Metode Blackbox Testing Dalam Pengujian Proyek Kemitraan: Penyesuaian Cetakan Kesepakatan Notadinas UMKM Dalam Proyek OSS RBA (Studi Kasus: PT Telkom Jakarta Selatan

1st Putri Ivani
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
putriivani04@gmail.com

2nd Renny Sukawati
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
renny@tass.telkomuniversity.ac.id

Anak Agung Gde Agung
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
agung@tass.telkomuniversity.ac.id

Persoalan regulasi perizinan usaha sering menjadi hambatan bagi pelaku usaha dan pemerintah di Indonesia, mengakibatkan birokrasi yang kompleks dan ketidakpastian hukum. Untuk mengatasi masalah ini, pemerintah mengembangkan sistem pelayanan berusaha berbasis elektronik yang dikenal sebagai *Online Single Submission* Berbasis Risiko (OSS-RBA). OSS-RBA bertujuan meningkatkan efisiensi, transparansi, dan akuntabilitas dalam proses perizinan usaha. Laporan ini meneliti kualitas fitur OSS-RBA menggunakan metode *black box testing* dengan teknik *equivalence partitioning*. *Black box testing* adalah metode pengujian perangkat lunak yang fokus pada validasi input dan output tanpa memperhatikan rincian internal system. Hasil pengujian menunjukkan bahwa OSS-RBA berfungsi dengan baik, memenuhi kebutuhan dan ekspektasi pengguna. Fitur-fitur seperti autentikasi pengguna, pencarian data kemitraan, dan penyimpanan informasi berjalan sesuai dengan harapan. Pengujian dilakukan dalam empat tahap: perencanaan, pembuatan test case, eksekusi pengujian, dan pelaporan hasil. Dengan menggunakan metode *black box testing* dan teknik *equivalence partitioning*, aplikasi OSS-RBA diuji secara menyeluruh untuk memastikan kinerja dan keandalannya sebelum implementasi publik

Kata kunci*— oss-rba, perizinan, *blackbox testing

I. PENDAHULUAN

Persoalan regulasi perizinan usaha menjadi salah satu tantangan utama yang dihadapi oleh para pelaku usaha dan pemerintah. Kompleksitas dan kerumitan regulasi seringkali menjadi hambatan dalam proses mendapatkan izin usaha, mengakibatkan birokrasi yang berat dan lamanya waktu yang dibutuhkan untuk mendapatkan persetujuan. Selain itu, peraturan yang ambigu atau bertentangan di berbagai tingkatan pemerintahan dapat menciptakan ketidakpastian hukum yang merugikan pelaku usaha. Sehingga persoalan ini dianggap menjadi penghambat pelaku usaha dalam menjalankan bisnis di Indonesia.

Pemerintah menciptakan sistem pelayanan berusaha berbasis elektronik sebagai langkah strategis untuk meningkatkan efisiensi, transparansi, dan kemudahan akses bagi pelaku usaha dalam mendapatkan izin berusaha. Sistem pelayanan berusaha berbasis elektronik disebut *Online Single Submission* Berbasis Risiko (OSS-RBA) yang didasarkan pada Peraturan Pemerintah Republik Indonesia No 24 Tahun 2018 tentang Pelayanan Perizinan Berusaha Terintegrasi Secara elektronik. Sistem ini juga membantu meningkatkan keterbukaan, karena informasi mengenai prosedur perizinan dan status pengajuan dapat diakses dengan mudah oleh para pelaku usaha. Selain itu, penggunaan system ini dapat meminimalkan potensi korupsi dan meningkatkan akuntabilitas, karena seluruh proses dapat tercatat secara transparan. Dengan demikian, OSS-RBA tidak hanya menciptakan lingkungan bisnis yang lebih kondusif tetapi juga mendukung visi pemerintah untuk mempercepat pertumbuhan ekonomi dan meningkatkan daya saing nasional.[1]

Saat ini OSS Berbasis Risiko (OSS-RBA) masih dalam tahap pengembangan dan perbaikan sistem dan fitur oleh PT. Telekomunikasi Indonesia Tbk. Agar aplikasi ini dapat digunakan sesuai dengan yang diharapkan maka harus dilakukan pengujian aplikasi. Untuk membuat aplikasi yang berkualitas perlu dilakukan pengujian, guna memastikan bahwa aplikasi tersebut memenuhi harapan dan tujuan pengguna. Pengujian aplikasi atau *testing* merupakan suatu metode dalam menilai tingkat fungsionalitas dalam suatu aplikasi perangkat lunak. *Testing* juga merupakan suatu alat ukur dalam menentukan atau meninjau kualitas, kinerja, ataupun keandalan dari suatu aplikasi perangkat lunak sebelum dilakukan implementasi di ranah publik [2]

Laporan ini bertujuan untuk mengetahui kualitas fitur yang ada pada OSS-RBA dengan menggunakan metode *black box*

testing. *Black box testing* memiliki banyak macam teknik pengujian salah satunya adalah teknik *equivalence partitioning*. *Teknik equivalence partitioning* yaitu teknik pengujian perangkat lunak yang melibatkan pembagian nilai input menjadi nilai-nilai valid dan tidak valid serta pemilihan perwakilan dari setiap data uji. Teknik ini mencoba yang mengungkap kesalahan pada sistem, sehingga sistem dapat berjalan sesuai dengan apa yang diharapkan. [3]

Berdasarkan latar belakang dan rumusan masalah yang telah diuraikan, penelitian ini memiliki beberapa batasan masalah yaitu:

1. Bagaimana mengetahui kualitas fitur yang ada dalam Kemitraan OSS-RBA sesuai dengan kebutuhan dan ekspektasi pengguna, serta dapat memudahkan pengguna dalam menggunakan aplikasi tersebut?
2. Bagaimana cara melakukan pengujian (*testing*) pada kemitraan OSS-RBA?

Tujuan dari laporan ini adalah mengetahui mengetahui kualitas OSS-RBA yang dikembangkan dengan menggunakan metode *black-box testing*. Pengujian dengan menggunakan *black-box testing* ini bertujuan untuk melihat program tersebut apakah sesuai dengan fungsi yang diinginkan tanpa mengetahui kode program yang dipakai. Pemilihan metode *black box* fokus kepada kebutuhan fungsional dari aplikasi, seseorang dapat mendefinisikan *test case* dan melakukan evaluasi pada kebutuhan fungsional aplikasi untuk mencari kesalahan dalam beberapa kategori yaitu fungsi yang tidak benar, kesalahan antarmuka, kesalahan struktur data, kesalahan performansi.

II. KAJIAN TEORI

A. Pengertian Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan proses yang digunakan untuk mengevaluasi kinerja atau kemampuan dari sebuah sistem, hal Ini dilakukan untuk memastikan bahwa sistem dapat berjalan dengan baik dan memenuhi kebutuhan yang telah ditetapkan sebelumnya. Melalui pengujian perangkat lunak, kita dapat memastikan bahwa perangkat lunak tersebut berfungsi dengan baik dan tidak mengalami kesalahan. Hal ini dilakukan untuk memastikan setiap bagian dari sistem dapat berjalan tanpa masalah dan sesuai dengan apa yang diinginkan oleh pengguna. [4]

B. Software Quality Assurance

Kualitas perangkat lunak bisa dilihat dari sejauh mana perangkat lunak tersebut memenuhi kebutuhan fungsional yang telah ditetapkan dan didokumentasikan sesuai dengan standar yang berlaku. perangkat lunak harus dapat melakukan fungsi-fungsi yang diinginkan oleh pengguna dengan baik, serta beroperasi dengan performa yang sesuai atau bahkan melampaui ekspektasi. [5]

C. Blackbox Testing

Pengujian blackbox testing fokus utamanya adalah memastikan kesesuaian *input* dan *output*, dan dilakukan tanpa memperhatikan rincian *internal* perangkat lunak. Tujuannya adalah mengidentifikasi kesalahan dalam berbagai aspek, termasuk fungsi yang tidak sesuai, antarmuka pengguna yang bermasalah, serta kesalahan dalam struktur data atau akses basis data *eksternal*. Dengan demikian, pengujian ini membantu menjamin bahwa perangkat lunak beroperasi sesuai dengan apa yang diharapkan. [6]

D. Tujuan Pengujian Perangkat Lunak

Tujuan pengujian perangkat lunak meliputi beberapa aspek penting. Pertama, pengujian bertujuan untuk meningkatkan kualitas sistem dengan memastikan bahwa perangkat lunak sesuai dengan persyaratan desain yang telah ditentukan. Proses *debugging*, yang merupakan bagian dari pengujian, dilakukan secara intensif untuk menemukan dan memperbaiki cacat desain yang mungkin ada. Kedua, pengujian juga berperan dalam validasi dan verifikasi perangkat lunak. Hal ini dilakukan dengan memberikan ukuran untuk semua faktor yang relevan. Pentingnya faktor-faktor ini dapat berbeda antara satu aplikasi dengan yang lainnya. Sebagai contoh, dalam sistem pengaturan dan pengendalian kereta api, keandalan dan integritas sangat penting untuk dipertahankan. Sementara pada sistem bisnis umum, faktor utama bisa berupa kegunaan dan pemeliharaan. Sedangkan untuk sistem yang tidak dirancang untuk digunakan secara luas atau berkelanjutan, mungkin faktor-faktor tersebut tidak terlalu penting. Pengujian yang dilakukan harus efektif dalam mengukur semua faktor ini sehingga kualitas perangkat lunak benar-benar dapat terlihat dan diuji secara nyata. Ketiga, pengujian perangkat lunak juga penting untuk menjaga kestabilan sistem. Keandalan perangkat lunak ini berkaitan erat dengan struktur dan jumlah pengujian yang telah dilakukan. Pengujian ini dapat berfungsi sebagai metode untuk mengambil sampel yang diperlukan untuk memperoleh data kegagalan dan estimasi kestabilan sistem. Meskipun tidak mungkin untuk menjamin bahwa suatu perangkat lunak benar-benar bebas dari cacat, pengujian yang sistematis dan terukur dapat membantu dalam memitigasi risiko dan memastikan bahwa sistem beroperasi sebagaimana mestinya. [7]

III. METODE

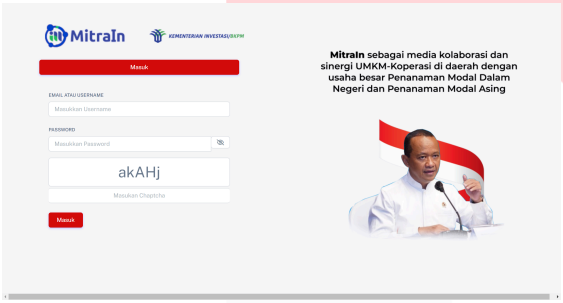
Metode yang digunakan pada laporan ini yaitu observasi. Observasi adalah teknik pengumpulan data dengan cara mengamati dan melihat langsung peristiwa dari perilaku subjek penelitian atau situasi pada tempat terjadi peristiwa. Dalam konteks *software quality assurance* (SQA), observasi menjadi alat yang sangat penting dalam memastikan kualitas perangkat lunak. *Software quality assurance* menggunakan teknik ini untuk mengamati proses pengembangan perangkat lunak, yang terdiri dari tahap desain, pengkodean, pengujian, dan implementasi. Dengan observasi langsung, SQA dapat mengidentifikasi potensi kesalahan atau masalah yang mungkin terjadi selama proses tersebut. Observasi juga memungkinkan SQA untuk memeriksa sejauh mana standar

kualitas telah dilaksanakan, serta untuk memberikan masukan kepada tim pengembang tentang bagian perangkat lunak yang memerlukan perbaikan atau perubahan. Dengan demikian, observasi tidak hanya membantu memastikan bahwa perangkat lunak berkualitas tinggi, tetapi juga memfasilitasi proses perbaikan berkelanjutan dalam pengembangan perangkat lunak. [8]

IV. HASIL DAN PEMBAHASAN

Hasil pengujian aplikasi kemitraan menunjukkan bahwa aplikasi berfungsi dengan baik sesuai dengan yang diharapkan. Sebagai contoh, gambar di bawah ini menggambarkan berbagai aspek aplikasi yang diuji, seperti kinerja, fungsionalitas, dan antarmuka pengguna, semuanya bekerja secara optimal tanpa adanya bug atau masalah signifikan.

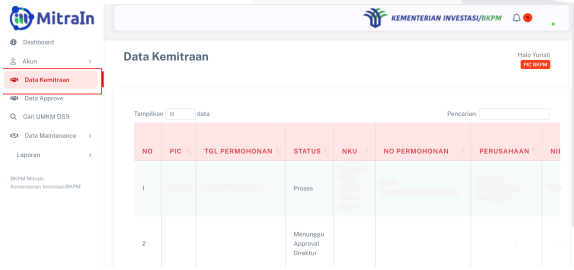
1. Berikut merupakan proses *login* yang berfungsi untuk masuk ke halaman alpikasi



Gambar 1

Hasil pengujian menunjukkan bahwa pengguna berhasil masuk ke aplikasi kemitraan OSS RBA tanpa mengalami kendala. Ini menandakan bahwa proses autentikasi dan keamanan aplikasi berjalan dengan baik, memberikan pengguna akses yang lancar ke fitur-fitur dalam aplikasi.

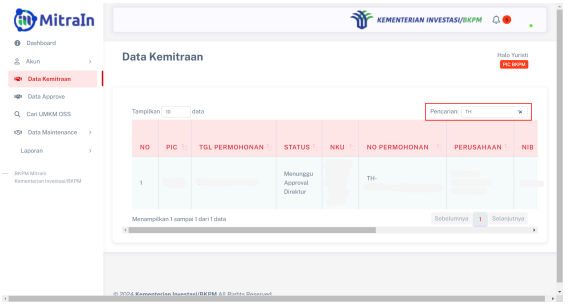
2. Berikut merupakan proses klik menu kemitraan



Gambar 2

Sistem berhasil menampilkan data kemitraan pada menu data kemitraan sesuai dengan yang diharapkan. Pengguna dapat dengan mudah mengakses informasi yang tentang kemitraan yang terdaftar, sehingga memudahkan proses manajemen dan pemantauan.

3. Berikut adalah proses *input* no permohonan pada kolom pencarian



Gambar 3

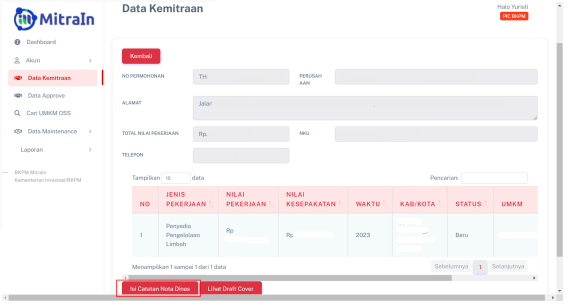
Sistem berhasil menampilkan data yang sesuai dengan nomor permohonan, sistem menunjukkan keakuratannya dalam mengambil dan menampilkan informasi berdasarkan input pengguna. Hal ini membantu meningkatkan efisiensi dalam pencarian dan pengelolaan data kemitraan.

4. Berikut adalah proses klik *icon* mata

Gambar 4

Icon mata berfungsi dengan baik, memungkinkan pengguna untuk melihat detail permohonan dengan mudah. Fitur ini membantu pengguna untuk memperoleh informasi tambahan atau rincian tentang permohonan yang relevan dengan cepat.

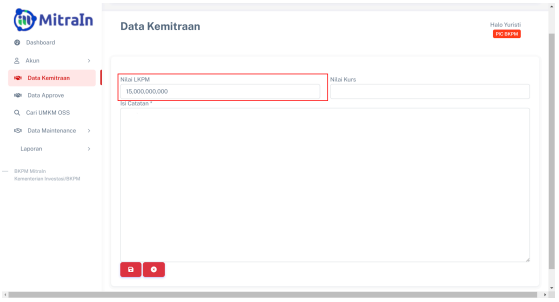
5. Berikut adalah proses klik isi catatan notadinas



Gambar 5

Sistem berhasil menampilkan kolom untuk mengisi catatan notadinas, memfasilitasi pengguna untuk melakukan catatan atau penandaan penting terkait dengan data kemitraan yang sedang dikelola. Ini memberikan fleksibilitas dan kontrol yang diperlukan kepada pengguna dalam mengelola informasi.

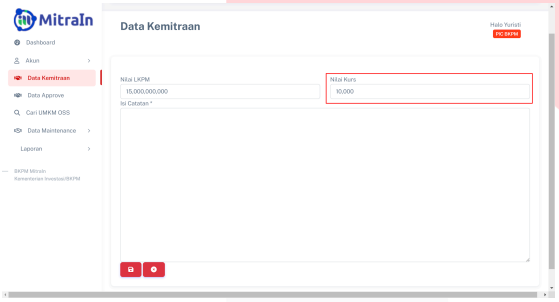
6. Berikut proses input nilai LKPM



Gambar 6

Melalui pengujian, sistem dapat menyimpan nilai LKPM sesuai dengan inputan pengguna secara akurat. Hal ini menunjukkan bahwa aplikasi dapat mengelola dan menyimpan data input pengguna dengan benar, mendukung kebutuhan pengguna dalam memasukkan informasi penting.

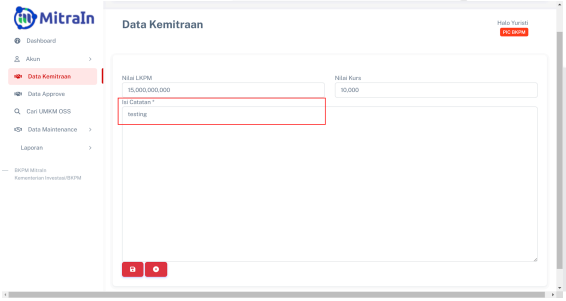
7. Berikut adalah proses *input* nilai kurs



Gambar 7

Sistem berhasil menyimpan nilai KURS sesuai dengan inputan pengguna, menunjukkan kemampuannya untuk mengelola berbagai jenis data dengan tepat dan konsisten.

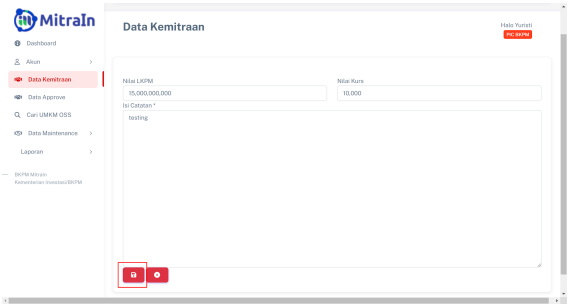
8. Berikut adalah proses *input* ”*testing*” catatan notadinas pada field isi catatan



Gambar 8

Sistem dapat menyimpan catatan yang telah diinput oleh pengguna dengan baik. Fitur ini memungkinkan pengguna untuk menyimpan informasi tambahan atau catatan yang diperlukan, mendukung proses manajemen data yang lebih efisien dan terorganisir.

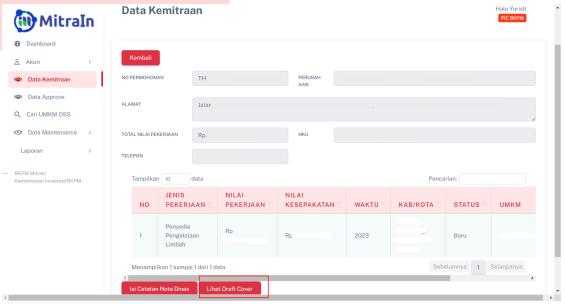
9. Berikut adalah proses klik button *save*



Gambar 9

Button save berfungsi dengan baik, memastikan bahwa perubahan yang dilakukan oleh pengguna disimpan dengan sukses. Ini memberikan jaminan kepada pengguna bahwa setiap perubahan yang mereka buat akan disimpan dengan aman dan tidak akan hilang.

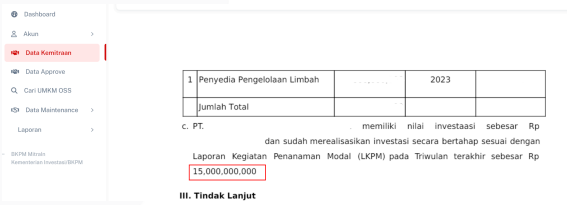
10. Berikut adalah proses klik lihat *draft cover*



Gambar 10

Button untuk melihat *draft cover* berfungsi dengan baik sehingga pengguna dapat melihat isi draft cover

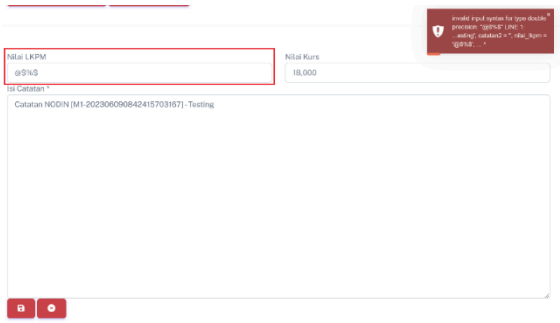
- Berikut adalah tampilan draft cover



Gambar 11

Pengguna dapat mengakses draft cover dengan mudah dan melihat perubahan yang telah mereka lakukan. Hal ini membantu dalam memantau dan meninjau perubahan yang telah dilakukan sebelumnya dengan cepat dan efisien.

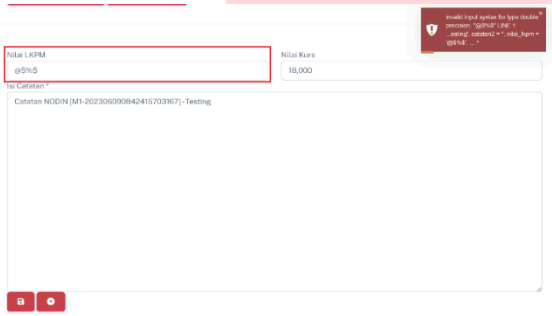
11. Berikut adalah proses *user* mengetikkan huruf pada nilai LKPM



Gambar 12

Ketika pengguna mengetikkan huruf pada nilai LKPM, sistem akan menampilkan pemberitahuan error. Fitur ini memastikan bahwa hanya nilai numerik yang diterima, mencegah input yang tidak valid dan menjaga integritas data

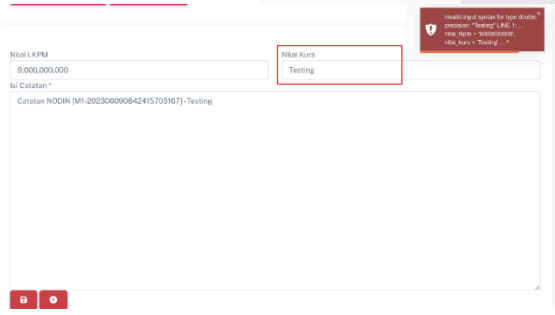
12. Berikut adalah proses *user* mengetikkan simbol pada nilai LKPM



Gambar 13

Ketika pengguna mengetikkan simbol pada nilai LKPM, sistem akan menampilkan pemberitahuan error. Hal ini membantu mencegah kesalahan input yang dapat mengganggu proses perhitungan atau analisis yang bergantung pada nilai LKPM yang valid.

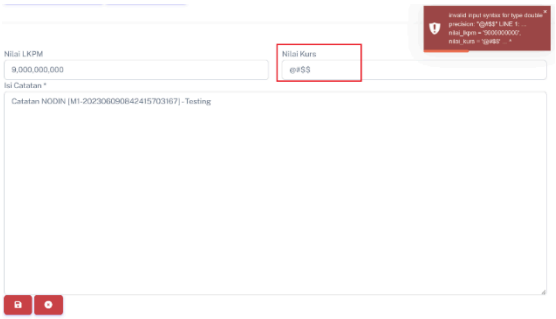
13. Berikut adalah proses mengetikkan huruf pada nilai kurs



Gambar 14

Saat pengguna mengetikkan huruf pada nilai kurs, sistem akan menampilkan pemberitahuan error. Fitur ini memastikan bahwa hanya angka yang dapat dimasukkan pada nilai kurs. Hal ini memudahkan pengguna dalam memasukkan data dengan benar dan menghindari kesalahan yang bisa mengganggu proses perhitungan atau analisis lebih lanjut.

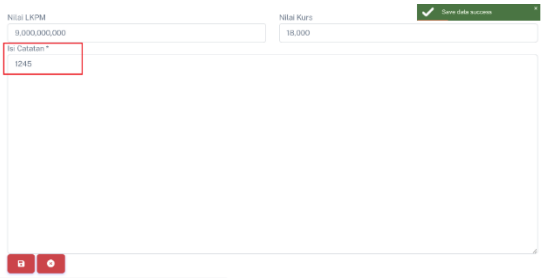
14. Berikut adalah proses *user* mengetikkan simbol pada nilai kurs



Gambar 15

Saat pengguna mengetikkan simbol pada nilai Kurs, sistem akan menampilkan pemberitahuan error. Fitur ini menjaga keakuratan dan validasi data yang dimasukkan, memastikan bahwa hanya angka yang diterima sebagai nilai kurs.

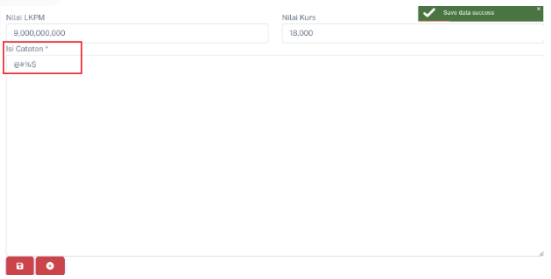
15. Berikut adalah proses mengetikkan angka 12345 pada *field* isi catatan



Gambar 16

Jika pengguna memasukkan angka 1245 pada field isi catatan, sistem akan menyimpan catatan tersebut. Fitur ini memungkinkan pengguna untuk menyimpan catatan numerik dengan mudah dan memastikan bahwa data yang dimasukkan dicatat dengan benar.

16. Berikut adalah proses user mengetikkan simbol @$% pada *field* isi catatan



Gambar 17

Saat pengguna memasukkan simbol seperti @$% pada field isi catatan, sistem akan menyimpan catatan tersebut. Hal ini menunjukkan bahwa

sistem fleksibel dalam menerima berbagai jenis input pada field catatan, memungkinkan pengguna untuk mencatat informasi yang mungkin mengandung karakter khusus sesuai kebutuhan mereka.

Berikut adalah langkah-langkah pengujian yang akan dilakukan untuk memverifikasi fungsi-fungsi utama dan memastikan bahwa aplikasi beroperasi sesuai dengan harapan dan kebutuhan pengguna

Table 1

No	Step	Expected Result	Status
	Buka http://kemitraan-oss-stg.apps.ocp4stg.oss.go.id/login_register	halaman login/register muncul	**PASSED**
1.	*Login*	Halaman utama atau *dashboard* akan ditampilkan	**PASSED**
2.	Klik menu data kemitraan	daftar data kemitraan ditampilkan	**PASSED**
3.	*Input* no permohonan pada kolom pencarian	sistem menampilkan data yang sesuai dengan nomor permohonan tersebut.	**PASSED**
4.	klik *icon* mata	Sistem menampilkan detail permohonan	**PASSED**
5.	klik isi catatan notadinas	Sistem menampilkan kolom untuk mengisi catatan notadinas	**PASSED**
6.	*Input* nilai LKPM	sistem menyimpan nilai tersebut sesuai dengan inputan	**PASSED**
7.	*Input* nilai kurs	sistem menyimpan nilai tersebut sesuai dengan inputan	**PASSED**
8.	*Input "testing* catatan notadinas" pada *field* isi catatan	sistem menyimpan catatan tersebut	**PASSED**
9.	Klik *button save*	Sistem menyimpan perubahan tersebut	**PASSED**
10.	Klik Lihat *draft cover*	sistem menampilkan draft cover yang telah diperbarui	**PASSED**
11.	*User* mengetikkan huruf pada nilai LKPM	Sistem menampilkan pemberitahuan error	**PASSED**
12.	*User* mengetikkan simbol pada nilai LKPM	Sistem menampilkan pemberitahuan error	**PASSED**
13.	*User* mengetikkan huruf pada nilai Kurs	Sistem menampilkan pemberitahuan error	**PASSED**
14.	*User* mengetikkan simbol pada nilai Kurs	Sistem menampilkan pemberitahuan error	**PASSED**
15.	*User* mengetikkan angka 1245 pada field isi catatan	sistem menyimpan catatan tersebut	**PASSED**
16.	*User* mengetikkan simbol @$% pada field isi catatan	sistem menyimpan catatan tersebut	**PASSED**

Di dalam Tabel diatas terbagi menjadi 3 yaitu:

a. Step

Step berfungsi untuk memastikan bahwa setiap tahap dalam pengembangan produk atau layanan memenuhi standar kualitas yang telah ditetapkan. Ini mencakup Langkah atau pengembangan untuk mengidentifikasi dan mengatasi cacat atau ketidaksesuaian sebelum sistem atau layanan mencapai tahap akhir.

b. Expected Result

Expected result adalah hasil yang diharapkan dari suatu proses. Tujuan atau standar yang ingin dicapai dan berfungsi

sebagai tolok ukur untuk mengevaluasi keberhasilan atau kegagalan suatu sistem.

c. Status

Status dalam *quality assurance* (QA) adalah indikator yang menunjukkan tahap atau kondisi saat ini dari aktivitas QA dalam suatu proyek atau proses. Status QA memberikan gambaran tentang seberapa jauh proses QA telah berjalan, masalah apa yang telah ditemukan, tindakan apa yang telah diambil untuk memperbaikinya, dan sejauh mana standar kualitas telah dipenuhi.

V. KESIMPULAN

1. Mengetahui kualitas fitur dalam kemitraan OSS-RBA

Untuk mengetahui kualitas fitur yang ada dalam Kemitraan OSS-RBA sesuai dengan kebutuhan dan ekspektasi pengguna, serta untuk memudahkan pengguna dalam menggunakan aplikasi tersebut, penting untuk melakukan pengujian secara menyeluruh. Pengujian ini mencakup berbagai aspek seperti fungsionalitas, kegunaan, dan kinerja fitur. Pengujian fungsional memastikan bahwa setiap fitur bekerja sesuai dengan spesifikasinya, sedangkan pengujian kegunaan melibatkan pengguna untuk memastikan bahwa fitur-fitur tersebut mudah digunakan dan memenuhi kebutuhan mereka. Pengujian kinerja memastikan bahwa aplikasi tetap responsif dan efisien dalam berbagai kondisi penggunaan. Dengan melibatkan pengguna dalam uji coba dan mendapatkan umpan balik langsung dari mereka, pengembang dapat menyesuaikan dan memperbaiki fitur-fitur yang ada untuk mencapai kualitas yang diharapkan.

2. Bagaimana cara melakukan pengujian

Pengujian pada Kemitraan OSS-RBA dilakukan dengan mengikuti langkah-langkah sistematis seperti *assign backlog*, *planning*, *test case development, testing environment set up, testing execution, user acceptance testing*. Pertama, pemagang perlu memahami *backlog* yang mencakup semua fitur dan fungsionalitas yang akan diuji. Setelah itu, mereka planning pengujian dengan menyusun strategi dan jadwal yang jelas. Pengembangan kasus uji melibatkan *test case development* yang mencakup semua kemungkinan penggunaan aplikasi. Selanjutnya, *testing environment set up* untuk memastikan hasil pengujian yang akurat. *Testing execution* dilakukan dengan menjalankan kasus uji yang telah disiapkan dan mencatat hasilnya. Terakhir, *user acceptance testing* melibatkan pengguna akhir untuk memastikan bahwa aplikasi memenuhi kebutuhan dan ekspektasi sebelum dirilis secara resmi.

REFERENSI

"KAK LELANG PENGEMBANGAN 2023 REV 3 FINAL SUBMIT (3)".

H. Mantik, "PERAN PENTING TESTING DAN QUALITY ASSURANCE DALAM SIKLUS PENGEMBANGAN SISTEM."

M. Sholeh, I. Gisfas, Cahiman, and M. A. Fauzi, "Black Box Testing on ukmbantul.com Page with Boundary Value Analysis and Equivalence Partitioning Methods," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Mar. 2021. doi: 10.1088/1742-6596/1823/1/012029.

M. Leo and A. Saputra, "Implementasi sistem keuangan desa (Siskeudes) di kecamatan muara sugihan menggunakan metode Black Box Testing," *Indonesian Journal of Data and Science (IJODAS)*, vol. 2, no. 3, pp. 148–157, 2021.

C. Senthilmurugan, S. Prakasam, and K. Kanchipuram, "A Literal Review of Software Quality Assurance," 2013.

A. Verma, A. Khatana, and S. Chaudhary, "A Comparative Study of Black Box Testing and White Box Testing," *Article in International Journal of Computer Sciences and Engineering*, 2017, doi: 10.26438/ijcse/v5i12.301304.

C. Mellon, "University 18-849b Dependable Embedded Systems Spring," 1999. [Online]. Available: http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/

N. Nasarudin, "Bab 5. Teknik Pengumpulan Data." [Online]. Available: www.globaleksekutifteknologi.co.id