IMPLEMENTASI PENGUJIAN *BLACKBOX* MENGGUNAKAN *AUTOMATION TESTING TOOLS* PADA *WEBSITE MONITORING* BALITA

Dinda Adisty Yudianto Putri, Nur Cahyo Wibowo, Prasasti Karunia Farista Ananto
Sistem Informasi, Universitas Pembangunan Nasional "Veteran" Jawa Timur
Jl. Rungkut Madya, Gn. Anyar, Kec. Gn. Anyar, Surabaya, Jawa Timur 60294
dndadisty@gmail.com

ABSTRAK

Sistem Informasi Balita (SIBAL) merupakan aplikasi berbasis *web* yang dirancang untuk membantu ahli gizi dalam memantau status gizi balita di Puskesmas Plamongan Sari. Pengujian terhadap aplikasi ini diperlukan untuk memastikan fungsionalitasnya berjalan sesuai spesifikasi sebelum diterapkan secara luas. Permasalahan yang dihadapi adalah keterbatasan pengujian sebelumnya yang hanya dilakukan secara manual dan terbatas pada skenario positif, sehingga berpotensi menyebabkan kesalahan atau kekurangan yang tidak terdeteksi. Penelitian ini bertujuan untuk menerapkan metode *blackbox testing* dengan *automation testing* tools, khususnya Katalon Studio, guna meningkatkan efisiensi dan akurasi pengujian. Metode yang digunakan mengikuti tahapan *Software Testing Life Cycle* (STLC), mulai dari analisis kebutuhan, perencanaan pengujian, pengembangan *test case*, persiapan lingkungan uji, eksekusi pengujian, hingga evaluasi hasil. Hasil pengujian menunjukkan bahwa seluruh 10 fitur utama aplikasi berfungsi dengan baik tanpa ditemukannya *bug*, dengan rata-rata waktu eksekusi sebesar 34,59 detik. Temuan ini membuktikan bahwa penggunaan *blackbox testing* dengan Katalon Studio efektif dalam memastikan kualitas perangkat lunak sebelum implementasi di lingkungan pengguna akhir.

Kata kunci : *Blackbox, Automation Testing, Katalon Studio, STLC, Monitoring, Balita*

1. PENDAHULUAN

Sistem Informasi Balita (SIBAL) merupakan salah satu implementasi dari program pencegahan gizi buruk pada balita di Indonesia yang telah diatur dalam Peraturan Menteri Kesehatan (PMK) No. 2 Tahun 2020. SIBAL adalah aplikasi berbasis *web* yang dirancang untuk mempermudah ahli gizi dalam memonitor status gizi balita, yang sebelumnya dilakukan secara manual dan menggunakan Microsoft Excel sebagai sarana rekapitulasi data [1].

Aplikasi ini memiliki fitur utama, seperti pencatatan data balita dan penghitungan status gizi berdasarkan parameter berat badan, tinggi badan, serta usia balita. Implementasi SIBAL bertujuan untuk meningkatkan efisiensi kerja ahli gizi dalam melakukan pemantauan serta penilaian status gizi balita. Oleh karena itu, diperlukan jaminan bahwa aplikasi ini memiliki fungsionalitas yang sesuai dengan kebutuhan pengguna melalui proses pengujian yang komprehensif dan sistematis.

Dalam tahap pengembangannya, SIBAL telah melalui proses pengujian manual oleh pengguna akhir menggunakan metode *User Acceptance Testing* (UAT) sebelum diterapkan di Puskesmas Plamongan Sari. Namun, pengujian yang dilakukan masih terbatas pada skenario positif dan hanya menguji fungsionalitas utama aplikasi. Keterbatasan ini menyebabkan pengujian bersifat subjektif serta tidak mencakup skenario negatif yang dapat dilakukan oleh pengguna. Kurangnya pengujian menyeluruh berpotensi berdampak pada akurasi hasil serta optimalitas penggunaan SIBAL sebagai alat pengukuran kesehatan di puskesmas.

Menurut Myers et al. [2], proses pengujian perangkat lunak dapat menghabiskan hingga 40% dari total waktu pengembangan. Oleh karena itu, pengujian aplikasi SIBAL menjadi krusial untuk mendeteksi kesalahan pada sistem sejak tahap awal, guna meminimalkan potensi kerugian akibat kegagalan sistem [3].

Pengujian perangkat lunak secara umum terbagi menjadi dua kategori utama, yaitu pengujian *blackbox* dan *whitebox*. Pengujian *blackbox* berfokus pada pemeriksaan fungsionalitas aplikasi tanpa mempertimbangkan struktur internal kode sumber, sedangkan pengujian *whitebox* menguji logika internal dari kode aplikasi [4].

Pengujian *blackbox* menjadi pilihan yang tepat untuk memastikan bahwa aplikasi SIBAL berfungsi sesuai dengan kebutuhan pengguna, karena teknik ini mengevaluasi sistem berdasarkan masukan dan keluaran yang diharapkan tanpa memerlukan pemahaman tentang implementasi kode [5].

Selain itu, pengujian blackbox dapat dilakukan secara otomatis untuk meningkatkan efektivitas dan efisiensi pengujian. Pengujian otomatis memiliki keunggulan dalam mengurangi *human error* serta mempercepat proses pengujian, sehingga lebih efisien dalam aspek waktu dan biaya [6].

Salah satu *tools* yang dapat digunakan untuk pengujian otomatis adalah Katalon Studio, yang mendukung pengujian berbasis *blackbox* dengan skenario uji yang dapat dijalankan secara sistematis dan berulang untuk memastikan fungsionalitas aplikasi berjalan sesuai spesifikasi [7].

Penelitian ini bertujuan untuk melakukan pengujian komprehensif terhadap aplikasi SIBAL dengan pendekatan pengujian *blackbox* yang didukung oleh Katalon Studio sebagai *tools* otomatisasi. Pengujian ini diharapkan dapat mengevaluasi aspek

fungsional aplikasi secara sistematis serta mengidentifikasi kesalahan atau kekurangan dari perspektif pengguna dengan lebih akurat dan cepat. Hasil pengujian akan disajikan dalam bentuk laporan yang dapat digunakan oleh pengembang sebagai dasar evaluasi dan perbaikan, guna meningkatkan kualitas aplikasi SIBAL agar lebih andal, ramah pengguna, dan bebas dari kesalahan sistem.

2. TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Dalam penelitian yang dilakukan oleh Zulianto et al. dalam Jurnal JEPIN, Volume 7, No. 3, halaman 370–378 (2021) [7], dilakukan pengujian otomatis pada aplikasi iPosyandu berbasis *web* menggunakan Katalon Studio untuk memastikan aplikasi berjalan sesuai spesifikasi. Hasil pengujian menunjukkan peningkatan efisiensi waktu eksekusi sebesar 2,54 kali dibandingkan pengujian manual. Studi serupa oleh Tempomona & Hendry dalam Jurnal Inovtek Polbeng Seri Informatika, Volume 7, No. 2, halaman 193–204 (2022) [8], menerapkan metode *blackbox testing* menggunakan Katalon Studio pada aplikasi absensi, menghasilkan laporan otomatis dalam format PDF dengan 15 test case, di mana 5 berhasil, 9 gagal, dan 1 tidak selesai.

Penelitian lainnya oleh Desyani et al. dalam Jurnal Informatika dan Rekayasa Perangkat Lunak, Volume 5, No. 1 (2023) [9], melakukan pengujian otomatis pada aplikasi Otten Coffee dengan metode *spy* dan *record* object di Katalon Studio, menghasilkan 10 test case dengan 4 yang berhasil. Berdasarkan penelitian-penelitian tersebut, metode *blackbox testing* dengan Katalon Studio terbukti efektif dalam meningkatkan efisiensi pengujian perangkat lunak.

2.2. Sistem Informasi Balita (SIBAL)

SIBAL atau Sistem Informasi Balita adalah sebuah aplikasi berbasis web yang dibuat untuk memudahkan proses *monitoring* gizi balita di Puskesmas Plamongan Sari. Aplikasi ini dirancang untuk membantu ahli gizi dalam menghitung status gizi balita berdasarkan parameter antropometri seperti berat badan, tinggi badan, dan usia balita. Tujuan dikembangkannya aplikasi ini adalah dapat mengurangi ketergantungan pada perhitungan manual sehingga proses *monitoring* gizi dapat dilakukan dengan lebih cepat dan akurat [1].

Wibisono telah mengembangkan aplikasi SIBAL menggunakan *framework* Laravel dengan arsitektur *Model View Controller* (MVC) agar dapat diakses dari berbagai perangkat yang terhubung dengan internet. Metode yang digunakan untuk mengembangkan aplikasi ini adalah metode *Prototyping* yang memungkinkan penyesuaian berdasarkan kebutuhan pengguna secara bertahap. Fitur utama pada aplikasi ini adalah pengguna dapat memasukkan data balita, mencari data balita, melihat detail data balita, melihat riwayat pengukuran balita, dan menambahkan data pengukuran balita.

2.3. *Blackbox Testing*

Metode *Blackbox Testing* adalah salah satu teknik pengujian yang melibatkan berbagai pendekatan, baik secara manual maupun otomatis. Fokus utama dari pengujian ini adalah untuk menguji setiap spesifikasi fungsional dari perangkat lunak. Penguji dapat menentukan berbagai kondisi *input* dan menguji bagaimana perangkat lunak berfungsi berdasarkan kondisi tersebut [2]. Jadi, penguji memastikan bahwa perangkat lunak berfungsi sesuai dengan kebutuhan pengguna tanpa perlu mengetahui detail kode program.

Pengujian *blackbox* juga mencakup pengujian berbasis skenario yang memungkinkan penguji untuk melihat bagaimana perangkat lunak berfungsi dalam kondisi dunia nyata. Dalam metode ini, penguji tidak hanya fokus pada input dan output, tetapi juga mengevaluasi bagaimana sistem bereaksi terhadap skenario penggunaan yang mungkin muncul selama operasi normal. Pengujian ini mengidentifikasi kesalahan yang berkaitan dengan fungsionalitas, seperti kesalahan dalam alur kerja, validasi data, atau kegagalan sistem dalam menangani *input* yang tidak valid. Dengan demikian, *blackbox* testing memberikan pandangan dari sudut pengguna untuk memastikan perangkat lunak memenuhi spesifikasi dan kebutuhan operasional yang diharapkan tanpa memperhatikan struktur internal atau kode yang mendasarinya [10].

2.4. *Automated Testing*

Pengujian otomatis atau *automated testing* adalah metode yang menggunakan perangkat lunak untuk menjalankan serangkaian pengujian secara otomatis. Metode ini bertujuan untuk mempercepat proses pengujian, mengurangi intervensi manual, dan meningkatkan akurasi pengujian. *Automated testing* sangat berguna bagi penguji dalam proyek perangkat lunak yang besar dan kompleks, di mana pengujian manual memerlukan waktu dan tenaga yang sangat besar. Penguji dapat lebih fokus pada analisis hasil dan pengembangan skenario pengujian yang lebih mendalam, karena pengujian otomatis memungkinkan mereka untuk mengulang uji yang sama dengan mudah dan cepat, memastikan konsistensi hasil serta meminimalkan *human error* [11].

Dustin juga menekankan pentingnya perencanaan dan manajemen dalam penerapan automated testing. Tidak semua pengujian dapat atau harus diotomatisasi, salah satu jenisnya yang sering diotomatisasi adalah pengujian fungsional. Dengan mengotomatiskan pengujian fungsional, penguji dapat secara efisien menguji berbagai skenario pengguna dan memastikan bahwa semua fitur perangkat lunak berjalan dengan baik. Penerapan *automated testing* yang tepat tidak hanya meningkatkan efisiensi pengujian, tetapi juga berkontribusi pada peningkatan kualitas perangkat lunak secara keseluruhan.

2.5. STLC

Desai dan Srivastava mengungkapkan pada bukunya yang berjudul SOFTWARE TESTING: A Practical Approach, bahwa Software Testing Life Cycle (STLC) didefinisikan sebagai rangkaian tahap terstruktur yang dilakukan selama proses pengujian perangkat lunak untuk memastikan bahwa produk perangkat lunak memenuhi standar kualitas dan fungsionalitas yang diharapkan [12].

STLC mencakup semua aktivitas yang dilakukan dari perencanaan pengujian hingga evaluasi hasil pengujian, dan melibatkan berbagai langkah seperti analisis kebutuhan, perencanaan pengujian, pengembangan *test case*, penyiapan lingkungan pengujian, pelaksanaan pengujian, dan penutupan siklus pengujian. Dengan mengikuti STLC, tim pengujian dapat secara sistematis mendeteksi dan mengatasi cacat dalam perangkat lunak, sehingga meningkatkan keandalan dan kualitas produk akhir.

STLC memiliki enam tahapan utama dalam pengujiannya, namun tahapan-tahapan ini tidak harus diikuti seluruhnya. Tahapan tersebut meliputi *Requirement Analysis*, *Test Planning*, *Test Case Development*, *Test Environment Setup*, *Test Case Execution*, dan *Test Cycle Closure*. Tahapan yang diikuti tergantung pada jenis perangkat lunak yang sedang dikembangkan dan sumber daya yang digunakan dalam melakukan pengujian.

2.6. *Functional Testing*

Dalam buku Instant Approach to Software Testing oleh Dr. Anand Nayyar [13], pengujian fungsional berfokus pada verifikasi apakah perangkat lunak berfungsi sesuai spesifikasi. Pengujian ini memastikan bahwa input yang diberikan menghasilkan output yang diharapkan serta mengevaluasi keandalan, akurasi, dan ketepatan sistem sesuai spesifikasi fungsional.

2.7. *Bug*

Menurut Peter Farrel-Vinay dalam "Manage Software Testing", bug adalah kesalahan dalam perangkat lunak yang menyebabkan program tidak berfungsi sebagaimana mestinya [14].

Bug dapat berupa kesalahan logika, *typo*, atau ketidaksesuaian dengan spesifikasi yang ditentukan, yang dapat menghambat kinerja aplikasi dan penggunaannya.

Farrel-Vinay menekankan pentingnya identifikasi dan pengelolaan bug dalam proses pengujian perangkat lunak. Pengujian yang tidak efektif dapat berdampak buruk pada kualitas perangkat lunak dan pengalaman pengguna. Oleh karena itu, penguji harus memiliki strategi yang tepat dalam mendeteksi, mendokumentasikan, dan mengelola bug untuk memastikan perangkat lunak berfungsi dengan baik dan berkelanjutan.

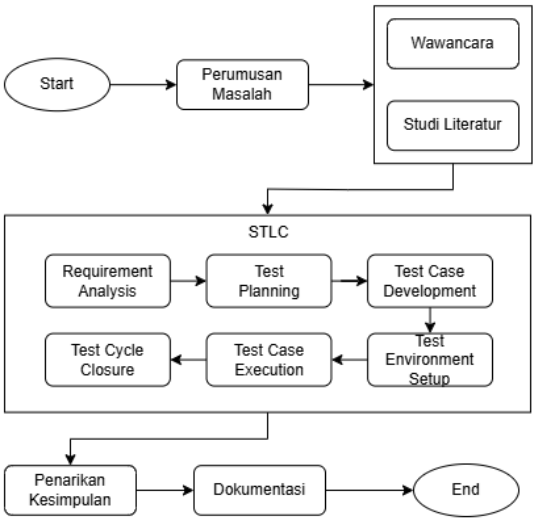
2.8. *Automation Testing Tools*

Tools pengujian otomatis dirancang untuk membantu penguji dalam melakukan pengujian perangkat lunak secara efisien. Shende menjelaskan bahwa penggunaan tools pengujian otomatis memungkinkan penguji untuk mengotomatiskan proses pengujian, yang mengurangi waktu dan upaya yang diperlukan untuk melakukan pengujian manual [15].

Tools-tools ini menawarkan berbagai fitur, termasuk pembuatan skrip pengujian, pelaporan hasil, dan integrasi dengan *tools* pengembangan lainnya, yang mempermudah penguji dalam mengelola dan menjalankan pengujian secara terstruktur. Namun, tidak semua tools pengujian otomatis cocok untuk setiap jenis aplikasi. Oleh karena itu, penguji perlu mempertimbangkan berbagai faktor, seperti jenis pengujian yang dilakukan dan lingkungan pengujian.

3. METODE PENELITIAN

Bab ini memuat seluruh tahapan yang dilakukan antaranya lain adalah pengumpulan data dilanjutkan dengan penerapan *framework Software Requirement Life Cycle* (STLC) yang divisualisasikan pada Gambar 1 berikut.



Gambar 1. Metode Penelitian

3.1. *Requirement Analysis*

Pada tahap *Requirement Analysis* dalam STLC, penguji menganalisis kebutuhan sistem yang akan diuji berdasarkan data yang dikumpulkan sebelumnya. Fokus utama adalah memahami fungsionalitas aplikasi SIBAL, seperti pendataan balita, penghitungan status gizi, dan pelaporan *real-time*. Hasil dari tahap ini adalah dokumen *Requirement Traceability Matrix* (RTM) yang merinci kebutuhan fungsional dan skenario pengujian.

Analisis dilakukan dengan mempertimbangkan spesifikasi dari pengembang dan pengguna untuk memastikan aplikasi bekerja optimal. Dokumen RTM yang dihasilkan menjadi acuan dalam proses pengujian

agar berjalan secara terstruktur dan sesuai dengan persyaratan yang telah ditetapkan.

3.2. *Test Planning*

Test Planning adalah tahap di mana penguji merencanakan seluruh proses pengujian berdasarkan analisis kebutuhan yang telah dilakukan. Pada tahap ini, penguji menentukan tujuan pengujian, skenario pengujian, serta *tools* yang akan digunakan selama proses pengujian aplikasi SIBAL. Hasil dari tahapan ini adalah dokumen *test plan* yang menjadi panduan bagi penguji dalam menjalankan pengujian.

Dalam penyusunan *test plan*, penguji mempertimbangkan jenis pengujian dan *tools* yang digunakan. Katalon Studio dan Excel dipilih untuk pengujian fungsional, sementara berbagai *tools* lainnya digunakan untuk aspek non-fungsional.

Test plan yang disusun dengan baik memastikan bahwa seluruh proses pengujian berjalan sesuai rencana. Hal ini membantu memastikan bahwa aplikasi SIBAL diuji secara menyeluruh sebelum diimplementasikan di Puskesmas Plamongan Sari.

3.3. *Test Case Development*

Test Case Development adalah tahap penting dalam pengujian perangkat lunak yang berfokus pada pembuatan *test case* dan pengumpulan *test* data. *Test case* dirancang untuk memverifikasi apakah aplikasi berfungsi sesuai spesifikasi yang ditetapkan, sehingga pengujian dapat dilakukan secara sistematis untuk mendeteksi kesalahan atau *bug* dalam aplikasi.

Tahapan ini mencakup pembuatan *test case* berdasarkan rencana dalam *Test Planning*, baik secara manual menggunakan Excel maupun otomatis menggunakan Katalon Studio. Penguji juga mengumpulkan *test data* yang representatif untuk memastikan pengujian mencerminkan penggunaan nyata oleh pengguna akhir.

Hasil dari tahap ini adalah *test case* dan *test data* yang siap digunakan dalam pengujian. Dengan *test case* yang terstruktur, pengujian berjalan lebih lancar, membantu mendeteksi kesalahan lebih awal, serta memastikan aplikasi SIBAL memenuhi semua persyaratan.

3.4. *Test Environment Setup*

Test Environment Setup adalah tahap penting dalam memastikan bahwa pengujian dilakukan di lingkungan yang merepresentasikan kondisi nyata penggunaan aplikasi. Penguji perlu menyiapkan perangkat keras dan perangkat lunak yang sesuai dengan kebutuhan pengujian serta menginstal dan mengkonfigurasi *tools* pengujian untuk mendukung proses pengujian yang efisien.

Proses ini mencakup penyusunan dan konfigurasi perangkat keras, perangkat lunak, serta memastikan tools pengujian siap digunakan. Selain itu, penguji juga perlu mengatur *test data* pada *tools* yang telah diinstal.

Hasil dari tahap ini adalah lingkungan pengujian yang siap digunakan dengan konfigurasi yang sesuai. Dengan lingkungan yang tepat, pengujian dapat berjalan lancar, menghasilkan data yang akurat, dan memungkinkan penguji mendeteksi potensi masalah sejak dini.

3.5. *Test Case Execution*

Test Case Execution merupakan tahap utama dalam pengujian di mana penguji menjalankan *test case* yang telah dibuat pada tahap sebelumnya. Pada tahap ini, penguji menjalankan pengujian terhadap fungsionalitas aplikasi untuk mendeteksi *bug* atau kesalahan yang mungkin muncul saat aplikasi dijalankan oleh pengguna.

Pengujian dilakukan sesuai dengan *test plan* yang telah disusun, dimulai dari pengujian fungsionalitas aplikasi, seperti penghitungan status gizi balita pada aplikasi SIBAL. Semua temuan selama pengujian dicatat dalam *bug report*, dan status *test case* diperbarui untuk mencerminkan hasil pengujian.

Hasil dari *Test Case Execution* meliputi pembaruan test case dengan hasil pengujian, bug report yang mencatat semua bug yang ditemukan, serta dokumen RTM yang menunjukkan status dari setiap *test case* yang telah dieksekusi. Semua hasil ini menjadi dasar evaluasi kualitas aplikasi yang diuji.

3.6. *Test Cycle Closure*

Test Cycle Closure adalah tahap akhir dalam siklus pengujian perangkat lunak, di mana seluruh hasil pengujian dievaluasi untuk memastikan semua *test case* telah dieksekusi sesuai rencana. Tahap ini bertujuan untuk menilai keberhasilan pengujian serta menyusun laporan akhir sebagai acuan perbaikan dan pengembangan lebih lanjut.

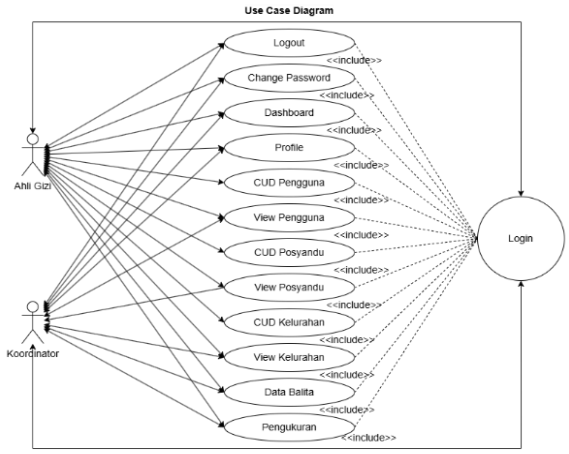
Langkah-langkahnya mencakup evaluasi hasil pengujian dan memastikan tidak ada *test case* yang terlewat. Selain itu, penguji menyusun *test report* yang merangkum hasil pengujian serta bug yang ditemukan.

Hasil dari tahap ini adalah *test report*, yang menyajikan ringkasan pengujian, hasil yang dicapai, serta temuan *bug*. Laporan ini memberikan gambaran kualitas aplikasi dan menjadi acuan evaluasi serta pengembangan lebih lanjut.

4. HASIL DAN PEMBAHASAN

4.1. Pengumpulan Data

Proses pengumpulan data dilakukan kepada *founder* dan *developer*. Hasil dari pengumpulan data ini adalah berupa *use case* yang akan digunakan sebagai kebutuhan pengujian. Dokumen yang telah didapatkan dicantumkan sebagai gambar di bawah ini.



Gambar 2. Use Case SIBAL

Gambar 2 merupakan *use case* penyesuaian dari hasil pengumpulan data yang telah dibuat oleh developer. Penulis melakukan penyesuaian dikarenakan dokumen ini akan digunakan sebagai kebutuhan *testing*, sehingga perlu disesuaikan agar dapat menjadi data yang optimal.

4.2. STLC

4.2.1. *Requirement Analysis*

Pada tahap ini, dilakukan penyusunan dokumen RTM (*Requirement Traceability Matrix*) sebagai hasil dari tahap *requirement analysis* untuk memastikan kebutuhan dalam data yang telah didapat sudah sesuai dan status pengujiannya dapat dilacak. Pengujian dilakukan oleh ahli gizi dan koordinator, yang berperan dalam memastikan bahwa fitur-fitur SIBAL berjalan sesuai dengan kebutuhan pengguna. Hasil penentuan prioritas untuk fitur SIBAL yang akan diuji adalah sebagai berikut.

Tabel 1. Prioritas Fitur

Fitur	Prioritas
Autentikasi	*High*
Dashboard	*High*
Profil	*Medium*
Akun Pengguna	*High*
Halaman Posyandu	*High*
Halaman Kelurahan	*High*
Data Balita	*High*
Pengukuran	*High*

RTM dibuat untuk setiap *role* yang akan dilakukan pengujian dalam hal ini yaitu *role* Ahli Gizi dan Koordinator. Variabel pada RTM antara lain *Business Requirement Document* (BRD), *Functional Requirement Document* (FSD), *Priority*, *Test Case Document*, serta *Result*.

4.2.2. *Test Planning*

Pada tahap *test planning* dilakukan pengerjaan dokumen *test plan* sebagai hasil dari tahap ini. Acuan yang digunakan untuk menyusun test plan adalah dokumen yang didapatkan di tahap pengumpulan data dan dokumen RTM (*Requirement Traceability Matrix*).

Acuan untuk dokumen *test plan* ini menggunakan *template* yang dikeluarkan oleh IEEE 829 (Institute of Electrical and Electronics Engineers) [16]. *Test plan* ini nantinya akan disetujui oleh dua pihak terkait yaitu QA dan *developer* untuk memastikan kesesuaian isi dokumennya. Test plan ini juga berisi kriteria-kriteria yang harus dipenuhi untuk melakukan proses *testing* pada tahap selanjutnya. Berikut adalah rincian *test plan* yang telah dibuat.

Tabel 2. Dokumen *Test Plan*

Modul	Fitur
Data Balita	Pengujian fungsionalitas tambah, ubah, dan hapus data balita
Pengukuran	Pengujian fungsionalitas tambah, ubah, dan hapus data pengukuran

Tabel 2 merupakan *test plan* yang dibuat menggunakan acuan pada tahap sebelumnya yaitu dokumen RTM.

4.2.3. *Test Case Development*

Test case disusun berdasarkan spesifikasi aplikasi yang telah ter-*deploy* dan dapat diakses oleh pengguna. Penyusunan *test case* dilakukan oleh QA dengan mempertimbangkan perspektif *end-user*, yaitu ahli gizi dan koordinator, guna memastikan pengujian yang relevan terhadap implementasi SIBAL [17].

Test case mencakup skenario, *test* data, serta status skenario yang nantinya akan diuji dengan status *passed* jika hasil sesuai ekspektasi sistem, dan *failed* jika tidak [18].

Pembuatan *test case* dilakukan dengan dua cara, yaitu manual dan otomatis. *Test case* manual disusun dalam Google Spreadsheet berdasarkan dokumen *test plan*, sedangkan *test case* otomatis diimplementasikan menggunakan automation testing *tools*. Salah satu *test case*nya adalah untuk modul Data Balita dan Pengukuran yang dapat dilihat pada Tabel 3 di bawah ini.

Tabel 3. *Test Case* Manual

TC ID	Nama *Test Case*	*Expected Result*
TC-01	Membuat data balita dengan inputan valid dan lengkap	Berhasil tambah data balita
TC-02	Membuat data balita dengan inputan invalid dan tidak lengkap	Gagal tambah data balita
TC-03	Mengubah data balita dengan inputan valid dan lengkap	Berhasil ubah data balita
TC-04	Mengubah data balita dengan inputan invalid dan tidak lengkap	Gagal ubah data balita
TC-05	Menghapus data balita	Berhasil hapus data balita
TC-06	Membuat data pengukuran dengan inputan valid dan lengkap	Berhasil tambah data pengukuran
TC-07	Membuat data pengukuran dengan inputan invalid dan tidak lengkap	Gagal tambah data pengukuran

TC ID	Nama *Test Case*	*Expected Result*
TC-08	Mengubah data pengukuran dengan inputan valid dan lengkap	Berhasil ubah data pengukuran
TC-09	Mengubah data pengukuran dengan inputan invalid dan tidak lengkap	Gagal ubah data pengukuran
TC-10	Menghapus data pengukuran	Berhasil hapus data pengukuran

Test case manual akan diimplementasikan menjadi *test case* otomatis menggunakan fitur *Record* di Katalon Studio, yang merekam setiap langkah pengujian. Kemudian, akan digunakan fitur *Play* untuk menjalankan *test step* agar menghasilkan status pada *test case* tersebut.

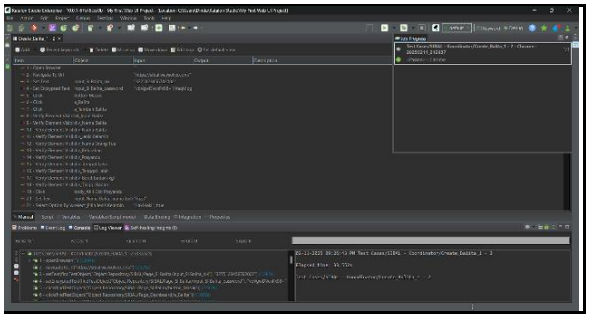
4.2.4. *Test Environment Setup*

Pada tahap *test environment setup*, dilakukan persiapan pengujian dengan menyiapkan laptop sebagai *hardware*, aplikasi SIBAL, serta menginstal Katalon Studio sebagai *automation tool* untuk pengujian fungsional.

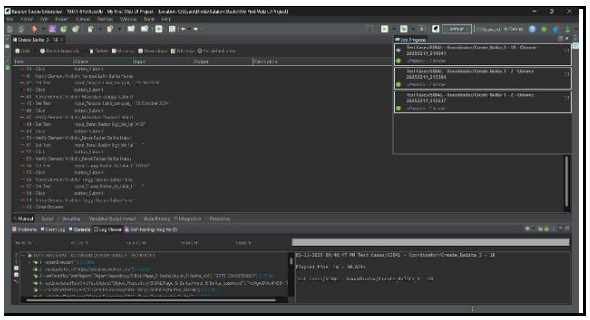
4.2.5. *Test Case Execution*

Pada tahap ini, dokumen *test case* akan terus diperbarui terutama pada kolom "Status" seiring berjalannya pengujian untuk mencatat skenario yang berhasil dan yang belum dieksekusi. Jika skenario berhasil, penguji akan melanjutkan ke skenario berikutnya.

Setiap elemen dalam pengujian diverifikasi menggunakan fitur *Verify Element Visible* untuk memastikan *actual result* sesuai dengan *test case*. Seluruh objek dan data *input* yang diklik atau diisi oleh penguji akan direkam oleh Katalon Studio sebagai langkah atau item pengujian. Hasil dari setiap langkah pengujian dapat dilihat di *tab log viewer*, sementara status eksekusi dapat dipantau melalui *job progress* di pojok kanan atas, yang menunjukkan apakah *test case passed* atau *failed*.



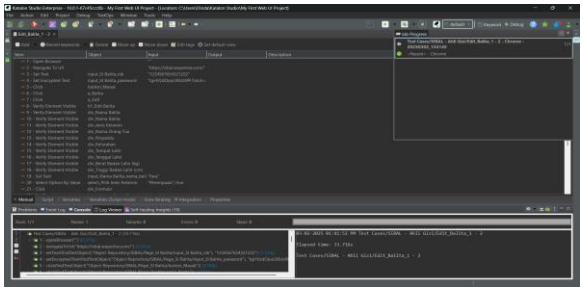
Gambar 3. Eksekusi *Test Case* TC-01



Gambar 4. Eksekusi *Test Case* TC-02

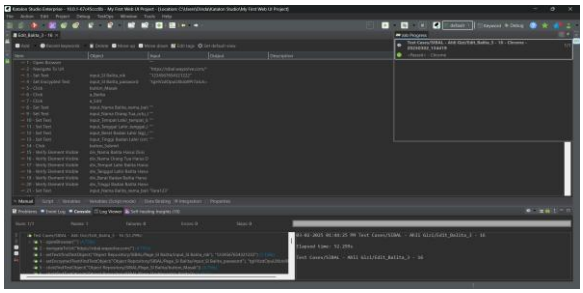
Gambar 3 menunjukkan bahwa *test case* Tambah Balita dengan Input Valid dengan ID TC-01 memiliki status *Passed* yang artinya pengguna dapat melakukan tambah balita dengan inputan valid sesuai dengan *expected result* pada Tabel 3.

Gambar 4 menunjukkan bahwa *test case* Tambah Balita dengan Input Invalid dengan ID TC-02 memiliki status *Passed* yang artinya pengguna tidak dapat melakukan tambah balita dengan inputan invalid sesuai dengan *expected result* pada Tabel 3.



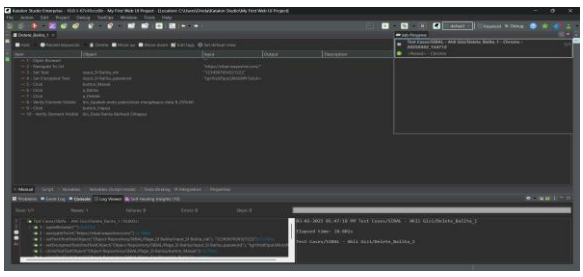
Gambar 5. Eksekusi *Test Case* TC-03

Gambar 5 menunjukkan bahwa *test case* Edit Balita dengan Input Valid dengan ID TC-03 memiliki status *Passed* yang artinya pengguna dapat melakukan edit balita dengan inputan valid sesuai dengan *expected result* pada Tabel 3.



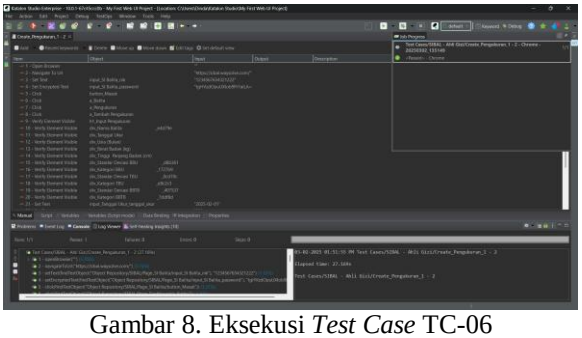
Gambar 6. Eksekusi *Test Case* TC-04

Gambar 6 menunjukkan bahwa *test case* Edit Balita dengan Input Invalid dengan ID TC-04 memiliki status *Passed* yang artinya pengguna tidak dapat melakukan edit balita dengan inputan invalid sesuai dengan *expected result* pada Tabel 3.



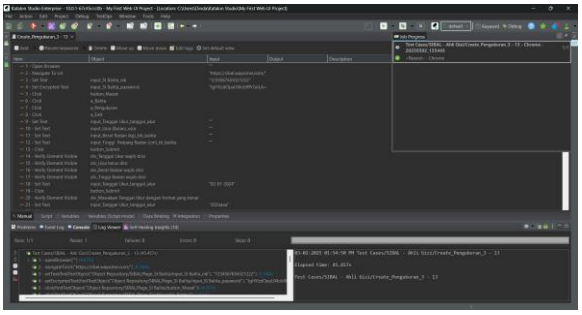
Gambar 7. Eksekusi *Test Case* TC-05

Gambar 7 menunjukkan bahwa *test case* Hapus Balita dengan ID TC-05 memiliki status *Passed* yang artinya pengguna dapat melakukan hapus balita sesuai dengan *expected result* pada Tabel 3.



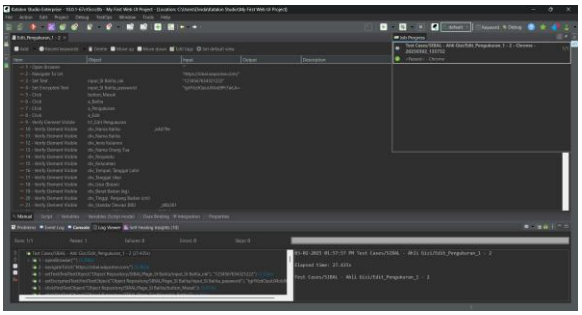
Gambar 8. Eksekusi *Test Case* TC-06

Gambar 8 menunjukkan bahwa *test case* Tambah Pengukuran dengan Input Valid dengan ID TC-06 memiliki status Passed yang artinya pengguna dapat melakukan tambah pengukuran dengan inputan valid. sesuai dengan *expected result* pada Tabel 3.



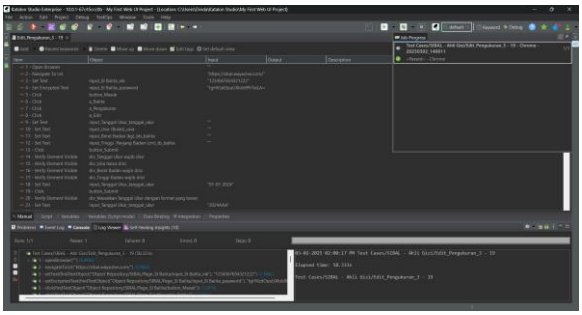
Gambar 9. Eksekusi *Test Case* TC-07

Gambar 9 menunjukkan bahwa *test case* Tambah Pengukuran dengan Input Invalid dengan ID TC-07 memiliki status *Passed* yang artinya pengguna tidak dapat melakukan tambah pengukuran dengan inputan invalid sesuai dengan *expected result* pada Tabel 3.



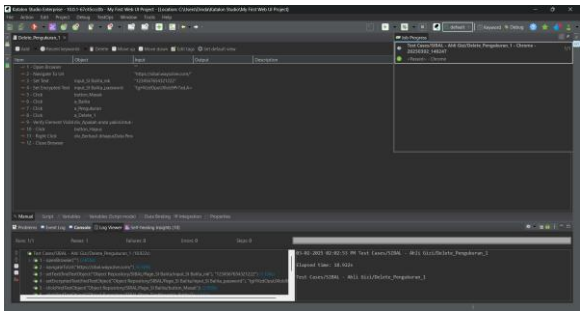
Gambar 10. Eksekusi *Test Case* TC-08

Gambar 10 menunjukkan bahwa *test case* Edit Pengukuran dengan Input Valid dengan ID TC-08 memiliki status *Passed* yang artinya pengguna dapat melakukan edit pengukuran dengan inputan valid sesuai dengan *expected result* pada Tabel 3.



Gambar 11. Eksekusi *Test Case* TC-09

Gambar 11 menunjukkan bahwa *test case* Edit Pengukuran dengan Input Invalid dengan ID TC-09 memiliki status *Passed* yang artinya pengguna tidak dapat melakukan edit pengukuran dengan inputan invalid sesuai dengan *expected result* pada Tabel 3.



Gambar 12. Eksekusi *Test Case* TC-10

Gambar 12 menunjukkan bahwa *test case* Hapus Pengukuran dengan ID TC-10 memiliki status *Passed* yang artinya pengguna dapat melakukan hapus balita sesuai dengan *expected result* pada Tabel 3.

4.2.6. *Test Cycle Closure*

Pada tahap ini, seluruh rangkaian pengujian telah diselesaikan. Tahap ini digunakan untuk menyajikan hasil dari seluruh proses pengujian, mulai dari *Test Planning* hingga *Test Case Execution*.

Berdasarkan tahapan *Test Case Execution*, dapat disimpulkan bahwa dari 10 fungsionalitas yang diuji, semuanya telah memenuhi *expected result* dengan tingkat keberhasilan 100%, serta tidak ditemukannya bug selama pengujian. Rata-rata waktu yang dibutuhkan untuk menguji 10 fungsionalitas tersebut adalah 34.59 detik. Hasil pengujian secara rinci dicantumkan pada Tabel 4.

Tabel 4. Hasil Pengujian

TC ID	Nama *Test Case*	Status
TC-01	Membuat data balita dengan inputan valid dan lengkap	*Passed*
TC-02	Membuat data balita dengan inputan invalid dan tidak lengkap	*Passed*
TC-03	Mengubah data balita dengan inputan valid dan lengkap	*Passed*

TC ID	Nama *Test Case*	Status
TC-04	Mengubah data balita dengan inputan invalid dan tidak lengkap	*Passed*
TC-05	Menghapus data balita	*Passed*
TC-06	Membuat data pengukuran dengan inputan valid dan lengkap	*Passed*
TC-07	Membuat data pengukuran dengan inputan invalid dan tidak lengkap	*Passed*
TC-08	Mengubah data pengukuran dengan inputan valid dan lengkap	*Passed*
TC-09	Mengubah data pengukuran dengan inputan invalid dan tidak lengkap	*Passed*
TC-10	Menghapus data pengukuran	*Passed*

5. KESIMPULAN DAN SARAN

Berdasarkan hasil pengujian yang telah dilakukan terhadap aplikasi SIBAL, seluruh fungsionalitas yang diuji mencapai tingkat keberhasilan 100% sesuai dengan *expected result*, tanpa ditemukan bug selama proses pengujian berlangsung. Pengujian dilakukan menggunakan pendekatan *blackbox* dengan *automation testing tools*, seperti Katalon Studio, untuk memastikan fungsionalitas sistem berjalan dengan baik. Rata-rata waktu eksekusi untuk menguji 10 fungsionalitas adalah 34,59 detik, dengan hasil pengujian yang tercatat dalam *test case execution* menunjukkan bahwa semua skenario pengujian berhasil dijalankan sesuai perencanaan.

DAFTAR PUSTAKA

[1] M. P. Wibisono, A. A. Arifiyanti, and R. Permatasari, “Rancang Bangun Sistem Informasi Controlling Status Gizi Balita Dengan Anthropometric Measurements (Studi Kasus : Puskesmas Plamongan Sari)”, Scientica, vol. 2, no. 10, pp. 404–411, Jul. 2024.

[2] G. J. Myers, C. Sandler, and T. Badgett, The Art of Software Testing. Hoboken, NJ, USA: John Wiley & Sons, 2011.

[3] W. Wibisono and F. Baskoro, “Pengujian perangkat lunak dengan menggunakan model behaviour UML,” Jurnal Ilmiah Teknologi Informasi, vol. 1, no. 1, pp. 43–50, 2002.

[4] S. Nidhra and J. Dondeti, “Black box and white box testing techniques—a literature review,” International Journal of Embedded Systems and Applications (IJESA), vol. 2, no. 2, pp. 29–50, 2012.

[5] A. Arfan and H. Hendrik, “Penerapan STLC dalam pengujian Black Box dengan automation testing tool (Studi kasus: PT. GIT Solution),” AUTOMATA, vol. 3, no. 2, 2022.

[6] J. Pan, “Software testing,” Dependable Embedded Systems, vol. 5, no. 2006, p. 1, 1999.

[7] A. Zulianto, A. Purbasari, N. Suryani, A. I. Susanti, F. R. Rinawan, and W. G. Purnama, “Pemanfaatan Katalon Studio untuk otomatisasi pengujian black-box pada aplikasi iPosyandu,” JEPIN (Jurnal Edukasi dan Penelitian Informatika), vol. 7, no. 3, pp. 370–378, 2021.

[8] V. T. Tempomona, “Penerapan Metode Blackbox Pada Perangkat Lunak Menggunakan Katalon Studio,” INOVTEK Polbeng-Seri Informatika, vol. 7, no. 2, pp. 193–204, 2022.

[9] T. Desyani, A. Syamsiana, E. U. Kobun, F. T. Ananda, and S. Priaji, “Otomatisasi Pengujian Aplikasi Web Otten Coffee Menggunakan Katalon Studio,” Jurnal Teknologi Sistem Informasi dan Aplikasi, vol. 6, no. 2, pp. 186–190, 2023.

[10] S. Desikan and G. Ramesh, Software Testing: Principles and Practice. India: Pearson Education, 2006.

[11] E. Dustin, J. Rashka, and J. Paul, Automated Software Testing: Introduction, Management, and Performance. Boston, MA, USA: Addison-Wesley Professional, 1999.

[12] S. Desai and A. Srivastava, Software Testing: A Practical Approach, 2nd ed. New Delhi, India: PHI Learning Pvt. Ltd., 2016.

[13] A. Nayyar, Instant Approach to Software Testing. India: BPB Publications, 2019.

[14] P. Farrell-Vinay, Manage Software Testing. USA: CRC Press, 2008.

[15] R. Shende, Software Automation Testing Tools for Beginners, 1st ed., 2012.

[16] School of Information Systems, "Pengertian Test Plan," BINUS University, May 3, 2017. [Online]. Available: https://sis.binus.ac.id/2017/05/03/pengertian-test-plan/. [Accessed: Mar. 2, 2025].

[17] Coursera Staff, "How to Write Test Cases: A Step-by-Step QA Guide," Coursera, Feb. 7, 2025. [Online]. Available: https://www.coursera.org/articles/how-to-write-test-cases. [Accessed: Mar. 2, 2025].

[18] "Pass/Fail Criteria," The Codest. [Online]. Available: https://thecodest.co/dictionary/passfail-criteria/. [Accessed: Mar. 2, 2025].