PENGUJIAN OTOMATIS APLIKASI *MOBILE* DENGAN TEKNIK *BLACK-BOX* MENGGUNAKAN APPIUM

Disusun Oleh:

N a m a : Andi Rivaldo Rambe

NIM : 18523151

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGUJIAN OTOMATIS APLIKASI *MOBILE* DENGAN TEKNIK *BLACK-BOX* MENGGUNAKAN APPIUM

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Andi Rivaldo Rambe

NIM : 18523151

Yogyakarta, 13 Oktober 2022

Pembimbing,

(Hanson Prihantoro Putro, S.T., M.T)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGUJIAN OTOMATIS APLIKASI *MOBILE* DENGAN TEKNIK *BLACK-BOX* MENGGUNAKAN APPIUM

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 10 November 2022

Tim Penguji

Hanson Prihantoro Putro, S.T., M.T. _____

Anggota 1

Mukhammad Andri Setiawan, S.T., M.Sc., Ph.D. _____

Anggota 2

Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Andi Rivaldo Rambe

NIM : 18523151

Tugas akhir dengan judul:

PENGUJIAN OTOMATIS APLIKASI MOBILE DENGAN TEKNIK BLACK-BOX MENGGUNAKAN APPIUM

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 13 Oktober 2022

(Andi Rivaldo Rambe)

HALAMAN PERSEMBAHAN

Laporan akhir ini saya persembahkan untuk orang tua yang tidak pernah berhenti memberi dukungan dan doa, yang mendidik dengan tulusnya, serta kasih sayang yang tiada kira.

HALAMAN MOTO

"Sic Parvis Magna. greatness from small beginning"

- Sir Francis Drake -

KATA PENGANTAR

Alhamdulillah, puji syukur kehadirat Allah Swt. yang telah melimpahkan rahmat dan taufiq serta hidayat-Nya sehingga laporan akhir yang berjudul "Pengujian Otomatis dengan Teknik *Black Box* Menggunakan Appium" dapat diselesaikan.

Laporan ini disusun untuk memenuhi persyaratan tugas akhir penjaluran magang untuk dapat memperoleh gelar sarjana pada Program Studi Informatika – Program Sarjana, Fakultas Teknologi Industri, Universitas Islam Indonesia. Pada kesempatan ini, turut diucapkan terima kasih kepada:

1. Kedua orang tua yang selalu memberikan do'a dan memberikan dukungan berupa semangat dan materi kepada penulis.
2. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku kepala Program Studi Informatika Universitas Islam Indonesia.
3. Bapak Hanson Prihantoro Putro, S.T., M.T., selaku dosen pembimbing yang telah memberikan bimbingan maupun arahan terkait penulisan laporan akhir ini.
4. Seluruh dosen dan staf Program Studi Informatika yang telah memberikan ilmu, pengalaman, dan bantuan selama perkuliahan,
5. Mas Farid Inawan, Co-Founder JALA selaku pembimbing lapangan yang telah memberikan panduan dan pembelajaran selama aktivitas magang.
6. Seluruh pimpinan dan rekan kerja di JALA yang telah memberikan kesempatan dan pengalaman magang.
7. Teman-teman masa perkuliahan yang telah memberikan pengalaman tak terlupakan.
8. *Last but not least, I wanna thank me, I wanna thank me for believing in me, I wanna thank me for doing all this hard work, I wanna thank me for never quitting,I wanna thank me for taking no days off, I wanna thank me for just being me at all the times.*

Demikian yang bisa disampaikan di penghujung kata pengantar dalam laporan akhir ini, semoga dari apa yang ditulis dapat memberikan pengetahuan, manfaat serta menjadi sumber referensi bagi siapa saja yang membutuhkannya

Yogyakarta, 13 Oktober 2022

(Andi Rivaldo Rambe)

SARI

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 pemanfaatan 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 berhasil dalam menemukan cacat dan error pada aplikasi Jala mobile dan dapat membantu tim pengembang menghemat waktu dan sumber daya

Kata kunci: Appium, *Automated Testing, Black box testing, Mobile Testing,* STLC.

GLOSARIUM

Actual result	hasil yang didapat setelah melakukan pengujian
Airtable	sebuah *platform* database online yang digunakan untuk menulis *test case*
Appium	sebuah program yang digunakan untuk melakukan pengujian otomatis
Automated testing	pengujian yang dilakukan secara otomatis menggunakan bantuan program pengujian lain
Black box testing	teknik pengujian yang berfokus pada fungsionalitas program
Bugs	cacat pada suatu program yang menyebabkan fitur tidak bekerja seperti seharusnya
Desired capabilities	nilai atau kunci yang ditulis dalam format JSON dan dikirim ke server Appium oleh klien setiap sesi otomasi dijalankan
Expected result	hasil yang diharapkan sebelum melakukan pengujian
STLC	siklus yang terdiri dari enam tahap dalam pengujian aplikasi
Test case	dokumen yang digunakan sebagai *tracking* proses pengujian
White box testing	teknik pengujian yang berfokus pada struktur internal dan kode program

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kualitas aplikasi merupakan salah satu perhatian utama dalam proses pengembangan aplikasi. Untuk memastikan kualitas aplikasi tetap memenuhi standar, maka perlu dilakukan pengujian aplikasi. Pengujian aplikasi adalah sebuah metode yang digunakan untuk memastikan fungsionalitas dari sebuah program aplikasi. Pengujian menjadi salah satu fase krusial dalam sebuah proses pengembangan aplikasi guna menjaga kualitas aplikasi yang dikembangakan dan memastikan aplikasi bebas dari cacat dan *bugs*.

Pengujian yang dilakukan dalam proses pengembangan aplikasi biasanya dilakukan berulang kali demi memastikan standar kualitas aplikasi sudah terpenuhi. Pengujian berulang yang dilakukan secara manual setiap kali memiliki resiko terjadinya *human error* dan menghabiskan banyak waktu. Pada pengembangan aplikasi dengan *resource* dan tim yang masih kecil, melakukan pengujian aplikasi berulang kali menjadi salah satu masalah dan seringkali dilakukan dengan seadanya karena waktu dan tenaga yang terbatas. Untuk mengatasi masalah tersebut, pengujian dapat dilakukan secara otomatis.

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* (Sharma M., Angmo R., 2014). Salah satu *tools* yang dapat digunakan untuk melakukan pengujian otomatis pada aplikasi *mobile* adalah Appium. Appium dipilih menjadi *tools* pengujian pada pelaksanaan magang karena bersifat *open-source* sehingga gratis untuk digunakan oleh siapa saja dan penguji merasa lebih familiar dengan penggunaan Appium dibanding dengan *automation tools* yang lain. Selain itu, Appium dapat mendukung banyak bahasa pemrograman seperti Java, Ruby, PHP, Node, dan Python.

Demi memenuhi standar aplikasi yang ingin dicapai, maka pengujian perlu dilakukan pada setiap struktur aplikasi. Untuk melakukan hal tersebut, terdapat dua teknik dalam pengujian aplikasi, yaitu teknik *black box testing* yang berfokus pada pengujian fungsionalitas aplikasi dan teknik *white box testing* yang berfokus pada pengujian struktur internal aplikasi.

Jala adalah sebuah perusahaan teknologi yang bergerak di bidang budidaya, khususnya budidaya udang (Jala Tech, 2020). 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. 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*.

Laporan akhir ini membahas pengujian otomatis yang dilakukan pada aplikasi Jala mobile. Pengujian pada aplikasi mobile dipilih untuk diangkat menjadi laporan akhir karena paling banyak dikerjakan selama kegiatan magang. Diharapkan laporan ini dapat memberikan pandangan tentang implementasi pengujian otomatis dalam melakukan uji fungsionalitas pada proses pengembangan aplikasi, guna mencapai kebutuhan standar aplikasi. Pengujian otomatis yang dilakukan dengan bantuan *testing tools* diharapkan mengurangi kemungkinan terjadinya *human error* pada proses pengujian dan meningkatkan efisiensi proses pengembangan. Diharapkan hasil yang diperoleh dapat menjadi sebuah solusi IT yang dibutuhkan bagi tim pengembang untuk menjaga kualitas aplikasi dan memastikan aplikasi yang dikembangkan bebas dari cacat dan *bugs.*

1.2 Ruang Lingkup Magang

Telah dilaksanakan aktivitas magang selama enam bulan di Jala Tech sebagai *software* QA *engineer.* Aktivitas yang dilakukan selama magang yaitu:

a. Mengikuti *onboarding* dan perkenalan dengan lingkungan perusahaan Jala Tech.
b. Melakukan pengujian otomatis pada aplikasi web Jala.
c. Melakukan pengujian manual pada aplikasi *mobile* Jala.
d. Melakukan pengujian otomatis pada aplikasi *mobile* Jala.

1.3 Tujuan

Tujuan yang diangkat sebagai bagian dari lingkup magang ini adalah untuk mengimplementasikan pengujian otomatis pada Aplikasi Jala mobile dengan menggunakan

teknik blackbox, dengan harapan dapat meningkatkan efisiensi pengujian aplikasi yang dilakukan pada proses pengembangan aplikasi Jala yang masih menggunakan proses manual.

1.4 Manfaat

1.4.1 Manfaat Bagi Keilmuan

Manfaat-manfaat yang didapat antara lain dapat menambah pengetahuan baru mengenai pengujian otomatis, menemukan kasus pengujian baru, dan menambah pengalaman mengerjakan proyek secara langsung yang berguna sebagai referensi pengerjaan proyek berikutnya.

1.4.2 Manfaat Bagi Perusahaan

Bagi perusahaan, tim pengembang aplikasi Jala dapat melakukan pengujian otomatis tanpa membuang banyak resource waktu yang terbatas. Hasil penelitian berhasil menemukan beberapa cacat dan *bugs* pada program. Temuan ini kemudian menjadi fokus utama tim untuk segera diubah dan diperbaiki guna menjaga kualitas aplikasi Jala.

1.5 Batasan Masalah

a. *Tool* yang digunakan untuk melakukan pengujian otomatis adalah Appium.
b. Teknik yang digunakan dalam penelitian adalah *black box testing*

1.6 Metode

Metode yang digunakan mengikuti siklus STLC (*Software Testing Life Cycle*). Ada enam tahap dalam siklus STLC yaitu:

1. *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.

2. *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.

3. *Test Case Development*

   Tahap ini meliputi pembuatan dokumen *test case* dan penulisan skrip pengujian otomatis.

4. *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*.

5. *Test Execution*

   Pada tahap ini penguji menjalankan skrip pengujian yang sudah ditulis, mencatat hasil pengujian pada *test case*, dan membandingkan hasil yang diharapkan dengan hasil yang didapat pada pengujian.

6. *Test Case Closure*

   Pada tahap ini penguji melaporkan hasil pengujian kepada tim, lalu mengevaluasi keberhasilan pengujian.

1.7 Sistematika Penulisan

1. BAB I PENDAHULUAN

   Bab ini berisi latar belakang masalah, ruang lingkup, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan metode penelitian untuk melakukan otomatisasi pengujian Aplikasi Jala mobile pada perusahaan Jala Tech.

2. BAB II LANDASAN TEORI

   Bab ini berisi penelitian dengan topik serupa dengan penelitian yang dibuat penulis dengan tujuan untuk mendukung isi dari penelitian.

3. BAB III PELAKSANAAN MAGANG

   Bab ini berisi berisi alur proses pengujian otomatis aplikasi mobile menggunakan Appium dari persiapan sampai pengujian.

4. BAB IV REFLEKSI MAGANG

   Bab ini membahas refleksi pelaksanaan magang sebagai Software QA Engineer yang melakukan pengujian otomatis pada aplikasi mobile menggunakan Appium berupa relevansi akademik dan pembelajaran magang.

5. BAB V PENUTUP

   Bab ini membahas kesimpulan dari pengujian otomatis yang dilakukan pada aplikasi Jala mobile. Bab ini juga berisi saran yang diberikan bagi peneliti lainnya yang akan melakukan penelitian serupa di masa depan.

BAB II
DASAR TEORI

2.1 Pengujian Aplikasi

Di dalam pengembangan suatu sistem aplikasi, harus dilakukan tahap pengujian aplikasi. Pengujian aplikasi adalah sebuah metode yang digunakan untuk memastikan produk aplikasi yang dikembangkan sesuai dengan kualitas yang diinginkan, dan memastikan aplikasi bebas dari cacat (Hamilton Thomas, 2022). Tujuan utama dilakukannya pengujian aplikasi adalah untuk mengidentifikasi *error*, *bugs*, cacat, dan mengidentifikasi kesalahan atau fitur yang tidak sesuai dengan persyaratan yang sebenarnya.

Pengujian aplikasi meliputi pengujian manual dan pengujian otomatis. Pengujian manual dilakukan dengan tangan untuk mempelajari apakah fitur dalam aplikasi berfungsi atau tidak. Sedangkan pengujian otomatis memanfaatkan skrip dan *testing tools* untuk menjalankan pengujian pada aplikasi. Pengujian yang dilakukan secara manual membutuhkan banyak sumber daya manusia dan kurang dapat diandalkan pada pengujian dengan skala besar, hal ini disebabkan pengujian tidak bisa dilakukan secara akurat setiap kali karena ada celah terjadinya *human error* pada setiap pengujian. Pengujian yang dilakukan secara manual menghasilkan proses pengujian yang lambat, membosankan, dan memakan banyak waktu. Dengan perkembangan dan meningkatnya jumlah aplikasi mobile pada saat ini, pengujian otomatis dapat menjadi sebuah solusi, dengan melakukan operasi yang sama setiap skrip dijalankan dapat mengeliminasi celah terjadinya *human error* dalam proses pengujian dan menghasilkan hasil pengujian yang lebih akurat. Kedua teknik pengujian ini dapat digunakan untuk mendukung satu sama lain, dimana pengujian otomatis dapat digunakan untuk menjalankan pengujian dalam jumlah besar dengan waktu yang singkat, dan pengujian manual dapat digunakan untuk menguji suatu fitur dengan memanfaatkan pengetahuan penguji untuk melakukan pengujian pada bagian yang dianggap lebih rawan terjadinya *bugs* dan *error*.

Berdasarkan tekniknya, pengujian aplikasi dibedakan menjadi dua yaitu *black box testing* dan *white box testing*. *Black box testing* merupakan pengujian perangkat lunak dari segi spesifikasi fungsional, sedangkan *white box testing* lebih berfokus pada struktur aplikasi seperti kode dan desain.

2.2 Pengujian Otomatis

Automation test atau pengujian otomatis adalah pengujian aplikasi menggunakan skrip atau kode yang memungkinkan program berjalan secara otomatis (Ryandhita Pietro, 2018). Sehingga dapat mengetahui apakah aplikasi berjalan sesuai dengan apa yang diharapkan. Penggunaan *automated testing* dapat dilakukan secara berulang, sehingga jika hasilnya tidak sama dengan yang diharapkan maka artinya aplikasi memiliki cacat dan *bugs*. Pengujian otomatis yang digunakan dengan tepat dapat meningkatkan kecepatan dan efisiensi proses pengujian aplikasi. Pengujian yang dilakukan secara otomatis juga dapat dilakukan kembali menggunakan skrip yang sama ketika aplikasi mendapat pembaruan.

Perbedaan utama antara pengujian yang dilakukan secara manual dengan pengujian otomatis dijelaskan pada poin-poin berikut:

- Pengujian manual dilakukan sendiri oleh seorang penguji atau QA (manusia) sedangkan pengujian otomatis dilakukan dengan menggunakan kode, skrip dan *tools* pengujian otomatis (komputer) yang dijalankan penguji.
- Pengujian manual kurang akurat karena kemungkinan terjadinya kesalahan oleh penguji atau *human error*, pengujian otomatis lebih akurat karena proses pengujian dijalankan berdasarkan kode dan skrip.
- Pada skala besar, pengujian otomatis dapat dilakukan lebih cepat dibandingkan pengujian manual

Automation testing membutuhkan pembuatan skrip pemrograman yang digunakan sebagai pengganti tenaga manusia ketika melakukan pengujian sistem aplikasi. Skrip tersebut akan menjalankan tahapan pengujian yang dilakukan penguji manual, sehingga ketika dijalankan penguji hanya perlu memantau proses dan menunggu hasil pengujian. Skrip dapat ditulis menggunakan berbagai bahasa pemrograman. Pengujian otomatis juga memerlukan *automation testing tools* untuk menjalankan skrip pengujian. Appium adalah salah satu *testing tools* yang dapat digunakan untuk melakukan pengujian otomatis pada platform mobile. Aktivitas yang dilakukan oleh penguji untuk melakukan pengujian otomatis meliputi menentukan batasan pengujian, menulis *test case*, mempersiapkan *automation tools*, menulis skrip pengujian, dan menjalankan skrip.

Ketika diaplikasikan, pengujian otomatis dapat membantu mengurangi penggunaan sumber daya pada proses pengembangan (Sikender Mohsienuddin Mohammad, 2015). Pada jurnal tersebut, dilakukan penelitian mengenai keuntungan dan manfaat yang didapatkan ketika mengotomasi pengujian aplikasi. Hasilnya menunjukkan bahwa pengujian aplikasi yang

diautomasi memiliki keuntungan seperti mempercepat penemuan *bugs* dengan akurat, menghemat waktu dan tenaga, skrip pengujian dapat dijalankan berulang kali, dan meningkatkan kualitas dan akurasi proses pengujian. Pengujian otomatis juga dapat menjalankan banyak kasus pengujian pada waktu yang singkat, sehingga menghemat penggunaan *resource* waktu dan SDM.

2.3 *Software Testing Life Cycle* (STLC)

Pengujian yang dilakukan pada penelitian ini mengikuti proses STLC (*Software Testing Life Cycle*) seperti diperlihatkan pada Gambar 2.1.

Gambar 2.1 Siklus STLC (Bugraptors, 2019)

STLC adalah enam urutan langkah dari aktivitas yang dilakukan pada pengujian aplikasi untuk memastikan standar kualitas aplikasi terpenuhi (Hamilton, 2022). 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. 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. Tahap ketiga adalah *test case development*, tahap ini meliputi pembuatan dokumen *test case* dan penulisan *script automation test*. 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*. Tahap kelima adalah *test execution*. Pada tahap ini penguji menjalankan skrip pengujian yang sudah ditulis, mencatat hasil pengujian pada *test*

case, dan membandingkan hasil yang diharapkan dengan hasil yang didapat pada pengujian. Tahap terakhir adalah *test case closure*. Pada tahap ini penguji melaporkan hasil pengujian kepada tim, lalu mengevaluasi keberhasilan pengujian.

2.4 Black Box Testing

Secara sederhana, pengujian aplikasi dapat dibedakan menjadi dua yaitu *black box testing* dan *white box testing. Black box testing* juga dapat disebut dengan pengujian fungsional, sebuah pengujian fungsional yang mendesain *test case* berdasarkan dengan informasi mengenai spesifikasi sebuah program. Dengan teknik *Black box*, penguji tidak memerlukan akses terhadap kode suatu program aplikasi. Pengujian *black box* tidak memperdulikan struktur internal program, dan hanya berfokus pada hasil keluaran dan respon program terhadap input dan perintah yang dijalankan. Singkatnya, pengujian *black box* 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 (Ndhira S, 2012). Teknik ini disebut *black box* atau "kotak hitam" karena penguji aplikasi tidak dapat melihat ke dalam kotak tersebut. Yang diketahui hanyalah informasi yang bisa dimasukkan ke dalam kotak, dan "kotak hitam" akan memberikan respon berupa keluaran atau *output*. Teknik *black box* dipilih untuk mengetahui *experience* penggunaan aplikasi dari sisi pengguna, dan fokus pada kebutuhan fungsional aplikasi.

Sebuah penelitian membandingkan penggunaan teknik *black box* dengan teknik *white box* pada pengujian aplikasi (Verma A., Khatana A., Chaundhary S., 2018). 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. Perbandingan kedua teknik dapat dilihat pada Tabel 2.1.

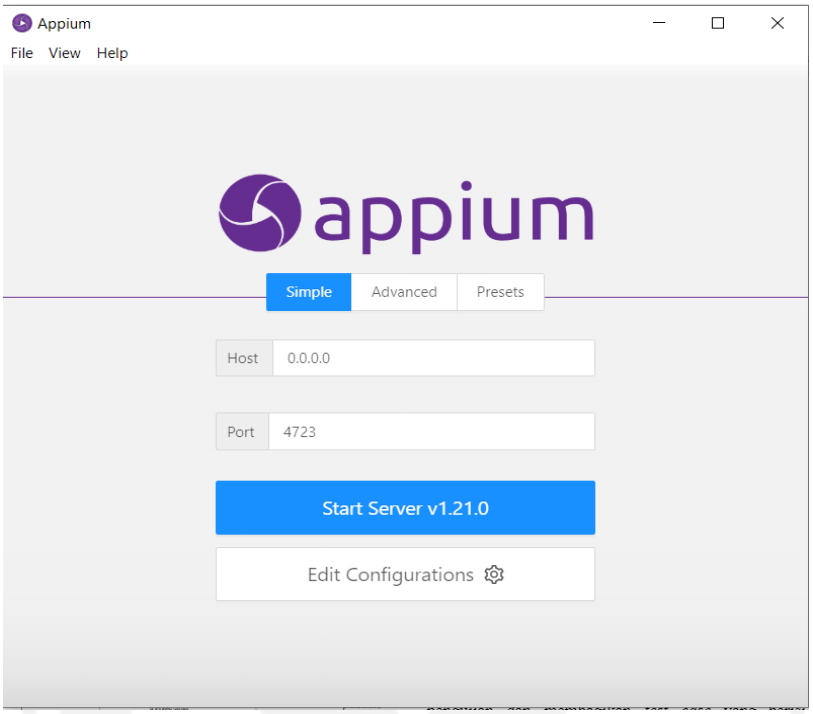
Tabel 2.1 Perbandingan teknik (Verma A., Khatana A., Chaundhary S., 2018).

Black box testing	***White box testing***
Disebut juga teknik pengujian berbasis spesifikasi	Disebut juga teknik pengujian struktural
Struktur internal aplikasi dan kemampuan *coding* tidak dibutuhkan	Memerlukan pengetahuan mengenai struktur internal aplikasi dan kemampuan *coding*
Berfokus pada fungsionalitas sistem	Berfokus pada struktur kode, cabang, perulangan, kondisi, dll.
Tidak dibutuhkan pengetahuan implementasi program	Membutuhkan pengetahuan mengenai implementasi program

2.5 Appium

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 (Hans M, 2015). Appium merupakan tools yang bersifat *open source* dan dapat digunakan untuk menjalankan skrip pengujian dan menguji tiga jenis aplikasi mobile yaitu aplikasi *native*, aplikasi web seluler, dan aplikasi hybrid di *smartphone* dan tablet Android dan iOS. Aplikasi *native* merupakan aplikasi yang dipasang dan dijalankan pada perangkat mobile, biasanya fungsionalitas aplikasi *native* perlu diuji pada beberapa perangkat berbeda seperti android dan IOS. Aplikasi web merupakan aplikasi yang dijalankan melalui Web *browser* pada perangkat mobile, sedangkan aplikasi hybrid adalah gabungan aplikasi *native* dan aplikasi Web.

Appium menggunakan protokol WebDriver dan memiliki komunitas pengembang yang kuat di belakangnya. Melalui Appium kita dapat memeriksa setiap elemen dalam aplikasi mobile untuk dapat mengidentifikasikan elemen mana yang penting untuk diotomasi dan bagaimana menghubungkan setiap elemen tersebut untuk membuat sistem otomatis yang dapat bekerja dengan baik. Tampilan Appium dapat dilihat pada Gambar 2.2



Gambar 2.2 Tampilan Appium

Penelitian yang sudah ada (Singh, S., Gadgil, R., & Chudgor, A., 2014), meneliti penggunaan Appium untuk pengujian otomatis pada *user interface* dari berbagai aplikasi *mobile* dengan menggunakan beberapa skrip. Penelitian dilakukan dengan membandingkan beberapa *tools* yang digunakan untuk pengujian otomatis yaitu Instruments, Uiautomator, Selenium, Monkey Talk, Robotium, dan Appium. 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#.

BAB III
PELAKSANAAN MAGANG

3.1 Aktivitas Magang

Kegiatan magang telah dilakukan selama lebih dari enam bulan dimulai dari Agustus 2021 hingga Maret 2022. Banyak aktivitas yang telah dilalui selama kurun waktu tersebut. Aktivitas-aktivitas tersebut dapat dilihat pada Tabel 3.1. Adapun penjelasan terkait detail aktivitas magang pada Tabel 3.1 dijelaskan pada sub bab berikutnya.

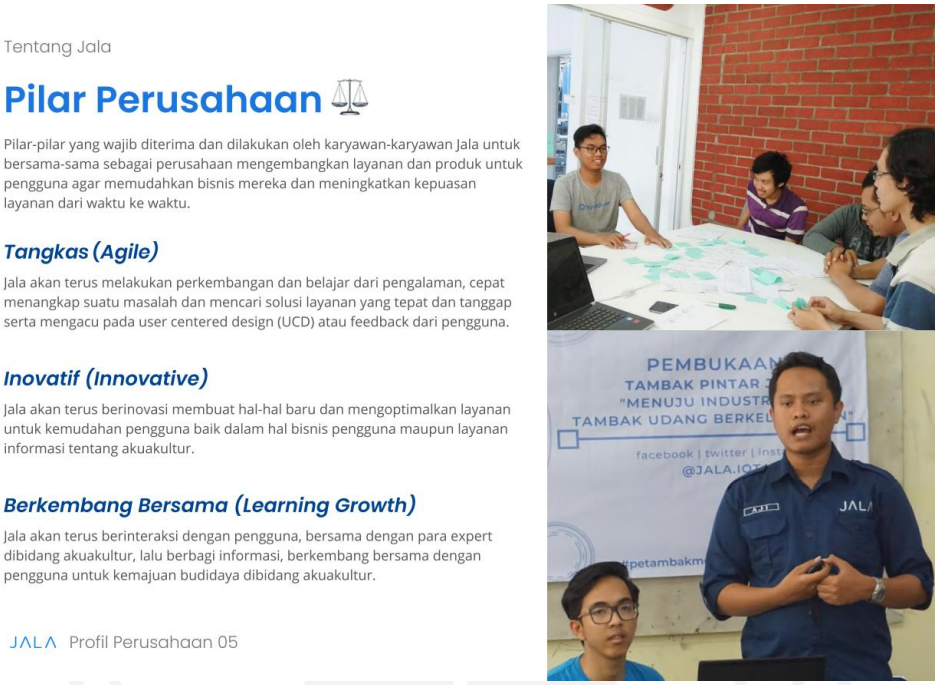
Tabel 3.1 Aktivitas Magang

Aktivitas	Durasi waktu	Rincian Aktivitas
On-boarding magang	1 hari	• Penjelasan mengenai profil perusahaan • Pemberian materi panduan magang di Jala
Pegenalan lingkungan pekerjaan magang	1 hari	• Pengenalan dengan *Supervisor* magang • Pengenalan dengan tim *software development* • Mengenalkan tim dengan *tools* yang dipakai untuk melakukan pengujian
Pengujian manual aplikasi Jala mobile	1 Bulan	• Menulis dokumen *test case* • Melakukan pengujian manual pada aplikasi Jala mobile
Pengujian otomatis aplikasi web Jala	1 Bulan	• Menulis dokumen *test case* • Bekerja sama dengan tim web developer pada pengujian fitur baru

		• Melakukan pengujian otomatis pada aplikasi Web Jala
Pengujian otomatis aplikasi Jala mobile	2 Bulan	• Menulis dokumen *test case* • Melakukan pengujian otomatis pada aplikasi Jala mobile
Membuat *repository* untuk *automation test* yang sudah dilakukan pada aplikasi Jala mobile.	1.5 Bulan	• Bekerja sama dengan tim mobile developer untuk menambahkan *id* pada setiap elemen di aplikasi Jala mobile • Membangun sebuah *repository* untuk semua skrip pengujian menggunakan Pytest
Melakukan *weekly meeting*	Setiap minggu	• Membahas progres dan kendala proyek dengan setiap anggota tim software developer

Pada hari pertama pelaksanaan magang, dilakukan *onboarding* untuk mengenalkan Jala Tech kepada peserta magang. Materi Jala Tech disampaikan oleh Arsanti Anastasia. *Onboarding* dilaksanakan secara daring melalui aplikasi Google Meet. Selain informasi mengenai Jala Tech, diberi juga informasi mengenai rutinitas umum yang dilakukan ketika melaksanakan magang di JALA, seperti *weekly meeting* setiap hari Senin dan Jumat.

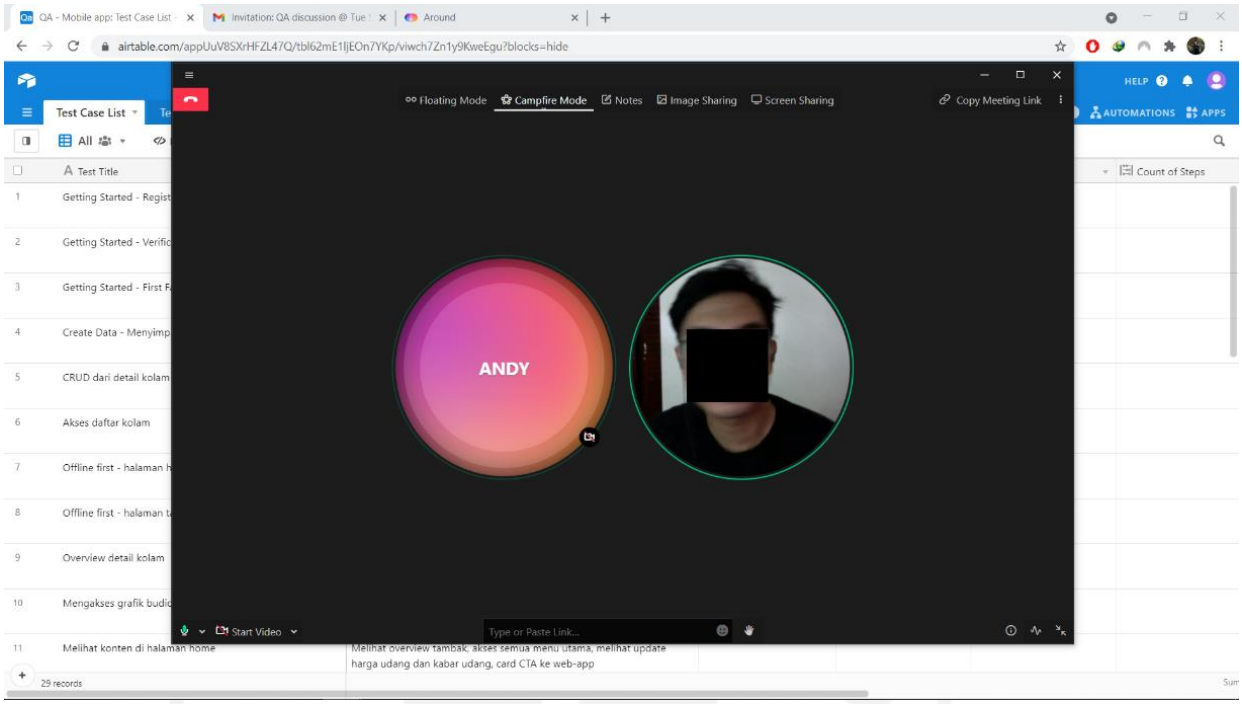
Materi yang diberikan adalah sejarah perusahaan JALA saat dibentuk hingga perkembangan perusahaan saat ini, tokoh-tokoh pendiri, struktur organisasi perusahaan, dan aturan umum selama magang di JALA. Gambar 3.1 memperlihatkan salah satu materi dari *onboarding.*



Gambar 3.1 Materi *Onboarding*

Pada hari kedua pelaksanaan magang, dilakukan *meeting* dengan *project manager* untuk mendiskusikan lingkup pekerjaan magang, *tools* yang digunakan, dan alur pekerjaan. Pertemuan dengan *project manager* diperlihatkan pada Gambar 3.2. *Project manager* juga memberikan dokumen catatan pengujian sebelumnya untuk dipelajari. Dokumen berisi hasil pengujian manual aplikasi Jala mobile. Karena sebelumnya Jala Tech belum memiliki divisi khusus untuk *quality assurance*, maka proyek magang akan dilakukan sendiri dengan bantuan tim proyek lain jika dibutuhkan.

Ada dua proyek yang dikerjakan selama menjalani magang sebagai software QA engineer di Jala. Proyek pertama adalah pengujian otomatis pada aplikasi mobile dengan menggunakan Appium. Proyek kedua adalah pengujian otomatis pada aplikasi web dengan menggunakan Selenium. Pekerjaan dari proyek pengujian otomatis aplikasi mobile dipilih untuk diangkat menjadi laporan akhir karena proyek pengujian aplikasi web dianggap kurang relevan. Pengujian pada aplikasi web yang dilakukan terhitung cukup minor dan hanya menguji satu fitur yang akan dirilis. Jika dipersentasekan, pengujian yang dilakukan pada aplikasi web hanya sekitar 16% dari keseluruhan pekerjaan magang.



Gambar 3.2 *Meeting* dengan *project manager*

3.2 Proyek Pengujian Aplikasi Jala Mobile

Di awal magang, *project manager* menugaskan untuk melakukan pengujian manual pada aplikasi Jala mobile. Pengujian dilakukan mengikuti *test case* baru yang ditulis sesuai dengan kriteria *test case* yang benar. Sebelumnya, dokumen hasil pengujian aplikasi Jala mobile hanya menunjukkan langkah dan hasil pengujian.

3.2.1 Inisiasi

Fokus utama pada proyek ini adalah untuk melakukan pengujian pada aplikasi Jala mobile. Dengan terus berkembangangnya fitur dan perubahan yang disediakan aplikasi Jala pada setiap *update*, maka semakin besar pula celah terjadinya *bugs* dan *error*. Proyek ini ditujukan untuk memastikan aplikasi Jala mobile bebas dari cacat dan *bugs* agar sesuai dengan standar kualitas yang diinginkan. Proyek ini berjalan sebagai bagian dari proyek pengembangan aplikasi JALA, yang terdiri dari tujuh orang. Terdiri dari tiga orang pada tim mobile *Developer*, dua orang pada tim UI/UX *designer*, dan satu orang QA *engineer*. Karena Jala Tech sebelum itu belum memiliki tim QA, maka proyek dilakukan sendiri dengan bantuan *supervisor* jika dibutuhkan.

3.2.2 Perencanaan

Proyek pengujian ini berkaitan erat dengan proyek pengembangan aplikasi Jala mobile. Tim pengembang biasanya mengirimkan versi *alpha* terbaru untuk dilakukan pengujian sebelum dirilis sebagai *update* di App Store dan Play Store. Lingkup pekerjaan proyek meliputi penulisan *test case*, pengujian otomatis, pengujian manual, dan pelaporan. Pengujian hanya dilakukan pada aplikasi Jala mobile versi terbaru dengan fitur yang diuji seperti registrasi akun, *login*, buat tambak, buat kolam, mulai siklus budidaya, mencatat data kualitas air, mencatat data pakan, dan beberapa fitur *input* lainnya.

Proyek direncanakan untuk berjalan dengan total waktu selama kurang lebih tiga bulan, dengan diantaranya diberikan proyek-proyek kecil ketika menunggu tim pengembang menyelesaikan fitur dan *update* terbaru. Pengujian akan dijalankan dengan aplikasi Jala pada sistem Android dan tanpa menggunakan emulator. *Software* yang akan digunakan pada proyek ini antara lain Microsoft Excel, Appium, dan Visual Studio Code. MS Excel digunakan untuk menulis tabel *test case* yang akan digunakan sebagai dokumen *tracking* selama pengujian. Appium digunakan untuk melakukan pengujian otomatis pada aplikasi mobile. Visual Studio Code digunakan untuk menulis skrip pengujian otomatis dengan bahasa Python.

Komunikasi dalam proyek pengujian ini biasanya hanya dilakukan dengan supervisor melalui Google Meet dan Whatsapp. Pada hari Senin dan Jumat, dilakukan *weekly meeting* sebagai laporan progres pekerjaan yang akan dan sudah dilakukan dalam satu pekan. Selain *supervisor*, komunikasi juga sesekali dilakukan dengan tim pengembang aplikasi untuk berdiskusi mengenai *update* terbaru, melaporkan *bugs* yang ditemukan, dan memberi masukan untuk meningkatkan kualitas aplikasi Jala mobile. Target kualitas dari proyek ini adalah ditemukannya semua cacat dan *bugs* pada aplikasi untuk memastikan aplikasi yang dirilis untuk publik lepas dari segala *error* dan cacat. Dengan resiko jika proyek ini gagal dijalankan, maka aplikasi yang dirilis bisa saja berisi beberapa *bugs* yang menghambat fungsionalitas aplikasi.

3.2.3 Eksekusi

Proyek pengujian aplikasi ini dilakukan dengan mengikuti siklus STLC (*Software Testing Life Cycle)*. Siklus STLC terdiri dari enam tahap proses atau fase, masing masing fase memiliki *entry criteria, exit criteria, activities,* dan *deliverable. Entry criteria* adalah ketentuan dan kondisi yang harus dipenuhi sebelum fase dapat dimulai. *Exit criteria* adalah ketentuan dan kondisi yang harus dipenuhi sebelum dapat lanjut ke fase berikutnya. *Activities* adalah

kegiatan yang dilakukan pada sebuah fase, dan *deliverable* adalah hasil yang didapat setelah menyelesaikan pengujian. Tahap siklus STLC dapat dilihat pada Gambar 2.1.

Di dalam magang, proses pengujian yang dilakukan meliputi diskusi dengan tim android developer mengenai rencana pengujian, menulis *test case*, menulis skrip, menjalankan skrip, dan membuat laporan hasil pengujian.

A. Requirement Analysis

Pada tahap pertama siklus STLC, *Stakeholders* atau pada proyek ini *supervisor* memberikan *requirement software* yang kemudian dievaluasi dan dianalisis oleh tim QA. Analisa dilakukan untuk mengetahui detail *software*, fitur yang dapat diuji, menganalisa fitur yang dapat diuji secara otomatis atau manual, dan detail pekerjaan yang akan dilakukan. Hasil dari fase ini adalah ditentukannya aplikasi yang diuji 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*. Tampilan aplikasi Jala mobile diperlihatkan pada Gambar 3.3.

Gambar 3.3 Tampilan *homescreen* Aplikasi Jala mobile

B. *Test Planning*

Tahap kedua adalah tahap perencanaan, pada tahap ini tim QA mempersiapkan rencana dan strategi pengujian berdasarkan *requirement* yang sudah ditentukan pada tahap sebelumnya. Strategi pengujian menyangkut *tools* yang akan digunakan, estimasi waktu, batas pengujian, pembagian peran, dan sumber daya yang tersedia. Hasil dari tahap ini adalah ditentukannya *tools* pengujian yang akan digunakan, yaitu Appium dan Visual Studio Code. Dengan estimasi waktu selama enam pekan. Pengujian yang dilakukan 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. Fitur cek anco merupakan fitur terbaru yang dapat digunakan untuk mencatat data pakan yang diberikan pada suatu jaring tangkur atau anco.

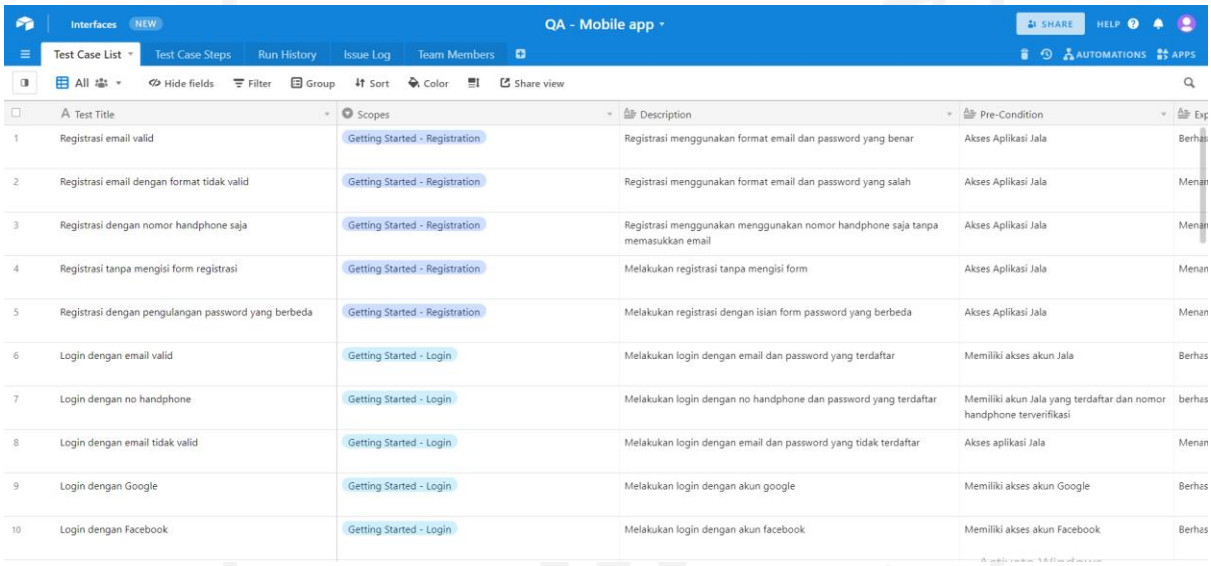
C. Test Case Development

Pada tahap ini, tim QA menulis *test case* yang akan digunakan pada proyek pengujian. *Test case* adalah dokumen yang menggambarkan input, tindakan, atau peristiwa dan respon yang diharapkan, untuk menentukan apakah fitur dari aplikasi bekerja dengan benar. *Test case* biasanya memiliki parameter *identifier*, deskripsi, kondisi pengujian atau *setup*, langkah langkah, dan hasil yang diharapkan, dan hasil yang didapat. Pada proyek ini *test case* ditulis menggunakan Microsoft Excel, namun pada awal kegiatan magang *test case* sempat ditulis menggunakan Airtable.

Airtable adalah platform *online* yang dirancang untuk membuat *spreadsheet* dan basis data, tempat semua data dapat disimpan, dirujuk, dan diambil (MarketTreading, 2021). Selama masa magang, airtable hanya digunakan pada satu bulan pertama. Karena *supervisor* saat itu, yaitu Syauqy Aziz ingin dapat memantau perkembangan penulisan *test case* secara *real-time* maka digunakanlah Airtable. Setelah *supervisor* berganti ke mas Farid Inawan, penulisan *test case* dilakukan di MS Excel. Penulisan *test case* pada Airtable memungkinkan kolaborasi dan dapat dipantau progressnya secara *real-time* oleh *supervisor*. Airtable juga *mobile-friendly* sehingga memudahkan penulisan ketika sedang tidak berada di depan komputer. Penggunaan Airtable selama pelaksanaan magang hanya digunakan pada proyek pengujian aplikasi Web Jala. Pada waktu itu, tim Web *developer* sedang mengembangkan fitur terbaru yaitu fitur cek anco. Sebelum dirilis pada Web app utama Jala, fitur cek anco perlu dilakukan pengujian untuk memastikan tidak ada *bug* dan *error* pada masa pengembangan. Karena waktu yang terbatas, maka *supervisor* pada saat itu ingin hasil pengujian dapat dipantau secara langsung agar jika

ditemukan *bug* dan *error* tim pengembang dapat bertindak cepat untuk memperbaiki masalah yang ditemukan, karena itu *test case* ditulis menggunakan Airtable. Setelah fitur cek anco dirilis, proyek beralih ke pengujian aplikasi mobile. Pada proyek pengujian aplikasi mobile, *supervisor* yang baru menyarankan agar penulisan *test case* dilakukan dengan Microsoft Excel karena faktor kemudahan dan lebih familiar. Proyek pengujian aplikasi mobile juga tidak memerlukan *progress* yang dapat dipantau secara *real-time* karena selama proyek berlangsung dilakukan *daily meeting* setiap hari.

Test case yang ditulis menggunakan Airtable dapat dilihat pada Gambar 3.4. Skenario yang ditulis pada *test case* pengujian meliputi *register*, *login*, membuat tambak dan kolam, memulai siklus budidaya, mencatat data kualitas air, dan mencatat data pakan.



Gambar 3.4 *Test case* pada Airtable

Perubahan *test case* ke format spreadsheet dapat dilihat pada Gambar 3.5. Pada versi yang ditulis pada Excel, dapat dilihat adanya perubahan pada isi *test case*. Selain itu, versi Excel ini juga melengkapi parameter utama test case, antara lain parameter *identifier*, deskripsi, kondisi pengujian atau setup, langkah-langkah, hasil yang diharapkan, serta hasil yang didapat.

Test Case Skenario *Register*

Pada skenario register, terdapat delapan kasus yang ditulis. *Test case* untuk skenario ini dapat dilihat pada Gambar 3.5. Skenario register memiliki satu kondisi normal pada kasus melakukan *register* dengan pengisian *form* yang benar dan lengkap, dan tujuh kondisi abnormal yaitu saat melakukan *register* dengan berbagai kondisi pengisian *form* yang tidak lengkap dan tidak benar. Salah satu contoh kondisi abnormal adalah melakukan *register* dengan memasukkan *password* yang salah.

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 3. Isi form daftar dengan benar dan lengkap sesuai ketentuan 4. Klik daftar sekarang	- Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Pengguna akan melihat halaman homepage	- Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Pengguna melihat halaman splash screen pemilihan tampilan yang diinginkan	Pass
TC2	Register	[Abnormal] Melakukan register dengan pengisian email yang salah tetapi pengisian password yang benar	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar 3. Isi form daftar dengan email yang tidak sesuai dengan ketentuan (ex: tanpa .com/.co.id) 4. Klik daftar sekarang	- Pengisian password dan konfirmasi 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 "email tidak valid"	Pass
TC3	Register	[Abnormal] Melakukan register dengan pengisian password dan konfirmasi password yang berbeda	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar 3. Isi form daftar dengan pengisian password dan konfirmasi password yang berbeda 4. Klik daftar sekarang	- Pengisian password dan konfirmasi 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 "isian harus sama dengan isian password"	Pass
TC4	Register	[Abnormal] Melakukan register tanpa pengisian form dan langsung menekan tombol "daftar sekarang"	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar 3. Klik daftar sekarang tanpa pengisian form	- 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 tidak sesuai ketentuan	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar 3. Isi form daftar dengan pengisian password dan konfirmasi password yang tidak sesuai dengan ketentuan 4. Klik daftar sekarang	- 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 dikosongkan	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar 3. Isi form daftar dengan pengisian seluruh form dengan benar tetapi password dikosongkan 4. Klik daftar sekarang	- Tombol "Daftar Sekarang" tidak bisa ditekan - Terdapat warning error warna merah "isian password harus diisi"	- Tombol "Daftar Sekarang" tidak bisa ditekan - Terdapat warning error warna merah "isian password harus diisi"	Pass
TC7	Register	[Abnormal] Melakukan register dengan pengisian email dikosongkan tetapi password yang benar	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar 3. Isi form daftar dengan pengisian seluruh form dengan benar tetapi email dikosongkan 4. Klik daftar sekarang	- Tombol "Daftar Sekarang" tidak bisa ditekan - Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Terdapat warning error warna merah "isian e-mail harus diisi"	- Tombol "Daftar Sekarang" tidak bisa ditekan - Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Terdapat warning error warna merah "isian e-mail harus diisi"	Pass
TC8	Register	[Abnormal] Melakukan register dengan pengisian form yang benar kecual nomor telepon yang tidak sesuai dengan ketentuan (11 atau 12 angka)	User belum terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik tombol daftar 3. Isi form daftar dengan pengisian seluruh form dengan benar kecuali nomor telepon yang tidak sesuai ketentuan (ex: hanya mengisi 4 nomor)	- Terdapat warning error warna merah "periksa nomor telepon harus 11 atau 12 angka"	- Terdapat pop up pesan server error	Pending

Gambar 3.5 *Test case* skenario *register*

Test case Skenario *Login*

Terdapat delapan kasus yang ditulis pada skenario *login*. Kasus dapat dilihat pada Gambar 3.6. Skenario *login* memiliki empat kondisi normal yaitu pada skenario kasus melakukan *login* dengan pengisian email dan *password* yang benar, *login* dengan nomor *handphone*, *login* dengan akun Gmail, dan *login* dengan akun Facebook. Skenario *login* memiliki tiga kondisi abnormal yaitu saat melakukan *login* dengan email dan *password* yang tidak benar, dan dua kasus melakukan *login* dengan pengisian salah satu *form* yang dikosongkan.

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 password dengan benar sesuai register 3. Klik masuk	- 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 email dan password dengan salah 3. Klik masuk	- 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 satu antara email dan password 3. Klik masuk	- 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 dan password dengan benar sesuai register 3. Klik masuk	- Pengguna akan melihat halaman homepage	- Pengguna akan melihat halaman homepage	Fail
TC13	Login	[Abnormal] Melakukan login dengan mengisi no handphone/password yang salah	User telah terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, mengisi salah satu antara no handphone dan password dengan salah 3. Klik masuk	- Terdapat pop up pesan kesalahan - Terdapat warning error warna merah penanda kesalahan	- Terdapat pop up pesan kesalahan	Pass
TC14	Login	[Abnormal] Melakukan login dengan mengosongkan salah satu antara no handphone dan password	User telah terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, mengosongkan salah satu antara no handphone dan password 3. Klik masuk	- 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
TC15	Login	[Normal] Melakukan login dengan menggunakan akun facebook	User telah terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik button facebook 3. Klik perintah selanjutnya "lanjutkan sebagai ..."	- Pengguna akan melihat halaman homepage	- Pengguna akan melihat halaman homepage	Fail
TC16	Login	[Normal] Melakukan login dengan menggunakan akun google	User telah terdaftar	1. Masuk kedalam aplikasi, ikutin panah pada splashscreen 2. Di halaman login, klik button google 3. Pilih akun google	- Pengguna akan melihat halaman homepage	- Pengguna akan melihat halaman homepage	Pass

Gambar 3.6 *Test case* Skenario *Login*

Test case Skenario Buat Tambak dan Kolam

Ada sebelas kasus yang ditulis pada skenario pengujian untuk membuat tambak dan kolam. Kasus-kasus dapat dilihat pada Gambar 3.7. Skenario ini memiliki lima kondisi normal, yaitu saat kasus membuat tambak dengan format pengisian data yang benar melalui *homepage* aplikasi, maupun menu *input.* Lalu kondisi normal berikutnya menyangkut pengisian data kolam dengan format pengisian yang benar. Skenario ini memilik enam kondisi abnormal, yaitu saat kasus membuat tambak dengan mengosongkan beberapa isian data yang diperlukan, dan membuat kolam dengan format pengisian data yang tidak sesuai.

TC17	Membuat tambak/farm dan kolam	[Normal] Melakukan buat tambak dengan pengisian form dengan benar dan lengkap melalui homepage	User telah terdaftar	1. Pada halaman utama klik buat tambak 2. Isi seluruh form buat tambak secara lengkap 3. Klik buat tambak	- Kolom zona waktu dapat langsung terisi - Pengguna akan melihat halaman buat kolam	- Kolom zona waktu, dan negara dapat langsung terisi - Pengguna akan melihat halaman buat kolam	Pass
TC18	Membuat tambak/farm dan kolam	[Normal] Melakukan buat tambak dengan pengisian form dengan benar dan lengkap melalui menu input	User telah terdaftar	1. Pada halaman utama klik menu input di bottom navigation 2. Pilih menu buat tambak 3. Isi seluruh form buat tambak secara lengkap 4. Klik buat tambak	- Kolom zona waktu dapat langsung terisi - Pengguna akan melihat halaman buat kolam	- Kolom zona waktu, dan negara dapat langsung terisi - Pengguna akan melihat halaman buat kolam	Pass
TC19	Membuat tambak/farm dan kolam	[Abnormal] Melakukan buat tambak dengan pengosongan beeberapa isian data	User telah terdaftar	1. Pada halaman utama klik buat tambak 2. Isi beberapa form dengan kosong 3. Klik buat tambak	- Tombol buat tambak tidak bisa ditekan - Terdapat warning error warna merah penanda form kosong	- Tombol buat tambak tidak bisa ditekan	Pass
TC20	Membuat tambak/farm dan kolam	[Abnormal] Melakukan buat tambak tetapi langsung mengklik tombol buat tambak	User telah terdaftar	1. Pada halaman utama klik buat tambak 2. Langsung klik tombol buat tambak tanpa mengisi isian form	- Tombol buat tambak tidak bisa ditekan - Terdapat warning error warna merah penanda form kosong	- Tombol buat tambak tidak bisa ditekan	Pass
TC21	Membuat tambak/farm dan kolam	[Normal] Melakukan buat kolam setelah selesai membuat tambak dengan mengisi nominal kolam yang dibutuhkan	User telah terdaftar	1. Membuka halaman buat kolam, melanjutkan dari halaman buat tambak 2. Mengisi nominal kolam yang ingin dibuat dari tambak tadi 3. Klik buat kolam	- Pengguna akan melihat halaman detail buat kolam	- Pengguna akan melihat halaman detail buat kolam	Pass
TC22	Membuat tambak/farm dan kolam	[Normal] Melakukan buat kolam setelah selesai membuat tambak dengan mengisi nominal kolam yang dibutuhkan dari halaman homepage	User telah terdaftar	1. Pada halaman utama klik menu input di bottom navigation 2. Pilih menu buat kolam 3. Menentukan tambak mana yang ingin dibuat kolam 4. Mengisi nominal kolam yang ingin dibuat dari tambak 5. Klik buat kolam	- Pengguna akan melihat halaman detail buat kolam	- Pengguna akan melihat halaman detail buat kolam	Pass
TC23	Membuat tambak/farm dan kolam	[Abnormal] Melakukan buat kolam setelah selesai membuat tambak dengan mengosongkan isian pada buat kolam	User telah terdaftar	1. Membuka halaman buat kolam, melanjutkan dari halaman buat tambak 2. Mengosongkan nominal kolam yang ingin dibuat dari tambak tadi 3. Klik buat kolam	- Tombol buat kolam tidak bisa ditekan - Terdapat warning error warna merah penanda form kosong	- Tombol buat kolam tidak bisa ditekan	Pass
TC24	Membuat tambak/farm dan kolam	[Abnormal] Melakukan buat kolam tetapi langsung mengklik tombol buat kolam	User telah terdaftar	1. Membuka halaman buat kolam, melanjutkan dari halaman buat tambak 2. Tanpa mengisi nominal buat kolam, langsung klik buat kolam	- Tombol buat kolam tidak bisa ditekan - Terdapat warning error warna merah penanda form masih ada yang kosong	- Tombol buat kolam tidak bisa ditekan	Pass
TC25	Membuat tambak/farm dan kolam	[Normal] Melakukan buat kolam setelah menentukan nominal jumlah kolam, mengisi nama kolam dan luas kolam pada halaman berikutnya	User telah terdaftar	1. Melihat halaman detail buat kolam 2. Mengisi nama kolam dan detail luas kolam yang dibutuhkan 3. Klik buat kolam	- Pengguna melihat halaman tambak yang sudah ada detail kolamnya	- Pengguna melihat halaman tambak yang sudah ada detail kolamnya	Pass

Gambar 3.7 *Test case* Skenario Buat Tambak dan Kolam

Test Case Skenario Mulai Siklus Budi Daya

Ada lima kasus yang pada skenario ini. Kasus-kasus tersebut ditunjukkan pada Gambar 3.8. Skenario mulai siklus budidaya memiliki dua kondisi normal yaitu saat kasus melakukan mulai siklus budi daya dengan pengisian data sesuai dengan format yang benar melalui menu *homepage* dan menu *input*. Skenario ini memiliki tiga kondisi abnormal, yaitu saat memulai siklus budi daya dengan pengisian data tidak sesuai dengan format yang benar.

TC28	Memulai Siklus Budidaya	[Normal] Melakukan mulai siklus budidaya melalui tombol mulai siklus di atas halaman utama dengan pengisian informasi benar dan lengkap	User telah terdaftar	1. Pada halaman utama diatas, klik tombol mulai siklus 2. Mengisi seluruh informasi dengan benar dan lengkap 3. Klik mulai siklus	- Di halaman utama terdapat pemberitahuan mengenai siklus yang dimulai
TC29	Memulai Siklus Budidaya	[Normal] Melakukan mulai siklus budidaya melalui menu input pada halaman utama dengan pengisian informasi benar dan lengkap	User telah terdaftar	1. Pada halaman utama di bottom navigation, klik menu input 2. Pilih menu mulai siklus 3. Mengisi seluruh informasi dengan benar dan lengkap 4. Klik mulai siklus	- Di halaman utama terdapat pemberitahuan mengenai siklus yang dimulai
TC30	Memulai Siklus Budidaya	[Abnormal] Melakukan mulai siklus budidaya dengan pengisian beberapa informasi yang tidak sesuai dengan ketentuan	User telah terdaftar	1. Pada halaman utama diatas, klik tombol mulai siklus 2. Mengisi beberapa informasi tidak sesuai dengan ketentuan (ex, pada penentuan total tebar diisi tanda titik atau tanda koma) 3. Klik mulai siklus	- Terdapat warning error warna merah jika total harus berupa nominal
TC31	Memulai Siklus Budidaya	[Abnormal] Melakukan mulai siklus budidaya dengan beberapa pengisian informasi dikosongkan	User telah terdaftar	1. Pada halaman utama diatas, klik tombol mulai siklus 2. Mengosongkan beberapa informasi mulai siklus 3. Klik mulai siklus	- Tombol mulai siklus tidak bisa ditekan - Terdapat warning error warna merah penanda form masih ada yang kosong
TC32	Memulai Siklus Budidaya	[Abnormal] Melakukan mulai siklus budidaya dengan langsung mengklik tombol mulai siklus tanpa pengisian informasi	User telah terdaftar	1. Pada halaman utama diatas, klik tombol mulai siklus 2. Klik mulai siklus tanpa mengisi informasi memulai siklus budidaya	- Tombol mulai siklus tidak bisa ditekan - Terdapat warning error warna merah penanda form masih ada yang kosong

Gambar 3.8 *Test Case Skenario* Mulai Siklus Budi Daya

Test Case Skenario Data Kualitas Air

Ada empat kasus yang ditulis pada skenario mencatat data kualitas air. Kasus-kasus tersebut dapat dilihat pada Gambar 3.9. Skenario ini memiliki satu kondisi normal yaitu kasus saat mencatat data kualitas air dengan pengisian data yang lengkap. Skenario ini memiliki tiga kondisi abnormal, yaitu pada kasus pengisian data yang tidak sesuai format, beberapa informasi yang dikosongkan, dan semua informasi yang dikosongkan.

TC33	Mencatat Data Kualitas Air (Online)	[Normal] Melakukan pencatatan data kualitas air dengan pengisian seluruh informasi lengkap	User telah terdaftar	1. Pada halaman utama di bottom navigation, klik menu input 2. Pilih menu kualitas air 3. Mengisi seluruh informasi dengan benar dan lengkap 4. Klik simpan	- Pop up data tersimpan - Terlempar ke halaman utama
TC34	Mencatat Data Kualitas Air (Online)	[Abnormal] Melakukan pencatatan data kualitas air dengan pengisian beberapa informasi tidak sesuai ketentuan	User telah terdaftar	1. Pada halaman utama di bottom navigation, klik menu input 2. Pilih menu kualitas air 3. Mengisi beberapa informasi tidak sesuai ketentuan (ex: pengisian pH, Suhu dengan tanda titik atau tanda koma) 4. Klik simpan	- Tombol simpan tidak bisa ditekan - Terdapat warning error warna merah penanda form ada yang tidak sesuai ketentuan
TC35	Mencatat Data Kualitas Air (Online)	[Abnormal] Melakukan pencatatan data kualitas air dengan pengisian beberapa informasi dikosongkan	User telah terdaftar	1. Pada halaman utama di bottom navigation, klik menu input 2. Pilih menu kualitas air 3. Mengosongkan beberapa informasi 4. Klik simpan	- Tombol simpan tidak bisa ditekan - Terdapat warning error warna merah penanda form ada yang kosong
TC36	Mencatat Data Kualitas Air (Online)	[Abnormal] Melakukan pencatatan data kualitas air dengan langsung menekan tombol simpan data tanpa pengisian informasi	User telah terdaftar	1. Pada halaman utama di bottom navigation, klik menu input 2. Pilih menu kualitas air 3. Klik simpan tanpa mengisi informasi data kualitas air	- Tombol simpan tidak bisa ditekan - Terdapat warning error warna merah penanda form ada yang kosong

Gambar 3.9 *Test Case Skenario* Data Kualitas Air

Test Case Skenario Data Pakan

Pada skenario ini, ditulis empat kasus seperti yang dapat dilihat pada Gambar 3.10. Skenario ini memiliki satu kondisi normal yaitu pada kasus mencatat data pakan dengan mengisi data yang diperlukan dengan lengkap dan benar, dan tiga kondisi abnormal yaitu pada kasus mencatat data pakan dengan mengisi data dengan tidak sesuai ketentuan, mengosongkan beberapa data, dan mengosongkan seluruh data yang dibutuhkan.

TC36	Mencatat Data Kualitas Air (Online)	[Abnormal] Melakukan pencatatan data kualitas air dengan langsung menekan tombol simpan data tanpa pengisian informasi	User telah terdaftar	1. Pada halaman utama di bottom navigation, klik menu input 2. Pilih menu kualitas air 3. Klik simpan tanpa mengisi informasi data kualitas air	- Tombol simpan tidak bisa ditekan - Terdapat warning error warna merah penanda form ada yang kosong
TC37	Mencatat Data Pakan (Online)	[Normal] Melakukan pencatatan data pakan dengan pengisian seluruh informasi lengkap	User telah terdaftar	1. Pada halaman utama di bottom navigation, klik menu input 2. Pilih menu pakan 3. Mengisi seluruh informasi dengan benar dan lengkap 4. Klik simpan	- Pop up data tersimpan - Terlempar ke halaman utama
TC38	Mencatat Data Pakan (Online)	[Abnormal] Melakukan pencatatan data pakan dengan pengisian beberapa informasi tidak sesuai ketentuan	User telah terdaftar	1. Pada halaman utama di bottom navigation, klik menu input 2. Pilih menu pakan 3. Mengisi beberapa informasi tidak sesuai ketentuan (ex: pengisian pakan dengan tanda titik atau tanda koma) 4. Klik simpan	- Tombol simpan tidak bisa ditekan - Terdapat warning error warna merah penanda form ada yang tidak sesuai ketentuan
TC39	Mencatat Data Pakan (Online)	[Abnormal] Melakukan pencatatan data pakan dengan pengisian beberapa informasi dikosongkan	User telah terdaftar	1. Pada halaman utama di bottom navigation, klik menu input 2. Pilih menu pakan 3. Mengosongkan beberapa informasi 4. Klik simpan	- Tombol simpan tidak bisa ditekan - Terdapat warning error warna merah penanda form ada yang kosong
TC40	Mencatat Data Pakan (Online)	[Abnormal] Melakukan pencatatan data pakan dengan langsung menekan tombol simpan data tanpa pengisian informasi	User telah terdaftar	1. Pada halaman utama di bottom navigation, klik menu input 2. Pilih menu pakan 3. Klik simpan tanpa mengisi informasi data kualitas air	- Tombol simpan tidak bisa ditekan - Terdapat warning error warna merah penanda form ada yang kosong

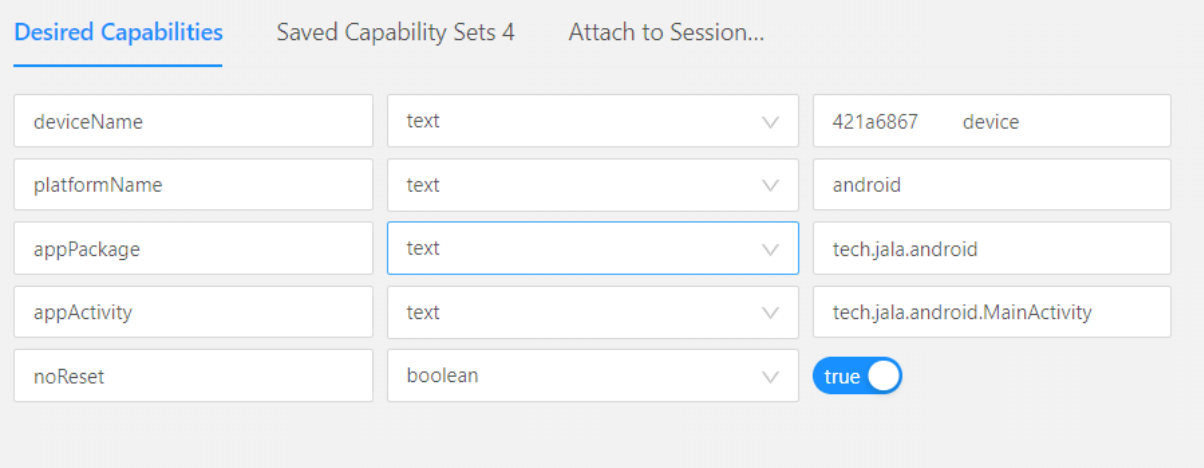
Gambar 3.10 *Test Case Skenario* Data Pakan

Dengan demikian, terdapat 48 kasus uji yang terdiri 40 kasus uji yang diotomasi, dan delapan kasus uji yang dilakukan secara manual. Delapan kasus uji yang dilakukan secara manual terdiri dari kasus yang sama pada skenario data kualitas air dan data pakan. Perbedaan dengan *test case* yang diperlihatkan adalah delapan kasus tersebut dilakukan dengan kondisi perangkat tidak terhubung dengan internet, sehingga tidak bisa diotomasi dengan Appium. Semua kasus uji ini akan dieksekusi menggunakan Appium untuk menguji aplikasi Jala.

D. *Test Environment Setup*

Tahap selanjutnya adalah test environment setup. Pada tahap ini, tim QA melakukan pengujian *environment* untuk memastikan *environment* dapat berjalan dengan baik. Lalu mempersiapkan kondisi *hardware* dan *software* yang akan digunakan dalam pengujian dan melakukan *smoke test*. *Smoke test* adalah adalah pengujian yang dilakukan hanya berfokus pada fungsi utama sebuah aplikasi, dan biasanya mengutamakan kasus positif. Karena pada tahap ini, pengujian hanya dilakukan untuk memastikan aplikasi dapat berjalan dengan semestinya. Aktivitas yang dilakukan meliputi memastikan aplikasi Jala sudah *up-to-date*, memastikan Appium dapat berjalan, dan melakukan *setup* pada aplikasi sesuai dengan *precondition* yang sudah ditentukan pada *test case*. 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. Di dalam magang, proses pengujian pada tahap ini meliputi mempersiapkan Appium, mempersiapkan perangkat android, dan mempersiapkan aplikasi sesuai dengan *pre-condition* yang ditulis pada *test case*.

Pada tahap ini juga aplikasi Appium dipersiapkan. Setelah memulai server appium, hal berikutnya yang harus dilakukan sebelum melakukan pengujian adalah menentukan *desired capabilities. Desired capabilities* merupakan kumpulan kunci dan nilai yang dikodekan dalam objek JSON, yang dikirim dari klien Appium kepada server setiap klien melakukan *request* untuk melakukan sesi otomasi. Ada berbagai macam *capabilities* yang dapat didukung Appium, contoh *capabilities* yang digunakan pada proyek dapat dilihat pada Gambar 3.11 yang diakses saat awal pembuatan proyek. *Desired capabilities* ini dapat direpresentasikan dalam bentuk JSON seperti diperlihatkan pada Gambar 3.12.

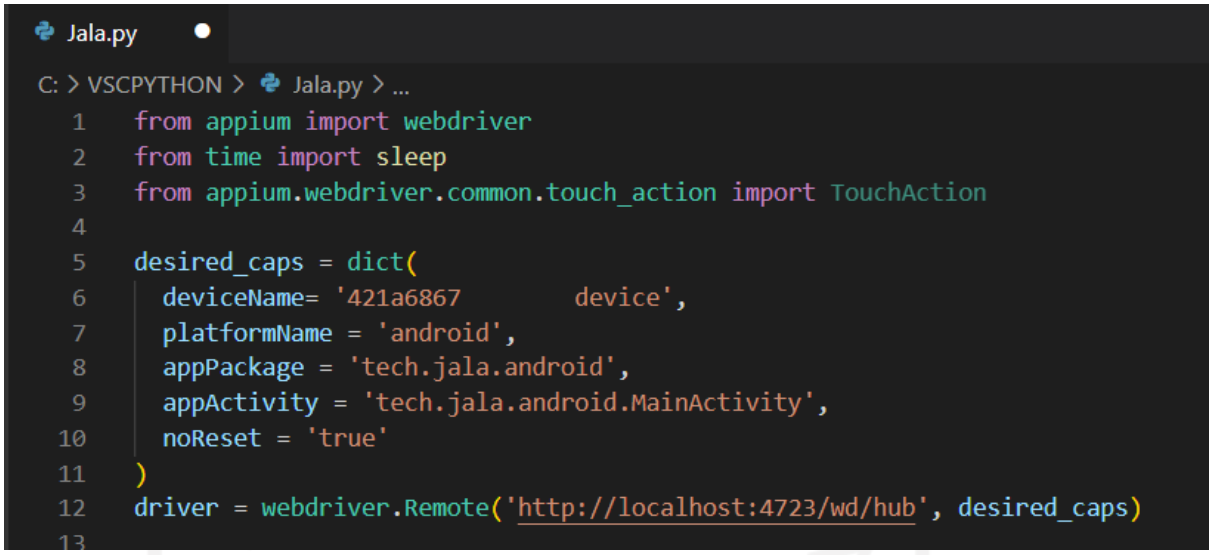


Gambar 3.11 Tampilan *desired capabilities* pada Appium

```
{
  "deviceName": "421a6867 device",
  "platformName": "android",
  "appPackage": "tech.jala.android",
  "appActivity": "tech.jala.android.MainActivity",
  "noReset": true
}
```

Gambar 3.12 Representasi *desired capabilities* dalam bentuk JSON

Dengan *desired capabilities* yang ditulis, penguji memberi intruksi kepada *driver* untuk memulai sesi *automation* untuk aplikasi “tech.jala.android” pada perangkat ”421a6867 device” dengan nama “android”, dan dengan ketentuan setiap kali sesi dijalankan *cache* dan data aplikasi tidak di-*reset*. *Desired capabilities* kemudian dituliskan pada VSCode agar skrip nanti dapat dijalankan pada perangkat dan aplikasi yang sudah disiapkan. Penulisan *desired capabilities* pada file Jala.py dalam VSCode dapat dilihat pada Gambar 3.13. Jala.py adalah file python yang berisi skrip untuk menjalankan aplikasi Jala pada perangkat android menggunakan Appium.



Gambar 3.13 *Desired capabilities* yang ditulis pada VSCode

E. Test Execution

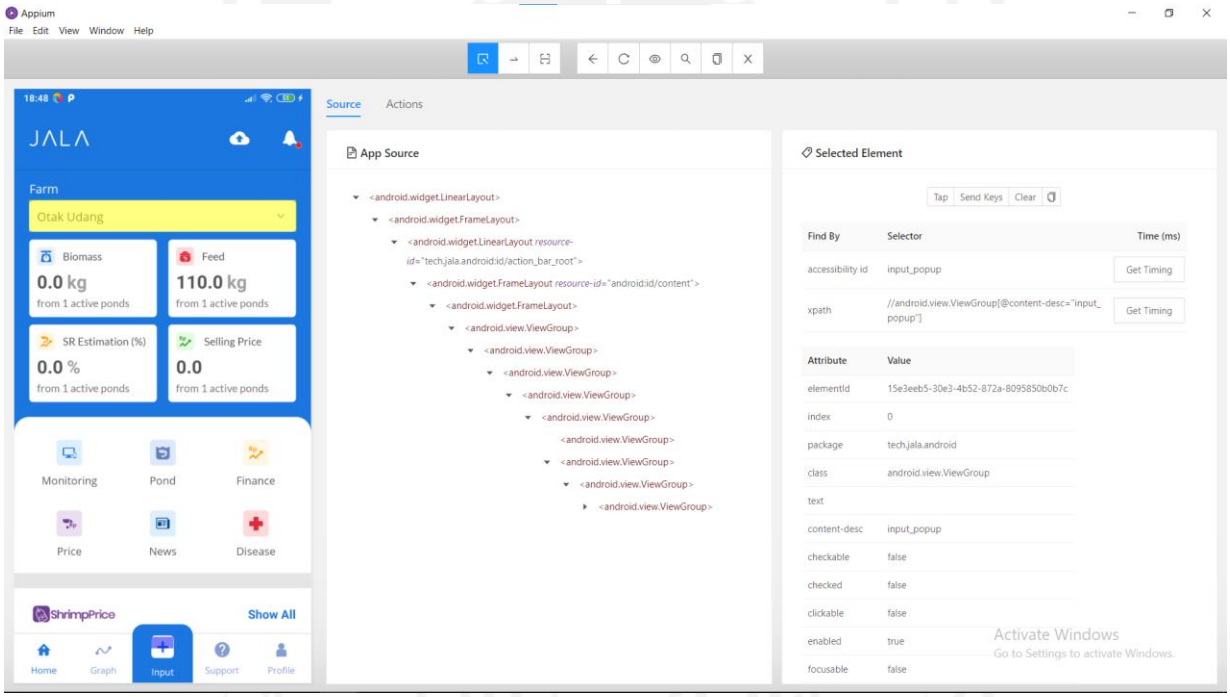
Pada tahap *test execution*, setelah persiapan selesai dilakukan, pekerjaan dilanjutkan dengan melakukan pengujian pada aplikasi Jala mobile menggunakan aplikasi Appium sesuai dengan *test case* yang sudah ditulis. Jika sebuah kasus dapat dijalankan dan memiliki hasil sesuai dengan *expected result* yang tertulis, maka status kasus tersebut adalah *pass* atau berhasil. Jika kasus yang dijalankan tidak sesuai dengan hasil yang diharapkan, maka terdapat *bug* dan terjadi *error*. *Bug* dan *error* yang ditemukan kemudian dicatat pada *test report* untuk disampaikan kepada tim pengembang untuk diperbaiki.

Pada tahap ini, server Appium digunakan untuk menghubungkan eksekusi skrip dengan perangkat Android yang digunakan. Setelah aplikasi Appium diatur pada langkah *test environment setup*, tampilannya diperlihatkan pada Gambar 3.14. Pada bagian kiri gambar terdapat tampilan layar yang merupakan tampilan yang sedang ditampilkan pada perangkat atau emulator yang digunakan. Pada bagian tengah merupakan letak elemen yang sedang dipilih dalam hierarki elemen. Sedangkan pada bagian kanan merupakan detail dari elemen yang dipilih seperti *id*, xpath, atribut, dan nilai. Penggunaan Appium juga 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.

Contohnya adalah pada salah satu kasus pengujian pada fitur login, yaitu login menggunakan akun Facebook. Ketika menjalankan kasus ini, aplikasi membuka situs Facebook dalam in-app browser untuk memasukkan data login. Namun saat in-app browser dibuka, Appium tidak dapat membaca elemen di dalam browser sehingga pengujian otomatis gagal dilakukan. Namun setelah dipelajari lebih lanjut, berdasarkan jawaban pengguna pada situs forum stackoverflow (2019). Hal ini dapat dilakukan dengan cara menambah kode untuk beralih ke tampilan webview. Namun pada kegiatan magang, solusi ini belum bisa diimplementasikan pada proyek karena masih dirasa kompleks dengan waktu yang terbatas dan kasus lain yang perlu diuji.

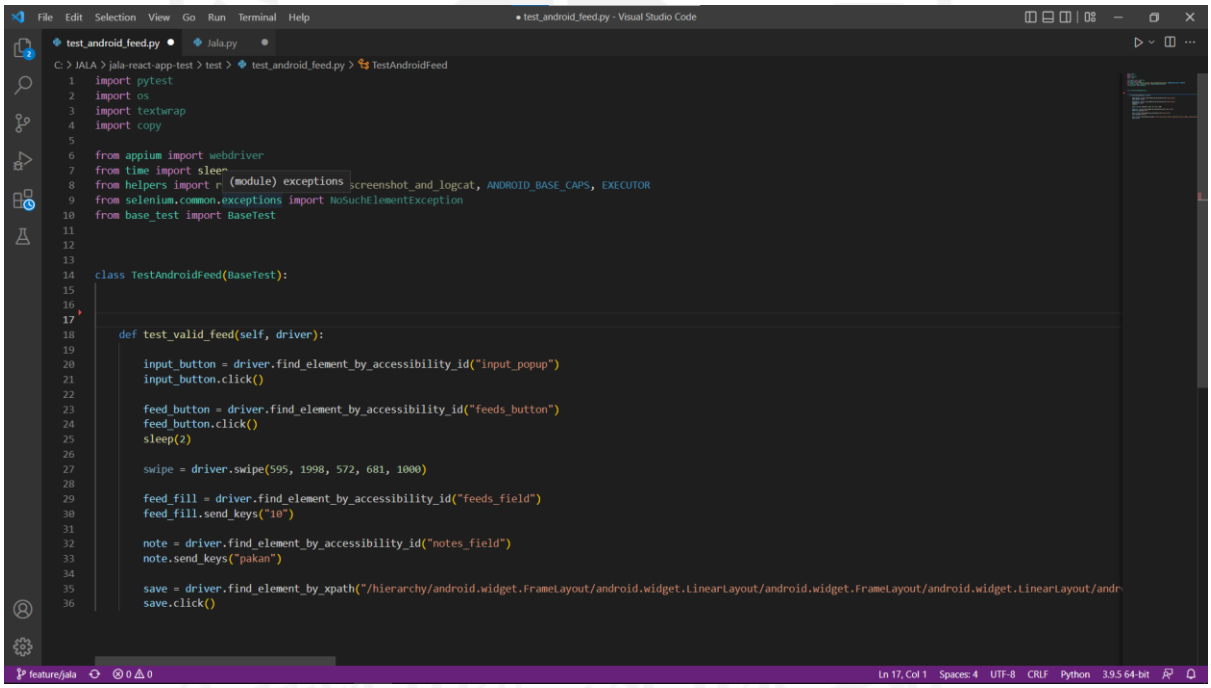


Gambar 3.14 Penggunaan Appium

Pada tahap ini skrip pengujian yang sudah ditulis dijalankan, skrip ditulis pada Visual Studio Code (VSCode) dengan bahasa Python. Contoh salah satu skrip yang digunakan untuk menguji halaman *input* data pakan yang tersedia pada file test_android_feed dapat dilihat pada Gambar 3.15. Ruang kerja pada VSCode diperlihatkan pada Gambar 3.16 yang memperlihatkan pengujian pada *method* test_valid_feed, dalam kelas TestAndroidFeed():.

```
test_input_feed(self, driver):

input_button = driver.find_element_by_accessibility_id("input_popup")
input_button.click()

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

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

input_notes = driver.find_element_by_accessibility_id("notes_field")
input_notes.send_keys("Input feed test")

confirm_button = driver.find_element_by_accessibility_id("notes_field")
confirm_button.click()
```

Gambar 3.15 Skrip pengujian *input* data pakan



Gambar 3.16 *Workspace* pada VSCode

Ketika menjalankan skrip, 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 3.2. 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 3.3.

Tabel 3.2 Hasil Pengujian

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

Tabel 3.3 *Error report*

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 dengan hasil *output* kolam yang tersimpan adalah " "	Sebaiknya diberi pesan pengingat bahwa detail kolam tidak boleh kosong.
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.

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.

F. *Test Cycle Closure*

Pada fase terakhir siklus STLC, pengujian sudah selesai dilakukan. Fase ini berfokus pada penyelesaian *test case* dan persiapan laporan. Tim QA melakukan *meeting* untuk menganalisis dan mendiskusikan proses pengujian yang sudah dilakukan, dan mengevaluasi strategi apa yang dapat digunakan untuk meningkatkan kualitas pengujian berikutnya. Setelah itu QA mempersiapkan laporan hasil pengujian. Laporan ini berisi kesimpulan hasil dari pengujian yang telah dilakukan dan perbandingan antara hasil yang diharapkan, dan hasil yang didapat setelah pengujian dilakukan. Setelah laporan ditulis, maka dilakukan *meeting* dengan *supervisor* dan tim mobile *developer*. Pertemuan ini bertujuan untuk melaporkan dan menganalisa hasil pengujian dengan setiap anggota tim yang bersangkutan.

Pada fase ini *test case* sudah selesai ditulis, dan dikirim kepada *supervisor* untuk dibahas dengan tim pengembang aplikasi mobile. *Bugs* dan *error* yang ditemukan juga dilaporkan kepada tim pengembang aplikasi mobile Jala untuk kemudian ditemukan penyebab kegagalan dan diperbaiki pada *update* berikutnya. Ketika akan ada *update* terbaru, tim pengembang akan memberi informasi mengenai perubahan yang dilakukan pada aplikasi dan *bugs* yang diperbaiki. Setelah versi terbaru siap dirilis, aplikasi akan diuji ulang khususnya pada fitur yang ditambahkan dan fitur yang diperbaiki dari versi sebelumnya. Pengujian ini untuk memastikan bahwa tidak ada *bugs* pada fitur terbaru, dan memastikan fitur yang memiliki *bugs* pada versi sebelumnya sudah berfungsi dengan baik pada versi terbaru. Ringkasan 32 hasil *test case* yang ditulis (dari 48 *test case*) pada proyek ini diperlihatkan pada Tabel 3.4.

Tabel 3.4 Ringkasan *test case*

Tes Case Id	Test Case Description		Expected Result	Actual Result	Status
TC1	Register	[Normal] Melakukan register dengan pengisian form yang benar dan lengkap	- Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Pengguna akan melihat halaman homepage	- Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Pengguna melihat halaman splash screen pemilihan tampilan yang diinginkan	Pass
TC2	Register	[Abnormal] Melakukan register dengan pengisian email yang salah tetapi pengisian password yang benar	- Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Terdapat warning error "email tidak valid"	- Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Terdapat warning error "email tidak valid"	Pass
TC3	Register	[Abnormal] Melakukan register dengan pengisian password dan konfirmasi password yang berbeda	- Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Terdapat warning error "isian harus sama dengan isian password"	- Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Terdapat warning error "isian harus sama dengan isian password"	Pass
TC4	Register	[Abnormal] Melakukan register tanpa pengisian form dan langsung menekan tombol "daftar sekarang"	- 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 tidak sesuai ketentuan	- Terdapat warning error "password must be at least 5 characters"	- Terdapat warning error "password must be at least 5 characters"	Pass
TC6	Register	[Abnormal] Melakukan register dengan pengisian email yang benar tetapi password dikosongkan	- Tombol "Daftar Sekarang" tidak bisa ditekan - Terdapat warning error "isian password harus diisi"	- Tombol "Daftar Sekarang" tidak bisa ditekan - Terdapat warning error "isian password harus diisi"	Pass
TC7	Register	[Abnormal] Melakukan register dengan pengisian email dikosongkan tetapi password yang benar	- Tombol "Daftar Sekarang" tidak bisa ditekan - Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Terdapat warning error "isian e-mail harus diisi"	- Tombol "Daftar Sekarang" tidak bisa ditekan - Pengisian password dan konfirmasi password ditampilkan berupa titik - titik - Terdapat warning error "isian e-mail harus diisi"	Pass
TC8	Register	[Abnormal] Melakukan register dengan pengisian form yang benar kecual nomor telepon yang tidak	- Terdapat warning error "periksa nomor telepon harus 11 atau 12 angka"	- Terdapat pop up pesan server error	Fail

		sesuai dengan ketentuan (11 atau 12 angka)			
TC9	Login	[Normal] Melakukan login dengan mengisi email dan password akun yang benar	- Pengguna akan melihat halaman homepage	- Pengguna akan melihat halaman homepage	Pass
TC10	Login	[Abnormal] Melakukan login dengan mengisi email/password yang salah	- Terdapat pop up pesan kesalahan - Terdapat warning error penanda kesalahan	- Terdapat pop up pesan kesalahan	Pass
TC11	Login	[Abnormal] Melakukan login dengan mengosongkan salah satu antara email dan password	- Tombol "Masuk" tidak bisa ditekan - Terdapat warning error penanda kesalahan	- Tombol "Masuk" tidak bisa ditekan - Terdapat warning error penanda kesalahan	Pass
TC12	Login	[Normal] Melakukan login dengan mengisi no handphone dan password akun yang benar	- Pengguna akan melihat halaman homepage	- Tidak dapat login	Fail
TC13	Login	[Abnormal] Melakukan login dengan mengisi no handphone/password yang salah	- Terdapat pop up pesan kesalahan - Terdapat warning error penanda kesalahan	- Terdapat pop up pesan kesalahan	Pass
TC14	Login	[Abnormal] Melakukan login dengan mengosongkan salah satu antara no handphone dan password	- Tombol "Masuk" tidak bisa ditekan - Terdapat warning error penanda kesalahan	- Tombol "Masuk" tidak bisa ditekan - Terdapat warning error penanda kesalahan	Pass
TC15	Login	[Normal] Melakukan login dengan menggunakan akun facebook	- Pengguna akan melihat halaman homepage	- Terdapat pesan error dan gagal login	Fail
TC16	Login	[Normal] Melakukan login dengan menggunakan akun google	- Pengguna akan melihat halaman homepage	- Pengguna akan melihat halaman homepage	Pass

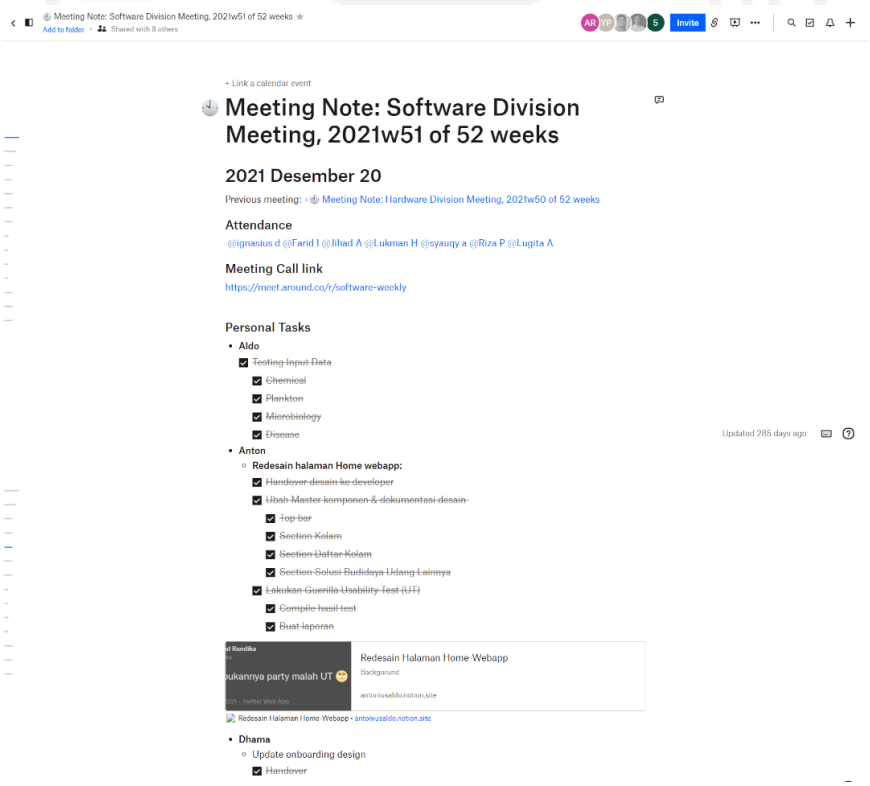
TC17	Membuat tambak/farm dan kolam	[Normal] Melakukan buat tambak dengan pengisian form dengan benar dan lengkap melalui homepage	- Kolom zona waktu dapat langsung terisi - Pengguna akan melihat halaman buat kolam	- Kolom zona waktu, dan negara dapat langsung terisi - Pengguna akan melihat halaman buat kolam	Pass
TC18	Membuat tambak/farm dan kolam	[Normal] Melakukan buat tambak dengan pengisian form dengan benar dan lengkap melalui menu input	- Kolom zona waktu dapat langsung terisi - Pengguna akan melihat halaman buat kolam	- Kolom zona waktu, dan negara dapat langsung terisi - Pengguna akan melihat halaman buat kolam	Pass
TC19	Membuat tambak/farm dan kolam	[Abnormal] Melakukan buat tambak dengan pengosongan beberapa isian data	- Tombol buat tambak tidak bisa ditekan - Terdapat warning error penanda form kosong	- Tombol buat tambak tidak bisa ditekan	Pass
TC20	Membuat tambak/farm dan kolam	[Abnormal] Melakukan buat tambak tetapi langsung mengklik tombol buat tambak	- Tombol buat tambak tidak bisa ditekan - Terdapat warning error penanda form kosong	- Tombol buat tambak tidak bisa ditekan	Pass
TC21	Membuat tambak/farm dan kolam	[Normal] Melakukan buat kolam setelah selesai membuat tambak dengan mengisi nominal kolam yang dibutuhkan	- Pengguna akan melihat halaman detail buat kolam	- Pengguna akan melihat halaman detail buat kolam	Pass
TC22	Membuat tambak/farm dan kolam	[Normal] Melakukan buat kolam setelah selesai membuat tambak dengan mengisi nominal kolam yang dibutuhkan dari halaman homepage	- Pengguna akan melihat halaman detail buat kolam	- Pengguna akan melihat halaman detail buat kolam	Pass
TC23	Membuat tambak/farm dan kolam	[Abnormal] Melakukan buat kolam setelah selesai membuat tambak dengan mengosongkan isian pada jumlah kolam	- Tombol buat kolam tidak bisa ditekan - Terdapat warning error penanda form kosong	- Tombol buat kolam tidak bisa ditekan	Pass
TC24	Membuat tambak/farm dan kolam	[Abnormal] Melakukan buat kolam tetapi langsung mengklik tombol buat kolam	- Tombol buat kolam tidak bisa ditekan - Terdapat warning error penanda form masih ada yang kosong	- Tombol buat kolam tidak bisa ditekan	Pass
TC25	Membuat tambak/farm dan kolam	[Normal] Melakukan buat kolam setelah menentukan nominal jumlah kolam, mengisi nama kolam dan luas kolam pada halaman berikutnya	- Pengguna melihat halaman tambak yang sudah ada detail kolamnya	- Pengguna melihat halaman tambak yang sudah ada detail kolamnya	Pass

TC26	Membuat tambak/farm dan kolam	[Abnormal] Melakukan buat kolam setelah menentukan nominal jumlah kolam, mengosongkan beberapa isian nama kolam dan luas kolam pada halaman berikutnya	- Terdapat warning error penanda form masih ada yang kosong	Kolam tetap bisa dibuat	Fail
TC27	Membuat tambak/farm dan kolam	[Abnormal] Melakukan buat kolam setelah menentukan nominal jumlah kolam, langsung mengklik buat kolam tanpa isi detail	- Terdapat warning error penanda form masih ada yang kosong	- Pengguna melihat halaman tambak yang sudah ada detail kolamnya tetapi form yang kosong tetap terlihat	Fail
TC28	Memulai Siklus Budidaya	[Normal] Melakukan mulai siklus budidaya melalui tombol mulai siklus di atas halaman utama dengan pengisian informasi benar dan lengkap	- Di halaman utama terdapat pemberitahuan mengenai siklus yang dimulai	- Di halaman utama terdapat pemberitahuan mengenai siklus yang dimulai	Pass
TC29	Memulai Siklus Budidaya	[Normal] Melakukan mulai siklus budidaya melalui menu input pada halaman utama dengan pengisian informasi benar dan lengkap	- Di halaman utama terdapat pemberitahuan mengenai siklus yang dimulai	- Di halaman utama terdapat pemberitahuan mengenai siklus yang dimulai	Pass
TC30	Memulai Siklus Budidaya	[Abnormal] Melakukan mulai siklus budidaya dengan pengisian beberapa informasi yang tidak sesuai dengan ketentuan	- Terdapat warning error jika total harus berupa nominal	- Terdapat warning error jika total harus berupa nominal	Pass
TC31	Memulai Siklus Budidaya	[Abnormal] Melakukan mulai siklus budidaya dengan beberapa pengisian informasi dikosongkan	- Tombol mulai siklus tidak bisa ditekan - Terdapat warning error penanda form masih ada yang kosong	- Tombol mulai siklus tidak bisa ditekan	Pass
TC32	Memulai Siklus Budidaya	[Abnormal] Melakukan mulai siklus budidaya dengan langsung mengklik tombol mulai siklus tanpa pengisian informasi	- Tombol mulai siklus tidak bisa ditekan - Terdapat warning error penanda form masih ada yang kosong	- Tombol mulai siklus tidak bisa ditekan	Pass

3.2.4 Pemantauan

Pemantauan dalam proyek dilakukan melalui *weekly meeting* dan *daily meeting*. *Weekly meeting* dilakukan pada hari Senin pukul 10.00 pagi dan Jumat pukul 16.00 sore, daily meet dilakukan setiap hari kecuali Senin dan Jumat pada pukul 10.00 pagi. Pelaksanaan *weekly meet* diikuti oleh seluruh anggota tim pengembangan aplikasi mobile dan Web, tim data *science, chief technology officer*, dan *software manager* Jala. Sedangkan *daily meet* hanya diikuti oleh tim QA dan *supervisor*.

Weekly meet yang dilakukan pada hari Senin digunakan untuk membahas target progres pekerjaan proyek pada minggu tersebut, dan *weekly meet* pada hari Jumat digunakan untuk membahas progres pekerjaan proyek pada minggu tersebut. *Daily meet* digunakan sebagai diskusi singkat mengenai kendala dan progres dalam pelaksanaan proyek pengujian aplikasi Jala mobile. Pada *daily meet*, progres harian dibahas dengan melakukan *screen share* dan menampilkan dokumen excel yang terbaru dan menampilkan hasil pengujian yang sudah dilakukan dengan memutar video. Pemantauan progres proyek juga dilakukan melaui Dropbox Paper, sebuah dokumen ruang kerja *online* yang dapat digunakan untuk berkolaborasi dan sebagai *tracking* progres suatu proyek seperti yang diperlihatkan pada Gambar 3.17.



Gambar 3.17 Dropbox Paper

3.2.5 Penutupan

Pada akhir pelaksanaan proyek sekaligus akhir pelaksanaan magang, penulis melakukan presentasi yang dapat dihadiri oleh seluruh karyawan Jala Tech yang bersedia. Materi presentasi berisi penjelasan mengenai pekerjaan yang dilakukan selama pelaksanaan magang. Setelah melakukan presentasi, setiap orang diberi kesempatan bertanya mengenai pelaksanaan kegiatan magang, misalnya mengenai proyek yang dilakukan, atau pelajaran yang didapat selama magang. Setelah presentasi selesai dilaksanakan, maka kegiatan magang pun dinyatakan selesai dilakukan.

BAB IV
REFLEKSI MAGANG

Bab ini menjelaskan refleksi pelaksanaan magang yang terdiri dari relevansi akademik dan pembelajaran magang. Relevansi akademik berkaitan dengan kesenjangan antara teori pada Bab II dibandingkan dengan pelaksanaan di lapangan, sedangkan pembelajaran magang memaparkan manfaat, tantangan, dan hambatan yang diperoleh selama proses pengujian aplikasi mobile. Relevansi akademik menganalisis kesenjangan antara landasan teori yang telah dibahas di Bab II dengan pelaksanaan di lapangan selama magang. Beberapa hal yang akan dibahas, yaitu terkait pengujian aplikasi, pengujian otomatis, siklus STLC, *black box testing*, dan Appium.

4.1 Pengujian Aplikasi

Materi mengenai pengujian aplikasi diajarkan pada mata kuliah Pengujian Perangkat Lunak di semester enam perkuliahan Program Studi Informatika – Program Sarjana. Pengujian aplikasi dalam teori yang dipelajari pada perkuliahan membantu memberikan pemahaman mengenai tujuan pengujian aplikasi, jenis jenis pengujian, *tools* pengujian, dan cara melakukan pengujian aplikasi.

Pada praktiknya di lapangan, teori yang sudah didapat membantu sebagai dasar untuk mengembangkan kemampuan pada pekerjaan magang sebagai *quality assurance engineer*. Pengujian aplikasi yang dilakukan selama pelaksanaan magang memberi pemahaman baru mengenai tahap-tahap melakukan pengujian dalam sebuah proyek pengembangan aplikasi.

Mata kuliah PPL sudah mengajarkan teori dasar mengenai pengujian aplikasi. Namun, pada saat penulis menempuh mata kuliah PPL teori mengenai tahap dan *tools* yang digunakan dalam pengujian aplikasi dirasa masih kurang cukup jelas. Misalnya, ketika mempelajari *test case*, tidak diberikan contoh yang jelas bagaimana penulisan *test case* pada umumnya. Sehingga ketika menulis *test case*, masing-masing mahasiswa memiliki cara dan format *test case* tersendiri sesuai dengan contoh yang mereka dapat di internet. Contoh lainnya adalah ketika diajarkan mengenai pengujian otomatis, mata kuliah PPL tidak memberikan contoh *tools* yang dibutuhkan, maupun cara menjalankan *tools* tersebut. Sehingga setiap mahasiswa menggunakan *tools* pengujian masing-masing dan memiliki pengalaman belajar yang berbeda-beda walaupun mengikuti kelas yang sama.

Kurang detailnya pembelajaran ini sebenarnya bisa menjadi hal positif, jika melihat usaha setiap mahasiswa untuk belajar dan mengembangkan kemampuan mereka berdasarkan format *test case* dan *tools* pengujian mereka sendiri. Namun, akan lebih baik jika diberikan sedikit contoh dan format yang umum digunakan pada pengujian aplikasi di lapangan, lalu memberikan mahasiswa kesempatan untuk mencari *tools* lain yang ingin mereka gunakan. Pengetahuan mengenai *tools* yang umum digunakan tersebut tentunya akan membantu mahasiswa untuk mengembangkan kemampuan, dan dapat membantu jika suatu saat mahasiswa bekerja dengan tim yang menggunakan *tools* yang sudah diajarkan.

4.2 Pengujian Otomatis dengan *Tools*

Terjun langsung ke proyek dalam skala profesional tentunya memerlukan adaptasi, baik terkait pengetahuan maupun keterampilan teknis dan nonteknis. Karena Jala Tech pada saat itu belum memiliki divisi QA, maka pada awal kegiatan magang dilakukan pengenalan mengenai teknis pengujian aplikasi, tahap yang dilakukan, dan tools yang akan digunakan kepada tim *developer* dan *supervisor* di Jala Tech. Pengujian yang dilakukan selama pelaksanaan magang menggunakan pengujian manual dan otomatis pada platform Web dan mobile. Pengujian otomatis pada aplikasi Web menggunakan *tool* Selenium Webdriver, sedangkan pengujian otomatis pada aplikasi mobile menggunakan *tool* Appium. Proyek pengujian pada aplikasi Web lebih berfokus pada pengujian fungsionalitas fitur baru yang pada saat itu akan dirilis, yaitu fitur cek anco. Proyek pengujian pada aplikasi mobile lebih menjadi prioritas dan berfokus pada fungsionalitas seluruh fitur utama aplikasi Jala, sehingga topik ini dipilih menjadi laporan akhir.

Pengujian yang dilakukan secara otomatis lebih cepat dibanding dengan pengujian yang dilakukan secara manual. Hal ini disebabkan pengujian dapat dilakukan dengan menjalankan beberapa kasus sekaligus dengan hanya menambahkan beberapa baris kode. Jika dibandingkan dengan pengujian yang dilakukan secara manual, di mana penguji harus menjalankan setiap kasus secara manual dan berulang kali. Contohnya pada salah satu kasus pengujian yaitu kasus register, pengujian yang dilakukan secara otomatis dapat melakukan register dalam waktu kurang lebih 15 detik. Sedangkan pengujian yang dilakukan secara manual dapat memakan waktu lebih dari 40 detik karena harus mengisi setiap data pada *form* registrasi seperti nama, nomor hp, email, password, konfirmasi password, dan asal negara. Pengujian yang dilakukan secara otomatis juga mengurangi kesempatan terjadinya *human error* pada proses pengujian, sehingga dapat meningkatkan kualitas pengujian.

Penggunaan *tool* seperti Appium dapat membantu mempercepat proses pengujian dan membantu memudahkan dalam penulisan skrip pengujian otomatis dengan menampilkan *id* elemen yang akan dipilih. Namun pada awal pelaksanaan magang, sekitar lebih dari 80% elemen yang terdapat pada aplikasi Jala mobile tidak memiliki *id.* Elemen yang tidak memiliki *id* tetap dapat dijalankan pada pengujian otomatis menggunakan pengenal lain seperti XPath atau XML *Path Language.* Namun berdasarkan pengalaman selama pekerjaan proyek, penggunaan XPath tidak direkomendasikan untuk penulisan skrip pengujian otomatis yang akan digunakan pada jangka panjang. Sebab, XPath akan selalu berubah setiap tampilan *user interface* mendapat pembaruan. Jika XPath suatu elemen berubah, maka skrip pengujian yang ditulis berdasarkan versi aplikasi sebelumnya tidak lagi bisa digunakan. Untuk mengatasi ini, diberikan proyek baru untuk menambahkan *id* pada setiap elemen yang akan digunakan pada pengujian otomatis. Bekerja sama dengan tim mobile *developer*, aplikasi Jala sekarang memiliki pengenal *id* pada setiap elemennya.

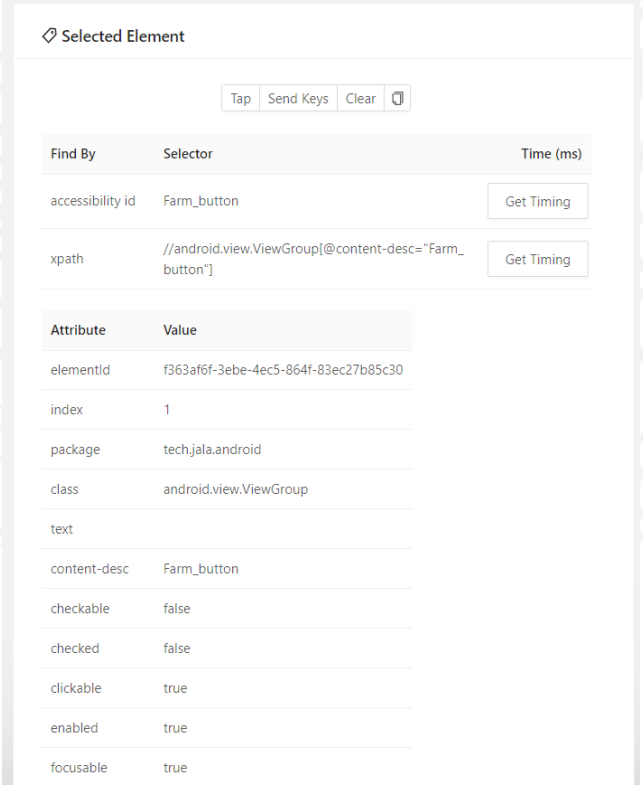
Sebenarnya ada banyak *automation testing tools* lain yang dapat digunakan untuk mengotomisasi pengujian mobile, misalnya Katalon, Kobiton, Espresso, Robotium, testRigor, dll. Appium dipilih menjadi *tool* pengujian yang digunakan pada proyek magang karena dirasa lebih familiar dibanding dengan *tool* pengujian otomatis lain. Penggunaan Appium juga tidak diajarkan secara spesifik dalam mata kuliah Pengujian Perangkat Lunak, namun dipelajari sendiri saat diberikan tugas melakukan pengujian otomatis pada aplikasi mobile dengan *tool*. Penggunaan Appium pada praktik di lapangan memberikan pengetahuan dan pengalaman baru mengenai pengujian otomatis pada platform mobile.

4.3 Penggunaan Appium

Pengujian otomatis perlu dilakukan menggunakan *automation testing tools* yang sesuai dengan pengujian yang akan dilakukan. *Tools* yang digunakan untuk melakukan pengujian pada platform Web biasanya tidak bisa digunakan untuk melakukan otomasi pada aplikasi mobile. *Tools* yang sesuai dapat membantu untuk mengotomasi pengujian yang dilakukan, mengurangi waktu yang dibutuhkan, dan meningkatkan efisiensi dengan mengurangi resiko terjadinya *human error* seperti ketika pengujian dilakukan secara manual oleh manusia. *Tools* pengujian yang digunakan selama kegiatan adalah Selenium dan Appium, dengan Selenium digunakan pada pengujian aplikasi Web dan Appium pada pengujian aplikasi mobile.

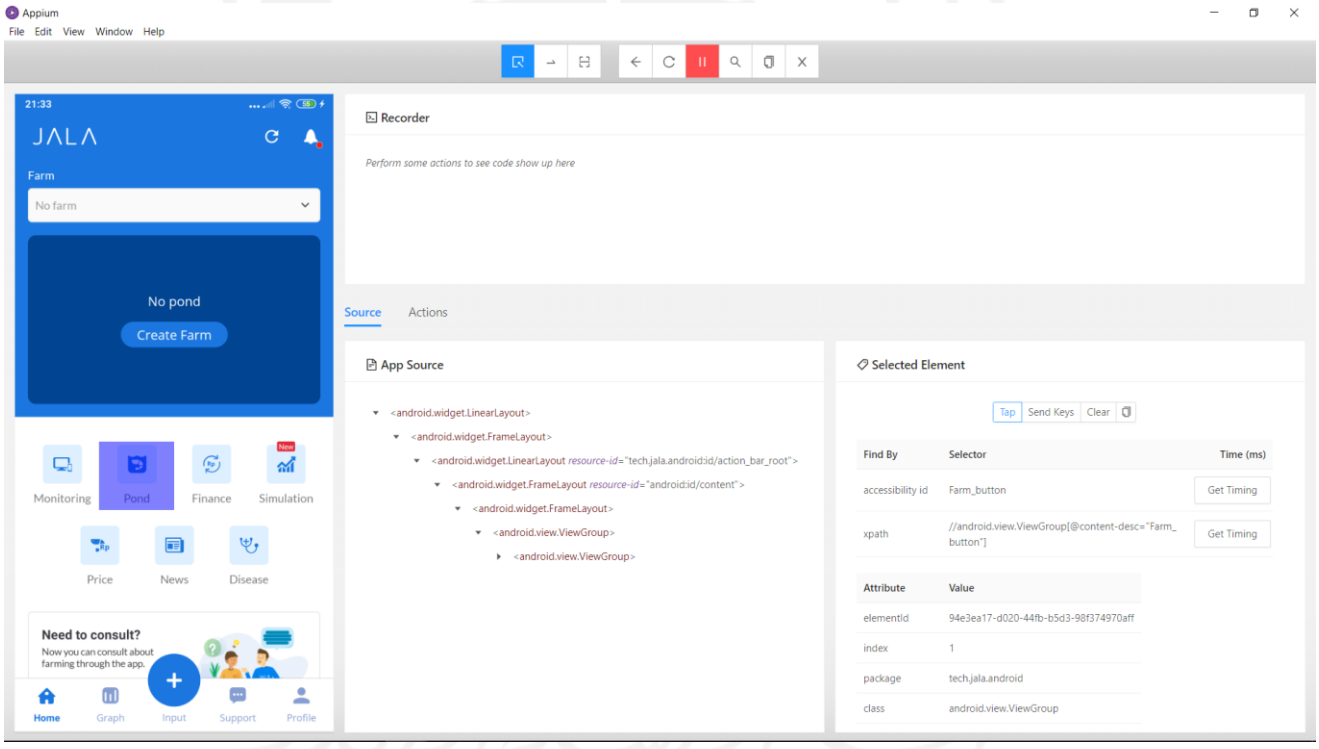
Appium dipilih menjadi *automation testing tools* pada proyek pengujian aplikasi Jala mobile karena bersifat *open-source* sehingga dapat digunakan secara bebas dan mudah untuk

digunakan. Penggunaan Appium dimulai dengan menentukan *desired capabilities* terlebih dahulu, seperti yang sudah ditampilkan Gambar 3.12 pada bab sebelumnya. Ketika pada tahap pengujian GUI atau *Graphical User Interface*, Appium *inspector* akan mengidentifikasi elemen yang dipilih dan menampilkan informasi seperti nama elemen, tipe, Xpath, dan kelas. Tampilan informasi elemen yang diberikan Appium *inspector* ditampilkan pada Gambar 4.1. Appium dapat digunakan untuk merekam tahap-tahap yang dijalankan pada Appium *inspector*. Rekaman ini berbentuk skrip dalam bahasa pemograman yang kita pilih, dan dapat dijalankan untuk menjalankan kembali langkah yang dilakukan pada Appium *inspector*. Informasi yang ditampilkan oleh Appium *inspector* digunakan sebagai panduan pada penulisan skrip untuk memberikan perintah kepada server Appium mengenai elemen yang harus dipilih atau diklik. Contohnya ketika menulis perintah untuk mengklik menu "Pond" pada layar utama aplikasi Jala seperti yang diperlihatkan pada gambar Gambar 4.2 Appium *inspector* Untuk memberi perintah klik, maka dilakukan dengan *command* `Elements.click()`. Pada kasus ini, yang ditulis di dalam skrip adalah `driver.find_element_by_accessibility_id("Farm_button").Click()`.



Gambar 4.1 Informasi elemen pada Appium *inspector*

Penggunaan Appium dalam proyek pengujian aplikasi Jala mobile membantu menghasilkan pengujian otomatis dan berhasil menemukan beberapa *bugs* dan *error* pada aplikasi Jala. Berdasarkan teori yang dikaji pada bab dua, disebutkan salah satu kelebihan Appium dibanding *tools* pengujian lainnya adalah dengan mendukung banyak bahasa pemrograman. Kelebihan ini sangat membantu pada proses pengerjaan proyek, dengan aplikasi Jala mobile yang ditulis menggunakan Javascript, Appium tetap mampu bekerja meskipun skrip pengujian ditulis dalam bahasa Python. Tetapi Appium juga bukan merupakan tools pengujian yang sempurna, kekurangan yang paling dirasakan selama pelaksanaan pengujian adalah ketidakmampuan Appium untuk menguji fitur yang beralih dari aplikasi *native* ke aplikasi Web. Misalnya pada fitur menu *news* pada aplikasi Jala mobile, ketika diklik aplikasi membuka halaman berita pada situs Web Jala. Pengujian ini tidak dapat dilanjutkan karena Appium *inspector* tidak dapat mendeteksi elemen yang berada pada halaman Web.



Gambar 4.2 Appium *inspector*

4.4 Penerapan STLC

Pengerjaan proyek ini memanfaatkan metode STLC, yang merupakan enam urutan langkah dalam proses pengujian aplikasi. Implementasi metode STLC pada proyek merupakan pengalaman pertama penulis menggunakan metode ini. Metode STLC membawa banyak kemudahan dalam sebuah proyek pengujian aplikasi. Dengan memanfaatkan metode STLC

proses pengujian dirasa menjadi lebih terstruktur dibandingkan sebelum metode diterapkan. Dengan konsep kriteria *entry* dan *exit* yang jelas, progres proyek pengujian dapat dipantau dengan lebih mudah. Metode ini juga memudahkan penguji dalam melakukan pengujian, dengan langkah-langkah yang sudah ditetapkan, penguji bisa fokus pada satu tahap terlebih dahulu lalu ketika sudah selesai dilanjutkan ke tahap berikutnya.

4.5 Implementasi Teknik *Black Box*

Berdasarkan teori yang dikaji, penerapan teknik *black box* disebutkan hanya berfokus pada pengujian fungsionalitas dan tidak memerlukan pengetahuan mengenai struktur internal pada aplikasi yang diuji. Pada praktiknya di lapangan, teknik *black box* berhasil diterapkan tanpa pengetahuan sama sekali mengenai struktur internal aplikasi Jala mobile. Penerapan teknik ini berhasil menemukan beberapa *bugs* dan cacat pada aplikasi Jala mobile. Namun ketika pengujian beralih dari manual ke otomatis, ada salah satu kriteria teknik *black box* berdasarkan teori yang menjadi kurang relevan, yaitu tidak diperlukannya kemampuan *coding*. Penulisan skrip yang dibutuhkan untuk menjalankan pengujian otomatis memerlukan pemahaman dasar mengenai kode dan bahasa pemrograman.

Selain itu pada pelaksnaan magang, pengujian otomatis yang dijalankan menggunakan skrip juga membutuhkan informasi mengenai *id* dari elemen yang akan dijalankan yang hanya dapat dilihat dengan melihat kedalam stuktur kode tampilan aplikasi. Tapi ketidakcocokan dengan teori yang dikaji ini dapat diatasi jika penguji memiliki daftar *id* setiap elemen yang akan dipakai pada penulisan skrip.

BAB V
PENUTUP

5.1 Kesimpulan

Dari pekerjaan yang telah dilakukan yaitu Pengujian Otomatis dengan Teknik *Black Box* Menggunakan Appium, telah berhasil dicapai tujuannya, dengan indikator keberhasilan yaitu setiap kasus pengujian pada *test* case berhasil dijalankan. Sesuai pembahasan dalam refleksi, pengujian otomatis membantu mempercepat proses pengujian aplikasi dan meningkatkan efisiensi pengujian. 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 *bugs* dan cacat pada aplikasi Jala mobile. Sedangkan Appium berhasil membantu proses otomatisasi sehingga pengujian otomatis dapat dilakukan dengan lebih mudah jika melakukan pengujian berulang, dibandingkan dengan melakukan pengujian secara manual setiap kali aplikasi diuji.

5.2 Saran

Saran yang dapat diberikan setelah melakukan proyek magang adalah perlunya dikembangkan *tools* pengujian yang dapat melakukan *automated test* pada *browser* yang dibuka di dalam aplikasi. Saran berikutnya adalah perlu dilakukan kajian untuk mengukur efisiensi penggunaan Appium, terlebih jika dibandingkan dengan *tools* lain sehingga dapat dipilih *tools* pengujian yang tepat untuk kasus yang tepat.

DAFTAR PUSTAKA

Hamilton, T. (2022). What is Software Testing? Definition. *www.guru99.com/software-testing-introduction-importance.html*.

Hamilton. Thomas. (2022). STLC (Software Testing Life Cycle) Phases, Entry, Exit Criteria. *https://www.guru99.com/software-testing-life-cycle.html*.

Hans M. (2015). *Appium Essenstials*. Packt Publishing Ltd.

Mohammad, S. M. (2015). *Automation Testing in Information Technology*. Retrieved from http://www.ijcrt.org

Nidhra, S. (2012). Black Box and White Box Testing Techniques - A Literature Review. *International Journal of Embedded Systems and Applications*, *2*(2), 29–50. doi:10.5121/ijesa.2012.2204

Ryanditha, P. (2018). Perbedaan Manual Testing dan Automated Testing. *https://medium.com/skyshidigital/perbedaan-manual-testing-dan-automated-testing-9d13373a36e*.

Sharma, M., & Angmo, R. (2014). *Web based Automation Testing and Tools*. Retrieved from http://www.ijcsit.com

Singh Ȧ, S., Gadgil Ȧ, R., & Chudgor Ȧ Ȧ, A. (2014). *Automated Testing of Mobile Applications using Scripting Technique: A Study on Appium*. Retrieved from http://inpressco.com/category/ijcet

Verma, A., Khatana, A., & Chaudhary, S. (2017). *A Comparative Study of Black Box Testing and White Box Testing*. Retrieved from http://www.ijcseonline.org

FrianH. (2019). Appium: How to switch in Mobile App from native App to Mobile Browser and run URL on Browser in mobile. *https://stackoverflow.com/questions/57521505/appium-how-to-switch-in-mobile-app-from-native-app-to-mobile-web-browser-and-ru*

LAMPIRAN