Terbit *online* pada laman : http://teknosi.fti.unand.ac.id/

Jurnal Nasional Teknologi dan Sistem Informasi

| ISSN (Print) 2460-3465 | ISSN (Online) 2476-8812 |

Studi Kasus

Uji Kualitas Website Admin Travel Booking Menggunakan Halstead's Metric dan Equivalence Partitioning

Fauziah Nur Syifa [a], Trisana Nurul Anzali Nabil [b], Derrel Hendi Arifin [c], Rangga Sidik [d,]*

[a]*Universitas Komputer Indonesia, Jl. Dipati Ukur 112-114, Kota Bandung, 40132, Indonesia*
[b]*Universitas Komputer Indonesia, Jl. Dipati Ukur 112-114, Kota Bandung, 40132, Indonesia*
[c]*Universitas Komputer Indonesia, Jl. Dipati Ukur 112-114, Kota Bandung, 40132, Indonesia*
[d]*Universitas Komputer Indonesia, Jl. Dipati Ukur 112-114, Kota Bandung, 40132, Indonesia*

INFORMASI ARTIKEL

Sejarah Artikel:
Diterima Redaksi: 19 Juni 2024
Revisi Akhir: 29 April 2025
Diterbitkan *Online*: 04 Mei 2025

KATA KUNCI

Testing,
Halstead's Metric,
Equivalence Partitioning,
Travel System

KORESPONDENSI

E-mail: rangga.sidik@email.unikom.ac.id*

A B S T R A C T

Tripease is a travel booking admin website designed to enable admins with management roles for train tickets, hotel tickets, and user data. Blackbox testing ensures the software meets the set requirements and specifications and detects functional and performance errors. Whitebox testing is done to check the software's logical path by examining the software's logical structure and detecting errors, such as logical errors and script understanding errors. This research discusses testing the travel booking admin website using two methods: Halstead's Metric and Equivalence Partitions. This research aims to determine the success rate of features and predict the appearance of bugs on the website. White box testing with Halstead's Metric shows that the add train feature has the highest prediction of bugs (45.7%), while the add user feature is the lowest (13.4%). The average prediction of the occurrence of bugs in all features is 25.59%. Black box testing with Equivalence Partitions shows that the add user and edit user features have a low success rate because they do not succeed in the input validity test item. The add train and login features have a 100% success rate. Based on the test results, it is recommended that the code be refactored and the input validity of each feature improved.

1. PENDAHULUAN

Tripease merupakan sebuah website admin travel booking. Website ini dirancang dengan tujuan utama untuk memungkinkan seorang pengguna dengan peran admin untuk mengelola manajemen tiket kereta api, tiket hotel, serta data pengguna aplikasi. Dengan kata lain, Tripease bertujuan untuk menyediakan platform yang efisien dan efektif bagi admin untuk mengelola berbagai aspek dari layanan travel booking. Dalam website ini, admin diberikan kemampuan untuk melihat, menambah, mengedit, dan menghapus tiket kereta api dan hotel yang tersedia. Selain itu, mereka juga dapat melihat dan mengelola data pengguna aplikasi. Dengan demikian, Tripease memungkinkan admin untuk memiliki kontrol penuh atas layanan travel booking, sehingga mereka dapat memastikan bahwa layanan tersebut berjalan dengan lancar dan efisien. Dalam konteks pengujian perangkat lunak, kita akan melakukan pengujian blackbox dan whitebox pada penelitian ini. Pengujian blackbox adalah teknik pengujian di mana kita memeriksa fungsionalitas perangkat lunak. Dengan kata lain, kita tidak tertarik pada bagaimana perangkat lunak bekerja, tetapi hanya pada apa yang dilakukannya. Pengujian whitebox adalah teknik pengujian di mana kita memeriksa struktur internal perangkat lunak, bukan hanya fungsionalitasnya dari perspektif pengguna [1]. Dengan kata lain, kita tidak hanya tertarik pada apa yang dilakukan perangkat lunak, tetapi juga bagaimana cara kerjanya. Tujuan utama dari pengujian blackbox adalah untuk memastikan bahwa perangkat lunak memenuhi semua persyaratan dan spesifikasi yang ditetapkan [2]. Melalui metode ini, kita dapat mendeteksi adanya kesalahan seperti kesalahan fungsional dan kesalahan kinerja yang dapat mengakibatkan kekecewaan pengguna. Pengujian blackbox sangat penting dalam siklus pengembangan perangkat lunak. Dengan melakukan pengujian ini, kita dapat memastikan bahwa perangkat lunak memenuhi semua kebutuhan pengguna. Selain itu, pengujian ini juga membantu kita dalam memperbaiki kesalahan sebelum perangkat lunak dirilis [3]. Dengan demikian, pengujian blackbox membantu kita dalam memastikan bahwa perangkat lunak memenuhi semua persyaratan dan spesifikasi yang ditetapkan, sehingga meningkatkan kepuasan pengguna dan keberhasilan proyek secara keseluruhan [4]. Tujuan utama dari pengujian whitebox adalah untuk memeriksa jalur logis perangkat lunak dengan memeriksa struktur logis perangkat lunak [5]. Melalui pendekatan ini, kita dapat mendeteksi adanya kesalahan seperti kesalahan logis dan kekeliruan pemahaman script yang dapat mengakibatkan kegagalan pada perangkat lunak. Dengan demikian, pengujian whitebox membantu kita dalam

https://doi.org/10.25077/TEKNOSI.v11i1.2025.67-77

meningkatkan kualitas dan keandalan perangkat lunak. Metode pengujian whitebox yang kami gunakan pada penelitian ini adalah menggunakan halstead metric, yang bertujuan untuk menghitung *number of bugs expected in the program*. *Number of bugs expected in the program* adalah untuk mengetahui prediksi bug yang muncul dalam program. Dengan mengetahui kemungkinan kemunculan bug dari program tersebut, dapat memberikan gambaran kualitas ketahanan program aplikasi sistem admin travel booking tersebut.

Penelitian sebelumnya yang menggunakan metode metrik Halstead telah dilakukan oleh Fredy Nenda Pranata dkk [6]. Penelitian tersebut bertujuan untuk mendeteksi cacat pada program sedini mungkin agar menjamin kualitas perangkat lunak dan juga untuk mengevaluasi serta melakukan pengukuran jumlah alur melalui program. Hasil dari penelitian tersebut adalah perhitungan akurasi didapatkan bahwa sistem yang diteliti untuk metode *Halstead's Volume* sebesar 87.5% dan metode *Cyclomatic Complexity* sebesar 100%. Hal ini menunjukan bahwa validitas algoritma kedua metode adalah valid. Kecacatan dari sistem yang diteliti adalah tidak dapat mendeteksi *operator* dalam kode sumber yang berasal dari data *import library* java [6]. Penelitian tersebut dilakukan untuk mengetahui akurasi sistem yang dibuat, namun belum mencapai pengujian untuk dapat menghitung estimasi kemunculan bug.

Selain itu penelitian yang dilakukan oleh Alfredo Alvaress Dace, Muhammad Harya Daffa, Yohanes Yeremias Dala Sula dan Fahri Rahman dalam penelitian yang berjudul "Pengujian Sistem Aplikasi Seleksi Sales Menggunakan Metode *Black Box* Teknik *Equivalence Partitions*". Penelitian ini bertujuan untuk memverifikasi bahwa program sesuai dengan fungsionalitas program yang dituju tanpa mengetahui kode sumber yang digunakan. Membuktikan bahwa dengan penerapan pengujian black box terutama dalam penggunaan teknik *Equivalence Partitioning*, beberapa fitur pada aplikasi yang diuji mengalami beberapa kesalahan fungsi. Fungsi tersebut tidak sesuai dengan spesifikasi yang diinginkan. Pengujian ini juga menunjukkan bahwa terdapat kesalahan pada formulir aplikasi Sistem Seleksi Best Selling [7].

Dari penelitian-penelitian tersebut maka penggunaan metode whitebox dan black box dapat digunakan untuk dapat saling melengkapi kekurangan hasil pengujian. Secara fungsional, program aplikasi dapat diuji kualitas dengan menggunakan metode black box *Equivalence Partitioning*. Namun bug pada program bisa saja terjadi dan tidak terprediksi secara fungsional. Oleh karena itu pengujian white box dilakukan untuk mendeteksi kemungkinan bug muncul. Pendeteksian bug yang muncul dilakukan melalui sebuah perhitungan estimasi yang dilakukan pada metode *Halstead's metric*. Pengujian pada aplikasi website admin travel booking dilakukakan untuk menguji kualitas aplikasi dari segi fungsional dan kompleksitas. Dengan pengggabungan dua metode tersebut diharapkan dapat memberikan gambaran kualitas perangkat lunak yang lebih jelas. Dengan gambaran kualitas yang jelas maka pengembang dapat mengidetifikasi resiko kecacatan pada perangkat lunak dan memberikan mitigasi yang jelas sebagai dukungan terhadap implementasi aplikasi sistem tersebut. Selain itu hasil pengujian dapat digunakan sebagai dasar pertimbangan pengembang sebelum tahapan rilis dilakukan.

1.1. Pengujian Perangkat Lunak

Metode pengujian adalah metode untuk menguji perangkat lunak dengan data yang dapat menguji perangkat lunak secara detail dan memiliki kemungkinan tinggi dalam menemukan kesalahan [8]. Pengujian pada penelitian ini menggunakan *white box testing* dan *black box testing*. *White box testing* adalah metode pengujian yang memiliki tujuan utama adalah untuk memeriksa jalur logis perangkat lunak dengan memeriksa struktur logis perangkat lunak [9]. Melalui pendekatan ini, kita dapat mendeteksi adanya kesalahan seperti kesalahan logis dan kekeliruan pemahaman script yang dapat mengakibatkan kegagalan pada perangkat lunak. Metode *black box* testing merupakan metode yang hanya menguji aplikasi tanpa mengetahui detailnya, seperti kode atau script dan memeriksa hasil berdasarkan input yang diberikan [10]. Metode pengujian blackbox hanya berdasarkan sisi fungsi, antarmuka, dan alurnya saja yang diuji tanpa menguji berdasarkan source code dari perangkat lunak.

1.2. Halstead's Metric

Halstead's metric adalah pengukuran yang dikembangkan untuk mengukur kompleksitas modul suatu program langsung dari kode sumber [11]. *Halstead's Metric* merupakan salah satu metode pengujian dari *white box testing.* Pengukuran dilakukan dengan menentukan ukuran kuantitatif kompleksitas dari operator dan operand dalam modul sistem. Operator merupakan simbol-simbol yang digunakan untuk melakukan operasi tertentu. Seperti operator aritmatika (+-*/ %), operator assignment (memberikan nilai pada variabel), operator perbandingan, operator logika, operator bitwise, operator ternary [12]. Operand merupakan nilai asal yang digunakan di dalam proses operasi[13]. *Halstead's metric* dapat digunakan untuk mengukur besarnya *defect*, dan evaluasi *software* untuk pengujian selanjutnya. Pada *Halstead's metric* terdapat beberapa enam jenis komponen yaitu :

1. *Length of program*

Length of program adalah hasil kalkulasi dari jumlah total operator dan operan yang muncul pada program yang sedang kita uji. Persamaan *length of program* dihitung dengan rumus berikut ini:

$$N = N1 + N2 \quad (1)$$

2. *Vocabulary of the program*

Vocabulary of the program adalah jumlah total operator dan operand unik yang muncul dalam program. Persamaan Vocabulary of the program dirumuskan pada Persamaan

$$n = n1 + n2 \quad (2)$$

3. *Volume of the program*

Volume dalam *Halstead metric* di gunakan untuk mengetahui volume program. Persamaan *Volume of the program* dirumuskan pada Persamaan

$$V = N \times \log2\ n \quad (3)$$

4. *Difficulty*

https://doi.org/10.25077/TEKNOSI.v11i1.2025.67-77

Difficulty dalam Halstead metric di gunakan untuk mengetahui kesulitan dan pengembangan program. Persamaan *Difficulty* dirumuskan pada Persamaan.

$$D = n1\ 2 \times N2\ n2 \quad (4)$$

5. *Effort*

Effort dalam Halstead metric di gunakan untuk mengetahui sumber daya yang digunakan untuk pengembangan program. Persamaan *Effort* dirumuskan pada Persamaan

$$E = D \times V \quad (5)$$

6. *Number of bugs expected in the program.*

Number of bugs expected in the program dalam Halstead metric di gunakan untuk mengetahui prediksi bug pada program [14]. Persamaan *Number of bugs expected in the program* dirumuskan pada Persamaan

$$B = V/3000 \quad (6)$$

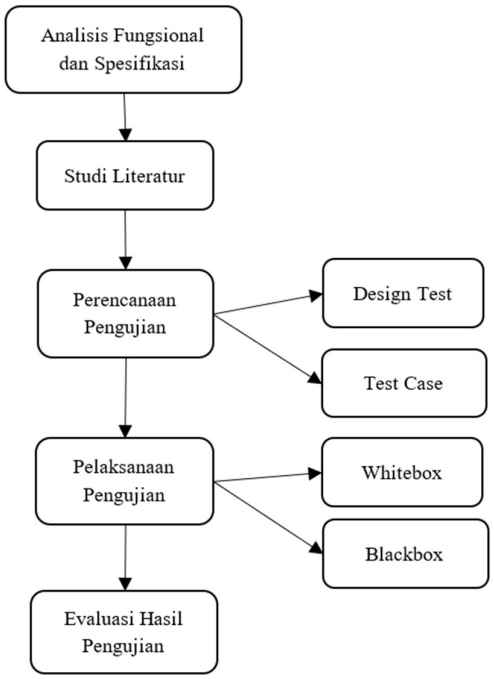
1.3. Equivalence Partitions

Metode yang digunakan pada uji perangkat lunak ini yaitu *Equivalence Partitions*, teknik tersebut menguji berdasarkan masukan pada tiap menu dengan cara menginputkan masukan yang telah dikelompokkan sesuai fungsinya [15]. Tahap awal yang akan dilakukan pada uji perangkat lunak ini yaitu memulai standar *grade partition input* dan *output*, hal itu dilakukan untuk mendapatkan data berupa hasil pengujian dengan menggunakan teknik *Equivalence Partitions* yang telah didokumentasikan [16]. Tabel *test case* mewakili semua kemungkinan kondisi yang dapat terjadi dalam sistem. Dengan menjalankan *test case* yang tercantum dalam tabel *test case tersebut*, penguji dapat memastikan bahwa sistem berfungsi dengan benar dalam semua kondisi. Jadi, Equivalence Partitioning adalah teknik pengujian black-box yang membagi domain input ke dalam kelas-kelas data yang menghasilkan hasil output yang sama.

2. METODE

Untuk mendapatkan hasil pengujian yang sesuai dengan yang diharapkan maka penerapan metode penelitian yang tepat harus dipilih secara matang. Pemilihan metode ini sangat penting karena dapat mempengaruhi validitas dan reliabilitas penelitian. Penelitian ini menggunakan metode penelitian kualitatif melalui pendekatan teknik pengujian *white box* dan *black box*. Strategi pengujian untuk white box menerapkan metode *Halstead's Metric* sedangkan strategi pengujian untuk *black box* menggunakan *Equivalence Partitioning*.

Adapun langkah-langkah penelitian yang dilakukan dapat dilihat pada Gambar 1. Penelitian ini dilakukan dengan empat tahapapan utama yaitu, Analisis Fungsional dan Spesifikasi, Perencanaan Pengujian, Pelaksanaan Pengujian, serta Hasil Pengujian.



Gambar 1. Tahapan Penelitian

1. Analisis Fungsional dan spesifikasi

   Langkah analisis spesifikasi dan fungsional merupakan tahap penting dalam pengujian perangkat lunak Website Admin Travel Booking ini. Analisis spesifikasi melibatkan pemahaman mendalam terhadap persyaratan sistem yang telah ditetapkan, termasuk fitur-fitur yang harus ada dan perilaku yang diharapkan dari sistem. Peneliti akan memeriksa dokumen spesifikasi sistem, seperti dokumen kebutuhan pengguna (*user requirement document*) dan dokumen desain sistem, serta melakukan

komunikasi dengan pemangku kepentingan terkait untuk memastikan pemahaman yang jelas terhadap kebutuhan sistem. Sementara itu, analisis fungsional akan fokus pada evaluasi fungsionalitas sistem, termasuk pengujian fitur-fitur kunci seperti login, setting, dan ganti password. Peneliti akan mengidentifikasi secara sistematis skenario pengujian untuk masing-masing fitur, menentukan input yang diperlukan, dan mengantisipasi hasil yang diharapkan.

2. Perencanaan Pengujian
   Langkah perencanaan pengujian merupakan bagian penting dalam memastikan keberhasilan pengujian perangkat lunak sistem *Travel Booking* berbasis website. Perencanaan dimulai dengan pembuatan desain tes yang komprehensif, yang mencakup identifikasi fitur-fitur utama yang akan diuji, metode pengujian yang akan digunakan, serta langkah-langkah yang akan diambil dalam pelaksanaan pengujian, termasuk pembuatan Flow *Graph*, analisis komplesitas logika, serta pencarian jalur logika. Selanjutnya, standarisasi hasil yang diharapkan adalah langkah kunci dalam menentukan kriteria kelulusan atau kegagalan pengujian. Peneliti akan menetapkan kriteria yang jelas dan terukur untuk mengevaluasi apakah fitur-fitur sistem berfungsi sebagaimana mestinya. Terakhir, merancang skenario kasus uji melibatkan pembuatan serangkaian langkah atau situasi yang mencakup penggunaan fitur-fitur sistem secara praktis dan realistis. Setiap skenario kasus uji akan dirancang untuk menguji berbagai kemungkinan interaksi pengguna dengan sistem serta menguji batasan dan kinerja sistem dalam berbagai situasi yang mungkin terjadi.
3. Pelaksanaan Pengujian
   Dalam tahap pelaksanaan pengujian, skenario kasus uji yang telah dirancang dengan cermat akan diimplementasikan dalam lingkungan pengujian yang sesuai dengan spesifikasi. Setiap skenario kasus uji akan dijalankan secara berurutan, dan hasil dari setiap pengujian akan dicatat dengan teliti. Selama proses pengujian, tim pengujian akan memonitor kinerja sistem dan mencatat setiap hasil yang diperoleh, termasuk hasil yang positif maupun negatif. Penting untuk memastikan bahwa setiap kasus uji dijalankan sesuai dengan prosedur yang telah ditetapkan dan bahwa lingkungan pengujian memenuhi persyaratan yang diperlukan untuk memastikan validitas hasil pengujian. Selama pelaksanaan pengujian, akan dilakukan evaluasi terhadap kesesuaian hasil yang diperoleh dengan hasil yang diharapkan, sebagaimana yang telah ditetapkan dalam rencana pengujian. Hasil yang diperoleh dari pengujian akan dianalisis secara menyeluruh untuk mengevaluasi kualitas dan kinerja sistem Travel Booking berbasis website, serta untuk mengidentifikasi potensi masalah atau kekurangan yang perlu diperbaiki.
4. Evaluasi Hasil
   Evaluasi pengujian merupakan tahap penting dalam proses pengembangan perangkat lunak sistem Travel Booking berbasis website. Pada tahap ini, hasil-hasil pengujian yang telah diperoleh akan dinilai secara menyeluruh untuk mengevaluasi kualitas dan kinerja sistem. Evaluasi ini mencakup penilaian terhadap hasil uji yang telah dilakukan, termasuk kemampuan sistem dalam memenuhi persyaratan fungsional dan spesifikasi yang telah ditetapkan. Setiap hasil uji akan dianalisis dengan teliti untuk menentukan apakah fitur-fitur sistem telah berfungsi sebagaimana mestinya, serta untuk mengidentifikasi potensi masalah atau kekurangan yang perlu diperbaiki. Selain itu, pada tahap evaluasi ini juga akan dilakukan perbandingan antara hasil uji yang diperoleh dengan hasil yang diharapkan berdasarkan rencana pengujian. Hal ini bertujuan untuk mengevaluasi sejauh mana sistem telah memenuhi ekspektasi dan kebutuhan pengguna. Berdasarkan hasil evaluasi ini, kesimpulan pengujian akan dibuat untuk merangkum temuan-temuan utama yang diperoleh selama proses pengujian. Kesimpulan ini akan mencakup ringkasan tentang kualitas dan kinerja sistem, serta rekomendasi untuk perbaikan atau pengembangan lanjutan jika diperlukan.

3. HASIL DAN PEMBAHASAN

3.1. Persiapan Pengujian

Sebelum memulai pengujian, sangat penting untuk dapat menetapkan tujuan yang jelas. Tujuan utama dari pengujian ini adalah untuk mengevaluasi kualitas dan kinerja website admin travel booking. Secara khusus pengujian ini berfokus pada identifikasi bug atau kesalahan potensial yang bisa muncul pada sistem aplikasi, serta menilai kepatuhan standar kualitas yang ditetapkan sesuai spesifikasi fungsional. Setelah menetapkan tujuan langkah selanjutnya adalah mengidetifikasi fitur kunci dari website admin travel booking yang akan diuji. Fitur-fitur tersebut meliputi login admin, manajemen pengguna (tambah dan edit pengguna), daftar pengguna, serta tambah kereta api. Dengan mengidentifikasi fitur-fitur ini, pengujian dapat difokuskan pada area penting dan relevan.

Langkah berikutnya adalah menyusun skenario pengujian yang mencakup berbagai situasi yang mungkin terjadi. Penyusunan skenario uji tersebut didasarkan pada spesifikasi fungsional yang telah ditetapkan sebelumnya. Skenario uji tersebut dirancang hanya untuk pengujian black box. Sedangkan untuk white box fokus pada identifikasi kemungkinan kemunculan bug.

Untuk mendapatkan hasil pengujian yang sesuai dengan spesifikasi, diperlukan lingkungan pengujian yang sesuai pula. Diperlukan konfigurasi perangkat lunak dan perangkat keras agar pengujian dapat berjalan sesuai dengan yang direncanakan pada *test plan*. Pengujian dilakukan pada lingkungan yang telah dipersiapkan yaitu:

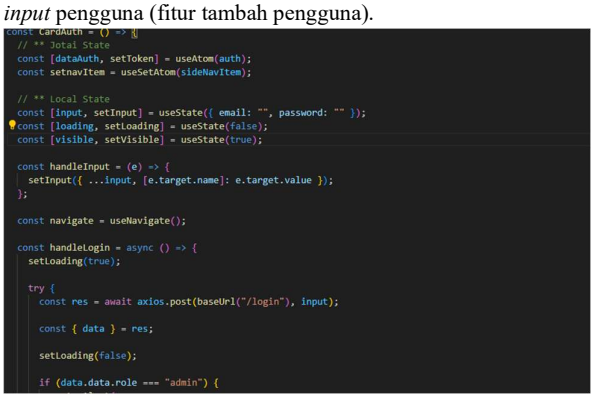
1. Perangkat Keras (*Hardware*)
   Perangkat keras yang digunakan adalah 1 unit laptop dengan spesifikasi:
   a) Processor : Intel® Core™ i7-10510U Processor 1.8 GHz (8M Cache, up to 4.9 GHz)
   b) Intel : 512GB M.2 NVMe™ PCIe® 3.0 SSD

https://doi.org/10.25077/TEKNOSI.v11i1.2025.67-77

2. Perangkat Lunak (*Software*)
    a) Visual Studio Code: *Software* penyunting kode sumber sistem aplikasi
    b) *Web Browser*: Dalam penelitian ini menggunakan Google Chrome
    c) Sistem Operasi: Windows 11

3.2. Perencanaan Pengujian Halstead's Metric

Pengujian *white box* merupakan pengujian perangkat lunak yang berfokus pada alur program dengan melihat sintaks dalam sumber kode program secara menyeluruh. Sehingga demikian, berikut ini pada Gambar 2 adalah hasil tangkapan layar dari kode sumber untuk website admin travel booking. Bagian tangkapan layar tersebut merupakan sebagian dari kode sumber pada laman *input* pengguna (fitur tambah pengguna).



Gambar 2. *Capture* Kode Sumber Laman Tambah Pengguna

Selain itu terlihat pada gambar 3, juga merupakan hasil tangkapan layar dari kode sumber untuk laman login. Dalam pengujian whitebox menggunakan teknik *Halstead's Metric*, perhitungan jumlah operator dan operand menjadi titik mula pengujian. Walaupun pengujian dengan teknik ini bukan pengujian yang mengedepankan alur program, namun teknik ini mampu digunakan untuk memprediksi jumlah *bug* yang mungkin muncul. Sehingga tujuan tester untuk dapat memprediksi kemungkinan bug dari pengujian *white box* dengan menjelajah alur program dapat ditemukan. Prediksi bug yang ditemukan merupakan hasil temuan dalam memperkirakan penggunaan operator dan operand yang terdapat dalam kode sumber. Sehingga apabila terdapat operator dan operand yang didefinisikan tetapi tidak digunakan atau sebaliknya, tidak didefinisikan namun digunakan dalam sintaks dapat menimbulkan potensi *bug* pada aplikasi tersebut.

Gambar 3. *Capture* Kode Sumber Laman Login

Untuk memudahkan pengembang melakukan *testing* dengan lebih efisien dan jelas, juga untuk menentukan tujuan serta target pengujian maka perlu dibuat terlebih dahulu *test plan*. Berikut *test plan* pada pengujian *white box* :

Tabel 1. *Test Plan* pada Pengujian *White Box*

Item Uji	Butir Uji
Fitur Tambah Pengguna	Untuk mengetahui prediksi bugs yang muncul saat menambahkan data pengguna
Fitur Edit Pengguna	Untuk mengetahui prediksi bugs yang muncul saat edit data pengguna
Fitur Tambah Kereta Api	Untuk mengetahui prediksi bugs yang muncul saat menambahkan data kereta api
Fitur Daftar Pengguna	Untuk mengetahui prediksi bugs saat web memunculkan daftar pengguna
Fitur Login	Untuk mengetahui prediksi bugs saat web memlakukan login

3.3. Pelaksanaan Pengujian White box

Setelah membuat *test plan*, selanjutnya adalah menghitung tiap-tiap komponen pada *Halstead's metric*. Berikut ini merupakan hasil dari penerapan uji *Halstead's metric* pada kode sumber website admin travel booking seperti terlihat pada tabel 1 di bawah ini. Aplikasi website admin travel booking mendapatkan hasil pengujian dimana untuk *vocabulary* terbanyak terdapat pada fitur tambah kereta, sedangkan untuk *length* (panjang program) juga ada pada fitur tambah kereta api. Begitupun untuk *volume*, fitur tambah kereta api menjadi yang tertinggi. Sedangkan untuk *Difficulty*, fitur tambah pengguna menjadi yang tertinggi. Artinya fitur tambah pengguna dari sisi kesulitan program dalam menjalankan operator dan operannya lebih rumit dibanding laman fitur lainnya.

Tabel 1. Hasil Pengujian *Halstead's Metric*

	Tambah Penggun a	**Edit**	**Daftar Penggun a**	**Login**	**Tambah Kereta Api**
Vocabulary	21	37	28	44	45
Length	124	84	114	197	265
Volume	545,5	437,5	548,2	1.076	1.371
Difficulty	19	11,05	7,5	8	16,04
Level	0,051	0,09	0,13	0,125	0,379
Effort	10.365,7	4.832	4.112	8.613	8.613
Time to Program	575.872	268,4	226,4	478,5	478,5
Number of Bugs Expected	0,134	0,1458	0,1828	0,3589	0,457

Jika dilihat pada hasil perhitungan *Time to program*, laman fitur tambah pengguna mempunyai nilai yang tertinggi ini bisa dikatakan bahwa waktu yang diperlukan untuk menerapkan pemrograman pada laman ini cenderung lebih lama dibandingkan dengan fitur lainnya. Berbanding terbalik, pada fitur daftar pengguna menjadi yang tercepat dengan hasil perhitungan yang paling rendah.

Dari tabel 1 diatas dapat disimpulkan bahwa setelah dilakukan pengujian menggunakan *Halstead's metric*, fitur tambah kereta api merupakan fitur yang memiliki tingkat prediksi kemunculan *bug* paling tinggi dari fitur yang lain yaitu sebesar 0,457 (45,7%). Sedangkan fitur tambah pengguna merupakan fitur dengan prediksi kemunculan bug yang paling kecil yaitu sebesar 0,134 (13,4%). Untuk lebih jelasnya lihatlah pada tabel 2 berikut untuk dapat melihat rata-rata prediksi kemunculan bugs pada setiap fitur dengan menggunakan formula statistika :

Tabel 2. Presentase Hasil Uji

Halaman	Prediksi Munculnya Bug
Tambah Pengguna	13,4%
Edit Pengguna	14,58%
Tambah Kereta Api	45,7%
Daftar Pengguna	18,28%
Login	35,89%
Jumlah Fitur	5
Rata-rata	**25,59%**

Setelah dilakukannya pengujian menggunakan *Halstead's metric* rata-rata prediksi kemunculan bugs pada setiap fitur yang ada adalah sebesar 25,59%. Hal yang bisa dilakukan untuk mengurangi tingginya tingkat prediksi kemunculan bugs adalah dengan *refactoring code*. *Refactoring code* adalah sebuah usaha untuk memperbaiki struktur kode jika ditemukan sebuah kode yang kompleks dan sulit dipahami, hal ini dapat membantu mengurangi potensi munculnya bugs dan menjadi mudah di *maintenance* [17].

3.4. Perencanaan pengujian Equivalence Partitioning

Pengujian dilakukan hanya pada fitur-fitur yang dianggap kritis dan mempunyai bugs lebih banyak muncul di tahapan debugging. Dalam hal ini, ruang lingkup dan batasan pengujian pada tahapan ini meliputi: 1) pengujian ini hanya menguji beberapa fitur yang dipilih yaitu fitur login, tambah pengguna, tambah kereta api, dan edit pengguna. 2) hanya berfokus pada fungsionalitas dari setiap fitur yang telah didefinisikan ditahapan analisis spesifikasi dan fungsi dari sistem aplikasi yang dikembangkan. 3) pada pengujian *blackbox* hanya mencakup penggunaan strategi *functional testing* dan tidak tidak melibatkan kode sumber.

Spesifikasi fungsional yang terdapat pada Tripease yaitu:

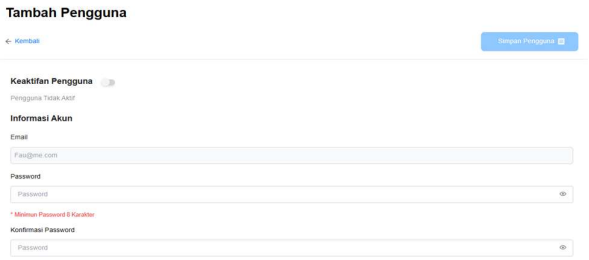
1. Dashboard : Memiliki fungsi untuk menampilkan rekapan tentang berapa jumlah pengguna, hotel, dan transaksi yang terjadi
2. Stasiun : Memiliki fungsi/fitur untuk melihat stasiun yang ada, menambahkan stasiun, mengedit stasiun, menghapus data stasiun
3. Kereta api : Memiliki fungsi/fitur untuk melihat kereta api yang ada, menambahkan kereta api, mengedit kereta api, menghapus data kereta api
4. Pengguna : Memiliki fungsi/fitur untuk melihat pengguna yang ada, menambahkan pengguna, mengedit pengguna, menghapus data pengguna
5. Tiket kereta api : Memiliki fungsi/fitur untuk melihat tiket kereta api yang ada, menambahkan tiket kereta api, menghapus tiket kereta apii
6. Pesanan kereta api : Memiliki fungsi/fitur untuk melihat pesanan yang terjadi, detail pemesan
7. Pesanan hotel : memiliki fungsi/fitur untuk melihat pesanan hotel yang terjadi, detail pemesan.

Berikut ini terlihat pada Tabel 3 deskripsi modul uji sebagai bagian dari perencanaan *test*.

Tabel 3. Modul Uji

Modul Uji	Tujuan	Input yang diperlukan	Output yang diharapkan
Tambah Pengguna	Untuk menambahkan data baru pengguna	Data pengguna	Dapat menambahkan data pengguna
Edit Pengguna	Untuk mengedit data pengguna	Data pengguna	Dapat mengedit data pengguna
Tambah Kereta Api	Untuk menambah data kereta api	Data kereta api	Dapat menambahkan data kereta api
Login	Untuk mengakses sistem	Data pengguna	Dapat melakukan login

Adapun antarmuka untuk setiap fitur yang diuji dapat dilihat pada Gambar 4 sampai dengan Gambar 7. Gambar tersebut merupakan hasil tangkapan layar laman web untuk fitur tambah pengguna, Edit Pengguna, Tambah kereta api, dan login.



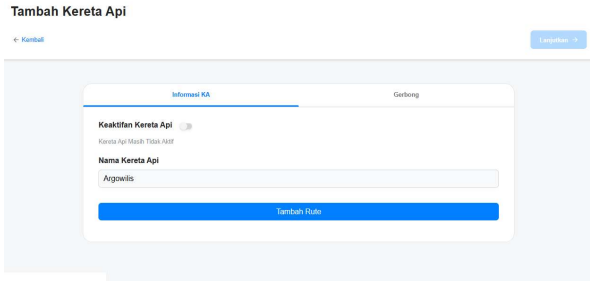
Gambar 4. Halaman Tambah Pengguna

Gambar 4 memperlihatkan halaman web untuk melayanai aktifitas tambah pengguna pada sistem aplikasi. Pengujian pada halaman ini adalah membuat partisi masukan yang tersimpan pada basisdata dan partisi masukan yang gagal tersimpan pada basis data serta validitas masukan. Halaman tersebut dirancang sesuai dengan spesifikasi fungsional dari halaman tambah pengguna. user diharuskan mengisi form yang dibutuhkan oleh sistem. Semua form bersifat *mandatory*, tombol tambah pengguna dapat di klik dan data tersimpan ke dalam sistem jika semua form inputan telah terisi dengan benar, kemudian akan diarahkan ke halaman daftar pengguna.

https://doi.org/10.25077/TEKNOSI.v11i1.2025.67-77

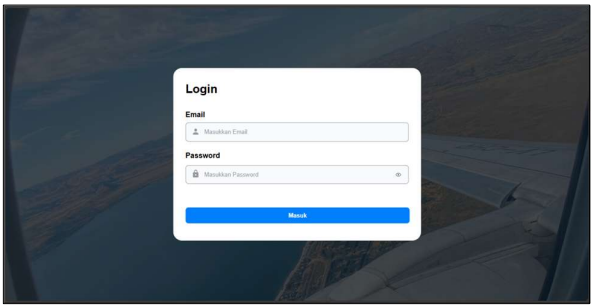
Gambar 5. Halaman Edit Pengguna

Pengujian pada halaman edit pengguna (lihat Gambar 5), sama halnya dengan halaman tambah pengguna, membagi pengujian kedalam partisi validitas masukan serta keberhasilan dalam menyimpan data pada basisdata.



Gambar 6. Halaman Tambah Kereta Api

Test case yang dirancang untuk menguji halaman tambah kereta api seperti terlihat pada Gambar 6 tidak berbeda dengan partisi pada fitur-fitur sebelumnya. Pada halaman ini diuji terkait validitas inputan dan keberhasilan fitur untuk mengantarkan hasil yang sesuai dengan harapan. Begitupun untuk halaman login pada Gambar 7. Pengujian *black box* menggunakan teknik *equivalence partitioning* membatas lingkup pengujian berdasarkan partisi yang bisa dibentuk sesuai dengan spesifikasi fungsional pada laman login.



Gambar 7. Halaman Login

Test plan merupakan sebuah list atau dokumen yang berisi tujuan serta target pengujian dalam ruang lingkup iterasi, item-item yang menjadi sasaran pengujian, pendekatan yang akan diambil, sumber daya yang dibutuhkan [18]. Berikut terlihat pada Tabel 4 *test plan* untuk masing-masing modul uji.

Tabel 4. Test Plan pada Pengujian Black Box

Item Uji	Butir Uji
Fitur Tambah Pengguna	Validitas Inputan Keberhasilan Fitur
Fitur Edit Pengguna	Validitas Inputan Keberhasilan Fitur
Fitur Tambah Kereta Api	Validitas Inputan Keberhasilan Fitur
Fitur Login	Validitas Inputan Keberhasilan Fitur

3.5. Pelaksanaan Pengujian Black box

Setelah menyusun *test plan* dan mempersiapkan lingkungan pengujian, langkah selanjutnya adalah melakukan eksekusi *test case* pada setiap fitur. *Test case* adalah sebuah rincian informasi mengenai tahapan pengujian, komponen yang akan diuji, bagaimana cara mengujinya, data apa yang digunakan dan apa hasil yang di harapkan [19]. Pada *test plan* yang telah didefinisikan sebelumnya, fokus pengujian hanya pada 2 jenis butir uji untuk setiap fitur yang akan kami uji, yaitu validitas inputan dan keberhasilan fitur. Validitas inputan mempunyai arti yaitu sejauh mana ketepatan serta kecermatan inputan dalam melakukan fungsinya [20].

Eksekusi pengujian didasarkan pada hasil dari *test plan*. Didalam *test case* didefinisikan secara detail terkait item yang diuji, langkah pengujian, serta hasil yang diharapkan. Pada pelaksanaan test dilakukan perbandingan antara hasil yang diharapkan dengan hasil yang diciptakan dari setiap skenario yang dijalankan. Bernilai valid jika sesuai dengan hasil yang diharapkan dan bernilai Invalid jika tidak sesuai dengan hasil yang dirapakan.
Setelah melakukan eksekusi *test case*, hasil uji yang didapatkan yaitu seperti berikut pada Tabel 6:

Tabel 5. Hasil Pengujian Equivalence Partitions

Halaman	Jumlah *Test Case*	*Test Case* Berhasil	*Test Case* Tidak Berhasil	Butir Uji Pada Test Plan	
				Validitas Inputan	Keberhasilan Fitur
Tambah Pengguna	12	9	3	×	✓
Edit Pengguna	7	5	2	×	✓
Tambah Kereta Api	8	8	0	✓	✓
LogIn	5	5	0	✓	✓

Dari Tabel 6 diatas bisa kita lihat fitur tambah pengguna memiliki kegagalan *test case* yang paling banyak yaitu dari 12 *test case* ada 3 *test case* yang tidak berhasil. Fitur tambah pengguna memiliki validitas inputan yang tidak berhasil. Kemudian ada fitur tambah kereta api dan fitur login yang memiliki tingkat keberhasilan 100% pada validitas inputan dan keberhasilan fitur. Hal yang dapat dilakukan utuk meningkatkan keberhasilan fitur adalah dengan memperbaiki kesalahan pada setiap butir *test case* yang tidak berhasil.

Berdasarkan hasil skenario kasus uji dengan menggunakan *Halstead's Metric* dan *Equivalance Partitioning* dapat diketahui

bahawa sistem yang dibangun masih terdapat celah kecacatan. Berdasarkan hasil pengujian didapatkan (lihat Tabel 7 berikut):

Tabel 7. Simpulan Hasil Uji

Halaman	Keberhasilan pada Fitur		Prediksi Munculnya Bugs
	Validitas Inputan	Keberhasilan Fitur	
Tambah Pengguna	×	✓	13,4%
Edit Pengguna	×	✓	14,58%
Tambah Kereta Api	✓	✓	45,7%
Daftar Pengguna	-		18,28%
Login	✓	✓	35,89%

Dapat disimpulkan dari hasil pengujian yang dilakukan, bahwa pada tingkat keberhasilan fitur, fitur tambah pengguna dan edit pengguna memiliki tingkat keberhasilan yang rendah karena keduanya tidak berhasil pada butir uji validitas inputan. Pada prediksi munculnya bugs, fitur tambah kereta api merupakan fitur yang tinggi prediksi kemunculan bugs nya. Fitur daftar pengguna tidak bisa kami uji menggunakan black box dengan metode equivalance partitions karena halaman daftar pengguna tidak memiliki kolom inputan. Dengan dilakukanya pengujian pada website tersebut, kami dapat menyarankan perbaikan untuk melakukan *refactoring code* dan memperbaiki validitas inputan yang ada pada setiap fitur.

4. KESIMPULAN

Sesuai dengan tujuan penelitian yaitu untuk memverifikasi bahwa program website admin travel booking telah sesuai dengan spesifikasi fungsional, dapat disimpulkan bahwa dengan menggunakan dua pendekatan yaitu *white box* dan *black box*, kualtas *software* dapat diuji dari *script* dan logika program serta ketapatan fungsi dari masukan. Berdasarkan hasil uji menggunakan *Halstead's metric* secara fungsional program masih bisa berjalan dengan baik hanya saja ditemukan kemungkinan munculnya *bug* sebesar 0,457 pada fitur tambah kereta api. Namun berdasarkan hasil uji menggunakan *Equivalence Partitioning* mendapatkan hasil uji dengan tingkat keberhasilan 100%. Dengan demikian pengujian yang dihasilkan oleh *Halstead's metric* merupakan prediksi terhadap kemungkinan munculnya *bug* pada program walaupun secara fungsional, program aplikasi dapat bejalan sesuai dengan spesifikasi masukan yang telah ditetapkan. Walaupun hasil pengujian telah menunjukan hasil yang baik diperlukan perluasan cakupan pengujian untuk membantu pengembang untuk mendapatkan kualtas perangkat lunak yang lebih handal. Pengujian bisa dilakukan untuk seluruh fitur yang terdapat pada program aplikasi website admin travel booking sebelum dilakukan tahapan implemntasi dan rilis.

DAFTAR PUSTAKA

[1] Dhaifullah, I. R., Salsabila, A. A., & Yaqin, M. A. (2022). Survei Teknik Pengujian Software. Journal Automation Computer Information System, 2(1), 31-38.

[2] Debiyanti, D., Sutrisna, S., Budrio, B., Kamal, A. K., & Yulianti, Y. (2020). Pengujian Black Box pada Perangkat Lunak Sistem Penilaian Mahasiswa Menggunakan Teknik Boundary Value Analysis. Jurnal Informatika Universitas Pamulang, 5(2), 162-166.

[3] Prabowo, M. (2020). Metodologi Pengembangan Sistem Informasi. LP2M Press IAIN Salatiga.

[4] Fahrurrozi, I., & Azhari, S. N. (2012). Proses Pemodelan Software dengan metode waterfall dan extreme programming: studi perbandingan. Jurnal Online STMIK EL Rahma, 1-10.

[5] Sulistyanto, H. (2017). Urgensi Pengujian pada Kemajemukan Perangkat Lunak dalam Multi Perspektif. Komuniti: Jurnal Komunikasi dan Teknologi Informasi, 6(1), 65-74.

[6] Marlina, L. A., Harliana, H., & Wibowo, S. S. (2023). Pengujian Sistem Informasi Perpustakaan Dengan Teknik Equivalence Partitioning di SMA Nurul Muttaqin Albarokah. Journal Automation Computer Information System, 3(2), 137-145.

[7] Pranata, F. N., Kom, S., Pradana, F., & Astoto, T, "Pengembangan Sistem Perhitungan Kompleksitas Kode Sumber Berdasarkan Metrik *Halstead* Dan *Cyclomatic* Complexity", Program Doktoral, Universitas Brawijaya, Kediri, 2016.

[8] Hasibuan, A. N., & Dirgahayu, T. (2021). Pengujian dengan Unit Testing dan Test case pada Proyek Pengembangan Modul Manajemen Pengguna. AUTOMATA, 2(1).

[9] Alfisahrin, Sa'Diyah N. N, (2012). "Pendekatan White Box Testing Untuk Menentukan Kualitas Perangkat Lunak Dengan Menggunakan Bahasa Pemrograman C++." Jurnal Khatulistiwa Informatika, vol. 14, no. 1, doi:10.31294/p.v14i1.3380.

[10] A. A. Arwaz, T. Kusumawijaya, R. Putra, K. Putra, and A. Saifudin, "Pengujian Black Box pada Aplikasi Sistem Seleksi Pemenang Tender Menggunakan Teknik Equivalence Partitions," J. Teknol. Sist. Inf. dan Apl., vol. 2, no. 4,p. 130, 2019, doi: 10.32493/jtsi.v2i4.3708.

[11] Atmaja, R. G., Priyambadha, B., & Pradana, F. (2019). Pembangunan Kakas Bantu Untuk Mengukur Maintainability Index Pada Perangkat Lunak Berdasarkan Nilai Halstead Metrics dan McCabe's Cyclomatic Complexity: English. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 3(3), 2167-2172.

https://doi.org/10.25077/TEKNOSI.v11i1.2025.67-77

[12] Hanief, S., Jepriana, I. W., & Kom, S. (2020). Konsep Algoritme dan Aplikasinya dalam Bahasa Pemrograman C++. Penerbit Andi.
[13] Setiawan, D. (2017). Buku sakti pemrograman web: html, css, php, mysql & javascript. Anak Hebat Indonesia.
[14] Hadiansyah, F. (2020). Optimasi Maintainability Menggunakan Metode Clean Code pada Sistem Informasi Museum Mandhilaras (Doctoral dissertation, Universitas Komputer Indonesia).
[15] Adi, R. P., Koswara, Y., Tashika, J., Devi, Y., & Saifudin, A. (2020). Pengujian Black Box pada Aplikasi Pertokoan Minimarket Menggunakan Metode Equivalence Partitioning. Jurnal Teknologi Sistem Informasi dan Aplikasi ISSN, 2654, 3788.
[16] Cantika, P. D. (2017). Rancang Bangun Aplikasi E-learning Untuk Pembelajaran Agama Islam Berbasis Android (Studi Kasus MIN 6 Bandar Lampung).
[17] Longsari, K. (2017). The Impact of Design Patterns in Refactoring Technique to Measure Performance Efficiency. Unpublished Master Degree Thesis for Software Engineering.
[18] Dhaifullah, I. R., Salsabila, A. A., & Yaqin, M. A. (2022). Survei Teknik Pengujian Software. Journal Automation Computer Information System, 2(1), 31-38.
[19] Kosasih, Y., & Cahyono, A. B. (2021). Automation Testing Tool Dalam Pengujian Aplikasi The Point Of Sale. AUTOMATA, 2(1).
[20] Saputra, A. (2020). CAMI: Aplikasi Uji Validitas dan Reliabilitas Instrumen Penelitian Berbasis Web. Yayasan Ahmar Cendekia Indonesia.

NOMENKLATUR

N1 = total semua operator yang muncul
N2 = total semua operan yang muncul
n = Vocabulary of the program
n1 = jumlah operator unik
n2 = jumlah operand unik
V = Volume of the program
N = nilai kalkulasi length of the program
n = nilai kalkulasi vocabulary of the program
D = Difficulty
E = Effort
D = nilai dari kalkulasi Difficulty
V = nilai kalkulasi Volume of the program
B = Number of bugs expected in the program
V = dari kalkulasi Volume of the program

BIODATA PENULIS

Fauziah Nur Syifa
Saat ini sedang menempuh pendidikan di Universitas Komputer Indonesia jurusan Sistem Informasi. Memiliki minat dalam pengembangan sistem informasi khususnya dalam bidang *frontend development.*

Trisana Nurul Anzali Nabil
Saat ini sedang menjalani pendidikan jurusan Sistem Informasi di Universitas Komputer Indonesia. Memiliki minat dibidang *data mining*.

Derrel Hendi Arifin
Saat ini sedang menjalani pendidikan jurusan Sistem Informasi di Universitas Komputer Indonesia. Memiliki minat dalam bidang pemograman sistem informasi khususnya *backend development*

Rangga Sidik
Saat ini menjabat sebagai dosen di Program Studi Sistem Informasi, Universitas Komputer Indonesia, memiliki gelar magister Sistem informasi dan Master of Computer Engineering. Minat penelitian utama berkisar pada pengembangan sistem informasi inovatif untuk meningkatkan produktivitas organisasi, Software Testing, Knowledge Management dan pengembangan KMS, serta Penerapan Enterprise Arsitektur pada organisasi.

LAMPIRAN

Contoh *test case* untuk pengujian halaman daftar pengguna

Id Pengujian	**Deskripsi**	***Test Case***	**Test Step**	**Hasil yang Diharapkan**
TC_TambahPengguna_01	Mengisikan semua filed inputan dengan sesuai	Email : billyjoe@gmail.com Password : “testing1234” Konfirmasi Password : “testing1234” Nama Pengguna : “Billy Joe” Tanggal Lahir : “12/01/2010” Nomor Handphone : “089630016”	Mengisi semua form inputan kemudian klik tombol simpan pengguna	Menampilkan popup konfirmasi bahwa data telah benar

Id Pengujian	Deskripsi	*Test Case*		Test Step	Hasil yang Diharapkan
TC_Tambah Pengguna_02	Mengosongkan inputan email	Email : “ ”		Mengisi semua form inputan kecuali inputan email kemudian klik tombol simpan pengguna	Tombol tambah pengguna tidak bisa di klik
TC_Tambah Pengguna_03	Mengosongkan inputan password	Password : “ “		Mengisi semua form inputan kecuali inputan password kemudian klik tombol simpan pengguna	Tombol tambah pengguna tidak bisa di klik
TC_Tambah Pengguna_04	Mengosongkan inputan konfirmasi password	Konfirmasi password : “ “		Mengisi semua form inputan kecuali inputan konfirmasi password kemudian klik tombol simpan pengguna	Tombol tambah pengguna tidak bisa di klik
TC_Tambah Pengguna_05	Mengosongkan inputan nama pengguna	Nama pengguna : “ “		Mengisi semua form inputan kecuali inputan nama pengguna kemudian klik tombol simpan pengguna	Tombol tambah pengguna tidak bisa di klik
TC_Tambah Pengguna_06	Mengosongkan inputan tanggal lahir	Tanggal lahir : “ “		Mengisi semua form inputan kecuali inputan tanggal lahir kemudian klik tombol simpan pengguna	Tombol tambah pengguna tidak bisa di klik
TC_Tambah Pengguna_07	Mengosongkan inputan nomor handphone	Nomor handphone : “ “		Mengisi semua form inputan kecuali inputan handphone kemudian klik tombol simpan pengguna	Tombol tambah pengguna tidak bisa di klik
TC_Tambah Pengguna_08	Menginputkan email yang tidak sesuai dengan format email	Email : “admin~.com”		Mengisi semua form inputan dan inputan email diisi tidak sesuai kemudian klik tombol simpan pengguna	Tombol tambah pengguna tidak bisa di klik
TC_Tambah Pengguna_09	Mengosongkan inputan nama pengguna	Nama pengguna : “ “		Mengisi semua form inputan kecuali inputan nama pengguna kemudian klik tombol simpan pengguna	Tombol tambah pengguna tidak bisa di klik

https://doi.org/10.25077/TEKNOSI.v11i1.2025.67-77

Id Pengujian	Deskripsi	*Test Case*	Test Step	Hasil yang Diharapkan
TC_Tambah Pengguna_10	Menginputkan password kurang dari 8 karakter	Password : “ test “	Menginputkan semua inputan form dan mengisi password yang tidak sesuai kemudian klik tombol simpan pengguna	Menampilkan pemberitahuan “Minimun Password 8 Karakter”
TC_Tambah Pengguna_11	Menginputkan konfirmasi password yang tidak sesuai dengan password	Password : “ test 1234 ” Konfirmasi password : “ testtest “	Mengisi semua form inputan dan mengisi inputan konfirmasi password yang tidak sama dengan password kemudian klik tombol simpan pengguna	Menampilkan pemberitahuan “Password Tidak Sesuai”
TC_Tambah Pengguna_12	Menginputkan nomor handphone yang tidak sesuai	Nomor handphone : “0896 ”	Mengisi semua form dan mengisi form inputan nomor hanphone yang tidak sesuai kemudian klik tombol simpan pengguna	Menampilkan alert “Nomor handphone tidak sesuai”