Penerapan STLC dalam Pengujian *Automation* Aplikasi *Mobile* (Studi kasus: LMS Amikom Center PT.GIT Solution)

Aulizar Arfan
Program Studi Informatika
Universitas Islam Indonesia
D.I.Yogyakarta, Indonesia
18523199@students.uii.ac.id

Hendrik,M.Eng.
Program Studi Informatika
Universitas Islam Indonesia
D.I. Yogyakarta, Indonesia
hendrik@uii.ac.id

Abstra***ct*— PT.GIT Solution merupakan suatu perusahaan *startup* sekaligus menyediakan jasa *training* dan *certification center.* Aplikasi *e-learning* yang telah tersedia tidak memiliki sistem manajemen untuk menampung kliennya. Maka dari itu, perusahaan membutuhkan sistem manajemen belajar. Maka dari itu, perusahaan mengembangkan sistem manajemen belajar bernama LMS Amikom Center. Dalam pengembangan aplikasi, sering terjadi kesalahan atau tidak sesuai dengan apa yang dibutuhkan. Dengan demikian perlu adanya verifikasi dan validasi ketika masa pengembangan berlangsung. Pengujian *black box* secara automasi dapat mendeteksi secara dini kesalahan atau *bug* yang dialami ketika masa pengembangan berlangsung. Selain itu, dengan pengujian *black box* secara otomatis juga dapat lebih mudah terdokumentasi dan lebih minim terjadi resiko dari kesalahan manusia. Pengujian *black box* berfokus pada spesifikasi kebutuhan dan memiliki sudut pandang pengguna. Hal tersebut mampu mendeteksi sepenuhnya bagaimana perilaku pengguna dalam menggunakan *software*. Dengan adanya *Software Testing Life Cycle* (STLC) juga mampu membuat pengujian lebih teratur, sehingga fokus pengerjaan pengujian dapat lebih terkontrol dengan baik. Hasil dari pengujian menunjukkan bahwa dua dari lima fungsionalitas sudah dapat digunakan.**

***Keywords*—Pengujian, Katalon Studio, STLC.**

I. PENDAHULUAN

Teknologi telepon genggam atau sekarang yang dikenal sebagai *smartphone* merambah ke berbagai bidang dalam kehidupan masyarakat. Sekarang ini, *smartphone* memiliki peranan penting bagi penggunanya. *Software* yang beragam sebagai penunjang berbagai aktivitas pengguna, merupakan salah satu alasan *smartphone* banyak digunakan.

Sistem informasi yang dikembangkan juga beragam menyesuaikan kebutuhan pengguna. Sistem informasi yaitu suatu sistem yang menyediakan informasi untuk manajemen dalam mengambil keputusan dan juga untuk menjalankan operasional perusahaan, dimana sistem tersebut merupakan kombinasi dari orang-orang, teknologi informasi dan prosedur-prosedur yang terorganisasi yang berfungsi untuk manajemen [1].

PT.GIT Solution merupakan perusahaan yang bergerak di bidang pengembangan sistem informasi, *training* dan *certification center*. Selain mengembangkan sistem informasi dan edukasi, PT.GIT Solution juga mengembangkan *game*. PT.GIT Solution merupakan perusahaan di bawah naungan yayasan Amikom Yogyakarta.

Dalam rangka membangun sistem untuk memenuhi kebutuhan edukasi digital, perusahaan memerlukan sistem untuk manajemen pembelajaran. LMS Amikom Center merupakan sistem yang dibangun untuk manajemen belajar yang disesuaikan dengan kebutuhan Amikom Center. Amikom Center merupakan kanal pembelajaran daring milik Amikom untuk klien perusahaan.

Siklus pengembangan perangkat lunak memiliki beberapa bagian. Mulai dari fase *planning, testing* dan *deploying.* Semua bagian tersebut dilakukan dengan cara yang berbeda sesuai dengan kebutuhan. Model tersebut lebih dikenal sebagai *Software Development Lifecycle* (SDLC) [2].

Fase *testing* merupakan fase penting, karena pada tahapan tersebut bertujuan untuk memverifikasi kebutuhan pengguna dan memvalidasi kualitas perangkat lunak. Fase ini juga membantu pengembang untuk menemukan *bug*. Aktivitas pengujian pun memiliki fasenya sendiri yang lebih spesifik yaitu *Software Testing Life Cycle* (STLC).

Pengujian perangkat lunak dibagi menjadi dua, yaitu pengujian *black box* dan pengujian *white box.* Pengujian *black box* adalah pengujian berdasarkan spesifikasi kebutuhan dan tidak perlu terjun langsung untuk mengidentifikasi *code.* Pengujian *black box* merupakan pengujian berdasarkan perspektif pengguna. Sedangkan pengujian *white box* digunakan untuk mendeteksi *logic* yang *error* dalam *code* [3].

Fokus utama dalam melakukan pengujian *black box* adalah menemukan kesalahan fungsionalitas terhadap *software* tanpa memiliki pengetahuan, struktur dan implementasi kode. Sehingga pengujian *black box* sepenuhnya didasarkan pada persyaratan dan spesifikasi *software*. *Input* dan *output* pada pengujian *black box* akan menunjukkan apa yang dialami oleh pengguna. Hal tersebut menjadi alasan pengujian *black box* dapat mengidentifikasi secara akurat kebutuhan pengguna.

Dalam menghimpun validasi fungsionalitas suatu sistem, pengujian *black box* dapat dilakukan secara automasi. Pengujian automasi merupakan pengujian yang dilakukan dengan bantuan aplikasi. Aplikasi tersebut diantaranya adalah Katalon Studio.

Katalon Studio adalah aplikasi *automation testing tool* untuk membuat, menjalankan, dan mengelola pengujian menjadi efisien. Pengujian akan melalui dua tahap yaitu pembuatan *test case* dan eksekusi pengujian dengan menjalankan *test case*. Hasil dari eksekusi tersebut akan disimpan oleh Katalon Studio.

Pada penelitian ini, akan melakukan pengujian metode *black box* secara automasi dengan menggunakan Katalon Studio serta menerapkan *Software Development Lifecycle* (STLC). Pengujian dilakukan pada aplikasi *mobile* dengan *platform* Android. Sampel yang diambil adalah dari aplikasi *Learning Method System* (LMS) Amikom Center dengan

mengambil salah satu perspektif pengguna. Tujuannya adalah untuk mengidentifikasi kelayakan fungsionalitas dari LMS Amikom Center.

II. KAJIAN PUSTAKA

Penelitian [4] membeberkan bahwa fokus dari pengujian *black box* merupakan perincian fungsionalitas dari perangkat lunak. Kecenderungan dari pengujian *black box* memiliki beberapa poin diantaranya:

1. Menemukan fungsi yang tidak valid bahkan tidak ada.
2. Menemukan antarmuka yang tidak sesuai (*interface errors).*
3. Menemukan struktur data dan akses basis data yang salah.
4. Menemukan kesalahan dalam performansi.
5. Menemukan inisialisasi dan terminasi yang salah.

Penelitian [5] melakukan pengujian aplikasi berbasis android belajar tajwid. Dalam penelitian ini, pengujian dilakukan menggunakan *automation testing tool* yaitu Katalon Studio. Dalam penelitian tersebut menunjukan, pengujian *black box* tidak hanya dapat dilakukan secara manual saja, tetapi dapat dilakukan secara automasi. Penelitian menunjukkan, bahwa pengujian fungsionalitas fitur dilakukan dengan cara merekam *test case* dengan perangkat android yang telah terkonfigurasi dengan Katalon dan menjalankan langkah pengujian yang telah terekam. Terdapat beberapa aksi yang ada pada Katalon Studio yang dapat dilakukan selayaknya pengguna berinteraksi dengan *software.* Seperti *tap, swipe,* menambahkan teks dan lain sebagainya. Tetapi dalam penggunaan Katalon Studio masih terdapat beberapa keterbatasan. Namun secara fungsi sebagai *automation testing tool* telah terpenuhi.

Pengujian *black box* memiliki beragam metode atau teknik, antara lain:

1. *Equivalence Partitioning*
2. *Boundary Value Analysis*
3. *Decision Table Based Testing*
4. *State Transition*
5. *Error Guessing*

Selain itu, penelitian pengujian aplikasi prediksi kelulusan SNMPTN dengan menggunakan teknik *boundary value analysis* dalam kesimpulannya menyebutkan bahwa [4]:

1. Metode pengujian *black box* merupakan metode yang mudah digunakan.
2. Estimasi banyaknya data uji dapat dihitung melalui banyaknya field data entri yang akan diuji, aturan entri serta aturan batas atas dan bawah yang memenuhi.
3. Pengujian fungsionalitas masih dapat berjalan namun masih bisa menyebabkan data yang disimpan kurang valid.
4. Pengujian teknik *boundary value analysis* dapat mengetahui entri data mana yang perlu dilengkapi sehingga bisa mendapatkan data yang lebih akurat.

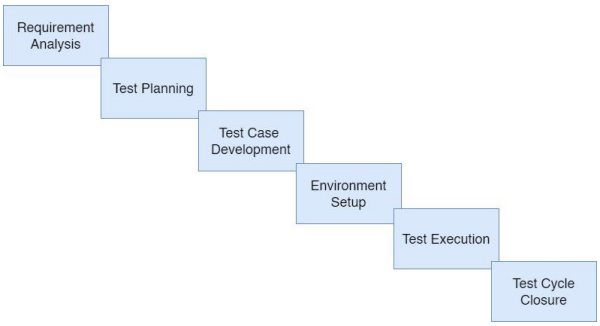
Teknik dalam pengujian *black box* lainnya adalah *equivalence partitions* dalam aplikasi seleksi kenaikan jabatan [6]. Pada pengujian tersebut menyebutkan bahwa pengujian *black box* menggunakan teknik *equivalence partition* dilakukan dengan membagi domain nilai masukan kedalam nilai kelompok valid dan tidak valid. Setelah itu, memilih salah satu nilai dari tiap kelompok yang lain. Penelitian tersebut berhasil menemukan *bug* dalam *test case* pada suatu *form*.

Pengujian *black box* yang dilakukan [7] melakukan teknik pengujian kombinasi. Teknik pengujian yang dikombinasikan adalah teknik *equivalence partitioning* dan *boundary value analysis.* Teknik yang dikombinasikan tersebut digunakan untuk mengidentifikasi ketidaksesuaian luaran sistem dengan kebutuhan pengguna.

Dari penelitian yang telah dilakukan sebelumnya, pengujian *black box* dengan cara manual dan *automation testing tool* maupun dengan teknik tertentu. Tetapi belum ada yang mengimplementasikan STLC. Maka pada penelitian kali ini, akan melakukan penerapan STLC sebagai manajemen pengujian.

III. METODOLOGI

Pada Gambar 1 Software Testing Life Cycle merupakan beberapa tahapan pada *Software Testing Life Cycle.*



Gambar 1 *Software Testing Life Cycle*

3.1 Requirement Analysis

Tahapan pertama yang dilakukan dalam *Software Testing Life Cycle* (STLC) yaitu menganalisis kebutuhan. Dalam tahapan ini, *tester* mengkaji persyaratan dimana identifikasi persyaratan yang dapat diuji berdasarkan sudut pandang pengujian. Adapun yang berinteraksi dengan *tester* diantaranya adalah *client*, *business analyst, technical leads*, *system architect* dan lain-lain dengan tujuan agar dapat memahami persyaratan secara rinci.

Persyaratan-persyaratan yang dimaksud dibagi menjadi dua kategori, yaitu persyaratan fungsional yang berisi hal apa saja yang harus dilakukan perangkat lunak serta persyaratan non fungsional yang berisi kinerja sistem atau ketersediaan keamanan. Kegiatan tersebut meliputi [8]:

1. Dokumen berisi kebutuhan pengguna dapat berupa *file brief.*
2. Membuat daftar jenis pengujian seperti fungsional, keamanan, kinerja dan lain-lain.
3. Menentukan fokus dan prioritas pengujian.

4. Membuat daftar rincian lingkungan uji yang akan dilakukan.
5. Memeriksa kelayakan otomasi apabila diperlukan dan menyiapkan laporan kelayakan otomatisasi.

3.2 Test Planning

Test planning merupakan tahapan dimana rincian setiap pengujian dituangkan pada siklus STLC. Tahap ini *tester* mengumpulkan setiap persyaratan yang diperlukan seperti analisis *file brief* untuk mengidentifikasi kebutuhan sistem serta biaya jika diperlukan, dan mempersiapkan upaya pengujian. Hasil dari tahap ini adalah *test plan* yang akan dituangkan ke dalam *test case*, dan perkiraan waktu pengujian. Setelah itu *tester* dapat memulai untuk mengembangkan butir uji [8]. Poin-poin dalam *test planning* sebagai berikut:

1. Menentukan cakupan dan arah proyek.
2. Mengidentifikasi daftar jenis pengujian.
3. Menentukan metode yang akan digunakan yaitu dengan *automation testing tools* atau pengujian manual.
4. Menentukan lingkungan pengujian.
5. Susun tahapan kegiatan atau prosedur pengujian.
6. Mempertimbangkan resiko siapa saja yang akan terlibat dalam pengujian.

3.3 Test Case Development

Pada tahap ini merupakan tahap pembuatan, verifikasi dan pengerjaan butir uji serta *test script* yang telah dipersiapkan pada *test planning.* Awal dari tahap ini adalah mengidentifikasi data uji serta ditinjau hingga kemudian dikerjakan ulang sesuai dengan persyaratan yang telah dipenuhi. [8]. Aktivitas yang dilakukan adalah sebagai berikut:

1. Membuat butir uji/*test case* dan skrip jika perlu.
2. Meninjau kasus dan skrip.
3. Membuat data pengujian.

Setelah tahap ini dilakukan, maka hasil yang diperoleh adalah *test case* dan data uji.

3.4 Environment Setup

Pada dasarnya tahap *environment setup* adalah untuk memastikan bahwa *environment test* yang akan dijalankan baik *hardware* maupun *software*, berjalan sesuai dengan rencana. Fase ini juga dapat dilakukan bersamaan dengan fase *test case development.* Fase ini merupakan fase independen atau tidak terikat dengan fase yang lain. Jika tim developer telah menyediakan *environment test,* maka *tester* tidak boleh ikut dalam aktivitas ini. *Tester* hanya wajib untuk melakukan *smoke test. Smoke test* adalah proses pengujian yang dilakukan untuk memastikan perangkat lunak telah stabil. Sementara untuk *tester, smoke test* dapat diartikan sebagai konfirmasi untuk pengujian lebih lanjut [8]. Aktivitas yang dilakukan sebagai berikut:

1. Memahami arsitektur, pengaturan lingkungan, dan mempersiapkan *software* dan *hardware* untuk kebutuhan lingkungan pengujian.
2. Data pengujian dan *setup test environment.*
3. Melakukan *smoke test.*

Hasil yang diperoleh dari fase ini adalah kesiapan lingkungan berikut dengan pengaturan data uji dan hasil dari *smoke test.*

3.5 Test Execution

Tahap ini merupakan tahap eksekusi berdasarkan *test plan* dan *test case* yang telah disiapkan sebelumnya. *Tester* akan memberikan keterangan *pass* untuk *test case* yang berhasil dieksekusi sesuai dengan harapan. Untuk *test case* yang tidak sesuai, maka akan diberi status *fail.* Temuan *bug* akan disampaikan kepada tim pengembang untuk dikoreksi. Pengujian ulang akan dilakukan apabila *bug* telah dikoreksi. Aktivitas yang dilakukan antara lain:

1. Menjalankan *test case* sesuai dengan rencana.
2. Hasil pengujian didokumentasikan untuk kepentingan laporan berikut dengan laporan *bug*.
3. Pengujian ulang dilakukan apabila telah selesai dikoreksi.

Hasil yang didapatkan dari aktivitas ini adalah hasil dari *test case* dan laporan *bug.*

3.6 Test Cycle Closure

Test cycle closer merupakan tahap penutupan dari STLC. Tahap ini berisi tentang pelaporan penyelesaian tes, pengumpulan matriks penyelesaian tes dan hasil tes[8]. *Tester* melakukan diskusi tim, evaluasi dan menganalisis pengujian. Aktivitas yang dilakukan pada tahap ini sebagai berikut:

1. Melakukan evaluasi terhadap waktu, cakupan proyek pengujian, biaya jika ada, perangkat lunak maupun perangkat keras, dan kualitas yang telah tercapai.
2. Laporan penutupan tes.
3. Analisis isu *bug* hasil dari pengujian serta tingkat keparahan *bug.*

IV. HASIL DAN PEMBAHASAN

4.1 Requirement Analysis

Requirement analysis yang dilakukan yaitu dengan berkumpulnya *stakeholder* pengembangan sistem manajemen pembelajaran sehingga terjalin suatu kesepakatan Hal tersebut merupakan langkah awal dari proses pembentukan butir uji. Secara keseluruhan LMS Amikom Center memiliki tiga pengguna yaitu mentor, *mentee* dan admin. *Mentee* adalah pengguna jasa perusahaan agar mendapatkan hak akses pembelajaran. Mentor memiliki peran untuk manajemen pembelajaran dengan memberikan materi, memberi tugas dan memberi kuis. Sedangkan admin berperan sebagai manajemen sistem LMS Amikom Center. Cakupan pengujian diambil dari salah satu pengguna sistem yaitu *mentee*. Kesepakatan yang terbentuk akan menghasilkan *file brief.* Sehingga akan menjadi dasar utama selama pengujian berlangsung. Berikut merupakan rincian kebutuhan *mentee:*

1. Sistem dapat melakukan registrasi.
2. Sistem dapat melakukan autentikasi.
3. Sistem dapat menampilkan isi materi dalam kelas.
4. Sistem dapat mengunggah tugas.
5. Sistem dapat digunakan untuk mengerjakan kuis.

4.2 Test Planning

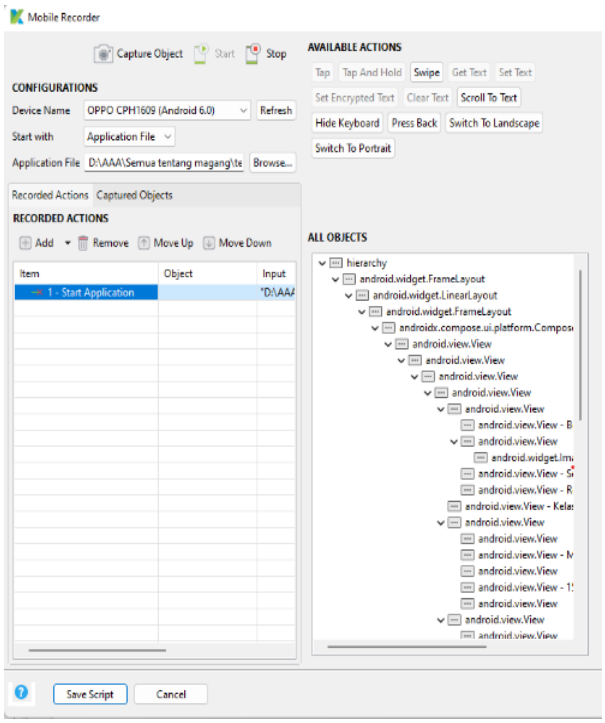
Pengujian akan dilakukan secara automasi. Hal ini didasari dengan pengujian automasi dapat mempersingkat waktu ketika memerlukan pengujian regresi. Pengujian automasi dilakukan menggunakan *automation testing tool* Katalon Studio. Cakupan yang diambil adalah pengguna dengan sudut pandang *mentee.*

Berikut merupakan rincian butir uji:

1. Pengujian fungsionalitas registrasi.
2. Pengujian fungsionalitas autentikasi.
3. Pengujian tampilan isi materi kelas.
4. Pengujian fungsionalitas mengunggah tugas.
5. Pengujian fungsionalitas kuis.

4.3 Test Case Development

Pembuatan *test case* dilakukan dengan cara mengambil *object* melalui *record mobile* menggunakan Katalon Studio. Seperti pada Gambar 2 merupakan proses pengambilan *object* dengan menggunakan cara *record mobile.*



Gambar 2 *Mobile record*

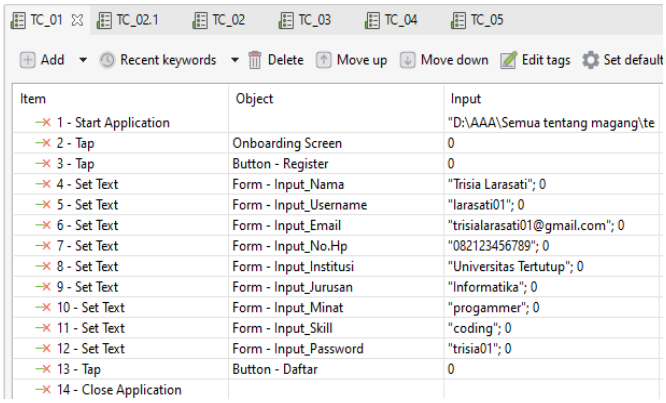
Object yang telah diambil dengan aksi tertentu seperti dalam *record mobile* akan tersimpan dalam *repository*. Sehingga ketika akan dieksekusi, hasil dari *mobile record* tersebut akan dijalankan sesuai dengan *test case*.

Tabel I menunjukkan rangkaian skenario pengujian yang akan digunakan di Katalon Studio. *Test case* yang dibuat adalah hasil dari tahap *requirement analysis*.

TABEL I DAFTAR TEST CASE

Id_Test Case	Nama Test Case
TC_01	Registrasi
TC_02	Login dan logout
TC_03	Tampilan isi materi
TC_04	Unggah tugas
TC_05	Kuis

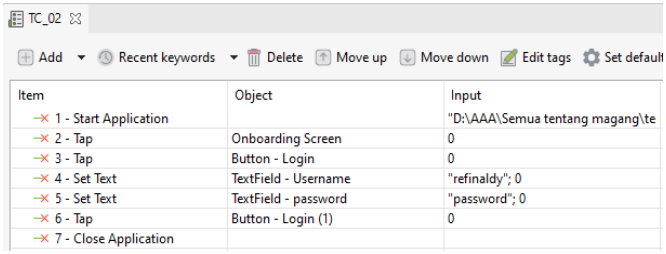
Gambar 3 menunjukkan pengambilan *test case* dari fitur registrasi untuk *mentee*.



Gambar 3 Test case registrasi

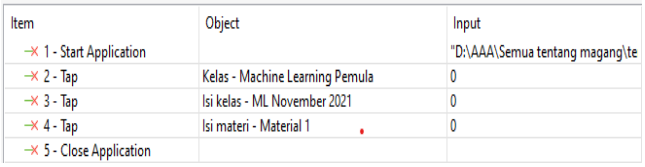
Pada gambar 3 menunjukkan pengujian fungsionalitas *mentee* harus mendaftarkan diri dengan mengisi formulir agar mendapatkan hak akses sebagai pengguna.

Pada *test case login*, tim pengembang memberikan akses akun *mentee* untuk kepentingan pengujian. Sehingga data *username* dan *password* yang diinput merupakan data yang berasal dari *file brief*. Gambar 4 merupakan *test case* yang dibuat untuk pengujian fitur *login*.



Gambar 4 *Test case login*

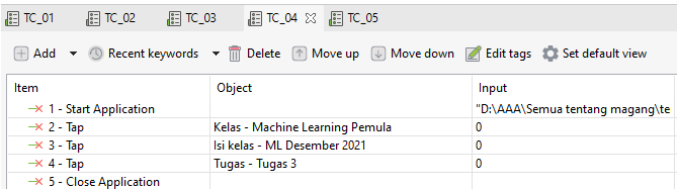
Gambar 5 merupakan *test case* menampilkan materi.



Gambar 5 *Test case* materi

Pengujian pada gambar 5 bertujuan apakah materi dari sisi pengguna lain yaitu mentor bisa tampil pada *mentee.*

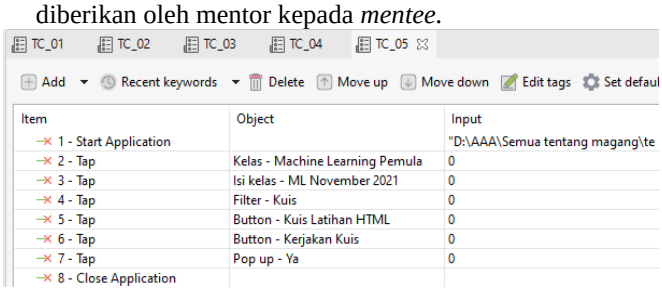
Gambar 6 berikut merupakan *test case* unggah tugas.



Gambar 6 *Test case* unggah tugas

Pengujian pada gambar 6 menguji fungsionalitas fitur unggah tugas.

Gambar 7 menunjukan *test case* kuis. Kuis akan diberikan oleh mentor kepada *mentee.*

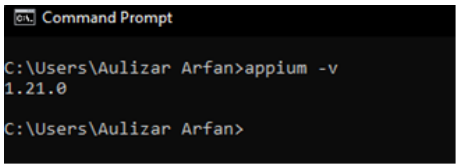


Gambar 7 *Test case k*uis

4.4 Environment Setup

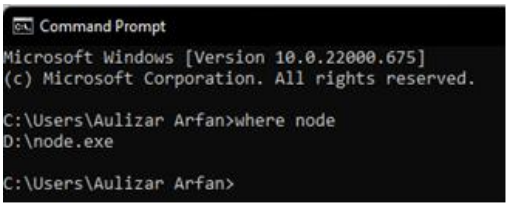
Katalon Studio memiliki beberapa kebutuhan yang harusdipenuhi untuk dapat menjalankan pengujian aplikasi *mobile.* Kebutuhan tersebut diantaranya adalah instalasi Appium dengan versi minimal 1.12.1, perangkat android dengan minimal *OS version* 6.0 dan instalasi Node.js. Gambar 8 dan gambar 9 merupakan cara mengetahui apakah Appium dan Node.js telah terpasang.

Gambar 8 menunjukkan Appium telah terpasang sekaligus mengetahui versi yang digunakan.



Gambar 8 Appium *version*

Gambar 9 menunjukan bahwa Node.js telah terpasang.



Gambar 9 *Node.js*

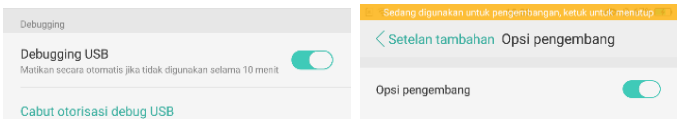
Spesifikasi perangkat yang digunakan untuk pengujian seperti pada Tabel II

TABEL II SPESIFIKASI PERANGKAT

Nama Perangkat	Oppo F3
Operation System (OS)	Android 6 (Marshmallow)
Chipset	Mediatek MT6750T
RAM	4 GB
USB	microUSB 2.0

Katalon Studio memberikan batasan agar pengujian dapat berjalan dengan minimal OS Android versi 6 atau disebut juga *Marshmallow*.

Konfigurasi antara Katalon dengan perangkat android dilakukan dengan cara menghubungkan perangkat menggunakan kabel USB. Setelah itu, perangkat android harus mengaktifkan opsi pengembang dan *debugging* USB seperti pada Gambar 10.



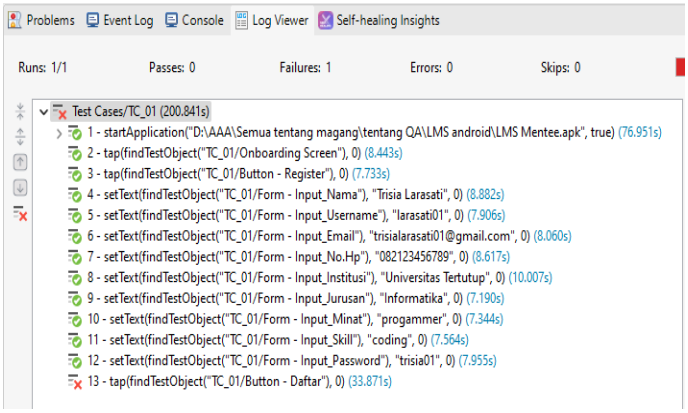
Gambar 10 *Opsi pengembang*

Mengaktifkan fitur opsi pengembang dan *debugging* USB merupakan langkah agar Katalon Studio dapat mendeteksi adanya perangkat. Hal tersebut dilakukan ketika akan menguji *software* dengan perangkat.

4.5 Test Execution

Setelah pembuatan *test case* selesai, maka *test case* tersebut siap untuk diujikan. Urutan pengujian dapat terlihat pada *log viewer*. Centang hijau yang berada di sebelah *test case* menandakan bahwa *test case* berstatus *pass* atau sesuai dengan fungsionalitasnya. Namun apabila silang merah menandakan bahwa *test case* berstatus *failed* atau gagal. Berikut merupakan hasil dari *test case* registrasi pada gambar 11.

Gambar 11 merupakan hasil dari *test case login.*



Gambar 11 *Log viewer* registrasi

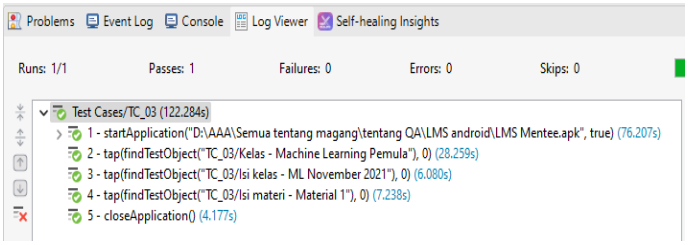
Gambar 11 menunjukkan bahwa hasil eksekusi *test case* login gagal. Terdapat *error* ketika klik *button* daftar. Sehingga pengguna tidak dapat terdaftar ke dalam sistem.

Gambar 12 merupakan hasil eksekusi dari *test case login.*

Gambar 12 Log viewer login

Gambar 12 menunjukkan hasil eksekusi sukses dijalankan. Sistem dapat mengenali pengguna, dalam hal ini adalah sebagai *mentee.*

Gambar 13 merupakan hasil dari *test case* tampilan isi materi.



Gambar 13 *Log viewer* tampilan materi

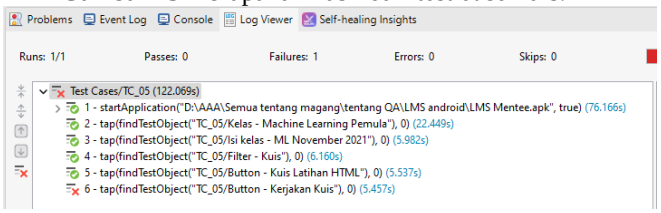
Berdasarkan hasil *test* pada gambar 13 menunjukkan bahwa pengujian berhasil dijalankan. *Mentee* dapat melihat materi yang diberikan oleh mentor.

Gambar 14 adalah hasil dari *test case* unggah tugas.

Gambar 14 *Log viewer* unggah tugas

Eksekusi *test case* yang ditunjukkan oleh gambar 14 gagal. *Software* gagal menampilkan tugas.

Gambar 15 merupakan hasil dari *test case* kuis.



Gambar 15 *Log viewer* kuis

Hasil dari gambar 15 adalah gagal. *Error* disebabkan dari respon ketika klik *button pop up* tidak dapat menampilkan kuis.

4.6 Test Cycle Closer

Rangkaian pengujian telah selesai dilakukan mulai dari tahap *planning* hingga *execution.* Berdasarkan pada tahap *test execution* dapat disimpulkan bahwa hasil dari lima fungsionalitas yang diuji, hanya dua fitur yang dapat memenuhi fungsionalitasnya. *Bug* ditemukan pada fitur registrasi pada *button* daftar, fitur unggah tugas tidak dapat menampilkan tugas, dan fitur kuis pada *button pop up* ketika akan memasuki sesi kuis berlangsung.

V. KESIMPULAN

Pengujian otomatis mampu mendeteksi sejauh mana fungsionalitas aplikasi. Hasil dari pengujian tersebut menunjukan bahwa dari lima aplikasi yang diujikan, hanya dua saja yang berjalan dengan lancar. Pengujian *black box* dapat dengan baik menguji fungsionalitas setiap fitur ketika saat masa pengembang berlangsung. Serta dengan bantuan *automation tool* yaitu Katalon Studio dapat dengan baik mendokumentasikan pengujian dan dapat meminimalisir terjadinya kesalahan manusia. Dengan adanya STLC juga mendukung pengujian menjadi lebih terstruktur dan mampu memberikan fokus yang lebih baik ketika setiap fase yang dilaluinya.

Rekomendasi untuk penelitian yang akan datang adalah lakukan pengujian regresi terhadap aplikasi yang sedang dikembangkan. Tujuannya adalah meninjau keefektifan Katalon Studio dalam melakukan pengujian regresi secara berkala.

REFERENCES

[1] Y. E. 'Irviani, R. 'Anggraeni, Pengantar sistem informasi. Yogyakarta: Penerbit ANDI, 2017.

[2] P. K. Ragunath, S. Velmourougan, P. Davachelvan, S. Kayalvizhi, and R. Ravimohan, "Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC)," 2010.

[3] S. Nidhra, "Black Box and White Box Testing Techniques - A Literature Review," International Journal of Embedded Systems and Applications, vol. 2, no. 2, pp. 29–50, Jun. 2012, doi: 10.5121/ijesa.2012.2204.

[4] M. S. Mustaqbal, R. F. Firdaus, and H. Rahmadi, "PENGUJIAN APLIKASI MENGGUNAKAN BLACK BOX TESTING BOUNDARY VALUE ANALYSIS (Studi Kasus : Aplikasi Prediksi Kelulusan SNMPTN)," 2015.

[5] D. Katarina and E. Windia Ambarsari, "STRING (Satuan Tulisan Riset dan Inovasi Teknologi) AUTOMATION TESTING TOOL DALAM PENGUJIAN APLIKASI BELAJAR TAJWID PADA PLATFORM ANDROID."

[6] *Nasrullah, B. Muslim, C. H. Wijaya, D. Pirmantara, and A. Saifudin, "Penerapan Teknik Equivalence Partitions pada Pengujian Aplikasi Seleksi Kenaikan Jabatan di PT Maju Makmur," Jurnal Teknologi Sistem Informasi dan Aplikasi, vol. 3, no. 4, p. 193, Oct. 2020, doi: 10.32493/jtsi.v3i4.5407.*

[7] T. Hidayat and M. Muttaqin, "Pengujian Sistem Informasi Pendaftaran dan Pembayaran Wisuda Online menggunakan Black Box Testing dengan Metode Equivalence Partitioning dan Boundary Value Analysis," 2018. [Online]. Available: www.ccssenet.org/cis

[8] H. 'Thomas, "STLC (Software Testing Life Cycle) Phases, Entry, Exit Criteria," Guru99, Apr. 2022.