Pengujian Otomatis Aplikasi *Mobile* dengan Teknik *Black-box* Menggunakan Appium

(Studi Kasus: Pengembangan Aplikasi Jala Mobile)

Andi Rivaldo Rambe
Jurusan Informatika
Universitas Islam Indonesia
Yogyakarta, Indonesia
18523151@students.uii.ac.id

Hanson Prihantoro
Jurusan Informatika
Universitas Islam Indonesia
Yogyakarta, Indonesia
hanson@uii.ac.id

***Abstract*—Pengujian adalah salah satu langkah paling penting dalam pengembangan sebuah aplikasi. Pengujian perlu dilakukan berulang kali demi memastikan aplikasi yang sedang dikembangkan bebas dari cacat dan *error*. Pengujian yang dilakukan berulang-ulang secara manual akan menghabiskan banyak waktu dan sumber daya yang diperlukan tim pengembang. Masalah tersebut membutuhkan sebuah solusi untuk memudahkan tim pengembang melakukan pengujian dengan lebih efektif dan efisien, seperti *automated testing tools*. Appium adalah sebuah alat pengujian yang memudahkan pengembang untuk melakukan pengujian secara otomatis. Penelitian ini ditujukan untuk mengetahui efektivitas Appium dalam pengujian aplikasi *mobile* pada aplikasi Jala Mobile. Penelitian dilakukan dengan melakukan pengujian otomasi pada aplikasi Jala Mobile dengan menggunakan teknik *black box testing* dan menerapkan siklus STLC. Hasil penelitian menyimpulkan bahwa pengujian otomatis yang dilakukan menggunakan Appium efektif dalam menemukan cacat dan *error* pada aplikasi Jala Mobile dan dapat membantu tim pengembang menghemat waktu dan sumber daya.**

Keywords— Appium, Automated Testing, Black box testing, Mobile Testing.

I. PENDAHULUAN

Pengujian aplikasi adalah salah satu fase paling penting dan krusial dalam sebuah siklus pengembangan aplikasi. Hal ini biasanya menghabiskan waktu 40% sampai 70% dari proses pengembangan [1]. Pengujian aplikasi merupakan sebuah metode yang digunakan untuk memastikan fungsionalitas dari sebuah program aplikasi. Singkatnya, pengujian aplikasi adalah langkah yang dilakukan untuk memastikan bahwa hasil yang diharapkan sesuai dengan hasil yang didapat dan memastikan aplikasi bebas dari cacat dan *bugs* [2].

Jala adalah sebuah perusahaan teknologi yang bergerak di bidang budidaya, khususnya budidaya udang. Dalam perkembangannya, Jala telah mengembangkan beberapa perangkat keras dan perangkat lunak untuk web dan *mobile* yang ditujukan kepada petambak udang sebagai solusi atas masalah yang seringkali dihadapi. Masalah tersebut muncul salah satunya yaitu sulitnya melakukan pengecekan manual pada setiap kolam dan tambak udang yang ada.

Jala mengembangkan perangkat keras bernama Baruno yang memanfaatkan teknologi IoT (*Internet Of Things)* dan memungkinkan pengguna untuk mengetahui kondisi dan melakukan *monitoring* pada tambak secara *real-time.* Baruno yang terhubung ke internet dapat mengukur empat parameter sekaligus yaitu suhu, salinitas, pH, dan *dissolved oxygen* (DO) atau oksigen terlarut. Dengan terhubung ke internet, data hasil pengukuran langsung dapat tersimpan pada *cloud* sehingga pengguna dapat mengakses data dan informasi kualitas air pada tambak menggunakan *gadget* baik *smartphone* maupun laptop.

Selain perangkat keras, Jala juga mengembangkan aplikasi budi daya tambak pada *platform* web dan *mobile* untuk membantu petambak mengelola data budi daya. Selain mengelola data, aplikasi Jala juga memiliki fitur lain yang dapat membantu petambak seperti rekomendasi berdasarkan data budidaya yang dicatat, informasi harga udang, kabar terbaru mengenai budidaya udang, dan lain sebagainya. Seiring ditambahnya fitur-fitur baru pada setiap *update* membuat aplikasi Jala semakin kompleks dan tidak luput dari *bugs* dan *error*. Untuk menemukan dan mengatasi masalah ini, dilakukanlah pengujian pada aplikasi budi daya tambak Jala.

Secara sederhana, pengujian aplikasi dapat dibedakan menjadi dua yaitu *black box testing* dan *white box testing*. *Blackbox testing* adalah pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program untuk mengetahui apakah fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Teknik *black box* dipilih untuk mengetahui *experience* penggunaan aplikasi dari sisi pengguna, dan fokus pada kebutuhan fungsional aplikasi.

Dengan banyaknya fungsi dan fitur yang akan diuji, pengujian dilakukan dengan memanfaatkan program Appium untuk menJalankan pengujian otomatis. Pengujian otomatis adalah penggunaan kakas pengujian (*testing tools* atau *testing framework*) dalam melakukan pengujian suatu perangkat lunak yang secara signifikan mengurangi waktu yang dibutuhkan untuk melakukan pengujian [3].

Makalah ini akan membahas pengujian menggunakan Appium dengan teknik *black box testing*. Hal ini bertujuan untuk menemukan cacat/*bugs* pada aplikasi Jala dan dengan harapan cacat dan *bugs* yang ditemukan dapat membantu pengembang untuk mengembangkan aplikasi Jala yang bebas cacat. Pada makalah ini tertulis kajian literatur yang terkait, kemudian dijelaskan metodologi yang dilakukan, lalu terdapat hasil, pembahasan, dan kesimpulan.

II. KAJIAN PUSTAKA

Kualitas aplikasi merupakan salah satu perhatian utama dalam pembangunan sebuah aplikasi. Penelitian yang dilakukan oleh Akanksha Verma dkk [4] membandingkan dua metode utama dalam pengujian aplikasi, yaitu teknik *black box testing* dan *white box testing*. Hasil penelitian menyimpulkan bahwa *Black box testing* memiliki kelebihan yaitu: 1) Pengujian dilakukan berdasarkan sudut pandang pengguna; 2) Pengujian dilakukan dengan memanfaatkan aplikasi pihak ketiga sehingga bebas dari bias *developer*; 3) Penguji tidak perlu memiliki skill *programming*; 4) Pengujian tetap efisien meskipun dilakukan pada sistem yang besar. Di sisi lain, teknik *white box* testing memiliki kelebihan di antaranya: 1) Pengujian dilakukan lebih menyeluruh; 2) Pengujian sudah dapat dilakukan sejak awal pengembangan aplikasi karena tidak bergantung dengan GUI; 3) *White box testing* memungkinkan penguji untuk membantu mengoptimasi kode program.

Dari sebuah penelitian [5] yang dikutip [6], *black box testing* dibandingkan dengan dua metode lain yaitu metode *white box testing* dan *grey box testing*. Hasil penelitian menunjukkan bahwa metode *black box testing* tidak melihat struktur ke dalam. Hal ini berbeda dengan *white box testing* yang melihat struktur dalam secara utuh. Pada *grey box testing*, struktur masih diperhatikan namun hanya secara parsial. Untuk metode *black box*, terdapat tujuh pendekatan yang dapat dilakukan. Pendekatan-pendekatan tersebut yaitu: 1) *Equivalence Partitions*; 2) *Boundary Value Analysis*; 3) *Fuzzing*; 4) *Cause Effect Graph*; 5) *Orthogonal Array Testing*; 6) *All Pair Testing*; 7) *State Transition Testing* [7].

Pada penelitian berikutnya terkait pengujian pada aplikasi bergerak [8], telah dilakukan pengujian dengan menggunakan pendekatan *equivalence partitions* pada aplikasi Petgram Mobile. Penelitian dilakukan demi menemukan *error* dan *bugs* pada aplikasi. Pendekatan *equivalence partitions* ini menguji berdasarkan masukan menu dengan menginputkan masukan yang telah dikelompokkan berdasarkan fungsinya. Hasil penelitian berhasil menemukan beberapa *error* pada *form register* dan *form* postingan, yang berarti kualitas aplikasi Petgram mobile masih perlu ditingkatkan lagi.

Selanjutnya sebuah penelitian melakukan pengujian aplikasi *mobile*, masih menggunakan metode *blackbox testing* [9]. Namun dalam hal ini, pengujian dilakukan pada aplikasi *action & strategy* dengan teknologi *phonegap*. Penelitian dilakukan guna mengetahui apakah fungsi-fungsi pada aplikasi telah berJalan sesuai dengan fungsinya dan guna mengevaluasi kesesuaian aplikasi dengan kebutuhan pengguna. Hasil penelitian menunjukkan bahwa pengujian ini dapat dilakukan sehingga tidak adanya *error* atau *bug* pada setiap proses pengujian fungsionalitas aplikasi dari Android maupun web.

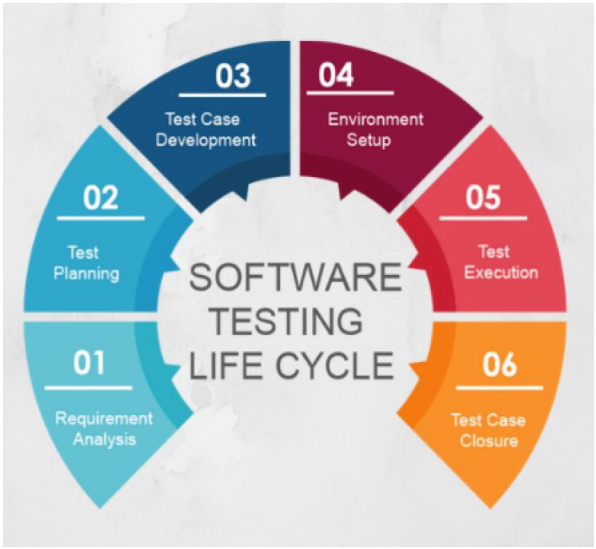
Pengujian aplikasi digunakan untuk menganalisis dan mengintrogasi sebuah program guna menemukan error dan kesalahan dalam program [10]. Berdasarkan metode pengujiannya, pengujian aplikasi dapat dibedakan menjadi pengujian manual dan pengujian otomatis. Dalam melakukan pengujian otomatis, diperlukan sebuah alat bantu untuk dapat mengeksekusi skenario pengujian secara otomatis. Appium adalah sebuah server HTTP yang ditulis menggunakan Node.js. Server Appium bekerja hampir sama seperti Selenium Webdriver, yaitu dengan menerima *request* dari *library* klien melalui JSON dan memproses *request* tersebut dengan cara yang berbeda-beda, tergantung dengan *platform* yang diJalankan [11]

Penelitian yang sudah ada [12], meneliti penggunaan Appium untuk pengujian otomatis pada *user interface* dari berbagai aplikasi *mobile* dengan menggunakan beberapa *script*. Hasil penelitian menyimpulkan bahwa dibandingkan dengan *tools automation* yang lain, Appium unggul pada beberapa hal seperti platform yang independen, mendukung pengujian *hybrid* pada web dan *mobile*, dan mendukung banyak bahasa pemrograman seperti Ruby, Python dan C#.

Makalah yang sudah ada telah melakukan pengujian dengan menggunakan teknik *black box* serta melakukan pengujian secara otomatis menggunakan Appium pada aplikasi *web* dan *mobile* seperti aplikasi Petgram mobile. Selanjutnya pada makalah ini akan dibahas pengujian otomatis dengan teknik *black box testing* menggunakan *tools* Appium pada aplikasi *mobile* Jala.

III. METODOLOGI

Pengujian yang dilakukan pada penelitian ini mengikuti proses STLC *(Software Testing Life Cycle)* seperti diperlihatkan pada Gambar 1.



Gambar 1. Siklus STLC [www.bugraptors.com]

STLC adalah enam urutan langkah dari aktivitas yang dilakukan pada pengujian aplikasi untuk memastikan standar kualitas aplikasi terpenuhi. Langkah pertama adalah *requirement analysis.* Pada langkah ini penguji berdiskusi dengan tim mengenai tujuan pengujian, teknik apa yang akan digunakan, dan detail aplikasi atau fitur yang akan diuji. Aplikasi yang diuji pada penelitian ini adalah Jala *mobile*, pengujian dilakukan untuk memastikan aplikasi Jala bebas dari *bug* dan telah memenuhi standar kualitas yang diharapkan. Pengujian yang dilakukan berupa *automated testing* dengan teknik *black-box* dengan pendekatan *equivalence partitions.*

Tahap kedua pada siklus STLC adalah *test planning.* Pada tahap ini penguji dan tim mempersiapkan *tools* yang akan digunakan, menentukan estimasi waktu pengujian, menentukan fitur yang akan diuji, dan menentukan penjadwalan pengujian setiap fitur. Pada penelitian ini *tools* yang digunakan adalah Appium dan Visual Studio Code, dengan estimasi pengujian selama enam pekan. Fitur yang diuji meliputi fitur *register*, menu *login*, fitur pembuatan tambak dan kolam, fitur memulai siklus budidaya, dan

beberapa fitur untuk mengisi data kolam seperti anco, pakan, dan kualitas air. Penelitian dilakukan dengan jadwal menguji satu fitur untuk satu pekan.

Tahap ketiga adalah *test case development.* Tahap ini meliputi pembuatan dokumen *test case* dan penulisan script *automation test. Test case* yang dibuat untuk penelitian ini ditulis menggunakan *software* Microsoft Excel, dan *script* pengujian ditulis pada Visual Studio Code menggunakan bahasa pemrograman Python. Pengujian yang dilakukan menggunakan *teknik black box*, yang berarti pengujian hanya dilakukan dengan mengamati hasil input dan output program dan penguji tidak perlu mengetahui struktur kode perangkat lunak.

Tahap keempat adalah *environment setup*. Pada tahap ini penguji mempersiapkan perangkat yang digunakan untuk pengujian, dan mempersiapkan kondisi aplikasi yang akan diuji sesuai dengan yang tertulis pada *test case*. Pengujian yang dilakukan pada penelitian ini menggunakan perangkat Android sebagai perangkat yang diuji.

Tahap kelima adalah *test execution.* Pada tahap ini penguji menJalankan *script* pengujian yang sudah ditulis, mencatat hasil pengujian pada *test case*, dan membandingkan hasil yang diharapkan dengan hasil yang didapat pada pengujian. Pada penelitian ini, 43 dari 48 pengujian berhasil dilakukan dan mendapat hasil yang sesuai dengan yang diharapkan.

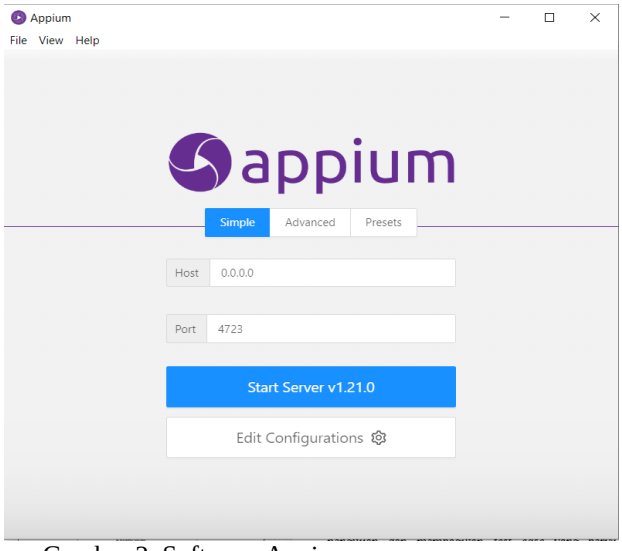
Tahap terakhir adalah *test case closure.* Pada tahap ini penguji melaporkan hasil pengujian kepada tim, lalu mengevaluasi keberhasilan pengujian.

IV. HASIL DAN PEMBAHASAN

1. *Requirement Analysis*

Tahap *requirement analysis* yang dilakukan pada penelitian ini meliputi *meeting* bersama tim untuk membahas aplikasi yang akan diuji dan teknik pengujian yang akan digunakan. Aplikasi yang diuji adalah aplikasi Jala pada platform *mobile* menggunakan teknik *black box testing* dan pengujian dilakukan secara *automated*.

2. *Test Planning*



Gambar 2. Software Appium

Tahap *planning* yang dilakukan meliputi persiapan *tools* dan *software* yang akan digunakan, seperti Appium dan Visual Studio Code. Setelah *tools* siap, penguji berdiskusi dengan tim mengenai fitur apa yang akan diuji. Fitur yang direncanakan untuk diuji pertama kali adalah fitur *register*. Setelah fitur *register* selesai diuji, maka siklus STLC diulang kembali untuk pengujian fitur berikutnya. *Tools* Appium yang digunakan diperlihatkan pada Gambar 2.

3. *Test Case Development*

Pada tahap *test case development* penguji menulis dokumen *test case* yang berisi 48 kasus pengujian, dengan enam skenario yang meliputi skenario *register*, *login*, buat tambak dan kolam, mulai siklus budidaya, catat kualitas air, dan skenario catat data pakan. Setiap skenario dibuat berdasarkan setiap fitur yang akan diuji, dan berisi beberapa kasus pengujian. Karena pengujian dilakukan menggunakan teknik *black box*, kasus uji didapat dari analisis bersama tim seputar proses input dan output dari aplikasi Jala mobile. *Test case* yang ditulis berupa tabel yang berisi informasi dan gambaran proses pengujian akan dilakukan. Parameter yang terdapat dalam *test case* adalah id, deskripsi, kondisi pengujian, langkah pengujian, hasil yang diharapkan, dan hasil yang didapat. *Test case* untuk penelitian ini dapat dilihat pada Tabel 1.

Pada tabel tersebut terdapat kolom *test case id* yang berisi identitas dari setiap kasus yang akan diuji. Kemudian terdapat kolom *test case description* yang menjelaskan deskripsi mengenai kasus pengujian. Kolom selanjutnya adalah *precondition* yang menjelaskan setup atau kondisi apa saja yang harus dipenuhi sebelum pengujian dapat dilakukan. Selanjutnya terdapat kolom *test step*, berisi langkah-langkah yang dilakukan untuk melakukan pengujian pada kasus tersebut. Kemudian terdapat kolom *expected result* yang menjelaskan hasil apa yang diharapkan ketika pengujian telah dilakukan. Kolom terakhir adalah *actual result*, menjelaskan hasil apa yang didapat setelah pengujian berhasil dilakukan.

Tabel 1. Test case yang digunakan

Tes Case Id	Test Case Description		Pre-Condition	Test Step	Expected Result	Actual Result	Status
TC1	Register	[Normal] Melakukan register dengan pengisian form yang benar dan lengkap	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar	password ditampilkan berupa titik - titik - Pengguna akan melihat halaman homepage	- Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Pengguna melihat halaman splash	Pass
TC2	Register	[Abnormal] Melakukan register dengan pengisian email yang salah tetapi pengisian	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar	password ditampilkan berupa titik - titik - Terdapat warning error warna merah "email tidak valid"	- Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Terdapat warning error warna merah	Pass
TC3	Register	[Abnormal] Melakukan register dengan pengisian password dan konfirmasi password yang	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar	password ditampilkan berupa titik - titik - Terdapat warning error warna merah "isian harus sama dengan isian password"	- Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Terdapat warning error warna merah	Pass
TC4	Register	[Abnormal] Melakukan register tanpa pengisian form dan langsung menekan tombol	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar	- Tombol "Daftar Sekarang" tidak bisa ditekan	- Tombol "Daftar Sekarang" tidak bisa ditekan	Pass
TC5	Register	[Abnormal] Melakukan register dengan pengisian email yang benar tetapi password yang	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar	- Terdapat warning error warna merah "password must be at least 5 characters"	- Terdapat warning error warna merah "password must be at least 5 characters"	Pass
TC6	Register	[Abnormal] Melakukan register dengan pengisian email yang benar tetapi password	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar	ditekan - Terdapat warning error warna merah "isian password harus diisi"	- Tombol "Daftar Sekarang" tidak bisa ditekan - Terdapat warning error warna merah	Pass
TC7	Register	[Abnormal] Melakukan register dengan pengisian email dikosongkan tetapi password	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar	ditekan - Pengisian password dan konfirmasi password ditampilkan berupa titik - titik	- Tombol "Daftar Sekarang" tidak bisa ditekan - Pengisian password dan konfirmasi	Pass
TC8	Register	[Abnormal] Melakukan register dengan pengisian form yang benar kecual nomor telepon	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar	- Terdapat warning error warna merah "periksa nomor telepon harus 11 atau 12 angka"	- Terdapat pop up pesan server error	Pass
TC9	Login	[Normal] Melakukan login dengan mengisi email dan password akun yang benar	User telah terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, mengisi email dan	- Pengguna akan melihat halaman homepage	- Pengguna akan melihat halaman homepage	Pass
TC10	Login	[Abnormal] Melakukan login dengan mengisi email/password yang salah	User telah terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, mengisi salah satu antara	- Terdapat pop up pesan kesalahan - Terdapat warning error warna merah penanda kesalahan	- Terdapat pop up pesan kesalahan	Pass
TC11	Login	[Abnormal] Melakukan login dengan mengosongkan salah satu antara email dan password	User telah terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, mengosongkan salah	- Tombol "Masuk" tidak bisa ditekan - Terdapat warning error warna merah penanda kesalahan	- Tombol "Masuk" tidak bisa ditekan - Terdapat warning error warna merah penanda kesalahan	Pass
TC12	Login	[Normal] Melakukan login dengan mengisi no handphone dan password akun yang benar	User telah terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, mengisi no handphone	- Pengguna akan melihat halaman	- Pengguna akan melihat halaman	Fail

4. Environment Setup

Pada tahap *environment setup*, penguji mempersiapkan perangkat Android yang akan dilakukan *automated testing*, lalu mengunduh aplikasi Jala mobile pada perangkat dan memastikan bahwa aplikasi yang terunduh sudah versi terbaru. Setelah proses *download* selesai, penguji melakukan *setup* pada aplikasi sesuai dengan *precondition* yang tertulis pada *test case* kasus yang akan diuji, misalnya ketika akan menguji skenario *login*, penguji harus memastikan bahwa akun yang digunakan pada pengujian sudah terdaftar dan *username* dan *password* yang akan dimasukkan sudah benar. Aplikasi Jala pada Android diperlihatkan pada Gambar 3.

Gambar 3. Aplikasi Jala pada perangkat Android

5. Test Execution

Pada tahap ini *script* pengujian ditulis pada Visual Studio Code, dan ditulis menggunakan bahasa pemrograman Python. Setelah selesai ditulis, *script* diJalankan. Contoh salah satu script yang digunakan untuk menguji halaman *input* data pakan dapat dilihat pada kode program berikut:

```
test_input_feed(self, driver):

input_button =
driver.find_element_by_accessibility_id("inp
ut_popup")
input_button.click()

feeds_button =
driver.find_element_by_accessibility_id("fee
ds_button")
feeds_button.click()

feeds_field =
driver.find_element_by_accessibility_id("fee
ds_field")
feeds_field.send_keys("15")

input_notes =
driver.find_element_by_accessibility_id("not
es_field")
input_notes.send_keys("Input feed test")

confirm_button =
driver.find_element_by_accessibility_id("not
es_field")
confirm_button.click()
```

Ketika menJalankan *script*, penguji mencatat hasil pengujian pada *test case* dan mencatat apakah pengujian berhasil dilakukan atau gagal, dan membandingkan hasil

pengujian suatu kasus sesuai atau tidak sesuai dengan hasil yang diharapkan. Hasil pengujian ditulis pada tabel seperti pada Tabel 2. Tabel menunjukkan dari 48 kasus yang diuji, 43 kasus berhasil diJalankan dan sesuai dengan yang diharapkan, dan terdapat lima kasus yang tidak dapat diJalankan atau hasilnya tidak sesuai yang diharapkan. Kasus yang tidak lolos pengujian dapat dilihat pada Tabel 3.

Kasus-kasus yang tidak dapat diJalankan disebabkan oleh *bugs* pada aplikasi Jala Mobile. Contohnya pada salah satu kasus dalam skenario login, kasus *login* menggunakan akun Facebook. Pengujian tidak dapat dilakukan karena aplikasi Jala tidak merespon ketika tombol *login* dengan Facebook dipilih. Ketika ditemukan kegagalan dalam pengujian seperti ini, penguji mencatat detail yang menyebabkan kegagalan untuk kemudian dilaporkan kepada tim pengembang untuk dilakukan perbaikan.

Tabel 2. Hasil pengujian

Project Name	Test Case Jala (Mobile)		
Pass	43	Number of test cases	48
Fail	5		

Tabel 3. Kasus-kasus bermasalah

Nama kasus	Masalah	Catatan kepada pengembang
Melakukan register dengan pengisian form yang benar kecuali nomor telepon yang tidak sesuai dengan ketentuan (11 atau 12 angka).	Aplikasi tidak menampilkan *pop-up* peringatan terdapat kesalahan dalam pengisian *form*, *pop-up* yang muncul adalah server *error*.	Seharusnya diberikan *pop-up* atau *warning* bahwa pengguna mengisi *form* dengan format yang tidak sesuai.
melakukan *login* dengan mengisi nomor telepon dan *password* akun yang benar.	*Placeholder* pada *form login* tertulis "isi email/no telp", tetapi *button login* tidak dapat ditekan ketika *form* diisi dengan nomor telepon, dan terdapat *warning error*.	Jika *login* hanya bisa dilakukan menggunakan *email*, sebaiknya *placeholder* diubah.
Login menggunakan akun Facebook.	Aplikasi menunjukkan pesan *error* dan gagal *login* ketika *button login* dengan Facebook ditekan.	Fitur tidak berfungsi, sebaiknya segera diperbaiki.
Melakukan buat kolam dengan mengosongkan keseluruhan detail kolam.	Ketika tidak mengisi keseluruhan detail kolam, data kolam masih tersimpan	Sebaiknya diberi pesan pengingat bahwa detail kolam tidak boleh kosong.
	dengan hasil *output* kolam yang tersimpan adalah " "	
Melakukan mulai siklus budidaya dengan pengisian beberapa informasi yang tidak sesuai dengan ketentuan.	Ketika total tebar, umur, dan lama persiapan diisi dengan format yang tidak sesuai, seperti menggunakan titik, koma, dan simbol, data masih tersimpan dan siklus dijalankan.	*Form* seharusnya hanya bisa diisi dengan format yang sesuai, karena dapat menyebabkan data yang membingungkan pengguna.

Pada tahap ini, server Appium digunakan untuk menghubungkan eksekusi *script* dengan perangkat Android yang digunakan. Penggunaan Appium terbatas pada pengujian menggunakan teknik *black box* dan tidak bisa digunakan dalam pengujian teknik *white box* maupun *grey box*. Penggunaan teknik *black box* dapat dilihat pada tahap ini, dengan pengujian yang dilakukan berada pada level terluar dan berfokus pada perilaku program.

Penggunaan Appium sangat membantu dalam pembuatan *automated test*. Dengan mendukung pengujian pada Android dan IOS dan mendukung banyak bahasa pemrograman, Appium menjadi program yang fleksibel dan dapat digunakan oleh siapa saja. Namun salah satu kekurangan program Appium yang dirasakan selama penelitian adalah keterbatasan Appium dalam menguji aplikasi *hybrid*. Appium tidak dapat digunakan pada situasi ketika program yang diuji beralih dari *native app* ke web *app* dan sebaliknya.

6. *Test Case Closure*

Setelah semua tahap sudah dilalui, penguji melakukan *meeting* dengan tim untuk melaporkan hasil pengujian dan membagikan *test case* yang berisi informasi keberhasilan pengujian. Laporan disampaikan melalui *meeting* terkait keberhasilan pengujian dan masalah yang dihadapi. Setelah selesai melaporkan, proses pengujian kembali kepada tahap pertama untuk menguji fitur lain atau proyek lainnya.

V. KESIMPULAN

Telah berhasil dilakukan pengujian aplikasi mobile Jala dengan teknik *Black-box* menggunakan aplikasi Appium. Pengujian dilakukan dengan mengikuti siklus STLC, yaitu urutan langkah dari aktivitas yang dilakukan pada pengujian aplikasi untuk memastikan standar kualitas aplikasi terpenuhi. Penggunaan teknik *black-box* berhasil membantu penguji menemukan beberapa *bug* dan cacat pada aplikasi Jala mobile. Sedangkan Appium berhasil membantu proses otomatisasi sehingga pengujian dapat dilakukan dengan lebih efektif dan efisien ketika melakukan pengujian berulang, dibandingkan dengan melakukan pengujian secara manual setiap kali aplikasi diuji. Kekurangan dari pengujian yang dilakukan menggunakan Appium adalah keterbatasan ketika aplikasi yang sedang diuji membuka

in-app browser, Appium tidak dapat membaca elemen di dalam browser sehingga *automated testing* gagal dilakukan.

Saran yang dapat diberikan setelah melakukan pekerjaan ini adalah perlunya dikembangkan *tools* pengujian yang dapat melakukan *automated test* pada *in-app browser*. Saran berikutnya adalah perlu dilakukan kajian untuk mengukur efisiensi penggunaan Appium, terlebih jika dibandingkan dengan *tools* lain sehingga dapat dipilih *tools* yang tepat untuk kasus yang tepat.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] Sharma, M., & Angmo, R. (2014). Web based automation testing and tools. International Journal of Computer Science and Information Technologies, 5(1), 908-912.

[3] Barus, A. C., & Siburian, L. (2019). Studi Perbandingan Alat Pengujian Otomatis untuk Aplikasi Android. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), 6(6).

[4] Cholifah, W. N., Yulianingsih, Y., & Sagita, S. M. (2018). Pengujian Black Box Testing pada Aplikasi Action & Strategy Berbasis Android dengan Teknologi Phonegap. STRING (Satuan Tulisan Riset dan Inovasi Teknologi), 3(2), 206-210.K. Elissa, "Title of paper if known," unpublished.

[5] Verma, A., Khatana, A., & Chaudhary, S. (2017). A comparative study of black box testing and white box testing. *Int. J. Comput. Sci. Eng*, *5*(12), 301-304.

[6] A. Bansal., 2014. A Comparative Study of Software Testing Techniques. Int. J. Comput. Sci. Mob. Comput., vol. 36, no. 6, pp. 579–584

[7] Jaya, T. S. (2018). Pengujian aplikasi dengan metode blackbox testing boundary value analysis (studi kasus: kantor digital Politeknik Negeri Lampung). *Jurnal Informatika: Jurnal Pengembangan IT*, *3*(1), 45-48.

[8] Khan, M. (2011). Different approaches to black box testing techniques for finding errors. *International Journal of Software Engineering & Applications (IJSEA)*, *2*(4).

[9] Sasongko, B. B., Malik, F., Ardiansyah, F., Rahmawati, A. F., Adhinata, F. D., & Rakhmadani, D. P. (2021). Pengujian Blackbox Menggunakan Teknik Equivalence Partitions pada Aplikasi Petgram Mobile. *Journal ICTEE*, *2*(1), 10-16.

[10] Mohammad, S. M. (2015). Automation Testing in Information Technology. International Journal of Creative Research Thoughts (IJCRT), ISSN, 2320-2882.

[11] Hans, M. (2015). Appium Essentials. Packt Publishing Ltd.

[12] Singh, S., Gadgil, R., & Chudgor, A. (2014). Automated testing of mobile applications using scripting technique: A study on Appium. *International Journal of Current Engineering and Technology (IJCET)*, *4*(5), 3627-3630.