur dari *regression testing* dapat di tunjukan dalam gambar 1.



Gambar 1. Alur *Regression Testing*

2.2.1. *Regresssion Testing Manually*

Regression testing manually adalah proses pengujian yang dilakukan oleh manusia (tester) untuk memverifikasi atau memastikan perubahan yang dilakukan tidak merusak fungsionalitas yang sudah ada sebelumnya. Pengujian ini dilakukan untuk mengevaluasi aplikasi dari setiap bagian-bagian yang sudah berjalan dengan baik tetap berfungsi setelah ada perubahan. Manual regression testing sering digunakan dalam skenario di mana pengujian otomatis yang memerlukan interaksi langsung pengguna untuk mengevaluasi pengalaman mereka [8].

Tujuan dari *regression testing manually* adalah:

a. Memastikan fungsionalitas tidak terganggu. Dengan memastikan ketika ada perubahan yang dilakukan seperti bug fixing atau penambahan feature baru tidak mengganggu atau tidak merusak fungsi-fungsi yang lain, terutama Ketika aplikasi antarmuka pengguna yang [9].
b. Menemukan bug tidak terduga. Dengan memeriksa setiap fungsi-fungsi, terkadang perubahan tersebut dapat menimbulkan bug [10].
c. Pengujian interaksi pengguna. Pengguna dapat menilai pengalaman mereka ketika menggunakan aplikasi [11].

2.2.2. *Regresssion Testing Automation*

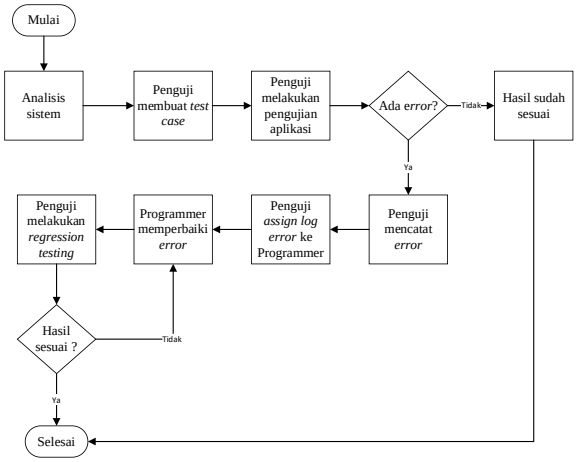
Regression testing automation adalah yang dilakukan oleh sistem dengan bantuan *tools* atau framework. Testing ini dilakukan memiliki kemampuan secara teknis, karena dalam proses ini penguji akan menulis *script* atau kode untuk menjalankannya [12]. Penggunaan regression testing automation semakin populer dalam industri perangkat lunak karena dapat mengurangi waktu pengujian dan meningkatkan cakupan pengujian [13].

Alat pendukung untuk melakukan *regression testing* secara otomatis sebagai berikut:

a. Selenium
   Dalam dokumentasi selenium, selenium adalah seperangkat alat untuk otomatisasi peramban web yang menggunakan teknik terbaik yang tersedia untuk mengontrol *instance* peramban secara jarak jauh dan mensimulasikan interaksi pengguna dengan peramban. Selenium memungkinkan pengguna untuk mensimulasikan aktivitas umum yang dilakukan oleh pengguna akhir, seperti memasukkan teks ke dalam kolom, memilih nilai dari daftar *drop-down*, mencentang kotak, dan mengklik tautan dalam dokumen [14].
b. Appium
   Dalam dokumentasi Appium, Appium adalah proyek sumber terbuka dan ekosistem perangkat lunak terkait yang dirancang untuk memfasilitasi otomatisasi antarmuka pengguna (UI) pada berbagai platform aplikasi, termasuk perangkat mobile (iOS, Android). Appium bertujuan untuk mendukung otomatisasi antarmuka pengguna (UI) pada berbagai platform yang berbeda (mobile, web, desktop, dan sebagainya) [15].
c. JUnit
   JUnit adalah sebuah framework pengujian (testing framework) yang digunakan untuk menguji unit-unit terkecil dari kode dalam aplikasi Java. JUnit memungkinkan pengembang untuk menulis dan menjalankan pengujian otomatis pada kode mereka, memastikan bahwa bagian-bagian kecil dari aplikasi (unit) bekerja sesuai harapan dan mengidentifikasi potensi masalah secara dini dalam pengembangan perangkat lunak [16].

3. METODE PENELITIAN

Metode penelitian ini menggunakan *Regression Testing* untuk pengujian ulang terhadap perbaikan *log error* untuk memastikan perubahan tidak mempengaruhi fungsi lain.



Gambar 2. Kerangka Kerja Algoritma Pengujian

a. Mulai
Penguji mulai menentukan teknik yang digunakan dalam pengujian.
b. Analisis sistem
Proses analisis cara kerja sistem dan proses analisa terhadap:
- Input apa yang akan digunakan
- Proses apa yang terjadi
- Output yang diharapkan

c. Penguji membuat *test case*
Penguji membuat skenario test (*test case*) yang akan digunakan untuk menguji aplikasi. Kasus uji ini berfungsi untuk memverifikasi apakah setiap fitur berfungsi sesuai dengan hasil yang diharapkan.
d. Penguji melakukan pengujian aplikasi
Setelah pembuatan *test case* selesai, penguji melakukan pengujian pada aplikasi yang dikembangkan dengan menggunakan test case yang sudah dibuat sebelumnya untuk mendeteksi kesalahan atau *error* pada aplikasi. Dalam proses pengujian, jika penguji menemukan kesalahan (*error*) maka penguji melanjutkan pencatatan *log error*. Jika dalam proses pengujian tidak menemukan adanya *error*, maka hasil *test case* yang diharapkan sudah sesuai.
e. Penguji mencatat *error*
Jika dalam pengujian ditemukan *error*, maka penguji akan mencatat detail kesalahan serta menganalisis kategori dari *error* yang ditemukan.
f. Penguji *assign log error* ke programmer
Setelah pencatatan *error* dilakukan, penguji assign *log error* ke programmer yang akan memperbaiki *error* tersebut. Saat *log error* sudah di *assign* ke programmer, secara otomatis akan muncul tenggat waktu untuk *regression testing* yang disesuaikan dengan klasifikasi prioritas.
g. Penguji melakukan *Regression Testing*
Setelah programmer memperbaiki error, penguji melakukan *regression testing* untuk memastikan bahwa perbaikan tidak menyebabkan masalah baru pada aplikasi. Setelah *regression testing*, dilakukan pengecekan apakah hasil pengujian sudah sesuai, jika regression testing tidak menemukan kesalahan, maka hasil yang diharapkan sudah sesuai dengan *test case*. Jika hasil *regression testing* menemukan *error*, proses akan kembali ke programmer untuk memperbaiki *error*, dan programmer melakukan perbaikan ulang kesalahan yang ditemukan dari hasil *regression testing*.
h. Selesai
Semua hasil pengujian *test case* dan perbaikan *log error* sudah sesuai dengan hasil yang diharapkan, maka proses pengujian dinyatakan selesai.

4. HASIL DAN PEMBAHASAN

4.1. Jenis Kategori *Log Error*

Kategori *log error* terbagi menjadi menjadi 4 kategori yaitu:

a. Fatal
Kesalahan yang menyebabkan seatu fungsi sistem atau aplikasi tidak dapat melanjutkan proses selanjutnya, sehingga sistem tidak dapat digunakan. Kategori ini termasuk kategori kritis karna dapat menghambat proses utama dari sistem.
b. Major
Kesalahan yang mempengaruhi fungsi utama sistem tetapi tidak sampai membuat sistem berhenti sepenuhnya. Dampak dari kesalahan ini serius, namun sistem masih dapat digunakan
c. Minor
Kesalahan yang mempengaruhi fungsi sekunder atau tambahan, tetapi terdampak langsung pada fungsi utama aplikasi. Dampak dari kesalahan ini tidak menggangu pengguna secara signifikan.
d. Kosmetik
Kesalahan yang hanya berkaitan dengan tampilan atau estetika tanpa mempengaruhi fungsi aplikasi. Dampak tidak kritis, namun dapat mempengaruhi profesionalitas user pengguna.

4.2. Klasifikasi *Priority*

Klasifikasi prioritas perbaikan log kesalahan dan pengujian regresi ditentukan oleh kebutuhan perusahaan dan kualitas aplikasi. Klasifikasi *priority* digunakan untuk menentukan Tingkat urgensi perbaikan dari suatu *log error* dalam aplikasi.

Tabel 1. Daftar Waktu Penyelesaian *Log Error*

No	Priority	Waktu Penyelesaian (Hari)
1	High	1
2	Medium	2
3	Normal	3
4	Low	4

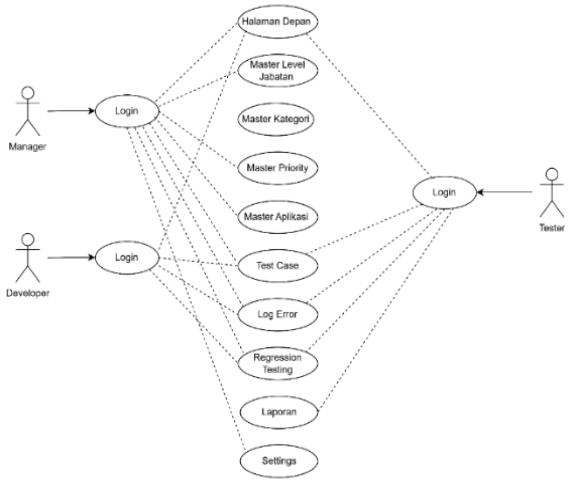
a. *High*
Kesalahan yang menggangu proses alur utama pada aplikasi dan harus segera untuk diperbaiki dan di uji ulang.
b. *Medium*
Kesalahan yang berdampak pada beberapa fungsi aplikasi tidak dapat digunakan, perbaikan dilakukan setelah prioritas high selesai dan pengujian ulang yang tidak mendesak.

c. Normal
Kesalahan yang memiliki dampak kecil pada pengguna dan tidak mempengaruhi fungsi utama aplikasi. Perbaikan dan pengujian ulang perlu dilakukan setelah prioritas medium.
d. *Low*
Kesalahan yang sangat kecil yang tidak memiliki dampak nyata pada fungsi aplikasi. Perbaikan dan pengujian ulang dapat dilakukan dalam jangka waktu yang cukup lama.

4.3. Pemodelan Perangkat Lunak

4.3.1. *Use Case* Diagram

Diagram *use case* menggambarkan interaksi antara aktor dan sistem, sedangkan diagram *activity* menggambarkan interaksi antara aktor dan sistem.



Gambar 3. *Use Case* Diagram

Pada *use case* diagram terdapat 3 aktor sebagai berikut:

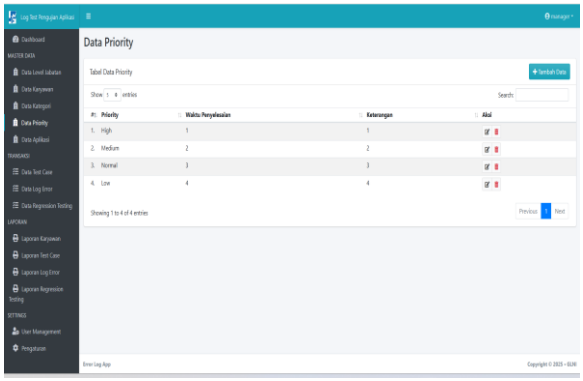
a. Manager
Manager sebagai aktor yang dapat akses semua menu, mengelola data master (master level jabatan, master karyawan, master kategori, master *priority*, master aplikasi), akses menu transaksi *test Case*, *log error*, *regression testing*, laporan dan kelola *settings*.
b. Tester
Tester sebagai aktor yang dapat akses serta mengelola menu transaksi *test case*, *log error*, *regression testing* dan laporan.
c. Developer
Developer sebagai aktor yang dapat akses menu transaksi *test case*, kelola *log error* dan *regression testing*.

4.4. Hasil Penelitian

Rincian berikut mewakili output yang dihasilkan oleh tampilan aplikasi yang dibangun:

4.4.1. Tampilan Menu Data *Priority*

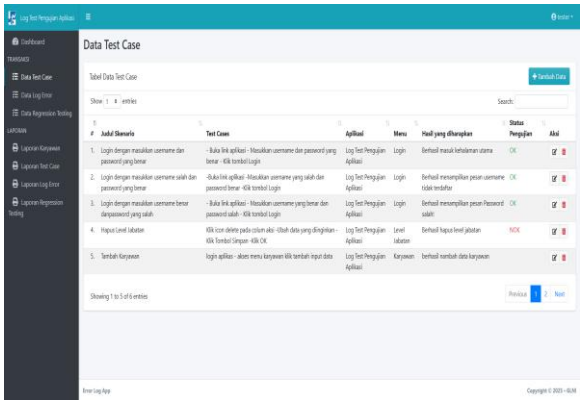
Pada halaman Data *Priority*, sistem akan menampilkan data waktu penyelesaian perbaikan *log error* sesuai *priority*. Pada halaman *priority* terdapat fungsi untuk menambah data, mengubah, menghapus dan mencari data *priority*.



Gambar 4 . Tampilan Layar Data *Priority*

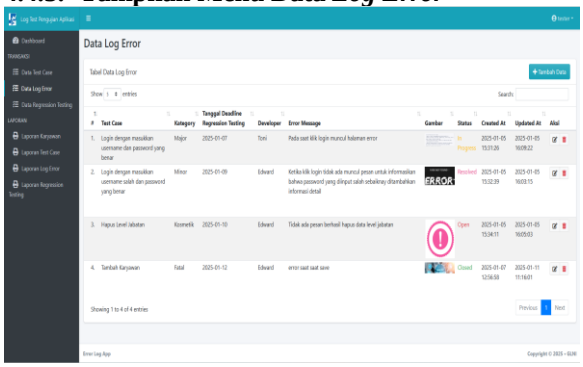
4.4.2. Tampilan Menu Data *Test Case*

Pada halaman Data *Test Case*, sistem akan menampilkan data scenario test (*test case*) pengujian dari aplikasi. Pada halaman *test case* terdapat fungsi untuk menambah data, mengubah, menghapus dan mencari data *test case*.



Gambar 5. Tampilan Layar Data *Test Case*

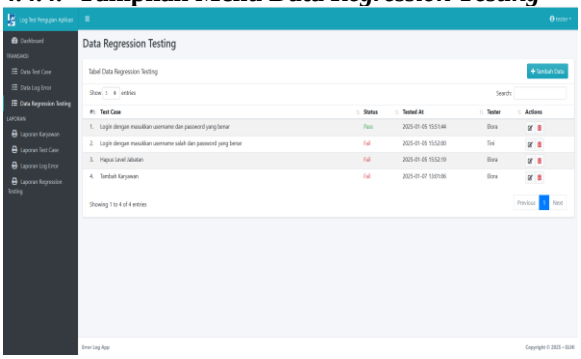
4.4.3. Tampilan Menu Data *Log Error*



Gambar 6. Tampilan Layar Data *Log Error*

Pada halaman Data *Log Error*, sistem akan menampilkan data *log error* dari pengujian aplikasi yang sudah dilakukan. Pada halaman *log error* terdapat fungsi untuk menambah data, mengubah, menghapus dan mencari data *log error*.

4.4.4. Tampilan Menu Data *Regression Testing*



Gambar 7. Tampilan Layar Data *Regression Testing*

Pada halaman Data *Regression Testing*, sistem akan menampilkan data regresi yang sudah dilakukan terhadap perbaikan *log error*. Pada halaman *regression testing* terdapat fungsi untuk menambah data, mengubah, menghapus dan mencari data *regression testing*.

4.5. Pengujian dengan *Black Box Testing*

Sistem yang dikembangkan menjalani evaluasi melalui pendekatan *black-box testing*, yang mengharuskan penilaian sistem hanya berdasarkan masukan dan keluarannya, tanpa menyelidiki arsitektur internalnya. Temuan dari pengujian sistem diuraikan di bawah ini:

Tabel 2. Pengujian *black box testing* menu data *priority*

No	Fungsi	*Test Cases*	Hasil yang diharapkan	Status Pengujian
1	Tambah data klasifikasi *priority*	• Klik menu Data Klasifikasi *Priority* • Klik tombol Tambah Data • Input data yang diinginkan • Klik tombol Simpan	• Berhasil masuk kehalaman data klasifikasi *priority* • Berhasil menampilkan *form* tambah klasifikasi p*riority* • Berhasil menambahkan data klasifikasi *priority*	Sesuai harapan
2	Ubah data klasifikasi *priority*	• Klik icon *edit* pada colum aksi • Ubah data yang diinginkan • Klik tombol Simpan, untuk menyimpan perubahan	• Berhasil menampilkan *form* ubah • Berhasil *update* data klasifikasi *priority*	Sesuai harapan
3	Hapus data klasifikasi *priority* yang belum digunakan	• Klik icon *delete* pada colum aksi • Pada *form* konfirmasi hapus, klik OK untuk melanjutkan penghapusan data *priority*	• Berhasil menampilkan *form* konfirmasi hapus • Berhasil hapus data kalsifikasi *priority*	Sesuai harapan
4	Hapus data klasifikasi *priority* yang sudah digunakan	• Klik icon *delete* pada colum aksi • Pada *form* konfirmasi klik OK untuk melanjutkan penghapusan data *priority*	• Berhasil menampilkan *form* konfirmasi hapus • Muncul pesan “Data tidak dapat dihapus karena masih digunakan”	Sesuai harapan
5	*Search* data klasifikasi *priority*	• Input data yang akan dicari pada field *column search* • Klik enter	Berhasil menampilkan data klasifikasi *priority* yang dicari, jika data tersedia	Sesuai harapan

Tabel 3. Pengujian *black box testing* menu data *test case*

No	Fungsi	*Test Cases*	Hasil yang diharapkan	Status Pengujian
1	Tambah data *test case*	• Klik menu Data *Test Case* • Klik tombol Tambah Data • Input data yang diinginkan • Klik tombol Simpan	• Berhasil masuk kehalaman data *test case* • Berhasil menampilkan *form* tambah *test case* • Berhasil menambahkan data *test case*	Sesuai harapan
2	Ubah data *test case*	• Klik icon *edit* pada *column* aksi • Ubah data yang diinginkan • Klik tombol Simpan, untuk menyimpan perubahan	• Berhasil menampilkan *form* ubah • Berhasil *update* data *test case*	Sesuai harapan
3	Hapus data *test case* yang belum digunakan	• Klik icon *delete* pada *column* aksi • Pada *form* konfirmasi hapus, klik OK untuk melanjutkan penghapusan data *test case*	• Berhasil menampilkan *form* konfirmasi hapus • Berhasil hapus data *test case*	Sesuai harapan
4	Hapus data *test case* yang sudah digunakan	• Klik icon *delete* pada *column* aksi • Pada *form* konfirmasi hapus, klik OK untuk melanjutkan penghapusan data *test case*	• Berhasil menampilkan *form* konfirmasi hapus • Muncul pesan “Data tidak dapat dihapus karena masih digunakan”	Sesuai harapan

No	Fungsi	*Test Cases*	Hasil yang diharapkan	Status Pengujian
5	*Search* data *test case*	• Input data yang akan dicari pada field *column search* • Klik enter	Berhasil menampilkan data *test case* yang dicari, jika data tersedia	Sesuai harapan

Tabel 4. Pengujian *black box testing* menu data *log error*

No	Fungsi	*Test Cases*	Hasil yang diharapkan	Status Pengujian
1	Tambah data *log error* tanpa *upload* gambar	• Klik menu Data L*og Error* • Klik tombol Tambah Data • Input data yang diinginkan • Klik tombol Simpan	• Berhasil masuk kehalaman data *log error* • Berhasil menampilkan *form* tambah *log error* • Berhasil menambahkan data *log error* • Waktu *deadline regression testing* otomatis terisi sesuai *priority* yang dipilih	Sesuai harapan
2	Tambah data *log error* dengan *upload* gambar	• Klik menu Data L*og Error* • Klik tombol Tambah Data • Input data yang diinginkan • *Upload* gambar *Log Error* • Klik tombol Simpan	• Berhasil masuk kehalaman data *log error* • Berhasil menampilkan *form* tambah *log error* • Berhasil *upload* gambar • Berhasil menambahkan data *log error* • Waktu *deadline regression testing* otomatis terisi sesuai *priority* yang dipilih	Sesuai harapan
3	Ubah data *log error*	• Klik icon *edit* pada *column* aksi • Ubah data yang diinginkan • Klik tombol Simpan, untuk menyimpan perubahan	• Berhasil menampilkan *form* ubah • Berhasil *update* data *log error*	Sesuai harapan
4	Hapus data *log error*	• Klik icon *delete* pada *column* aksi • Pada *form* konfirmasi hapus klik OK untuk melanjutkan penghapusan data *log error*	• Berhasil menampilkan *form* konfirmasi hapus • Berhasil hapus data *log error*	Sesuai harapan
5	*Search* data *log error*	• Input data yang akan dicari pada field *column search* • Klik enter	Berhasil menampilkan data *log error* yang dicari, jika data tersedia	Sesuai harapan

Tabel 5. Pengujian *black box testing* menu data *regression testing*

No	Fungsi	*Test Cases*	Hasil yang diharapkan	Status Pengujian
1	Tambah data *regression testing*	• Klik menu Data Regression Testing • Klik tombol Tambah Data • Isikan data yang diinginkan • Pilih status regresi • Klik tombol Simpan	• Berhasil masuk kehalaman data *regression testing* • Berhasil menampilkan *form* tambah *regression testing* tambah data • Berhasil *regression testing*	Sesuai harapan
2	Ubah data *regression testing*	• Klik icon *edit* pada *column* aksi • Ubah data yang diinginkan • Klik tombol Simpan, untuk menyimpan perubahan	• Berhasil menampilkan *form* ubah • Berhasil *update* data *regression testing*	Sesuai harapan
3	Hapus data *regression testing*	• Klik icon *delete* pada *column* aksi • Pada *form* konfirmasi hapus klik OK untuk melanjutkan penghapusan data *regression testing*	• Berhasil menampilkan *form* konfirmasi hapus • Berhasil hapus data *regression testing*	Sesuai harapan
4	*Search* data *regression testing*	• Input data yang akan dicari pada field *column search* • Klik enter	Berhasil menampilkan *regression testing* yang dicari, jika data tersedia	Sesuai harapan

Berdasarkan hasil pengujian yang dilakukan pada menu sistem, dapat disimpulkan bahwa seluruh fungsi yang diuji menunjukkan hasil yang sesuai dengan yang diharapkan. Secara keseluruhan, aplikasi ini telah teruji dengan baik dan berfungsi sesuai dengan tujuan pengujian, memberikan jaminan bahwa sistem yang dikembangkan memiliki kualitas dan kestabilan yang memadai dalam penggunaan.

5. KESIMPULAN

Berdasarkan hasil komprehensif dialog yang telah dilakukan, penelitian ini mengembangkan Digitalisasi Proses Pengujian Aplikasi Berbasis Analisis *Log Error* Dengan Metode *Regression Testing*. Tujuan utama dari penelitian ini adalah untuk meningkatkan efektivitas dalam pencatatan proses pengujian aplikasi dan memperbaiki kesalahan(*log error*) pada aplikasi yang telah diujikan. Beberapa poin penting dari penelitian ini adalah: Digitalisasi Pengujian Meningkatkan Efisiensi: Sistem digital yang dikembangkan untuk pencatatan *test case*, *log error*, dan *regression testing* dapat meningkatkan efisiensi dalam dokumentasi dan pengujian aplikasi. Proses yang sebelumnya dilakukan secara manual dapat diotomatisasi, mengurangi kesalahan pengguna Pencetakan Laporan dari Proses Pengujian: sistem yang dirancang dapat dapat mencetak laporan hasil pengujian aplikasi secara otomatis. Laporan yang terstruktur ini mempermudah komunikasi antara tim penguji dengan project manager. Kolaborasi Tim Lebih Efisien: Dengan penerapan sistem yang terintegrasi, kolaborasi antara tim penguji dan tim programmer menjadi lebih efisien. Tim programmer dapat mengakses *log error* secara *real-time*, tanpa harus menunggu laporan manual, sehingga mempercepat proses perbaikan dan pengujian ulang. Ini berpotensi mengurangi waktu tunggu antara kesalahan baru dan perbaikan**.** Monitoring dan Pengawasan Lebih Terstruktur: Project manager dapat memantau progres pengujian dan pengembangan aplikasi secara *real-time*, yang mempermudah pengawasan dan evaluasi. Sistem digital memberikan visibilitas yang lebih baik terhadap status penguji.

DAFTAR PUSTAKA

[1] Umar, M. A., & Zhanfang, C. (2019). *Fundamentals of Software Testing*. Wiley & Sons.

[2] M. Anderson and J. Smith, System Debugging and Log Analysis for Software Engineers. Springer, 2021.

[3] Y. F. Putri, "Automation Regression Testing pada Aplikasi Teman Diabetes dengan Menggunakan Metode Blackbox Testing," Skripsi, Program Studi Sistem Informasi, Fakultas Teknologi Industri, Universitas Atma Jaya Yogyakarta, 2020.

[4] Nursaid, F. F. Brata, and A. P. Kharisma, "Pengembangan Sistem Informasi Pengelolaan Persediaan Barang Dengan ReactJS Dan React Native Menggunakan Prototype (Studi Kasus: Toko Uda Fajri)," Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 4, no. 8, pp. 964X, 2020.

[5] I. Tvoroshenko and H. Maksimenko, "Research of regression and modular testing of web applications," Editorial Board, vol. 406, 2021.

[6] S. Luginawati and A. P. Wahyu, "Pengujian Perangkat Lunak Menggunakan Metode Regression Testing pada Aplikasi Pembelajaran Matematika untuk Siswa Tingkat SMP Berbasis Android," Syntax Idea, vol. 5, no. 1, 2023.

[7] N. B. Ali et al., "On the search for industry-relevant regression testing research," Empirical Software Engineering, vol. 24, no. 4, pp. 2020–2055, 2019.

[8] A. Singh, G. Kaur, and R. Malik, "Manual vs Automated Regression Testing: A Comparative Analysis," International Journal of Software Testing, vol. 14, no. 3, pp. 89–102, 2021.

[9] R. Patel and K. Desai, "Analyzing Manual Regression Testing in Complex Software Systems," Journal of Computer Applications, vol. 29, no. 5, pp. 99–110, 2022.

[10] P. Kumar and R. Sharma, "Manual Testing Strategies for Regression Testing in Agile Development," Software Engineering Journal, vol. 10, no. 3, pp. 78–89, 2020.

[11] S. Ramesh and P. Gupta, "User Interaction and Manual Regression Testing: Challenges and Best Practices," Journal of Human-Computer Interaction, vol. 26, no. 1, pp. 67–81, 2021.

[12] L. Zhang and H. Li, "Automation in Regression Testing: Trends and Future Directions," Software Engineering and Automation Journal, vol. 22, no. 6, pp. 143–159, 2021.

[13] R. Ahmed, K. Sharma, and S. Verma, "Automated Regression Testing: A Review on Techniques and Tools," International Journal of Software Engineering, vol. 15, no. 2, pp. 45–60, 2023.

[14] M. Rahman and Y. Chen, "Selenium-based Automated Testing for Web Applications: A Comparative Study," Journal of Software Testing & Validation, vol. 19, no. 2, pp. 34–49, 2022.

[15] S. Mehta and A. Roy, "Appium for Mobile Application Testing: An Overview," International Journal of Mobile Computing, vol. 17, no. 4, pp. 55–72, 2021.

[16] T. Brown, L. Williams, and M. Johnson, "JUnit Framework and Its Role in Regression Testing," Journal of Software Quality Assurance, vol. 28, no. 1, pp. 112–126, 2023.