Pengujian Fungsional *Website Crusher Report* Berbasis *Machine Learning* Menggunakan Metode *Robustness Testing*

Chelsea Ayu Adhigiadany[1], Kartika Maulida Hindrayani[2], Dwi Arman Prasetya[3]
[1,2,3]Universitas Pembangunan Nasional "Veteran" Jawa Timur Jl. Rungkut Madya No. 1, Kecamatan Gunung Anyar, Surabaya
kartika.maulida.ds@upnjatim.ac.id

Abstrak: *Website* dan *Machine Learning* menjadi kebutuhan penting perusahaan dalam rangka meningkatkan efektivitas kinerja. Salah satu implementasi integrasi *website* dengan *Machine Learning* adalah *website Crusher Report* milik PT XYZ. *Website* yang dirancang dengan memanfaatkan LARS, PostgreSQL, dan Flask ini sudah diuji secara ketangkasan model dalam memprediksi. Penelitian ini bertujuan untuk menguji keandalan *website Crusher Report* sebagai *user interface* milik PT XYZ menggunakan pendekatan *Black Box Testing* dengan metode *Robustness Testing*. Skenario pengujian yang digunakan yaitu dengan memberikan *input* diluar ketentuan *website*. Hasil pengujian menunjukkan bahwa *website* mampu menangani seluruh *input* tidak valid dengan baik melalui notifikasi kesalahan dan pengaturan nilai *input* otomatis, menghasilkan tingkat keberhasilan pengujian sebesar 100%. Temuan ini menunjukkan bahwa *website Crusher Report* efektif dalam mendeteksi dan mengelola kesalahan input, serta layak digunakan sebagai platform pendukung operasional *crusher* PT XYZ.
Kata Kunci : Pengujian *Website, Machine Learnin, Robustness Testing*

Abstract: *Website and Machine Learning have become important needs of companies to improve performance effectiveness. One of the implementations of website integration with Machine Learning is PT XYZ's Crusher Report website. This website, which was designed by utilizing LARS, PostgreSQL, and Flask, has been tested regarding model agility in predicting. This research aims to test the reliability of the Crusher Report website as a user interface owned by PT XYZ using the Black Box Testing approach with the Robustness Testing method. The test scenario is used to provide input outside the website provisions. The test results show that the website can handle all invalid inputs properly through error notifications and automatic input value settings, resulting in a test success rate of 100%. This finding shows that the Crusher Report website is effective in detecting and managing input errors, and is feasible to use as a support platform for PT XYZ crusher operations.*
Keywords: *Website Testing, Machine Learning, Robustness Testing*

1. Pendahuluan

Pada era digital saat ini, *website* menjadi salah satu kebutuhan perusahaan. *Website* dinilai mampu meningkatkan efektivitas perusahaan karena dapat dimanfaatkan sebagai sarana komunikasi internal maupun eksternal perusahaan. Pemanfaatan *website* sebagai sarana komunikasi internal dapat berperan sebagai sarana monitoring pekerjaan divisi, atau bahkan sebagai alat pendukung dalam pengambilan keputusan. Dengan adanya *website*, aktivitas yang sebelumnya dilakukan secara terpisah kini dapat diintegrasikan dalam satu platform.

Selain *website*, pemanfaatan *Machine Learning* dalam meningkatkan efektivitas perusahaan juga semakin pesat. *Machine Learning* merupakan salah satu cabang dari *Artificial Intelligence* (Idhom dkk., 2023). *Artificial Intelligence* merupakan suatu penciptaan sistem komputer yang mampu meniru kecerdasan manusia berdasarkan

kerangka berpikir yang telah disesuaikan dengan pola berpikir manusia dalam menyelesaikan permasalahan yang dihadapinya (Idhom dkk., 2023; Prasetya dkk., 2020). *Machine Learning* dapat digunakan dalam proses klasterisasi, prediksi, dan klasifikasi (Damaliana dkk., 2024; Muhaimin dkk., 2023; Prasetya dkk., 2025).

Website Crusher Report milik PT XYZ merupakan contoh *website* yang mengimplementasikan *Machine Learning* di dalamnya. Seperti halnya penelitian yang dilakukan oleh Lisanthoni, dkk pada *website* XportID (Lisanthoni dkk., 2025), *website Crusher Report* menggunakan Flask sebagai *backend*, PostgreSQL sebagai RDBMS, dan *Least Angle Regression* (LARS) sebagai metode *Machine Learning* dalam memprediksi durasi operasi *crusher*. PostgreSQL dipilih sebagai RDBMS pada *website Crusher Report* karena kelebihannya yang stabil dan efisien untuk operasi CRUD pada data berukuran besar dibanding dengan MySQL sebagai RDBMS yang umum digunakan pada website (Lutfi dkk., 2022). Metode LARS dipilih sebagai metode *Machine Learning* yang digunakan karena metode tersebut memiliki keunggulan berupa seleksi fitur yang diperoleh dari nilai koreksi absolut terdekat dengan fitur target (W. Zhao dkk., 2017). Selain pemilihan PostgreSQL dan LARS yang didasarkan pada penelitian terdahulu, pemilihan Flask sebagai *backend* juga didasarkan pada penelitian terdahulu. Kornienko, dkk dalam penelitiannya menggunakan Flask untuk mempercepat pengembangan aplikasi karena di dalam Flask telah terdapat kumpulan kode program yang digunakan untuk pengembangan *website* (Kornienko dkk., 2021). Tidak hanya itu, Zhao, dkk dalam penelitiannya juga menggunakan Flask dalam membangun *website* yang memanfaatkan *Big Data* (Z. Zhao dkk., 2023).

Pengujian yang harus dilalui dalam perancangan *website Crusher Report* yaitu pengujian terhadap kinerja model dan fungsi dari *website* sebagai *user interface*. Pengujian terhadap metode *Machine Learning* umumnya menggunakan pengujian akurasi dengan koefisien determinasi (R^2) dan nilai RMSE (Hindrayani dkk., 2020). Hasil yang didapatkan dari pengujian tersebut yaitu nilai R^2 sebesar 92,7% dan RMSE sebesar 1862,86. Dari hasil kedua pengujian tersebut, model yang dihasilkan dinilai baik dalam memprediksi durasi operasi *crusher*. Pengujian terkait fungsi *website Crusher Report* sebagai *user interface* belum dilakukan.

Pengujian terkait fungsi *website Crusher Report* sebagai *user interface* dapat dilakukan dengan pendekatan *Black Box Testing*. Diyasa, dkk menggunakan pendekatan *Black Box Testing* untuk melakukan pengujian terhadap kesalahan fungsional (Diyasa & Putra, 2023; Putri dkk., 2025). Pendekatan *Black Box Testing* memiliki beberapa metode, yaitu *Equivalence Partitioning, Boundary Value Analysis, Fuzzing Testing, Cause Effect Graph, Orthogonal Array Testing, All Pair Testing, Robustness Testing,* dan *State Transition Testing* (Triady dkk., 2023). Berdasarkan kebutuhan terkait deteksi kesalahan fungsional dari *website Crusher Report*, metode *Robustness Testing* dipilih sebagai metode pengujian *website* sebagai *user interface*. Pengujian dengan *Robustness Testing* dapat dilakukan dengan memberikan masukan diluar ketentuan yang telah dibuat pada *website Crusher Report*.

Metode *Robustness Testing* dipilih dalam penelitian ini karena fokus pengujian berada pada kemampuan *website Crusher Report* dalam menangani kesalahan input yang sering terjadi pada penggunaan sistem operasional harian. Berbeda dengan metode *Black Box Testing* lainnya, *Robustness Testing* secara spesifik menguji respon sistem terhadap input yang berada di luar ketentuan, seperti kesalahan format, ketidaklengkapan data, dan penggunaan notasi numerik yang tiak sesuai.

Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk mengevaluasi fungsional *website Crusher Report* milik PT XYZ sebagai *user interface* melalui pendekatan *Black Box Testing* dengan metode *Robustness Testing*. Evaluasi

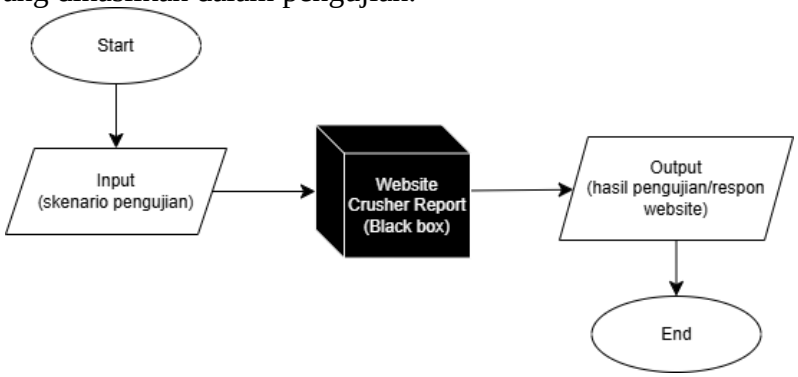
difokuskan pada kemampuan sistem dalam menangani *input* yang tidak sesuai ketentuan, menampilkan notifikasi kesalahan secara tepat, serta menjaga konsistensi data yang disimpan dan diporses oleh sistem. Hasil pengujian ini diharapkan dapat memberikan gambaran terkait tingkat ketahanan *website* terhadap kesalahan *input* serta mendukung kelayakan penggunaannya sebagai platform pendukung operasi *crusher*.

2. Metode Penelitian

Black Box Testing merupakan metode pengujian perangkat lunak yang berfokus pada evluasi fungsional sistem berdassarkan kesesuaian antara *input* dan *output* tanpa mengakses struktur internal atau kode program (Desikan & Ramesh, 2006; Myers dkk., 2012; Pressman, 2016). Pendekatan ini relevan digunakan pada *website* berbasis *Machine Learning* karena pengujian difokuskan pada validasi *input* dan respon sistem terhadap kesalahan pengguna yang secara langsung dapat memengaruhi kualitas data serta hasil prediksi model. Salah satu metode dari pengujian *Black Box Testing* yaitu *Robustness Testing*. *Robustness Testing* merupakan pengujian perangkat lunak dengan memberikan *input* diluar ketentuan sistem untuk mendeteksi peringatan dari kesalahan yang ditimbulkan (Abdi & Nursari, 2022). *Robustness Testing* dilakukan untuk menentukan proses atau bagian dari perangkat lunak yang terdeteksi gagal untuk dilakukan pemulihan (Naik & Tripathy, 2008). Adapun cara pengujiannya dengan memberikan *input* di luar ketentuan sistem, seperti melewati batas dari ketentuan, tipe data yang tidak sesuai, hanya mengisi beberapa dari keseluruhan *form* yang wajib diisi, atau menggunakan kombinasi tanda yang tidak sesuai dengan ketentuan *website*.

Pengujian dilakukan pada lima halaman *website Crusher Report* yang memiliki form *input*, yaitu halaman *input* lokasi *crusher* utara, selatan, barat, timur, serta halaman prediksi durasi operasi *crusher*. Adapun skenario pengujian yang digunakan meliputi pengisian form dengan *input* tidak lengkap, pengisian *field* numerik dengan kombinasi tanda koma dan titik, penggunaan tanda titik sebagai penanda desimal, serta pengisian angka desimal dengan lebih dari dua digit di belakang koma.

Berdasarkan skenario tersebut, penelitian ini menggunakan 18 *test case*, yang terdiri atas empat *test case* pada masing-masing halaman *input* lokasi *crusher* dan tiga *test case* pada halaman prediksi durasi operasi *crusher*. Setiap *test case* dievaluasi menggunakan kriteria keberhasilan (*pass*/valid) dan kegagalan (*fail*/tidak valid). Pengujian dinyatakan *pass* apabila *website* mampu mendeteksi *input* tidak valid, menampilkan notifikasi kesalahan yang sesuai, serta mencegah data tidak sesuai ketentuan tersimpan atau diproses oleh sistem. Sebaliknya, pengujian dinyatakan *fail* apabila *website* menerima *input* tidak valid tanpa memberikan peringatan atau validasi. Adapun siklus dari pengujian ini seperti pada Gambar 1 yang terdiri dari *input* pengujian dan *output* yang dihasilkan dalam pengujian.



Gambar 1. Alur Pengujian Black Box Testing

3. Hasil dan Pembahasan

3.1 Hasil

3.1.1 Halaman *input* lokasi *crusher* utara

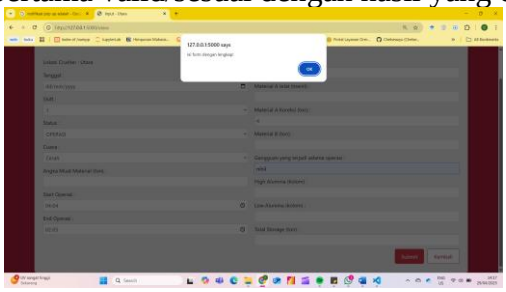
Pada halaman *input* lokasi *crusher* utara, terdapat beberapa *field* yang wajib diisi. Masing-masing *field* memiliki ketentuan yang berbeda sesuai dengan *input* yang diinginkan. Skenario pengujian yang dilakukan pada halaman *input* lokasi *crusher* utara meliputi pengisian beberapa *field*, mengisi *field* tipe numerik dengan kombinasi tanda titik dan koma, mengisi *field* tipe numerik dengan tanda titik sebagai penanda desimal, dan mengisi *field* tipe numerik dengan angka di belakang koma lebih dari 2. Adapun hasil dari pengujian *Robustness Testing* pada halaman *input* lokasi *crusher* seperti pada Tabel 1.

Tabel 1. Hasil Pengujian Halaman *Input* Lokasi *Crusher* Utara

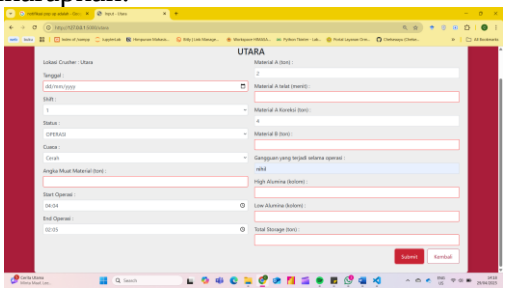
No	Skenario Pengujian	*Test Case*	Hasil yang Diharapkan	Keterangan (Valid/Tidak valid)
1	Hanya mengisi beberapa *field*	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* utara 3. Isi beberapa *field* 4. Klik tombol *submit*	Muncul notifikasi *pop up* dengan pesan "Isi form dengan lengkap!", *field* yang belum terisi akan berwarna merah.	Valid
2	Mengisi *field* tipe numerik dengan kombinasi tanda koma dan titik	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* utara 3. Isi *field* tipe numerik dengan kombinasi tanda koma dan titik	Tanda kedua yang digunakan akan tidak muncul pada *field* pengisian	Valid
3	Mengisi *field* tipe numerik dengan tanda titik sebagai penanda desimal	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* utara 3. Isi field tipe numerik dengan tanda titik sebagai penanda desimal	Muncul notifikasi *pop up* dengan pesan "*Please enter a valid value. The two nearest valid value are* 12 *and* 12,01"	Valid
4	Mengisi *field* tipe numerik dengan angka di belakang koma lebih dari 2	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* utara 3. Isi *field* tipe numerik dengan angka di	Muncul notifikasi "*Please enter a valid value. The two nearest valid value are* (nilai *input* dengan 2 angka di belakang koma) *and* (nilai *input* dengan angka kedua di	Valid

	belakang koma lebih dari 2	belakang koma ditambah 1" pada *field* yang salah
4.	Klik tombol *submit*	

Berdasarkan Tabel 1, halaman *input* lokasi *crusher* utara dilakukan pengujian dengan empat skenario. Skenario pertama yaitu memasukkan *input* pada beberapa *field* yang tersedia. Hasil yang diharapkan dari skenario tersebut adalah muncul notifikasi *pop up* pada bagian atas *website* dengan pesan "Isi form dengan lengkap!". Selain itu, bingkai pada *field* yang kosong akan berubah warna menjadi merah. Adapun hasil dari pengujian seperti pada Gambar 2 dan Gambar 3. Dari kedua gambar tersebut, dapat disimpulkan bahwasanya skenario pengujian pertama valid/sesuai dengan hasil yang diharapkan.



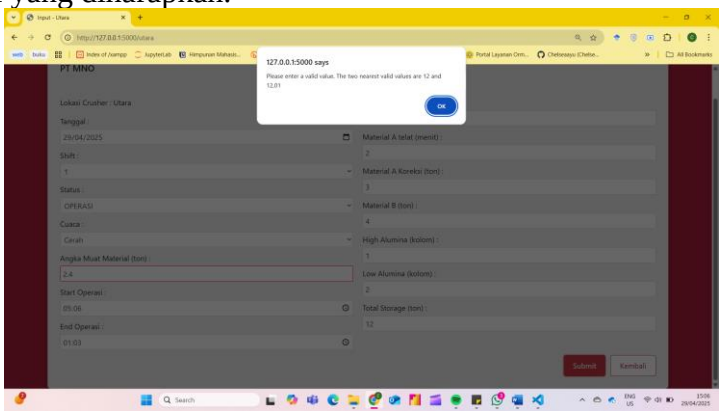
Gambar 2. Notifikasi Pop Up Pengujian Pertama



Gambar 3. Bingkai Field Yang Berubah Menjadi Warna Merah Pada Pengujian Pertama

Skenario pengujian yang kedua yaitu mengisi *field* tipe numerik dengan kombinasi tanda koma dan titik. Hasil yang diharapkan yaitu tanda kedua yang diinputkan tidak tercetak pada halaman *website* meskipun diinputkan berulang kali. Dari pengujian kedua, didapatkan hasil valid yang berarti hasil pengujian sesuai dengan hasil yang diharapkan.

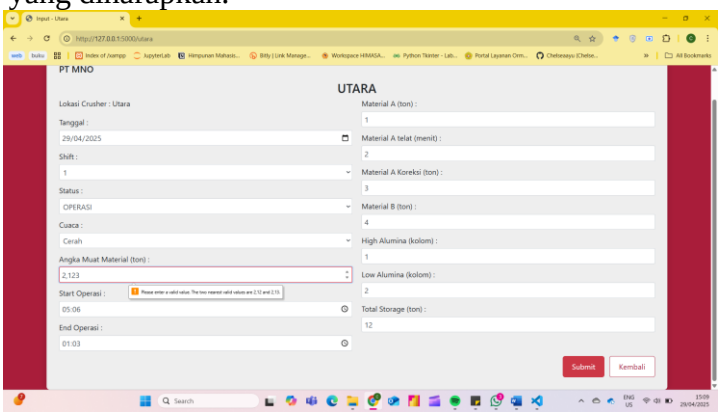
Skenario pengujian ketiga yaitu menuliskan angka desimal dengan tanda titik sebagai penanda desimal pada *field* bertipe numerik. Hasil yang diharapkan yaitu muncul notifikasi *pop up* "*Please enter a valid value. The two nearest valid value are* 12 *and* 12,01". Selain itu, bingkai pada *field* yang salah akan berubah warna menjadi merah. Adapun hasil pengujian seperti pada Gambar 4. Dari Gambar 4 dapat disimpulkan bahwa skenario pengujian ketiga valid/sesuai dengan hasil yang diharapkan.



Gambar 4. Notifikasi Pop Up Pada Pengujian Kedua

Skema pengujian keempat yaitu menuliskan *input* dengan angka di belakang koma lebih dari dua pada *field* bertipe numerik. Hasil yang diharapkan

yaitu muncul notifikasi "*Please enter a valid value. The two nearest valid value are* (nilai *input* dengan 2 angka di belakang koma) *and* (nilai *input* dengan angka kedua di belakang koma ditambah 1)" pada *field* yang salah dan bingkai berubah warna menjadi merah. Adapun hasil dari pengujian seperti pada Gambar 5. Dari Gambar 5 dapat disimpulkan bahwa skenario pengujian keempat valid/sesuai dengan hasil yang diharapkan.



Gambar 5. Notifikasi Kesalahan Field Pada Pengujian Keempat

3.1.2 Halaman *input* lokasi *crusher* selatan

Halaman kedua yang dilakukan pengujian *Robustness Testing* merupakan halaman *input* lokasi *crusher* selatan. Karena halaman *input* lokasi *crusher* selatan memiliki kesamaan *field* yang wajib diisi, skenario pengujian yang diberikan juga sama dengan skenario yang diberikan pada halaman *input* lokasi *crusher* utara. Adapun hasil dari pengujian halaman *input* lokasi *crusher* selatan seperti pada Tabel 2.

Tabel 2. Hasil Pengujian Halaman *Input* Lokasi *Crusher* Selatan

No	Skenario Pengujian	*Test Case*	Hasil yang Diharapkan	Keterangan (Valid/Tidak valid)
1	Hanya mengisi beberapa field	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* selatan 3. Isi beberapa *field* 4. Klik tombol *submit*	Muncul notifikasi *pop up* dengan pesan "Isi form dengan lengkap!", *field* yang belum terisi akan berwarna merah.	Valid
2	Mengisi *field* tipe numerik dengan kombinasi tanda koma dan titik	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* selatan 3. Isi *field* tipe numerik dengan kombinasi tanda koma dan titik	Tanda kedua yang digunakan akan tidak muncul pada *field* pengisian	Valid

3	Mengisi *field* tipe numerik dengan tanda titik sebagai penanda desimal	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* selatan 3. Isi field tipe numerik dengan tanda titik sebagai penanda desimal	Muncul notifikasi *pop up* dengan pesan "*Please enter a valid value. The two nearest valid value are* 12 *and* 12,01"	Valid
4	Mengisi *field* tipe numerik dengan angka di belakang koma lebih dari 2	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* selatan 3. Isi *field* tipe numerik dengan angka di belakang koma lebih dari 2 4. Klik tombol *submit*	Muncul notifikasi "*Please enter a valid value. The two nearest valid value are* (nilai *input* dengan 2 angka di belakang koma) *and* (nilai *input* dengan angka kedua di belakang koma ditambah 1" pada *field* yang salah	Valid

Seperti pada Tabel 2, skenario pengujian yang diberikan pada halaman *input* lokasi *crusher* selatan sama seperti pada halaman *input* lokasi *crusher* utara. Dari keempat pengujian yang dilakukan, didapatkan hasil bahwasanya keempatnya valid. Hal tersebut berarti bahwa hasil pengujian sesuai dengan hasil yang diharapkan. Terkait gambaran hasil pengujian halaman *input* lokasi *crusher* selatan sesuai dengan gambar hasil pengujian halaman *input* lokasi *crusher* utara.

3.1.3 Halaman *input* lokasi *crusher* barat

Tidak hanya halaman *input* lokasi *crusher* selatan yang memiliki kesamaan *field* dengan halaman *input* lokasi *crusher* utara, halaman *input* lokasi *crusher* barat juga memiliki kesamaan. Dengan begitu ketentuan pengisian *field* dan juga skenario pengujian yang diberikan juga sama. Adapun hasil dari pengujian halaman *input* lokasi *crusher* barat seperti pada Tabel 3.

Tabel 3. Hasil Pengujian Halaman *Input* Lokasi *Crusher* Barat

No	Skenario Pengujian	*Test Case*	Hasil yang Diharapkan	Keterangan (Valid/Tidak valid)
1	Hanya mengisi beberapa field	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* barat 3. Isi beberapa *field* 4. Klik tombol *submit*	Muncul notifikasi *pop up* dengan pesan "Isi form dengan lengkap!", *field* yang belum terisi akan berwarna merah.	Valid
2	Mengisi *field* tipe numerik dengan kombinasi	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* barat	Tanda kedua yang digunakan akan tidak muncul pada *field* pengisian	Valid

	tanda koma dan titik	3. Isi *field* tipe numerik dengan kombinasi tanda koma dan titik		
3	Mengisi *field* tipe numerik dengan tanda titik sebagai penanda desimal	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* barat 3. Isi field tipe numerik dengan tanda titik sebagai penanda desimal	Muncul notifikasi *pop up* dengan pesan "*Please enter a valid value. The two nearest valid value are* 12 *and* 12,01"	Valid
4	Mengisi *field* tipe numerik dengan angka di belakang koma lebih dari 2	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* barat 3. Isi *field* tipe numerik dengan angka di belakang koma lebih dari 2 4. Klik tombol *submit*	Muncul notifikasi "*Please enter a valid value. The two nearest valid value are* (nilai *input* dengan 2 angka di belakang koma) *and* (nilai *input* dengan angka kedua di belakang koma ditambah 1" pada *field* yang salah	Valid

Seperti pada Tabel 3, skenario pengujian yang diberikan pada halaman *input* lokasi *crusher* barat juga sama seperti pada halaman *input* lokasi *crusher* utara. Dari keempat pengujian yang dilakukan, didapatkan hasil bahwasanya keempatnya valid. Hal tersebut berarti bahwa hasil pengujian sesuai dengan hasil yang diharapkan. Terkait gambaran hasil pengujian halaman *input* lokasi *crusher* barat juga akan sesuai dengan gambar hasil pengujian halaman *input* lokasi *crusher* utara.

3.1.4 Halaman *input* lokasi *crusher* timur

Tidak hanya halaman *input* lokasi *crusher* selatan dan barat yang memiliki kesamaan *field* dengan halaman *input* lokasi utara, halaman *input* lokasi *crusher* timur juga memiliki kesamaan. Hal tersebut karena keempatnya berfungsi sebagai halaman *input* data *crusher* harian berdasarkan lokasi pada *database* PostgreSQL. Adapun hasil dari pengujian halaman *input* lokasi *crusher* timur seperti pada Tabel 4.

Tabel 4. Hasil Pengujian Halaman *Input* Lokasi *Crusher* Timur

No	Skenario Pengujian	*Test Case*	Hasil yang Diharapkan	Keterangan (Valid/Tidak valid)
1	Hanya mengisi beberapa field	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* timur 3. Isi beberapa *field*	Muncul notifikasi *pop up* dengan pesan "Isi form dengan lengkap!", *field* yang belum terisi akan berwarna merah.	Valid

		4. Klik tombol *submit*		
2	Mengisi *field* tipe numerik dengan kombinasi tanda koma dan titik	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* timur 3. Isi *field* tipe numerik dengan kombinasi tanda koma dan titik	Tanda kedua yang digunakan akan tidak muncul pada *field* pengisian	Valid
3	Mengisi *field* tipe numerik dengan tanda titik sebagai penanda desimal	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* timur 3. Isi field tipe numerik dengan tanda titik sebagai penanda desimal	Muncul notifikasi *pop up* dengan pesan "*Please enter a valid value. The two nearest valid value are* 12 *and* 12,01"	Valid
4	Mengisi *field* tipe numerik dengan angka di belakang koma lebih dari 2	1. Tekan *Input* pada bagian menu 2. Pilih lokasi *crusher* timur 3. Isi *field* tipe numerik dengan angka di belakang koma lebih dari 2 4. Klik tombol *submit*	Muncul notifikasi "*Please enter a valid value. The two nearest valid value are* (nilai *input* dengan 2 angka di belakang koma) *and* (nilai *input* dengan angka kedua di belakang koma ditambah 1" pada *field* yang salah	Valid

Halaman *input* lokasi *crusher* timur merupakan pengujian terakhir dengan skenario dan hasil yang sama seperti halaman *input* lokasi *crusher* utara. Berdasarkan Tabel 4, berarti bahwa hasil dari keempat pengujian pada halaman *input* lokasi *crusher* timur juga sesuai dengan hasil yang diharapkan dari skenario pengujian.

3.1.5 Halaman prediksi durasi *crusher*

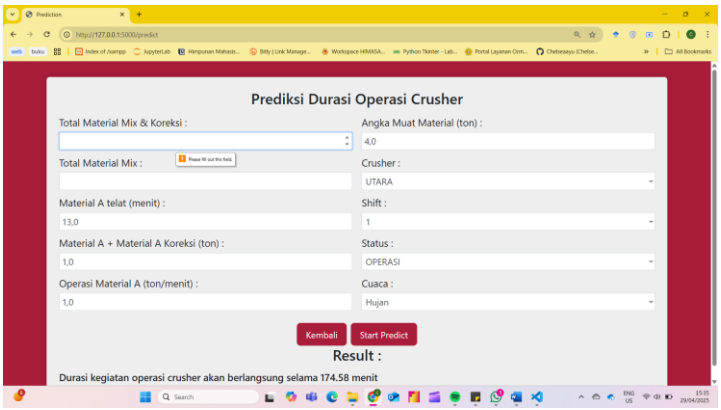
Halaman prediksi durasi operasi *crusher* juga memiliki *field* yang wajib diisi untuk menghasilkan prediksi durasi operasi *crusher*. Pengujian pada halaman prediksi durasi operasi *crusher* memiliki kemiripan dengan halaman lainnya, hanya saja pada halaman ini lebih ringkas dengan terdapat perubahan otomatis jika mengisi tanda lebih dari satu ataupun tidak sesuai dengan ketentuan. Adapun hasil dari pengujian halaman prediksi durasi *crusher* seperti pada Tabel 5.

Tabel 5. Hasil Pengujian Halaman Prediksi Durasi *Crusher*

No	Skenario Pengujian	*Test Case*	Hasil yang Diharapkan	Keterangan (Valid/Tidak valid)

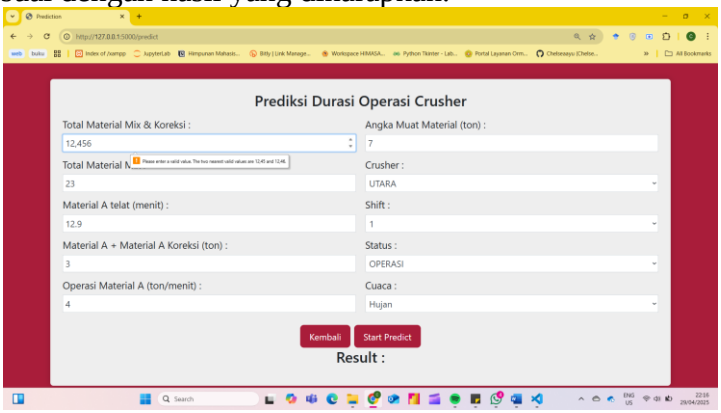
1	Hanya mengisi beberapa *field*	1. Tekan *Predict* pada bagian menu 2. Isi beberapa *field* 3. Tekan tombol *start predict*	Muncul notifikasi "*Please fill out this field*" pada *field* yang belum terisi	Valid
2	Mengisi *field* tipe numerik dengan angka di belakang koma lebih dari 2	1. Tekan *Predict* pada bagian menu 2. Isi *field* tipe numerik dengan kombinasi tanda koma dan titik 3. Tekan tombol *start predict*	Muncul notifikasi "*Please enter a valid value. The two nearest valid value are* (nilai *input* dengan 2 angka di belakang koma) *and* (nilai *input* dengan angka kedua di belakang koma ditambah 1" pada *field* yang salah	Valid
3	Mengisi *field* tipe numerik dengan tanda titik sebagai penanda desimal	1. Tekan *Predict* pada bagian menu 2. Isi *field* tipe numerik dengan tanda titik sebagai penanda desimal 3. Tekan tombol *start predict*	Tidak muncul notifikasi *error* melainkan sistem akan mengubah tanda titik menjadi koma	Valid

Berbeda dengan empat halaman sebelumnya, pengujian pada halaman prediksi durasi *crusher* hanya terdiri dari tiga pengujian seperti yang terlihat pada Tabel 5. Skenario pertama yaitu memasukkan *input* pada beberapa *field* yang tersedia. Hasil yang diharapkan dari skenario tersebut adalah muncul notifikasi pada *field* yang kosong dengan pesan "*Please fill out this field*". Adapun hasil dari pengujian seperti pada Gambar 6. Dari gambar tersebut, dapat disimpulkan bahwasanya skenario pengujian pertama valid/sesuai dengan hasil yang diharapkan.



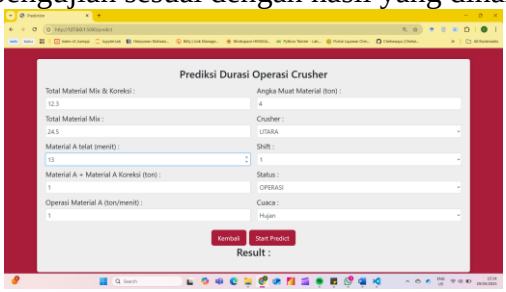
Gambar 6. Notifikasi Pada Field Yang Belum Terisi Pada Pengujian Pertama

Skenario pengujian kedua yaitu mengisi *field* tipe numerik dengan angka di belakang koma lebih dari satu. Hasil yang diharapkan yaitu muncul notifikasi "*Please enter a valid value. The two nearest valid value are* (nilai *input* dengan 2 angka di belakang koma) *and* (nilai *input* dengan angka kedua di belakang koma ditambah 1)" pada *field* yang salah. Adapun hasil dari pengujian kedua yaitu pada Gambar 7. Dari pengujian kedua, didapatkan hasil valid yang berarti hasil pengujian sesuai dengan hasil yang diharapkan.

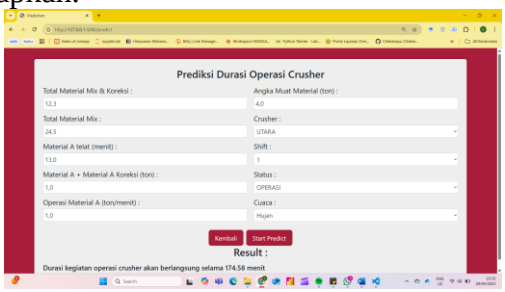


Gambar 7. Notifikasi Kesalahan Pada Field Yang Memiliki Angka Desimal Lebih Dari Dua

Skenario pengujian ketiga yaitu mengisi *field* tipe numerik dengan tanda titik sebagai penanda desimal. Hasil yang diharapkan yaitu tidak muncul notifikasi *error* melainkan sistem akan mengubah tanda titik menjadi tanda koma. Adapun hasil dari pengujian kedua yaitu Gambar 8 merupakan tampilan sebelum menekan tombol *start predict* dan Gambar 9 merupakan tampilan setelah menekan tombol *start predict*. Dari pengujian kedua, didapatkan hasil valid yang berarti hasil pengujian sesuai dengan hasil yang diharapkan.



Gambar 8. Sebelum Pengujian Ketiga



Gambar 9. Setelah Pengujian Ketiga

3.2 Pembahasan

Berdasarkan 18 skema pengujian yang diterapkan pada lima halaman *website Crusher Report*, hasil pengujian menunjukkan bahwa sistem mampu menangani seluruh skenario input tidak valid dengan menampilkan pesan kesalahan sesuai dengan ketentuan yang telah dirancang. Pengujian ini dilakukan menggunakan pendekatan *Black Box Testing* dengan fokus pada *Robustness Testing*, yaitu pengujian ketahanan sistem terhadap berbagai bentuk *input* yang tidak sesuai dengan spesifikasi fungsional sistem. Pendekatan ini sejalan dengan konsep pengujian berbasis spesifikasi yang dijelaskan oleh (Sommerville, 2016), di mana sistem diuji tanpa mempertimbangkan struktur internal kode program.

Notifikasi kesalahan pada pengujian kelengkapan *input* berperan penting dalam mencegah tersimpannya data bernilai *null* pada basis data PostgreSQL. Hal ini

sejalan dengan (Sommerville, 2016) yang menegaskan bahwa pengujian terhadap kondisi kesalahan (*error conditions*) merupakan bagian penting dari *validation testing* untuk memastikan keandalan sistem. Kelengkapan data menjadi krusial karena data operasional *crusher* harian digunakan sebagai *input* utama dalam proses prediksi durasi operasi. Model prediksi berbasis *Least Angle Regression* (LARS) membutuhkan sepuluh variabel *input*, sehingga ketidaklengkapan data berpotensi menghasilkan prediksi yang tidak valid dan menurunkan akurasi model, sebagaimana juga didukung oleh (Hastie dkk., 2009).

Pengujian penggunaan tanda koma sebagai penanda desimal menunjukkan bahwa sistem mampu menjaga konsistensi format numerik sesuai standar lokal yang digunakan. Hasil ini sejalan dengan temuan (Geron, 2019) yang menyatakan bahwa inkonsistensi format numerik dapat menyebabkan kesalahan pada proses pemrosesan data dan berdampak pada performa model *Machine Learning*. Dengan demikian, validasi format numerik menjadi bagian penting dalam menjaga kualitas data yang akan dianalisis.

Selain itu, pembatasan jumlah angka di belakang koma berfungsi untuk menjaga standarisasi nilai numerik sesuai kebijakan perusahaan.(Sommerville, 2016) menjelaskan bahwa sistem yang andal harus mampu membatasi dan mengendalikan variasi *input* agar tetap berada dalam batas yang dapat diterima oleh sistem. Pengaturan ini meningkatkan konsistensi data dalam basis data serta mendukung stabilitas proses analisis dan prediksi menggunakan model LARS.

Dengan demikian, hasil *Robustness Testing* pada *website Crusher Report* menunjukkan bahwa mekanisme validasi *input* yang diterapkan telah berjalan efektif dalam menjaga kualitas dan konsistensi data. Temuan ini konsisten dengan teori *Black Box Testing* dan *validation testing* yang dikemukakan oleh (Sommerville, 2016), serta mendukung keandalan *website Crusher Report* sebagai *user interface* operasional *crusher* PT XYZ yang terintegrasi dengan proses analisis dan prediksi berbasis *Machine Learning*.

4. Kesimpulan dan Saran

Berdasarkan hasil pengujian dengan menggunakan metode *Robustness Testing* pada *website Crusher Report* milik PT XYZ, didapatkan hasil bahwasanya *website* berhasil dalam menangani kesalahan *input* dengan 3-4 pengujian pada masing-masing halaman *website*. Pengujian pertama yaitu pada halaman *input* lokasi *crusher* utara. Dari empat pengujian, keempatnya mendapatkan hasil valid/sesuai dengan hasil yang diharapkan. Pengujian kedua yaitu pada halaman *input* lokasi *crusher* selatan. Dari empat pengujian, keempatnya mendapatkan hasil valid/sesuai dengan hasil yang diharapkan. Pengujian ketiga yaitu pada halaman *input* lokasi *crusher* barat. Dari empat pengujian, keempatnya mendapatkan hasil valid/sesuai dengan hasil yang diharapkan. Pengujian keempat yaitu pada halaman *input* lokasi *crusher* timur. Dari empat pengujian, keempatnya mendapatkan hasil valid/sesuai dengan hasil yang diharapkan. Pengujian kelima yang merupakan pengujian terakhir yaitu pada halaman prediksi durasi *crusher*. Dari tiga pengujian, ketiganya juga mendapatkan hasil valid/sesuai dengan hasil yang diharapkan. Dengan begitu, *website* dinilai dapat menangani kesalahan *input* dengan menampilkan pesan kesalahan sesuai dengan ketentuan dan dapat berfungsi dengan baik.

Daftar Pustaka

Abdi, N. F., & Nursari, S. R. C. (2022). Pengujian black box pada Website dengan Metode Robustness Testing (Studi kasus : Eiger Adventure). *Journal of Informatics and Advanced Computing (JIAC)*, *3*(2), 93–96.

Damaliana, A. T., Muhaimin, A., & Prasetya, D. A. (2024). FORECASTING THE OCCUPANCY RATE OF STAR HOTELS IN BALI USING THE XGBOOST AND SVR METHODS. *Journal Of Statistics*, *12*(1), 24–33. https://doi.org/https://doi.org/10.26714/jsunimus.12.1.2024.24-33

Desikan, S., & Ramesh, G. (2006). *Software Testing : Principle and Practice*. Pearson Education India.

Diyasa, I. G. S. M., & Putra, I. N. D. P. (2023). Blackbox Testing of Integrated Systems Evaluation of Learning in the OBE Ecosystem Based on Equivalence Partitions. *PROISRM*, *8*(1), 3–3.

Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (Second Edition). O'Reilly Media, Inc.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning : Data Mining, Inference, and Prediction* (Second Edition). Springer.

Hindrayani, K. M., Diyasa, I. G. S. M., Riyantoko, P. A., & Fahrudin, T. M. (2020). Studi Literatur Mengenai Prediksi Harga Saham Menggunakan Machine Learning. *Seminar Nasional Informatika Bela Negara (SANTIKA)*, *1*, 71–75. https://doi.org/https://doi.org/10.33005/santika.v1i0.20

Idhom, M., Prasetya, D. A., Riyantoko, P. A., Fahrudin, T. M., & Sari, A. P. (2023). Pneumonia Classification Utilizing VGG-16 Architecture and Convolutional Neural Network Algorithm for Imbalanced Datasets. *TIERS Information Technology Journal*, *4*(1), 73–82. https://doi.org/10.38043/tiers.v4i1.4380

Kornienko, D. V., Mishina, S. V., Shcherbatykh, S. V., & Melnikov, M. O. (2021). Principles of securing RESTful API web services developed with python frameworks. *Journal of Physics: Conference Series*, *2094*(3). https://doi.org/10.1088/1742-6596/2094/3/032016

Lisanthoni, A., Arman Prasetya, D., & Trimono, T. (2025). *XportID: Website for Clustering Indonesian Export Commodities by Destination Continent using Gaussian Mixture Model*. *9*(1), 104–118. https://doi.org/10.31764/jtam.v9i1.27500

Lutfi, M., Asrowardi, I., & Supriyatna, A. (2022). Migrasi Database Mysql Ke Postgresql Pada Aplikasi Sistem Evaluasi Dosen Oleh Mahasiswa (EDOM) Jurusan Ekonomi Dan Bisnis. *ROUTERS: Jurnal Sistem dan Teknologi Informasi*, 19–36. https://doi.org/10.25181/rt.v1i1.2699

Muhaimin, A., Wibowo, W., & Riyantoko, P. A. (2023). Multi-label Classification Using Vector Generalized Additive Model via Cross-Validation. *Journal of Information and Communication Technology*, *22*(4), 657–673. https://doi.org/10.32890/jict2023.22.4.5

Myers, G. J., Badgett, T., & Sandler, C. (2012). *The Art of Software Testing* (3 ed.). John Wiley & Sons.

Naik, K., & Tripathy, P. (2008). *Software Testing and Quality Assurance : Theory and Practice*. Wiley.

Prasetya, D. A., Nguyen, P. T., Faizullin, R., Iswanto, I., & Armay, E. F. (2020). Resolving the Shortest Path Problem using the Haversine Algorithm. *Journal of Critical Reviews*, *7*(1), 62–64. https://doi.org/10.22159/jcr.07.01.11

Prasetya, D. A., Sari, A. P., Idhom, M., & Lisanthoni, A. (2025). Optimizing Clustering Analysis to Identify High-Potential Markets for Indonesian Tuber Exports. *Indonesian*

Journal of Electronics, Electromedical Engineering, and Medical Informatics, *7*(1), 113–122. https://doi.org/10.35882/ijeeemi.v7i1.55

Pressman, R. S. (2016). *Software Engineering : A Practitioner's Approach* (7 ed.). McGraw-Hill.

Putri, A. E., Matahari, M., & Sahiruddin, S. (2025). Sistem Informasi Penjualan di Toko Lela Berbasis Website. *Jurnal PETISI*, *6*(1), 1–11. https://doi.org/https://doi.org/10.36232/jurnalpetisi.v6i1.311

Sommerville, I. (2016). *Software Engineering* (Tenth Edition). Pearson Education Limited.

Triady, D., Musdar, I. A., & Surasa, H. (2023). PENGUJIAN BLACKBOX PADA WEBSITE WORKER'S MENGGUNAKAN METODE EQUIVALENCE PARTITIONING. *Jurnal KHARISMA Tech*, *18*(1), 84–89. https://ccd-workers.com/

Zhao, W., Beach, T. H., & Rezgui, Y. (2017). Efficient least angle regression for identification of linear-in-the-parameters models. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *473*(2198). https://doi.org/10.1098/rspa.2016.0775

Zhao, Z., Li, Q., Zhao, Z., Zhang, H., Han, Q., & Ju, X. (2023). Design and Implementation of Power Big Data Platform. *Journal of Physics: Conference Series*, *2665*(1). https://doi.org/10.1088/1742-6596/2665/1/012001