CICES (*Cyberpreneurship Innovative and Creative Exact and Social Science*)
DOI: 10.33050/cices.v11i2.3514 ISSN: **2356-5209**
pp. 158~164 *Online* ISSN: **2655-3058**

Analisis Perbandingan *Test Case Prioritization* Menggunakan Algoritma *Greedy* dan Algoritma *Hill Climbing*

Dipika Syaiban Ainun*[1], Made Hanindia Prami Swari[2], Fawwaz Ali Akbar[3]
[1,2,3]Informatika, Ilmu Komputer, Universitas Pembangunan Nasional "Veteran" Jawa Timur, Surabaya
E-mail: *20081010226@student.upnjatim.ac.id

Abstrak

Software Testing merupakan salah satu langkah penting dalam memastikan suatu software tetap unggul serta dapat berjalan dan digunakan oleh masyarakat luas, software testing dapat dilakukan menggunakan Regression Testing untuk memastikan software tidak mengalami kerusakan atau terdapat bug baru setelah perubahan terjadi. Regression Testing digunakan untuk menghemat waktu dan biaya pengujian dengan menerapkan Test Case Prioritization yang berfungsi agar software tester dapat memilih test case yang paling penting untuk dilakukan pengujian berdasarkan penggunaan algoritma Greedy dan algoritma Hill Climbing yang digabungkan dengan parameter lain seperti Severity Rating, dan matriks Average Percentage Fault Detection (APFD). Fokus penelitian ini adalah mencari urutan prioritas test case berdasarkan hasil pengujian Regression Testing pada website demoblaze.com berdasarkan perbandingan algoritma Greedy dan algoritma Hill Climbing yang digabungkan dengan parameter Severity Rating berdasarkan nilai APFD tertinggi. Hasil dari penelitian menunjukkan penggunaan algoritma Hill Climbing lebih efektif dalam menentukan prioritas test case dengan memperoleh nilai APFD sebesar 0.91, sedangkan algoritma Greedy mendapatkan nilai APFD sebesar 0.9.

Kata Kunci— *Software Testing, Regression Testing, Test Case Prioritization, Algoritma Greedy, Algoritma Hill Climbing*

Abstract

Software Testing is one of the important steps in ensuring that a software remains superior and can run and be used by the wider community, software testing can be done using Regression Testing to ensure that the software is not damaged or there are new bugs after changes occur. Regression Testing is used to save time and testing costs by implementing Test Case Prioritization which functions so that software testers can select the most important test cases to be tested based on the use of the Greedy algorithm and the Hill Climbing algorithm combined with other parameters such as Severity Rating, and the Average Percentage Fault Detection (APFD) matrix. The focus of this research is to find the priority order of test cases based on the results of Regression Testing on the demoblaze.com website based on the comparison of the Greedy algorithm and the Hill Climbing algorithm combined with the Severity Rating parameter based on the highest APFD value. The results of the research show that the use of the Hill Climbing algorithm is more effective in determining the priority of test cases by obtaining an APFD value of 0.91, while the Greedy algorithm gets an APFD value of 0.9.

Keywords— *Software Testing, Regression Testing, Test Case Prioritization, Algoritma Greedy, Algoritma Hill Climbing*

DOI: 10.33050/cices.v11i2.3514 ISSN: **2356-5209**
pp. 158~164 *Online* ISSN: **2655-3058**

1. PENDAHULUAN

Perkembangan era digital memungkinkan banyak aktivitas manusia dilakukan jarak jauh perangkat lunak seperti *website* dan aplikasi, yang dapat diakses melalui *smartphone*, tablet, dan laptop. Penggunaan perangkat lunak telah meluas di berbagai bidang seperti industri, pendidikan, dan hiburan, yang meningkatkan kebutuhan akan perangkat lunak berkualitas tinggi. Salah satu contohnya adalah penggunaan *e-commerce* untuk meningkatkan aktivitas bisnis, memudahkan pelaku bisnis dalam memasarkan produk dan jasa. Data dari goodstats.id menunjukkan peningkatan signifikan pengguna *e-commerce* di Indonesia dari 93,42 juta pada tahun 2018 menjadi proyeksi 244,67 juta pada tahun 2027. Seiring dengan perkembangan cepat ini, pengembang perangkat lunak perlu memperhatikan kualitas produk mereka melalui proses pengujian perangkat lunak (*Software Testing*) untuk mencegah kecacatan dan meningkatkan pengalaman pengguna. Selama *software* dalam proses pengembangan, modifikasi fitur pada *software* sering terjadi khususnya pada jenis software yang dikembangkan menggunakan metode *agile development*, karena perubahan *requirement* dapat terjadi pada setiap fase pengembangan [1]. *Software Testing* adalah proses eksekusi program dengan tujuan menemukan kesalahan atau kecacatan pada sebuah program [2]. Software Testing bertujuan menemukan kesalahan dalam program, memastikan stabilitas dan fungsionalitas perangkat lunak, serta meningkatkan kepercayaan pelanggan.

Salah satu jenis pengujian penting adalah *Regression Testing*, *Regression Testing* adalah jenis pengujian dengan tujuan untuk memverifikasi perubahan yang telah dibuat terhadap versi terbaru dari perangkat lunak seperti *bug fixes, code merges*, migrasi sistem, dan *website*/aplikasi server untuk mengkonfirmasi bahwa fungsi program yang sudah ada tetap berfungsi sesuai dengan fungsional utamanya [2]. Tujuan lain dari Regression Testing yaitu memverifikasi bahwa perubahan pada perangkat lunak tidak menyebabkan *bug* baru [3]. *Regression Testing* memiliki 3 jenis teknik didalamnya, seperti *Test Case Minimizing*, *Test Case Selection*, dan *Test Case Prioritization* [4].

Test Case Prioritization merupakan proses pememilihan *test case* yang paling penting untuk dieksekusi [5]. *Test Case Prioritization* dapat didasarkan pada kode, kebutuhan, *bug*, waktu eksekusi, resiko, dan biaya. *Test Case Prioritization* meningkatkan kualitas perangkat lunak dan menghadirkan produk terbaik dalam waktu dan biaya yang terbatas [6]. *Test Case Prioritization* sangat penting dalam proses *Software Testing* karena memungkinkan *Software Tester* untuk mengoptimalkan waktu dan sumber daya yang tersedia. Dengan demikian, *Software Tester* dapat fokus pada *test case* yang paling penting dan efektif dalam menemukan bug atau masalah keamanan, sehingga dapat meningkatkan kualitas perangkat lunak dan memastikan bahwa perangkat lunak berfungsi dengan baik. Penggunaan *Test Case Prioritization* membuat *Software Tester* dapat meningkatkan peroses berjalanya Software Testing [7].

Severity Rating merupakan nilai yang digunakan untuk mengukur tingkat keparahan pada permasalahan yang ditemukan ketika menggunakan suatu sistem. Tingkat keparahan ini akan menjadi rekomendasi perbaikan untuk mengatasi masalah saat ini [8]. *Severity Rating* biasanya menggunakan skala numerik 0 hingga 4 sebagai penilaiannya, skala 0 dianggap tidak perlu dilakukan perbaikan pada suatu fitur, skala 1 dianggap tidak terlalu membutuhkan perbaikan karena masalah yang ada tidak mempengaruhi *experience* dari pengguna, skala 2 dianggap memerlukan perbaikan dengan tingkat prioritas yang rendah, skala 3 dianggap memerlukan perbaikan dengan prioritas yang tinggi, dan yang terakhir yaitu skala 4 dianggap bahwa perbaikan wajib dilakukan [9].

Algoritma Greedy merupakan implementasi dari konsep pencarian "pilihan terbaik berikutnya". Algoritma *Greedy* menjalankan prinsip di mana elemen-elemen dengan bobot tertinggi diambil terlebih dahulu, diikuti oleh yang memiliki bobot tertinggi kedua, dan seterusnya, hingga sebuah solusi dibentuk meskipun mungkin tidak optimal secara keseluruhan. Algoritma *Greedy* bekerja berdasarkan prinsip bahwa *test case* dengan bobot tertinggi akan diambil pertama kali, diikuti oleh bobot kedua, dan seterusnya. Bobot dihitung berdasarkan jumlah *fault* yang dideteksi oleh *test case* [4]. Algoritma *Greedy* tidak mempertimbangkan permasalahannya saat membuat solusi, sehingga memilih solusi terbaik saat ini di setiap proses tanpa mempertimbangkan situasi masalah secara keseluruhan.

DOI: 10.33050/cices.v11i2.3514 ISSN: **2356-5209**
pp. 158~164 *Online* ISSN: **2655-3058**

Algoritma Hill Climbing merupakan algoritma yang setiap urutan ujinya dianggap sebagai suatu keadaan, lalu algoritma ini akan bergerak secara berulang ke keadaan terbaik di antara semua keadaan saat ini [10].

Penelitian ini menggunakan *Regression Testing* dengan teknik *Test Case Prioritization* untuk mengidentifikasi dan mencaritahu perbaikan kerusakan yang perlu diprioritaskan pada perangkat lunak. *Test Case Prioritization* membantu mengoptimalkan waktu dan sumber daya dengan memilih *test case* yang paling penting. Penelitian ini membandingkan dua algoritma, *Greedy* dan *Hill Climbing*, untuk menentukan efektivitas prioritisasi test case berdasarkan nilai *Average Percentage Fault Detection* (APFD). Website demoblaze.com digunakan sebagai objek penelitian karena sifatnya yang dummy memungkinkan pengujian tanpa risiko bagi pengguna atau bisnis nyata.

Berdasarkan latar belakang tersebut, dapat dipaparkan rumusan masalah sebagai berikut:

1. Bagaimana melakukan pengujian *Regression Testing* pada pengembangan perangkat lunak.
2. Bagaimana implementasi teknik *Test Case Prioritization* berdasarkan nilai *severity*, algoritma *Greedy* dan algoritma *Hill Climbing*.
3. Bagaimana menentukan nilai efektivitas dari hasil pengujian *Test Case Prioritization* pada masing masing algoritma menggunakan *Average Percentage Fault Detection* (APFD).

2. METODE PENELITIAN

Penelitian ini menggunakan metode Campuran yaitu pendekatan penelitian yang menggabungkan atau mengasosiasikan bentuk kualitatif dan kuantitatif, pendekatan kualitatif digunakan pada proses wawancara untuk mendapatkan data dari *Severity Ratings*, sedangkan pendekatan kuantitatif digunakan pada proses eksperimen, yaitu penelitian dengan memanipulasi variabel dalam penelitian serta melihat pengaruhnya terhadap variabel lain. Penelitian ini dilakukan selama 4 bulan, dimulai sejak April 2024 hingga Agustus 2024.

Pengambilan sample pada penelitian ini memiliki 2 metode, metode yang pertama merupakan metode kualitatif yang dilakukan dengan wawancara kepada ahli pada bidang *software testing* dengan tujuan mendapatkan justifikasi dan validasi terhadap *Severity Rating* dalam temuan bug pada *website* demoblaze.com, dan metode yang kedua merupakan metode kuantitatif pada proses *Test Case Prioritization* dengan mencari nilai efektivitas terbaik dari perbandingan penggunaan 2 algoritma.

Tahapan dari penelitian ini dimulai dengan tahap analisis kebutuhan yang merupakan proses untuk mengidentifikasi dan memahami kebutuhan suatu sistem, proyek, atau produk serta mengetahui tujuan dibuatnya *website* demoblaze.com. Selanjutnya tahap identifikasi masalah dilakukan dengan tujuan menghasilkan titik temu yang menunjukkan adanya masalah dalam penelitian yang perlu ditinjau dari sisi keilmuan serta banyaknya masalah yang dapat diidentifikasi. Tahap selanjutnya yaitu pembuatan skenario pengujian yang berisikan langkah-langkah, dan *expected result* sebagai acuan dari proses pengujian *Regression Testing* yang dilakukan terhadap *website* demoblaze.com. Selanjutnya tahap pengujian *Regression Testing* dilakukan sesuai dengan alur dari seknario pengujian setelah skenario pengujian telah selesai dibuat. Tahap selanjutnya yaitu evaluasi hasil *Regression Testing* yang dilakukan dengan membandingkan *actual result* dengan *expected result* yang telah dibuat oleh peneliti pada proses pembuatan skenario pengujian. Metode kualitatif pada penelitian ini menggunakan teknik wawancara sebagai pengumpulan sampel yang nantinya akan digunakan sebagai penetapan *Severity Ratings* sebelum dilakukan proses *Test Case Prioritization*. Tahapan terakhir yaitu proses *Test Case Prioritization* dan evaluasi hasil dengan mencari nilai efektivitas terbaik berdasarkan matriks *Average Percentage Fault Detection* (APFD) pada penggunaan algoritma *Greedy* dan algoritma *Hill Climbing* yang dilakukan menggunakan pemrograman berbahasa *Python*.

DOI: 10.33050/cices.v11i2.3514 ISSN: **2356-5209**
pp. 158~164 *Online* ISSN: **2655-3058**

Tabel 1 *Severity rating*

Skala	Keterangan
0	Tidak perlu perbaikan
1	Terdapat masalah tetapi tidak mempengaruhi kenyamanan pengguna. Perbaikan tidak terlalu dibutuhkan
2	Perbaikan diperlukan dengan tingkat prioritas rendah
3	Perbaikan diperlukan dengan tingkat prioritas tinggi
4	Perbaikan wajib dilakukan

Berdasarkan Tabel 1 *Severity Rating* merupakan nilai yang digunakan untuk mengukur tingkat keparahan pada permasalahan yang ditemukan user ketika menggunakan sistem [9]. Tingkat keparahan ini berpengaruh untuk menjadi bahan rekomendasi perbaikan dari *Severity Rating* yang ada pada fault. skala *Severity Rating* dibagi menjadi 5 tingkatan, yaitu skala 0 merupakan skala yang paling rendah dan diartikan sebagai tidak perlu ada perbaikan, selanjutnya skala bernilai 1 yang diartikan sebagai perbaikan tidka terlalu dibutuhkan karena permasalahan yang ada tidak mempengaruhi kenyamanan pengguna, selanjutnya skala bernilai 2 yang berarti perbaikan perlu dilakukan dengan tingkat prioritas rendah, lalu skala bernilai 3 bermaksud perbaikan perlu dilakukan dengan tingkat prioritas tinggi, dan yang terakhir skala bernilai 4 yang berarti perbaikan wajib dilakukan.

Pengolahan dan analisis data dilakukan menggunakan pemrograman berbahasa *Python* yang akan membaca file Microsoft Excel berisikan temuan *bug*. Program akan mengolah data berdasarkan nilai *Severity Ratings* tertinggi dan melakukan proses *Test Case Prioritization* berdasarkan nilai efektivitas terbaik diantara penggunaan algoritma *Greedy* dan algoritma *Hill Climbing*.

3. HASIL DAN PEMBAHASAN

Tabel 2 Hasil *Severity rating* terhadap *faults*

Faults (f)	*Severity Rating*
f1	4
f2	4
f3	4
f4	3
f5	4
f6	1
f7	4
f8	2
f9	4

Tabel 2 menampilkan *Severity Rating* yang didapatkan oleh setiap *faults*. *Severity Rating* pada *faults* f1, f2, f3, f5, f7 mendapatkan *Severity Rating* senilai 4; *faults* f4 mendapatkan *Severity Rating* senilai 3; sedangkan *faults* f6 dan f8 mendapatkan *Severity Rating* senilai 2. Selanjutnya nilai *Severity Rating* dari setiap *faults* (f) akan dimasukkan kedalam tabel temuan *Bug* dari hasil *Regression Testing*.

Tabel 3 Temuan Bug dan Severity Rating

Test Cases (TC)	*Faults*								
	f1	f2	f3	f4	f5	f6	f7	f8	f9
TC1									
TC2	4	4							
TC3									

DOI: 10.33050/cices.v11i2.3514 ISSN: **2356-5209**
pp. 158~164 *Online* ISSN: **2655-3058**

TC4	4	4							
TC5									
TC6	4	4							
TC7	4	4	4						
TC8				3	4	2			
TC9					4	2	4		4
TC10			4	3	4				
--TC11						2			
TC12						2		2	
TC13	4	4			4				
TC14	4	4			4				
TC15									

Tabel 2 menampilkan hasil temuan *bug* pada *Regression Testing* menunjukkan bahwa pada TC1, TC3, TC5, dan TC15 tidak ditemukan bug pada proses pengujiannya; TC2, TC4, dan TC6 mendeteksi f1 dan f2; TC7 mendeteksi f1, f2, dan f3; TC8 mendeteksi f4, f5, dan f6; TC9 mendeteksi f5, f6, f7, f9; TC10 f3, f4, f5, dan f7; TC11 mendeteksi f6; TC12 mendeteksi f6 dan f8; TC13 dan TC14 mendeteksi f1, f2, dan f5.

Perhitungan Nilai Efektivitas

Perhitungan hasil Test Case Prioritization didapatkan sesuai dengan nilai *Average Percentage Fault Detection* (APFD). APFD digunakan untuk mengukur efektifitas dari prioritas perbaikan *test case* yang telah diuji. Nilai APFD berjarak 0 hingga 1, semakin tinggi nilai APFD menandakan semakin baik nilai pendeteksian Fault-nya [4].

$$APFD = 1 - \frac{(Tf1 + Tf2 + \ldots + Tfm)}{mn} + \frac{1}{2n} \quad (1)$$

Catatan: T merupakan test cases; n merupakan jumlah test cases; m merupakan jumlah fault; (Tf…Tfm) merupakan posisi *Test Case*(T) yang pertama kali mendeteksi *Faults*.

Tabel 4 Perbandingan Algoritma

Algoritma	APFD
Greedy	0,9
Hill Climbing	0,91

Tabel 4 menampilkan hasil perbandingan dari *Test Case Prioritization* dengan penggunaan algoritma *Greedy* dan algoritma *Hill Climbing*, dari penggunaan 2 algoritma tersebut, algoritma *Greedy* cukup efektif dalam memberikan prioritas perbaikan dengan mendapatkan nilai efektivitas sebesar 0.9, sedangkan algoritma *Hill Climbing* dapat lebih efektif dalam mendeteksi fault yang ada, serta dapat memberikan prioritas perbaikan berdasarkan urutan *test case*.

Penelitian ini menggunakan *website* demoblaze.com sebagai objek pengujian *Regression Testing*. Pengujian *Regression Testing* pada *website* demoblaze.com menghasilkan 9 jenis *faults* dari 15 *test case* yang dijalankan. *Test Case Prioritization* menggunakan algoritma *Greedy* menghasilkan nilai efektivitas sebesar 0.9 dengan pengurutan TC10-TC9-TC13-TC7-TC14-TC2-TC4-TC6-TC8-TC11-TC12-TC1-TC3-TC5-TC15, sedangkan algoritma *Hill Climbing* menghasilkan nilai efektivitas sebesar 0.91 dengan pengurutan TC10-TC13-TC9-TC7-TC14-TC2-TC4-TC6-TC8-TC11-TC12-TC1-TC3-TC5-TC15. Dengan nilai efetivitas yang didapatkan, nilai efektivitas sebesar 0.91 yang dimiliki oleh algoritma *Hill Climbing* dan 0.9 yang dimiliki oleh algoritma *Greedy* menunjukkan bahwa penggunaan APFD sebagai ukuran perhitungan nilai efektivitas cukup efektif [1] serta dapat membantu *software development* dalam menentukan urutan prioritas perbaikan. Nilai APFD tertinggi dapat digapai pada saat *test case* pertama dapat mendeteksi seluruh *faults*, sedangkan nilai APFD terendah dapat digapai pada saat *test case* n-1 tidak dapat mendeteksi *faults*.

DOI: 10.33050/cices.v11i2.3514 ISSN: **2356-5209**
pp. 158~164 *Online* ISSN: **2655-3058**

4. KESIMPULAN

Regression Testing dilakukan dengan menganalisis kebutuhan dari *software* agar *software tester* dapat memahami kebutuhan suatu sistem, proyek, ataupun produk, selanjutnya mengidentifikasi masalah dengan mencari tahu *behavior* dari suatu *software* agar *software tester* dapat menemukan celah-celah yang berkemungkinan akan ditemukan *bug/defect*, selanjutnya pembuatan skenario pengujian berdasarkan *behavior* dari *software* yang akan dilakukan pengujian, setelah skenario pengujian atau *test case* telah dibuat, maka proses *Regression Testing* akan dilakukan berdasarkan langkah-langkah pada *test case*, dan yang terakhir proses evaluasi hasil *Regression Testing* berdasarkan analisis kebutuhan awal dengan cara membandingkan *expected result* dengan *actual result*, dan menyimpulkan ketidaksesuaian hasilnya sebagai *bug/defect*.

Implementasi *Test Case Prioritization* dengan mencampurkan parameter *Severity Rating* kedalam penggunaan algoritma *Greedy* dan algoritma *Hill Climbing* cukup mempengaruhi proses prioritisasi, mencampurkan *Severity Rating* kedalam penggunaan algoritma *Greedy* dan algoritma *Hill Climbing* bertujuan agar proses *Test Case Prioritization* dapat dilakukan dengan memprioritaskan *test case* yang memiliki *fault* paling kritis terlebih dahulu agar perbaikan pada *software* tetap mengutamakan *fault* yang menjadi fokus utama dari kebutuhan suatu sistem, proyek, ataupun produk, selain itu proses *Test Case Prioritization* juga dilakukan berdasarkan nilai efektivitas dari pengurutan prioritasnya, hal ini bertujuan agar hasil *Test Case Prioritization* juga dapat memberikan daftar prioritas perbaikan pada *software* yang paling efektif untuk dilakukan oleh pihak pengembang.

Proses perhitungan nilai efektivitas pada *Test Case Prioritization* dapat dilakukan menggunakan rumus perhitungan *Average Percentage Fault Detection* (APFD) yang proses menentukan kebutuhan data perhitungannya diambil berdasarkan fn yang pertama kali ditemukan pada urutan prioritas, setiap fn yang ditemukan pertama kali pada urutan prioritas 1 akan dimasukkan kedalam rumus dengan poin 1, sedangkan fn yang ditemukan pertama kali pada urutan prioritas 2 akan mendapatkan poin 2, begitupun dengan fn yang ditemukan pertama kali apda urutan prioritas 3 akan diberikan poin 3, dan seterusnya. Penggunaan perhitungan *Average Percentage Fault Detection* (APFD) sebagai penentuan nilai efektivitas juga dapat disimpulkan bahwa semakin banyak fn yang pertama kali ditemukan pada prioritas 1, maka nilai efektivitasnya akan semakin tinggi

Berdasarkan penelitian ini, penggunaan algoritma *Hill Climbing* pada proses *Test Case Prioritization* lebih efektif dalam menemukan urutan Test Case berdasarkan prioritas perbaikannya jika dibandingkan dengan algoritma *Greedy*. Hal ini dapat membantu proses *software development* dalam menentukan prioritas perbaikan yang perlu dilakukan.

5. SARAN

Penelitian ini juga masih perlu ditinjau lebih lanjut, karena penggunaan objek pada penelitian ini masih berupa *website* demo yang tidak secara aktif digunakan dalam kehidupan sehari-hari, serta penggunaan perbandingan algoritma yang hanya melibatkan algoritma *Greedy* dan algoritma *Hill Climbing*, serta hanya menggunakan matriks *Average Percentage Fault Detection* (APFD). Namun, penelitian ini tetap dapat digunakan sebagai sarana pembelajaran untuk memahami teknik yang ada.

DAFTAR PUSTAKA

[1] Maspupah, A., Muharram, M. K., & Daeli, S. G. (2023). Analisis Efektifitas Algoritma FAST Menggunakan Metrik Average Percentage Fault Detection dan Waktu Eksekusi Pada Test Case Prioritization. *Journal of Information System Research (JOSH)*, *4*(2), 451–457. https://doi.org/10.47065/josh.v4i2.2822

[2] Luginawati, S., & Wahyu, A. P. (2023). Pengujian Perangkat Lunak Menggunakan Metode Regression Testing Pada Aplikasi Pembelajaran Matematika Untuk Siswa Tingkat SMP Berbasis Android Syntax Idea, 5(1). https://doi.org/10.36418/syntax-idea.v3i6.1227.

[3] Ali, N. bin, Engström, E., Taromirad, M., Mousavi, M. R., Minhas, N. M., Helgesson, D., Kunze, S., & Varshosaz, M. (2019). On the search for industry-relevant regression testing research. *Empirical Software Engineering*, *24*(4), 2020–2055. https://doi.org/10.1007/s10664-018-9670-1

[4] Rahmani, A., Min, J. L., & Maspupah, A. (2021). An empirical study of regression testing techniques. *Journal of Physics: Conference Series*, *1869*(1). https://doi.org/10.1088/1742-6596/1869/1/012080

[5] Bajaj, A., & Sangwan, O. P. (2019). A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms. *IEEE Access*, *7*, 126355–126375. https://doi.org/10.1109/ACCESS.2019.2938260

[6] Qasim, M., Bibi, A., Hussain, S. J., Jhanjhi, N. Z., Humayun, M., & Sama, N. U. (2021). Test case prioritization techniques in software regression testing: An overview. *International Journal of Advanced and Applied Sciences*, *8*(5), 107–121. https://doi.org/10.21833/ijaas.2021.05.012

[7] Khatibsyarbini, M., Isa, M. A., Jawawi, D. N. A., Hamed, H. N. A., & Mohamed Suffian, M. D. (2019). Test Case Prioritization Using Firefly Algorithm for Software Testing. *IEEE Access*, *7*, 132360–132373. https://doi.org/10.1109/ACCESS.2019.2940620

[8] Nasution, A. R., Hidayat, R., Manik, H. W. S., Assidiqie, M. F., & Ikhwan, A. (2023). Pengaruh Tampilan UI Dan UX Terhadap Kenyamanan Pengguna Pada Aplikasi OVO. *Jurnal Penelitian Dan Pengkajian Ilmiah Eksakta*, *2*(1), 81–84. https://doi.org/10.47233/jppie.v2i1.713

[9] Hadinegoro, A., Faticha, R., Aziza, A., & Mufhadhal, M. F. (2022). Analisis Pengaruh User Interface Dan User Experience Platform Online Menggunakan Metode Heuristik. *Jurnal Teknologi Informasi*, *17*(2).

[10] Lou, Y., Chen, J., Zhang, L., & Hao, D. (2019). A Survey on Regression Test-Case Prioritization. In *Advances in Computers* (Vol. 113, pp. 1–46). Academic Press Inc. https://doi.org/10.1016/bs.adcom.2018.10.001