SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI

IMPLEMENTASI *AUTOMATION TESTING* MENGGUNAKAN *TOOLS APPIUM* PADA APLIKASI *MOBILE ANDROID*: STUDI KASUS PADA APLIKASI *BALINK*

TUGAS AKHIR

Eka Amelia Putri
0110220147

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI
DEPOK
AGUSTUS 2024

SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI

IMPLEMENTASI *AUTOMATION TESTING* MENGGUNAKAN *TOOLS APPIUM* PADA APLIKASI *MOBILE ANDROID*: STUDI KASUS PADA APLIKASI *BALINK*

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana komputer

Eka Amelia Putri
0110220147

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI
DEPOK
AGUSTUS 2024

HALAMAN PERNYATAAN ORISINALITAS

Skripsi/Tugas Akhir ini adalah hasil karya penulis, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama : **Eka Amelia Putri**

NIM : **0110220147**

Depok, 25 Juni 2024

Eka Amelia Putri

HALAMAN PENGESAHAN

Skripsi/Tugas Akhir ini diajukan oleh :

Nama : Eka Amelia Putri

NIM : 0110220147

Program Studi : Teknik Informatika

Judul Skripsi : Implementasi *Automation Testing* Menggunakan *Tools Appium* Pada Aplikasi *Mobile Android*: Studi Kasus Pada Aplikasi *Balink*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer pada Program Studi Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri

DEWAN PENGUJI

Pembimbing	Penguji
Salman El Farisi, S.Kom, M.Kom	Zaki Imaduddin, S.T, M.Kom

Ditetapkan di : Depok

Tanggal : 24 Juli 2024

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan skripsi/Tugas Akhir ini. Penulisan skripsi/Tugas Akhir ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana komputer Program Studi Teknik Informatika pada Sekolah Tinggi Teknologi Terpadu Nurul Fikri Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi penulis untuk menyelesaikan skripsi/tugas akhir ini. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Allah SWT.
2. Orang tua dan semua anggota keluarga yang telah memberikan dorongan baik secara moril maupun materil dalam penyelesaian tugas ini.
3. Bapak Dr. Lukman Rosyidi, ST., MM., MT selaku Ketua Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
4. Ibu Tiffanny Nabarian, S.Kom M.T.I selaku Ketua Program Studi Teknik Informatika Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
5. Bapak Dr. Lukman Rosyidi, ST., MM., MT selaku Dosen Pembimbing Akademik yang telah membimbing penulis selama berkuliah di Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
6. Bapak Salman El Farisi, S.Kom, M.Kom selaku Dosen Pembimbing Tugas Akhir penulis dalam menyelesaikan penulisan ilmiah ini.
7. Para Dosen di lingkungan Sekolah Tinggi Teknologi Terpadu Nurul Fikri yang telah membimbing penulis dalam menuntut ilmu yang telah diberikan.
8. Manajer Alterra Academy dan staf yang telah menyisihkan waktu untuk menyediakan data yang diperlukan dalam penulisan tugas akhir ini.
9. Ibu, Ayah dan seluruh keluarga besar yang senantiasa terus memberikan support dan semangatnya pada proses penulisan Tugas Akhir ini sehingga bisa selesai tepat pada waktunya.
10. Kak Taufik dan Kak Afianur Hamzah yang selalu memberikan motivasi dan dukungannya dari segi materi maupun non materi sehingga saya dapat melanjukan dan menyelesaikan perkuliahan ini dengan tepat waktu.

11. Kakak-kakak Tim PKBM Nara Kreatif yang telah memberikan dukungan serta supportnya.
12. Member grup “UI UX” yang selalu mereminder terkait penulisan tugas akhir ini dan selalu memotivasi satu sama lain supaya bisa lulus bersama dan tepat waktu.
13. Seseorang yang tidak bisa disebutkan namanya yang tak kalah penting kehadirannya, terima kasih telah menjadi bagian dari proses penulisan tugas akhir ini. Berkontribusi banyak dalam penulisan tugas akhir ini, memberikan dukungan, semangat, pikiran serta tenaganya sehingga penulisannya ini dapat terselesaikan.

Dalam penulisan ilmiah ini tentu saja masih banyak terdapat kekurangan-kekurangan yang mungkin disebabkan oleh keterbatasan kemampuan dan pengetahuan yang penulis miliki. Walaupun demikian, penulis telah berusaha menyelesaikan penulisan ilmiah ini sebaik mungkin. Oleh karena itu apabila terdapat kekurangan di dalam penulisan ilmiah ini, dengan rendah hati penulis menerima kritik dan saran dari pembaca.

Akhir kata, penulis berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 24 Juli 2024

Eka Amelia Putri

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Sekolah Tinggi Teknologi Terpadu Nurul Fikri, saya yang bertanda tangan di bawah ini:

Nama : Eka Amelia Putri

NIM : 0110220147

Program Studi : Teknik Informatika

Jenis karya : Skripsi / Tugas Akhir

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada STT-NF **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty - Free Right*)** atas karya ilmiah saya yang berjudul :

Implementasi *Automation Testing* Menggunakan *Tools Appium* Pada Aplikasi *Mobile Android*: Studi Kasus Pada Aplikasi *Balink*

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini STT-NF berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 24 Juli 2024

Yang Menyatakan

Eka Amelia Putri

ABSTRAK

Nama : Eka Amelia Putri
NIM : 0110220147
Program Studi : Teknik Informatika
Judul : Implementasi *Automation Testing* Menggunakan *Tools Appium* Pada Aplikasi *Mobile Android*: Studi Kasus Pada Aplikasi *Balink*

Dalam era digital yang berkembang saat ini, aplikasi mobile menjadi bagian integral dari kehidupan sehari-hari. Diantaranya adalah aplikasi *Balink*, yang ditujukan untuk membantu wisatawan turis untuk lebih mengenak tentang Bali. Namun, dalam perkembangan aplikasi *mobile Balink* ini diperlukan pengujian untuk memastikan aplikasi benar-benar siap digunakan oleh pengguna. Untuk mengatasi ini pengujian otomatis menjadi penting. Pada pengujian ini memilih *Appium* sebagai aplikasi pengujian otomatis. *Appium* merupakan *tools automation open source* yang mendukung pengujian diberbagai *platform* dengan kelebihan bisa dipakai diberbagai aplikasi yang ingin dilakukan *testing*. Rumusan masalah mencakup pembuatan *test case*, eksekusi program, dan tingkat keberhasilan pembuatan *test script* pengujian otomatis. Tujuan dari penelitian ini adalah memahami proses dan teknik dalam pembuatan kode program otomatis serta mengevalusai tingkat keberhasilan pembuatan *test case* ke dalam kode program pengujian otomatis. Penelitian ini menggunakan metode kuantitatif dengan metode pengumpulan data yaitu studi literatur dan eksperimen. Hasil dari penelitian ini adalah pembuatan kode program otomatis terhadap implementasi pengujian aplikasi mobile Android mencapai tingkat keberhasilan 100%. Ini menandakan bahwa pembuatan *test script* berjalan sesuai harapan tanpa adanya masalah.

Kata kunci : Aplikasi *Mobile*, *Appium, Balink*, Pengujian Otomatis, Kuantitatif

ABSTRACT

Name : Eka Amelia Putri
NIM : 0110220147
Study Program : *Informatic Engineering*
Title : *Implementation Of Automation Testing Using Appium Tools On Android Mobile Applications: Case Study On The Balink Application*

In today's growing digital era, mobile applications are becoming an integral part of everyday life. Among them is the Balink application, which is intended to help tourist travelers to know more about Bali. However, in the development of this Balink mobile application, testing is required to ensure the application is truly ready to be used by users. To overcome this automated testing becomes important. In this test, Appium was chosen as the automated testing application. Appium is an open source automation tool that supports testing on various platforms with the advantage that it can be used in various applications that you want to test. The formulation of the problem includes the creation of test cases, program execution, and the success rate of creating automated test scripts. The purpose of this research is to understand the process and techniques in making automatic program code and evaluate the success rate of making test cases into automatic testing program code. This research uses quantitative methods with data collection methods, namely literature studies and experiments. The result of this research is the creation of automatic program code for the implementation of Android mobile application testing achieved a 100% success rate. This indicates that the creation of the test script runs as expected without any problems.

Key words : Appium, Automation Testing, Balink, Mobile Application, Quantitative

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Bab ini adalah bab pembuka mengenai gambaran umum tentang Tugas Akhir yang akan dibuat. Pada bab ini terdiri beberapa sub bab diantaranya yaitu latar belakang, rumusan masalah, tujuan penelitian dan manfaat penelitian, serta batasan masalah dan sistematika penulisan.

1.1 Latar belakang

Dalam era digital yang berkembang pesat saat ini, aplikasi mobile telah menjadi bagian integral dari kehidupan sehari-hari. Aplikasi mobile memungkinkan pengguna untuk melakukan berbagai tugas, seperti berkomunikasi, berbelanja, mengakses informasi dan banyak lagi. Bermacam-macam bentuk pengujian perlu dilakukan, baik fungsional maupun non fungsional. Untuk itu, proses *software testing* harus dilakukan dengan tujuan untuk mengidentifikasi *bug* atau *erorr* yang mungkin terjadi karena kesalahan dalam proses pemgembangan sebuah *software*[1]

Dalam pengembangan aplikasi mobile, *software testing* adalah metode yang digunakan oleh sebuah perusahaan untuk menentukan apakah suatu aplikasi sudah memenuhi seluruh *requirment* atau belum sebelum benar-benar digunakan oleh pengguna[2]. *Software testing* saat ini sudah banyak digunakan dalam berbagai sektor industri di bidang *Quality Engineer* maupun *Quality Assurance*. Artinya, sangat penting untuk mendeteksi *bug* sebelum berdampak lebih luas akibat *software testing*. Selain itu, pengujian juga dapat membantu *developer* untuk menentukan kualitas dan kinerja suatu aplikasi sebelum dipublikasikan.

Khususnya, pengujian otomatis pada aplikasi mobile Android menjadi semakin penting mengingat pangsa pasar yang luas dan keragaman perangkat yang mendukung sistem operasi tersebut. Namun, tantangan utama dalam pengujian otomatis aplikasi mobile Android adalah kebutuhan untuk mempertimbangkan berbagai faktor seperti perbedaan resolusi layar, versi sistem operasi, dan perangkat keras yang berbeda.

Appium adalah salah satu *tools* yang digunakan untuk melakukan pengujian otomatis pada aplikasi *mobile*. *Appium* adalah *tools open-source* yang dapat

digunakan oleh siapa saja dan mudah digunakan dibandingkan dengan *automation tools* lainnya yang dapat mendukung pengujian aplikasi mobile di berbagai *platform* termasuk *Android* dan iOS[3]. *Appium* banyak mendukung bahasa pemrograman yang luas, serta integrasi yang baik dengan berbagai *framework* pengujian.

Balink merupakan salah satu aplikasi mobile yang ditujukan untuk membantu wisatawan turis untuk lebih mengenal tentang Bali, dengan memberikan layanan penting kepada pengguna dalam berbagai aspek. Khususnya bagi wisatawan yang ingin belajar tentang *culture* yang ada di Bali dengan mengikuti *event-event* yang ada, serta membeli aneka produk Bali. Studi kasus pada implementasi *automation testing* menggunakan *Appium* pada aplikasi *Balink* akan memberikan wawasan konkret tentang bagaimana alat ini dapat diterapkan secara efektif dalam konteks pengujian aplikasi mobile yang kompleks.

Dengan pengujian ini, diharapkan dapat memberikan pengetahuan tentang implementasi *tools Appium* untuk pengujian otomatis dalam uji fungsionalitas pada pembuatan aplikasi *Balink*. Pengujian otomatis ini juga diharapkan dapat mengurangi kemungkinan terjadinya *human error* pada proses pengujian serta dapat meningkatkan efisiensi proses pengembangan. Hasil dari pengujian ini bisa menjadi sebuah solusi bagi tim pengembang untuk menjaga kualitas aplikasi yang bebas dari cacat dan *bugs*.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, penulis mencoba merumuskan permasalah yang akan dipecahkan dengan penelitian yang akan penulis lakukan, yaitu:

1. Bagaimana cara membuat *test script* pada pengujian aplikasi *Balink*?
2. Bagaimana proses *execution* pengujian pada pengujian aplikasi *Balink*?
3. Sejauh mana tingkat keberhasilan implementasi automation testing terhadap semua *test case* yang sudah dibuat?

1.3 Tujuan

Tujuan yang ingin diperoleh pada penelitian ini adalah:

1. Mengidentifikasi *bug*, kesalahan fungsional, dan ketidaksesuaian aplikasi Balink dengan spesifikasi yang sudah di tentukan.
2. Membuat *test scrip*t pengujian mobile android menggunakan aplikasi appium.
3. Menganalisis metode dan teknik untuk membuat *test script* pada aplikasi Appium.
4. Memahami proses eksekusi pengujian yang dilakukan pada aplikasi Balink menggunakan Appium.
5. Dapat mengevaluasi persentase tingkat keberhasilan *test case* yang dibuat kedalam *automation script*.

1.4 Manfaat Penelitian

Adapun manfaat yang ingin diperoleh dalam penelitian ini antara lain sebagai berikut:

1. Dapat Mendeteksi masalah dengan cepat disetiap iterasi pengembangan aplikasi Balink.
2. Menghemat waktu dan sumber daya dengan menjalankan skenario pengujian secara *automation testing.*

1.5 Batasan Masalah

1. Pengujian yang dilakukan yaitu pengujian *automation* pada mobile android.
2. *Tools* yang digunakan untuk pengujian yaitu *Appium*.
3. Pengujian ini menggunakan teknik *black box testing.*
4. Melakukan pengujian terhadap fitur kategori pencarian pakaian.
5. Melakukan pengujian terhadap fitur kategori pencarian kerajinan tangan.
6. Melakukan pengujian terhadap fitur kategori pencarian tas.
7. Melakukan pengujian terhadap fitur kategori pencarian masker khas bali.
8. Melakukan pengujian terhadap fitur transaksi untuk melihat *history* transaksi yang telah dilakukan.
9. Melakukan pengujian terhadap fitur transaksi untuk melihat detail transaksi *event*.
10. Melakukan pengujian terhadap fitur artikel untuk melihat daftar artikel yang tersedia.

11. Melakukan pengujian terhadap fitur artikel untuk melihat salah satu detail artikel yang ada.
12. Melakukan pengujian terhadap fitur *event* untuk melihat *event* dan membeli tiket yang tersedia.
13. Melakukan pengujian terhadap fitur *event* by ID.

1.6 Sistematika Penulisan

Pada penulisan tugas akhir ini dibagi menjadi beberapa bab sebagai gambaran yang lebih jelas dan sistematis, dengan urutan sebagai berikut:

BAB 1 PENDAHULUAN, bab ini merupakan bab pembuka mengenai gambaran umum tentang Tugas Akhir yang akan dibuat. Pada bab ini terdiri beberapa sub bab diantaranya yaitu latar belakang, rumusan masalah, tujuan penelitian dan manfaat penelitian, serta batasan masalah dan sistematika penulisan.

BAB II KAJIAN LITERATUR, bab ini menjelasakan mengenai definisi dasar-dasar konsep yang berkaitan pada dengan *Sofware testing serta* data atau teori-teori yang digunakan pada saat proses pengujian aplikasi.

BAB III METODOLOGI PENELITIAN, pada bab ini membahas mengenai tahapan penelitian, rancangan penelitian yang berisi beberapa metode penyelesaian masalah yang ada seperti jenis penelitian yang digunakan, metode pengumpulan data yang akan digunakan, serta implementasi evaluasi dan penarikan kesimpulan terhadap penelitian yang sudah dilakukan.

BAB IV IMPLEMENTASI DAN PENGUJIAN, pada bab ini menjelaskan tentang bagaimana implementasi *automation testing* pada aplikasi Balink menggunakan aplikasi *Appium* dan mengevaluasi persentase keberhasilan pembuatan script *automation testing.*

BAB V KESIMPULAN DAN SARAN, pada bab ini berisi penarikan kesimpulan dari Tugas Akhir yang sudah dibuat yaitu inti dari jawaban pada rumusan masalah dan saran untuk peneliti selanjutnya yang akan meneliti terkait topik implemntasi *automation testing* pada aplikasi mobile.

BAB II
KAJIAN LITERATUR

Bab ini menjelasakan mengenai definisi dasar-dasar konsep yang berkaitan dengan *Sofware testing* serta data atau teori-teori yang digunakan pada saat proses pengujian aplikasi.

2.1 *Software Development*

SDLC (*Software Development Life Cycle*) merupakan metodologi yang sistematis untuk mengelola proyek pengembangan sistem infomasi, yang mencakup serangkaian langkah yang harus diikuti dari konsep awal hingga penyelesaian proyek. Inti dari SDLC ini adalah menciptakan proses yang menghasilkan produk berkualitas tinngi dan efektif secara biaya, yang tidak hanya memenuhi tetapi juga berpotensi melbihi ekspektasi pengguna, sembari tetap berada dalam batasan anggaran dan waktu yang telas ditetapkan.

SDLC (*Software Development Life Cycle*) ini adalah rangkaian langkah yang terintegrasi dalam pembuatan perangkat lunak. Dalam setiap langkah ada keterkaitan yang erat, dimai dari pemasukan data awal, diikuti oleh proses pengolahan data tersebut sehingga menghasilkan hasil akhir sesuai dengan yang diharapkan. Berikut tahapan-tahapan dalam SDLC yang penting untuk dipahami sebagai bagian dari proses pengembangan perangkat lunak.

2.1.1 Planning

Perencanaan atau *planning* merupakan tahap awal dimana tim mengidentifikasi dan menetapkan cakupan atau domain pengaturan yang harus diperhatikan dalam proses pengmabngan proyek. Pada tahap ini, informasi yang diperlukan untuk pengembangan perangkat lunak dikumpulkan dari berbagai pihak terkait.

2.1.2 Analysis

Pada tahap ini, dilakukan analisis kebutuhan sistem secara fungsional. Ini mencakup pemahaman tentang masalah bisnis, tujuan proyek, dan fungsi utama yang akan dijalankan oleh perangkat lunak yang akan dikembangkan dan sebagainya.

2.1.3 Design

Pada tahap ini, ada beberapa aspek desain yang ditentukan yang akan digunakan yaitu:

- *Archiecture*: Bahasa pemrograman yang akan digunakan dan desain keseluruhan perangkat lunak.
- *User Interface*: Menentukan bagaimana ketika pengguna berinteraksi dengan perangkat lunak dan bagaimana perangkat lunak tersebut meresponnya.
- *Platform*: *platform* tempat perangkat lunak akan berjalan seperti Android, iOs, Linux dan lain-lain.
- *Security*: langkah-langkah untuk mengamankan sistem perangkat lunak seperti enkripsi lalu lintas SSL, perlindungan kata sandi, atau yang lain.

2.1.4 Develompent

Tahapan *development* atau pengembangan adalah fase terpanjang dalam pengembangan perangkat lunak. Pada tahap ini tim mulai menyusun sistem dengan menuliskan *script* menggunakan bahasa pemrograman yang dipilih.

2.1.5 Testing

Pada tahap ini, tim melibatkan *Software Quality Assurance* (QA) untuk menguji sistem yang telah dibangun. Pengujian dilakukan untuk memastikan bahwa perangkat lunak berfungsi sesuai dengan yang diharapkan. Jika ditemukan masalah, tim QA akan memberikan informasi kepada tim pengembang untuk dilakukan perbaikan.

2.1.6 Implementation dan Release

Setelah pengujian selesai dan tidak ada *bug* pada sistem, tahap implementasi dimulai. Tujuannya adalah untuk mengimplementasikan perangkat lunak di lingkungan produksi sehingga pengguna dapat mulai menggunakannya.

2.1.7 Maintenance

Pada tahap ini, tim akan melakukan pemeliharaan sistem dan rutin melakukan pembaruan agar perangkat lunak tetap optimal. Dalam hal ini meliputi perbaikan *bug*, peningkatan fitur dan lainnya.

2.2 *Software Testing*

Software testing adalah metode yang digunakan untuk memastikan bahwa produk aplikasi sesuai dengan kualitas yang diinginkan dan bebas dari cacat[4]. Dalam hal ini melibatkan pengujian manual dan otomatis. Pengujian manual dilakukan dengan mandiri/manual tanpa alat khusu untuk memeriksa apakah fitur aplikasi berfungsi dengan baik. Pengujian otomatis menggunakan *script* dan *tools* lain untuk menemukan cacat yang mungkin tidak terlihat. Terdapat dua teknik umum dalam pengujian aplikasi yaitu *white box testing* (fokus pada struktur aplikasi) dan *black box testing* (fokus pada aspek fungsional aplikasi)[5].

2.3 *Automation Testing*

Pengujian otomatis atau *Automation testing* adalah teknik pengujian perangkat lunak yang menggunakan *tools* khusus untuk menjalankan rangkaian kasus pengujian. Pengujian ini melibatkan pembuatan program (*script* pengujian) yang mengemulasi test case manual dalam bahasa pemrograman, baik dengan atau tanpa bantuan *tools* otomatis eksternal. Pengujian otomatis merupakan cara yang efektif untuk meningkatkan cakupan pengujian dan kecepatan eksekusi dalam pengujian aplikasi[6].

2.4 *Black Box Testing*

Pengujian *black box*, juga dikenal sebagai *behavioral testing*, dilakukan untuk mengamati hasil *input* dan *output* dari perangkat lunak tanpa memperhatikan struktur kode dibaliknya[7]. Pengujian ini biasanya dilakukan di akhir pembuatan perangkat lunak untuk memastikan bahwa perangkat lunak berfungsi dengan baik. Beberapa keuntungan dari pengujian *behavioral* (*black box*) meliputi:

1) Penguji tidak harus memiliki pengetahuan tentang suatu bahasa pemrograman.
2) Pengujian dilakukan berdasarkan sudut pandang pengguna untuk menemukan inkonsistensi dalam perangkat lunak.
3) Hubungan antara pengembang dan penguji terjalin dengan baik.
4) Penguji tidak perlu mengecek kode secara detail.
5) Penguji dan pengembang dapat bekerja secara mandiri tanpa harus melibatkan satu sama lain.

2.5 Appium

Appium adalah sebuah *server* HTTP yang ditulis menggunakan Node.js mirip dengan Selenium Webdriver. Cara kerja appium yaitu appium menerima permintaan dari *library klien* melalui JSON dan memperosesnya sesuai dengan *platform* yang digunakan. Appium adalah *tools open source* yang memungkinkan pengujian *script* untuk tiga jenis aplikasi mobile yaitu aplikasi *native*, aplikasi *web seluler* dan aplikasi *hybrid* pada perangkat Andorid dan IOS[3]. Dengan menggunakan protokol WebDriver Appium memungkinkan pengujian elemen-elemen dalam aplikasi mobile dan mengidentifikasi elemen yang perlu diotomasi untuk menciptakan sistem otomatis yang efisien.

2.6 Software Intelij IDEA

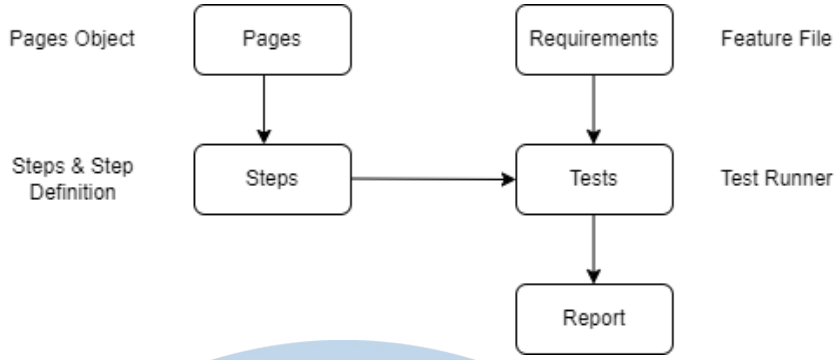
Intellij IDEA merupakan singkatan dari *integrated development environment* pertama kali di publikasi pada januari 2001 dan diusulkan sebagai aplikasi pengembang yang menggunakan bahasa pemrograman java pertama dengan pernavigasian dan perekstrusi kode program tingkat tinggi[8].

Intellij IDEA ini dikembangkan oleh *JetBrain*s yang dapat digunakan oleh pemgembang untuk mengembangkan programnya. *Intellij IDEA* ini juga merupakan salah satu tools pendukung pada pengujian aplikasi. Antar muka *intellij IDEA* ini dapat terhubung ke berbagai *platform*.

2.7 Serenity BDD

Serenity BDD (Behavior Driven Develpment) adalah *library open-source* yang digunakan untuk membantu *programer* dalam menulis kode program agar lebih bersih dan mudah di*maintenace*[9]. *Serenity* ini juga digunakan untuk menampilkan hasil *test* pengujian dalam bentuk diagram dan narasi laporan. *Serenity* ini dibentuk dengan menggabungkan *Automated Build Sever*.

Framework serenity ini biasanya digunakan pada aplikasi *web*, android maupun Api. Serenity ini juga dapat di integrsikan dengan berbagai *server automation testing* lainnya seperti *jenkins* yang dapat membantu dalam menjalankan pengujian aplikasi di latar belakan prosesnya.



Gambar 2.4 Arsitektur Dasar Serenity BDD

Serenity memiliki arsitektur seperti diatas, dengan dibuatnya arsitektur seperti atas membuat pengembang lebih mudah memahami dengan rancangan tersebut sehingga mudah untuk dieksekusi oleh tim penguji aplikasi[10].

2.8 Penelitian Terkait

Penelitian terkait mendetailkan penelitian-penelitian sebelumnya yang memiliki landasan atau konteks yang sama dengan penelitian yang sedang dikerjakan oleh penulis.

Tabel 2.1 Tahapan Penelitian

No	Nama dan Tahun	Judul	Topik	Subjek	Hasil
1	Rambe A, Prihantoro H, 2022	Automated Mobile Application Testing with Black-box Technique Using Appium	*Automation Testing*	Aplikasi	Aplikasi Jala
2.	Maximillian L, Kristianto R, Cendika D, 2024	Pengujian Aplikasi Mobile Web Wisata Bandung Menggunakan Blackbox Testing dan Katalon Berbasis Manual dan Automation Testing	*Software Testing*	Aplikasi	Aplikasi Mobile Web Wisata Bandung

3.	Uminingsih, Nur Ichsanudin M, Yusuf M, Suyara, 2022	Pengujian Fungsional Perangkat Lunak Sistem Informasi Perpustakaan Dengan Metode Black Box Testing Bagi Pemula	*Software Testing*	Aplikasi	Aplikasi Sistem Informasi Perpustakaan

1) Studi yang dilakukan oleh Andi Rivaldo Rambe dan Hanson Prihantoro (2022) berjudul "Automated Mobile Application Testing with Black-box Technique Using Appium". Mengevaluasi aplikasi jala dengan menggunakan Appium sebagai alat pengujian, yang dimana pada proses penelitian ini bertujuan untuk mengidentifikasi atau mendeteksi masalah cacat/*bug* pada aplikasi pada proses pengembangan aplikasi Jala. Hasil dari penelitian ini menunjukan bahwa penggunaan *black box testing* melalui Appium memungkinkan identifikasi *bug*, meningkatkan effisiensi dan memepercepat proses pengujian[4]. Perbedaan penelitian ini dengan penelitian penulis yaitu pada hasil akhir yang didapatkan, hasil akhir dari penelitian penulis yaitu tingkat keberhasilan dari implementasi *automation testing* menggunakan *tools Appium.*
2) Penelitian dilakukan oleh Louis Maximillian dan Ryan Putranda Kristianto (2024) yang berjudul "Pengujian Aplikasi *Mobile* Web Wisata Bandung Menggunakan *Blackbox Testing* dan Katalon Berbasis Manual dan *Automation Testing*". Pada penelitian ini penguji menggunakan teknik black*box* dan katalon sebagai *tools* yang digunakan untuk menguji Aplikasi *Mobile* Web Wisata Bandung. Penelitian ini berfokus pada pengujian manual dan automation pada Aplikasi *Mobile* Web Wisata Bandung untuk memastikan performa, keahlian dan kualitas aplikasi yang baik sebelum aplikasi diluncurkan ke publik. Penelitian ini memiliki hasil pengujian dengan mengevaluasi dua metode yang digunakan antara manual dan *automation* dengan memberikan keyakinan bahwa aplikasi siap untuk digunakan dengan kualitas yang baik. Pada penelitian ini pengujian manual mengevaluasi kesesuain aplikasi dengan harapan pengguna, sedangkan pengujian *automation* mengevaluasi dari segi efesiensi waktu yang digunakan pada saat pengujian[7].

Perbedaan penelitian ini dengan penelitian yang sedang penulis lakukan yaitu pada penggunaan *tools automation testing* yang dipilih, penulis menggunakan *Appium* sebagai *tools automation testing* sedangkan penelitian ini menggunakan *tools Katalon*.

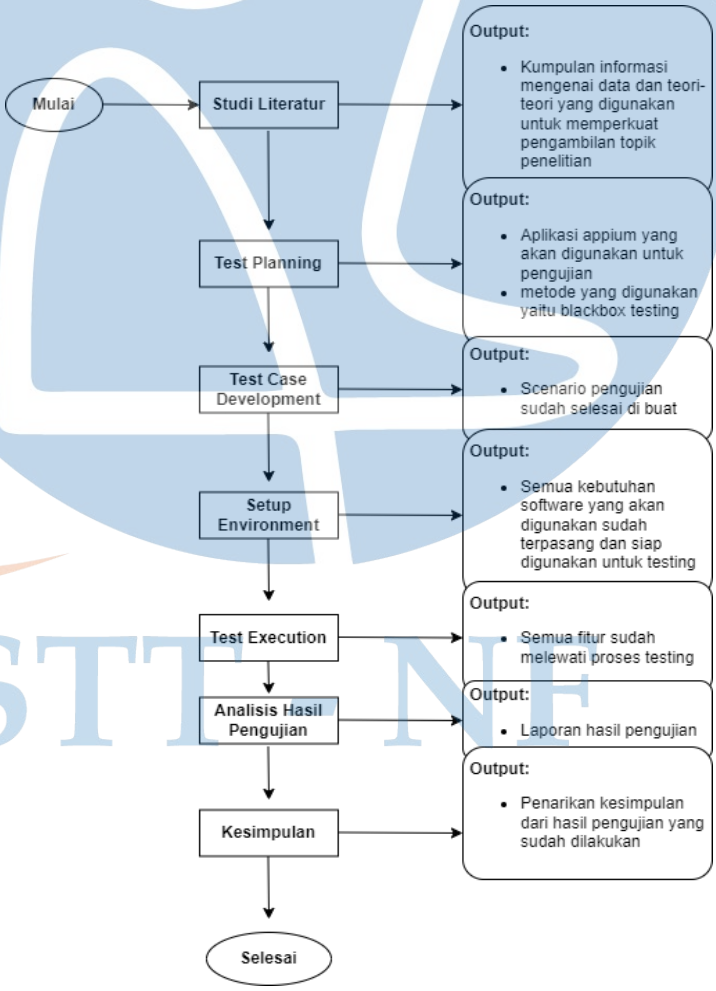
3) Penelitian dilakukan oleh Umiringsih, Muhamad Nur Ichsanudin, Muhammad Yusuf, Suraya (2022) yang berjudul "Pengujian Fungsional Perangkat Lunak Sistem Informasi Perpustakaan dengan Metode *Black-box Testing* Bagi Pemula". Penelitian ini berfokus pada pengujian fungsional pada aplikasi Sistem Informasi Perpustakaan untuk memberikan petunjuk kesalahan yang ada pada *system* aplikasi ataupun fitur-fitur aplikasi yang tidak sesuai. Hasilnya menunjukkan bahwa penggunaan metode *black box testing* dengan teknik *Equivalence Partition* sangat sesuai untuk pemula karena langkah-langkahnya yang lebih sederhana, dan pengujian aplikasi Sistem Informasi Perpustakaan mengungkapkan adanya *bug* atau kesalahan desain yang tidak sesuai dengan ekspektasi[11]. Perbedaan penelitian ini dengan penelitian yang sedang penulis lakukan yaitu pada tujuan penelitian yang dilakukan, penelitian ini berfokus pada tingkat keberhasilan penggunakan metode *black box testing* dengan teknik *Equivalence Partition*.

BAB III
METODOLOGI PENELITIAN

Pada bab ini membahas mengenai tahapan penelitian, rancangan penelitian yang berisi beberapa metode penyelesaian masalah yang ada seperti jenis penelitian yang digunakan, metode pengumpulan data yang dipilih, serta implementasi evaluasi dan penarikan kesimpulan terhadap penelitian yang sudah dilakukan.

3.1 Tahapan Penelitian

Tahapan ini berupa susunan atau langkah-langkah penelitian yang disusun sebagai pedoman pada saat penelitian. Langkah-langkah ini melibatkan studi literatur, *test planning*, *test case development*, *setup environment*, *test execution*, analisis hasil pengujian serta kesimpulan.



Gambar 3.1 Tahapan Penelitian

Gambar diatas menunjukan alur tahapan pengujian aplikasi *Android* Balink mengunakan aplikasi Appium dan metode *agile*. Berikut penjelasan mengenai tahapan-tahapan tersebut:

1. **Studi Literatur**: Memperlajari literatur terkait pengujian aplikasi Android menggunakan *tools* appium dengan metode *blackbox testing* dan teori-teori pendukung penelitian lainnya.
2. ***Test Planning***: Mempersiapkan *tools* dan *software* yang akan digunakan, serta menentukan fitur apa saja yang akan dilakukan pengujian. Tools yang digunakan yaitu Appium, Java, Intelij IDE, dan Android Studio dengan teknik yang digunakan pada pengujian ini yaitu Andoid UI Testing. Untuk fitur-fitur yang akan diuji ada fitur category, shop, event dan transaksi.
3. ***Test Case Develompent***: Menulis dokumen *test case*, setiap *test case* dibuat berdasarkan fitur yang akan dilakukan pengujian beserta *scenario* pengujiannya. *Test case* ini dibuat dalam bentuk tabel yang berisi informasi mengenai *scenario* pengujian.
4. ***Environment Setup***: aplikasi *Android* Balink yang akan dilakukan *automation testing* telah dipersiapkan dan *setup* aplikasi Appium yang akan digunakan untuk pengujian.
5. ***Test Execution***: Penulisan *script* pengujian pada Intelij IDE yang ditulis menggunakan bahasa pemrograman java. Setelah semuanya selesai di tulis *script* dijalankan.
6. **Analisis Hasil Pengujian**: Menganalisis seluruh hasil pengujian beserta metode yang digunakan dan melakukan evaluasi terhadap hasil *automation testing* yang telah dilakukan.
7. **Kesimpulan**: Tahap ini dilakukan penarikan kesimpulan terkait dengan hasil pengujian aplikasi Balink menggunakan Appium.

3.2 Rancangan Penelitian

Rancangan penelitian ini dibuat untuk menjelaskan mengenai arah dan target penelitian. Dengan tujuan yang jelas dan terukur, maka penelitian dan pemecahan masalah berjalan dengan baik.

3.2.1 Jenis Penelitian

Penelitian evaluatif merupakan jenis penelitian yang digunakan pada penelitian ini. Penelitian evaluatif merupakan sebuah penelitian yang digunakan untuk menganalisis suatu program atau kegiatan untuk melihat sejauh mana kesuksesan kegiatan atau program tersebut telah dilaksanakan sesuai yang diharapkan atau tidak, dengan berpedoman pada prosedur ilmiah yang sistematis untuk menentukan sebuah hasil[12]. Dalam penelitian ini peneliti akan melakukan pengujian pada aplikasi Balink menggunakan aplikasi appium dan mengevaluasi tingkat keberhasilan pembuatan test case kedalam *script automation testing*.

3.2.2 Analisis Data

Analisis data yang digunakan untuk penelitian ini yaitu analisis kuantitatif. Konsep kuantitatif, seperti yang dijelaskan oleh Kasiram dalam buku Metodologi Penelitian Kuantitatif (2008), didefinisikan sebagai metode memperoleh pengetahuan dengan menggunakan data berupa angka sebagai alat untuk menganalisis keterangan tentang apa yang ingin diketahui. Dalam pengujian ini metode kuantitatif dapat digunakan untuk mendapatkan hasil dari implementasi penggunaan aplikasi Appium untuk automation testing pada aplikasi Balink. Output penelitian berupa hasil pengujian automation testing dari aplikasi Balink menggunakan Appium.

3.2.3 Metode Pengumpulan Data

Pengumpulan data dan informasi pada penelitian ini dirancang menggunakan berbagai metode dan Teknik yang bersumber dari berbagai sumber. Metode pengumpulan data yang diterapkan melibatkan studi literatur dan eksperimen. Pendekatan ini dipilih untuk memastikan keberagaman sumber informasi dan menyelidiki aspek-aspek yang komprehensif dalam konteks penelitian ini.

a) Studi Literatur

Tujuan dari studi literatur ini yaitu menjelaskan teori-teori yang relevan dengan penelitian *software testing.* Teknik ini dilakukan melalui membaca, mempelajari, dan menelaah literatur-literatur terkait pengujian aplikasi andorid,

mulai dari teknik, *tools* dan metode apa yang di pilih untuk melakukan pengujian. Pada tahap studi literatur ini data yang dicari yaitu terkait penggunaan aplikasi *Appium* sebagai *tools* pengujian, cara kerja *automation testing* menggunakan aplikasi *Appium* dan bagaimana pembuatan *script* serta *test execution* pengujian tersebut.

b) Observasi

Penelitian observasi adalah salah satu dari beberapa macam metode penelitian kuantitatif. Penelitian observasi merupakan teknik pengumpulan data yang memiliki ciri-ciri khusus jika dibandingkan dengan teknik lainnya. Observasi tidak hanya terbatas pada manusia, tetapi juga dapat diterapkan pada objek-objek alam lainnya. Melalui kegiatan observasi, peneliti dapat memahami perilaku dan makna dari perilaku tersebut. Penelitian observasi biasanya digunakan mengamati objek penelitian yang sedang dilakukan sehingga peneliti dapat menyimpulkan dari apa yang diamati secara alami. Pada penelitian ini observasi digunakan untuk memperoleh data-data terkait tingkat keberhasilan *automation testing* yang nantinya akan dianalisis pada tahap analisi data.

Pada penelitian ini observasi digunakan untuk mengamati hasil pengujian *automation* yang dilakukan sehingga penulis bisa mengambil makna dan dapat menyimpulkan keberhasilan *test execution* dari *automation testing* aplikasi *Balink* dari analisis data yang dilakukan.

3.2.4 Metode Pengujian

Pengujian ini bertujuan untuk mengidentifikasi secara dini bugs/error yang terdapat pada pengembangan fitur-fitur aplikasi guna menganalisis langkah-langkah perbaikan yang dapat diambil untuk mencapai hasil yang maksimal.

1) Test Execution

Pengujian aplikasi Android ini memiliki peranan penting dalam memastikan bahwa aplikasi yang sedang atau telah dikembangkan dapat beroperasi sesuai dengan fungsionalitas yang diharapakan[13]. Pada pengujian ini dilakukan dengan cara melakukan *test execution* atau menjalankan *script* pengujian dengan mengikuti langkah-langkah cara kerja *automation testing* menggunakan *Appium*. Dengan pengujian ini nantinya

akan didapatkan hasil dari eksekusi program yang telah buat. Dari hasil tersebut akan dilakukan analisis data untuk mendapatkan persentase keberhasilan eksekusi *test case* yang telah dibuat.

3.2.5 Metode Implementasi dan Evaluasi

Metode implementasi dan evaluasi dalam pengujian ini akan diterapkan pada tahap tugas akhir dengan fokus pada implementasi *automation testing* menggunakan Appium pada aplikasi Android Balink. Detail mengenail implementasi dan evaluasi akan dijelaskan lebih detail pada bab 4.

3.2.6 Lingkungan Pengembangan

Dalam pengembangan penelitian ini, beberapa tools yang telah dipilih akan digunakan untuk proses *automation testing* pada aplikasi Android Balink. Beberapa tools yang akan digunakan yaitu Intelij IDE, Android Studio, Java dan Appium. Pada pengujian aplikasi ini penulisan script pengujian didukung dengan bahasa pemrograman Java.

BAB IV
IMPLEMENTASI DAN EVALUASI

Pada bab ini menjelaskan tentang bagaimana implementasi *automation testing* pada aplikasi Balink menggunakan aplikasi Appium dan mengevaluasi persentase keberhasilan pembuatan script *automation testing*.

4.1 Analisis dan Perancangan

Analisis dan perancangan pengujian aplikasi *Balink* merupakan tahap awal untuk menjelaskan mengenai sistem aplikasi *Balink* dengan menampilkan antar muka sistem dari aplikasi balink berupa fitur-fitur yang akan diuji serta rancangan proses pembuatan *scenario* pengujian menggunakan *tools Appium*.

4.1.1 Analisis Sistem dan Perancangan *Test Case* Aplikasi *Balink*

Analisis sistem merupakan langkah awal atau teknik awal dalam proses perangcangan pengujian perangkat lunak yang ditujukan untuk mengevaluasi kebutuhan sistem apa saja yang harus dilakukan pengujian pada proses pengembangan suatu perangkat lunak.

Analisis sistem aplikasi *Balink* dimulai dengan memperkenalkan antarmuka aplikasi yang berupa tampilan fitur-fitur aplikasi *Balink* yang telah dibuat dan akan dilakukan pengujian. Setelah menguraikan antar muka sistem, perancangan *test case* menjadi sangat penting karena merupakan salah satu hal yang dibutuhkan untuk menguji fitur yang telah dibuat. *Test case* ini dibuat untuk menjalankan *scenario* pengguna serta langkah-langkah detail dalam pengujian fitur *Balink*.

Pada tahap ini, dijelaskan mengenai fitur aplikasi yang dirancang untuk *Balink* serta perancangan *test case* pada setiap fiturnya. Penjelasan mengenai fitur aplikasi ini mencakup tampilan aplikasinya, fungsinya serta isi dari setiap *test case* yang dibuat.

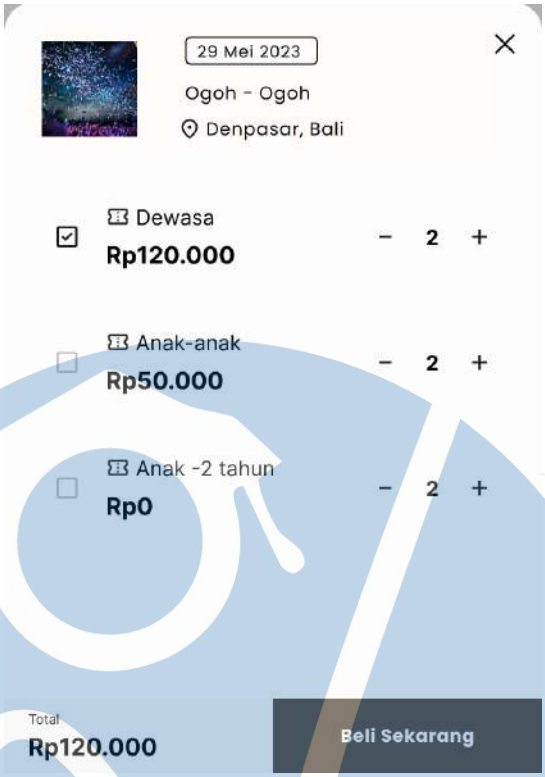
a. Halaman *Event*

Pada halaman ini akan menampilkan detail atau daftar *event-event* yang sedang berlangsung atau akan yang akan diadakan di Bali. Informasi yang disediakan berupa foto acara, deskripsi acara meliputi judul, waktu dan lokasi acara, serta harga yang tertera disetiap *event*-nya. Pengguna juga dapat

mencari event berdasarkan kriteria tertentu seperi waktu dan lokasi acara. Pada halaman ini juga terhubung dengan fitur transaksi untuk pemesanan dan pembayaran tiket suatu *event* yang ada di Bali.

Gambar 4.1 Halaman Event

Gambar 4.2 Halaman Detail Event



Gambar 4.3 Halaman Pesan Tiket Event

Halaman event ini perlu dilakukan pengujian karena halaman ini akan sering digunakan oleh pengguna untuk mencari *event-event* yang ada di Bali. Berikut *scenario* pengujian untuk halaman *event*:

Tabel 4.1 Rancangan *Test Case* Pada Fitur *Event*

ID	Deskripsi Pengujian	Hasil yang Diharapakan
TCEVENT01	Sebagai pengguna, saya ingin melihat semua daftar *event* dengan mengklik menu *event*.	Pengguna bisa melihat seluruh daftar data *event* pada halaman *event*.
TCEVENT02	Sebagai pengguna, saya ingin melihata data *event* secara lebih detai dengan mengklik salah satu *event* yang tersedia.	Pengguna bisa melihat salah satu *event* dengan lebih detail.

b. Halaman Kategori

Pada halaman Kategori ini akan menampilkan beberapa kategori produk yang tersedia dan biasa dijual dibali. Pada halaman ini pengguna dapat mencari atau melihat produk-produk khas bali sesuai dengan yang ingin kan

seperti tas, sepatu, baju dan aksesoris lainnya. Pengguna juga dapat membeli produk-produk yang tersedia.

Gambar 4.4 Halaman Kategori

Gambar 4.5 Halaman Detail Kategori

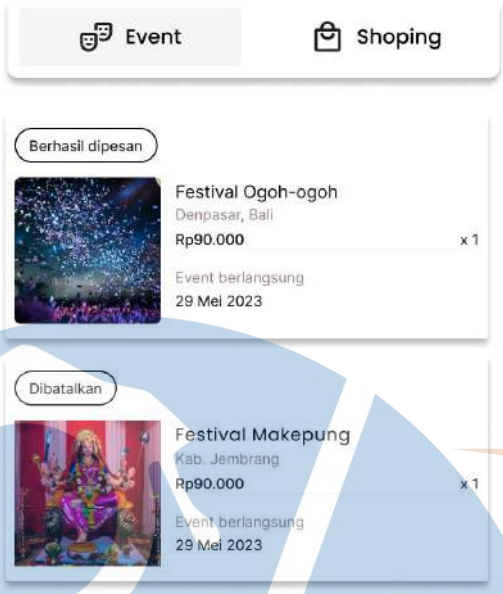
Halaman kategori juga perlu dilakukan pengujian karena pengguna akan sering berinteraksi dan menggunakan fitur kategori ini untuk membeli atau hanya sekedar melihat-lihat saja produk-produk khas Bali yang tersedia. Berikut scenario pengujiannya:

Tabel 4.2 Rancangan *Test Case* pada fitur kategori

ID	Deskripsi Pengujian	Hasil yang Diharapkan
TCCATEGORY01	Sebagai pengguna, saya ingin melihat seluruh kategori produk di aplikasi *Balink*.	Pengguna dapat melihat seluruh daftar kategori yang tersedia.
TCCATEGORY02	Sebagai pengguna, saya ingin melihat detail kategori produk pakaian dengan mengklik kategori pakaian.	Pengguna dapat melihat seluruh detail produk pakaian.
TCCATEGORY03	Sebagai pengguna, saya ingin melihat detail kategori produk tas dengan mengklik kategori tas.	Pengguna dapat melihat seluruh detail produk tas.
TCCATEGORY04	Sebagai pengguna, saya ingin melihat detail kategori produk kerajinan tangan dengan mengklik kategori kerajinan tangan.	Pengguna dapat melihat seluruh detail produk kerajinan tangan.

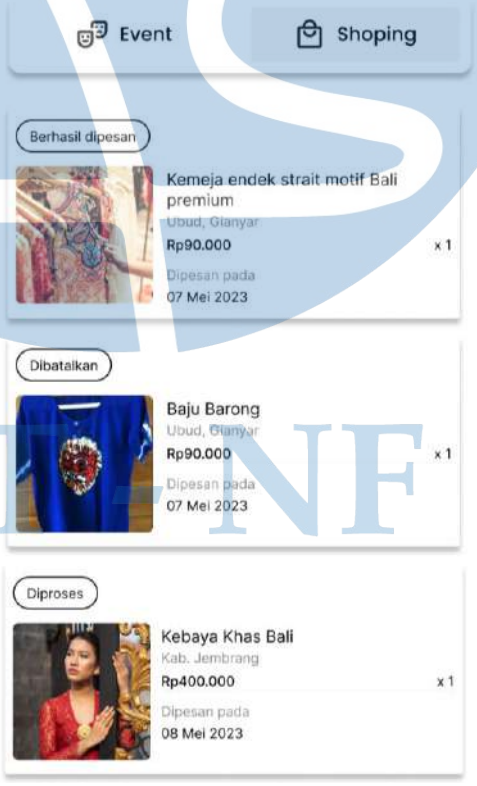
c. Halaman Transaksi

Halaman transaksi merupakan halaman yang akan sering digunakan oleh pengguna untuk melakukan transaksi pembelian tiket *event*, produk atau kerajinan-kerajinan lainnya yang ada di Bali. Pada halaman transaksi juga menampilkan daftar *history* transaksi yang pernah dilakukan yang melibatkan pembelian barang atau tiket. Setiap transaksi memberikan informasi meliputi tanggal, jam, barang yang dibeli, dan jumlah pembayaran yang sudah dilakukan.



Gambar 4.6 Halaman Transaksi Event

Halaman transaksi *event* ini digunakan untuk melihat daftar *event* yang telah dipesan, pada halaman ini juga pengguna dapat melihat *event* mana aja yang berhasil dan dibatalkan pemesanannya.

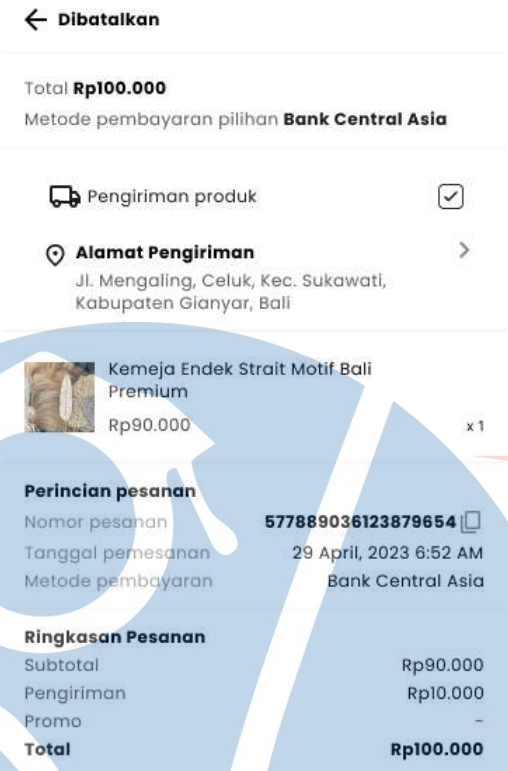


Gambar 4.7 Halaman Transaksi Produk

Halaman transaksi produk biasanya digunakan pengguna untuk melihat daftar produk atau barang yang dibeli, pada halaman ini juga menampilkan informasi mengenai update dari barang yang dipesan seperti masih diproses, berhasil ataupun dibatalkan pesanannya.

Gambar 4.8 Halaman Pemesanan Berhasil

Halaman ini menampilkan mengenai detail informasi terkait barang yang telah berhasil dipesan dan sudah dilakukan pembayaran. Biasanya informasi yang dicantumkan terkait harga, promo, ringkasan pesanar serta rincian pesanan seperti tanggal dan bank apa yang digunakan untuk pembayaran.



Gambar 4.9 Halaman Pemesanan Dibatalkan

Halaman transaksi digunakan untuk melakukan pembayaran pembelian barang atau produk sehingga diperlukan pengujian karena halaman ini akan sering digunakan oleh pengguna. Berikut scenario pengujiannya:

Tabel 4.3 Rancangan Test Case Pada Fitur Transaksi

ID	Deskripsi Pengujian	Hasil yang Diharapkan
TCTRANSAKSI01	Sebagai pengguna, saya ingin melihat seluruh daftar history transaksi yang telah dilakukan.	Pengguna dapat melihat seluruh daftar history transaksi
TCTRANSAKSI02	Sebagai pengguna, saya ingin melihat detail salah satu transaksi pembelian tiket *event* dengan mengklik salah satu history pembelian tiket *event.*	Pengguna dapat melihat detail transaksi pembelian tiket event.

d. Halaman Artikel

Pada halaman artikel ini menampilkan seluruh artikel tentang Bali yang ada pada aplikasi Balink. Informasi yang disampaikan pada halaman ini seperti judul artikel, foto terkait artikel serta deskripsi dari artikel tersebut. Fitur ini berada di dalam fitur profile.

Gambar 4.10 Halaman Artikel

Halaman artikel ini sering digunakan pengguna untuk membaca sejarah-sejarah yang ada di Bali. Pada halaman ini juga menampilkan detail setiap artikel yang ada dengan mengklik salah satu artikelnya sehingga halaman artikel ini sangat perlu untuk dilakukan pengujian untuk melihat apakah fungsi fitur berjalan dengan baik atau tidak.

Gambar 4.11 Halaman Detail Artikel

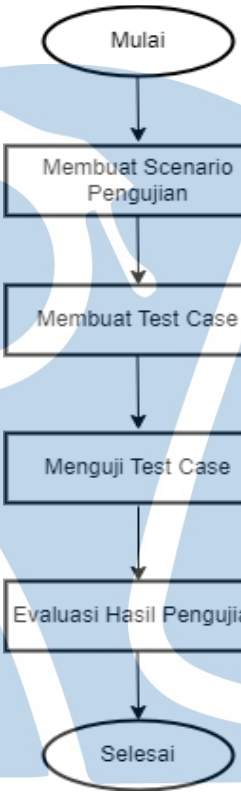
Berikut scenario pengujian untuk halaman artikel:

Tabel 4.4 Rancangan Test Case pada Fitur Artikel

ID	Deskripsi Pengujian	Hasil yang Diharapkan
TCARTIKEL01	Sebagai pengguna, saya ingin melihat seluruh artikel dengan mengklik fitur artikel	Pengguna bisa melihat seluruh artikel yang ada dihalaman artikel
TCARTIKEL02	Sebagai pengguna, saya ingin melihat detail salah satu artikel dengan mengklik salah satu artikel yang ada	Pengguna dapat melihat detail artikel yang ingin dilihat

4.1.2 Analisis Proses *Automation Testing*

Dalam melakukan automation testing, ada beberapa hal atau tahap yang perlu diperhatikan. Beberapa hal tersebut yaitu tahapan kegiatan yang wajib dilakukan pada pengujian automation ini. Tahapan-tahapan tersebut bisa dilihat pada diagram alir berikut:



Gambar 4.12 Diagram Alir Tahapan Pengujian

Penjelasan dari diagram alir:

1. Pembuatan *scenario* pengujian, pada tahap ini tahap perancangan *scenario* pada fitur yang akan dilakukan automation untuk mendeteksi *erorr*.
2. Pembuatan *test case*, pada tahap ini merupakan tahap menuliskan step-step pengujian pada halaman aplikasi yang akan dilakukan *automation testing*.
3. Menguji *Test*, pada tahap ini dilakukan *automation testing* yang berpacu pada *scenario* pengujian yang telah dibuat.
4. Evaluasi Hasil Pengujian, Pada tahap ini dilakukan untuk mengevaluasi hasil pengujian yang sudah didapat untuk mengetahui sejauh mana aplikasi ini bisa digunakan oleh pengguna.

4.1.3 *Test Plan Requirment*

Pada tahap ini pengujian dilakukan pada bagian-bagian secara menyeluruh secara satu per satu pada tampilan *Android* pengguna dengan menuliskannya didalam sebuah dokumen terperinci yang menguraikan fitur mana saja yang akan diuji dan seberapa prioritaskah fitur tersebut untuk dilakukan pengujian serta penjelasan dari pengujian yang akan dilakukan. Dokumen ini akan membantu pengembang dalam memahami kerangka kerja yang dirancang supaya lebih jelas dalam melakukan pengujian, apa saja yang diperlukan untuk memastikan berfungsinya perangkat lunak berjalan dengan baik.

Tabel 4.5 Skala Prioritas Pengujian

Fitur	Prioritas	Deskripsi
Event	Tinggi	Halaman *event* digunakan untuk memanajemen semua *event-event* yang ada di Bali
Kategori	Tinggi	Halaman Kategori dapat digunakan untuk memanajemen kategori produk yang ada di aplikasi *Balink*
Artikel	Tinggi	Halaman artikel dapat digunakan untuk pengguna melihat sejarah atau membaca sejajarah Bali aplikasi *Balink*
Transaksi	Tinggi	Halaman transaksi dapat digunakan untuk pembayaran dan melihat *history* transaksi yang sudah berhasil dan belum.

Halaman-halaman yang akan dilakukan *automation testing* sebagai berikut:

a. Sebagai pengguna, saya ingin melihat *event* dan membeli tiket *event* yang tersedia.
b. Sebagai pengguna, saya ingin melihat deskripsi dari salah satu *event*.
c. Sebagai pengguna, saya ingin melihat seluruh daftar kategori poduk.
d. Sebagai pengguna, saya ingin melihat daftar kategori produk kerajinan tangan.
e. Sebagai pengguna, saya ingin melihat daftar kategori tas.
f. Sebagai pengguna, saya ingin melihat daftar kategori Pakaian.

g. Sebagai pengguna, saya ingin melihat daftar *history* transaksi.
h. Sebagai pengguna, saya ingin melihat salah satu detail transasi tiket *event*.
i. Sebagai pengguna, saya ingin melihat seluruh produk yang ada dihalaman shop.
j. Sebagai pengguna, saya ingin melihat detail salah satu produk yang ada di halaman shop.

4.1.4 Perancangan Pengujian Arsitektur *Appium*

Appium adalah tools open-source yang digunakan QA engineer untuk melakukan automatisasi pengujian hanya pada aplikasi mobile. Appium sendiri memiliki kerangka cara kerjanya seperti berikut:

- *Library Client* mengubah perintah kode pengguna ke perintah *REST API*
- Permintaan ini di *send* ke *Server Appium* melalui *mobile JSON wire* protokol
- *Server Appium* menghubungkan keinginan ini ke *emulator android*
- Perintah-perintah ini ditafsirkan oleh *boostrap.jar* yang mengkonversiya menjadi format *UI automator* yang dapat dipahamani oleh *seluler*.
- Perintah *UI automatuor* kemudian dijalankan di emulator
- *Emulator* kemudian mengembalikan hasil dari perintah yang dilakukan *server Appium* melalui *boostrap.jar*.
- *Server Appium* meneruskan respon ini ke *client*.

4.1.5 *Overview Test Result*

Hasil pengujian otomatis menggunakan aplikasi Appium akan ditampilkan melalui tools *Serenity BDD (Behavior Driven Development)* dengan didefinisikan dalam bentuk diagram lingkaran yang menunjukkan distribusi hasil tes, serta diagram batang yang mengurutkan hasil tes berdasarkan hasil dan durasi tes.

1. ***Test Case* Berhasil Dieksekusi**

Gambar 4.13 *Test Case* Berhasil Dieksekusi

Gambar diatas merupakan contoh dari *overview test result* keberhasilan dari suatu *test case* yang telah dijalankan. Dari gambar diatas dapat diinformasikan ada 2 *test case* yang diuji, tidak ditemukan *bug* dan pengujian dapat berjalan dengan lancar sesuai rancangan. Laporan hasil pengujian tersebut menunjukkan bahwa dari 2 *test case*, semuanya berhasil (status '*passing*') dengan persentase 100%. Durasi pengujian otomatis untuk 2 *test* berkisar antara 10 hingga 30 detik ini tergantung dari jaringan yang dipakai.

Dengan demikian, laporan pengujian menggunakan *tools Appium* menunjukkan bahwa pengujian keseluruhan berjalan lancar tanpa menemukan *bug*. Selain itu, aplikasi *Balink* juga mampu mempertahankan performanya dengan menyelesaikan setiap tes dalam waktu yang sesuai. Hasil ini memberikan keyakinan bahwa aplikasi tersebut dapat beroperasi secara optimal di lingkungan produksi dengan respon yang cepat. Dengan hasil tersebut bisa dilaporkan kepada developer bahwa fitur yang dibuat bisa dikembangkan lagi atau bisa ke tahap *hosting* karena telah melewati tahap pengujian.

2. *Test Case Fail*

Gambar 4.14 *Test Case Fail*

Gambar 4.14 merupakan hasil *test result* dari pengujian *test case* yang telah dilaksanakan, sekilas gambarnya sama dengan 4.13 hanya terdapat penjelasan yang berbeda. Terdapat sebuah diagram lingkaran yang menunjukkan distribusi hasil tes. Warna berbeda mewakili status kasus uji: "*passed*" (berhasil dengan warna hijau), "*failed*" (gagal dengan warna orange). Persentase masing-masing status terlihat jelas. Dari gambar diatas dapat diinformasikan ada 29 *test case* yang telah dibuat tetapi hanya 5 *test case* yang berhasil dieksekusi, 24 lainnya *failed* atau tidak berhasil dieksekusi dengan persentase 17% *passing* dan 83% *fail*.

Kegagalan pada suatu *test case* yang melewati tahap pengujian biasanya terjadi karena adanya kesalahan pada saat mendevelop sebuat fitur pada aplikasi atau terjadi karena dari tim *Quality Engineer*-nya salah dalam mendefinisikan *scenario* yang diperlukan untuk pengujian.

Dengan demikian, hasil pengujian tersebut harus secepatnya dilaporkan kepada tim pengembang untuk dilakukan perbaikan pada fitur tersebut sebelum masuk ke tahap *hosting* atau publikasi.

4.2 Implementasi Pengujian

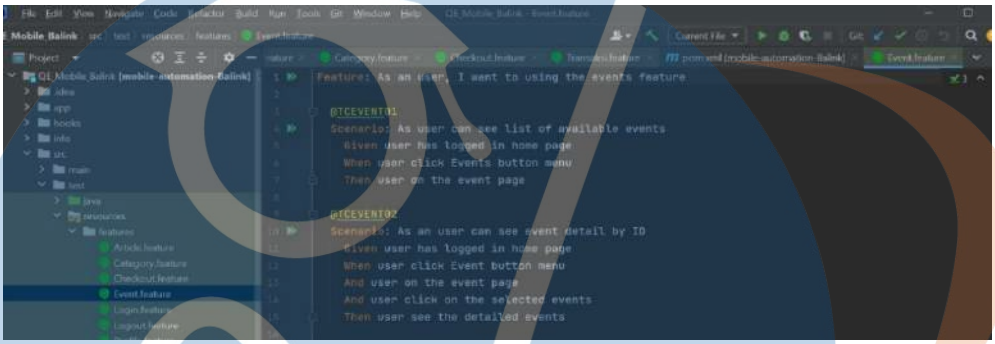
Tahap ini adalah tahap implementasi pengujian pada aplikasi *Android Balink*. Pada tahap ini akan dijelasakan mengenai penerapan aplikasi *Appium* dalam *Automation Testing* pada antarmuka aplikasi *Balink* serta menjelaskan secara terperinci mengenai langkah-langkah pengujian menggunakan arsiterktur pada

Appium dan memyajikan hasil *testing*-nya menggunakan *report* hasil pengujian yang ada pada *serenity BDD*.

4.2.1 Implementasi *Automation Testing* pada *User Interface* aplikasi *Balink*

Penerapan *automation testing* untuk antarmuka aplikasi *android Balink* terdapat beberapa *class java* untuk pengujian aplikasi, *class-class* tersebut akan dijelaskan sebagai berikut:

1. ***Feature/Scenario***

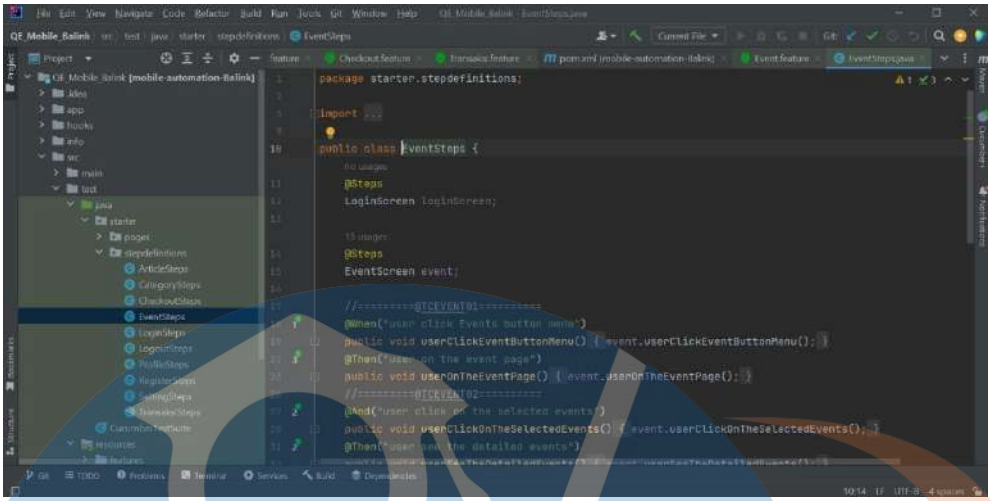


Gambar 4.15 Tampilan Scenario

Dalam penulisan kode *automation testing*, *scenario* pengujian ditulis dalam sebuah *file* yang diberi nama *feature* yang berisi *scenario-scenario* yang telah dibuat. Biasanya dalam pembuatan *scenario* pengujian di bagi menjadi 2 yaitu *scenario* positif dan *scenario* negatif. *HappyFlow* atau *scenario* pengujian positif dibuat untuk *scenario* yang menguji apakah aplikasi berperilaku sesuai harapan ketika diberikan *inputan valid*. Contoh dari *inputan valid* atau *scenario* positif biasanya ketika memasukan kata sandi yang benar atau memasukan *username* yang benar atau rincian *inputan* lainnya yang berperilakuk pada pengujian positif.

Lain hal dari *scenario* positif, *ErrorFlow* atau *scenario* negatif dibuat untuk menguji apakah aplikasi tetap merespon seperti yang diharapkan ketika diberikan inputan negatif. Contoh inputan negatif yaitu seperti ketika pengguna memasukan kata sandi yang salah tetapi respon yang dihasilkan pengguna tetap bisa masuk ke dalam aplikasi padahal sudah jelas kata sandi yang dimasukkan salah. Oleh karena itu pada *scenario* negatif ini diperlukan keterkaitan validasi dari aplikasi terhadap data-data yang tidak *valid*.

2. *Step dan Stepdefinitions*

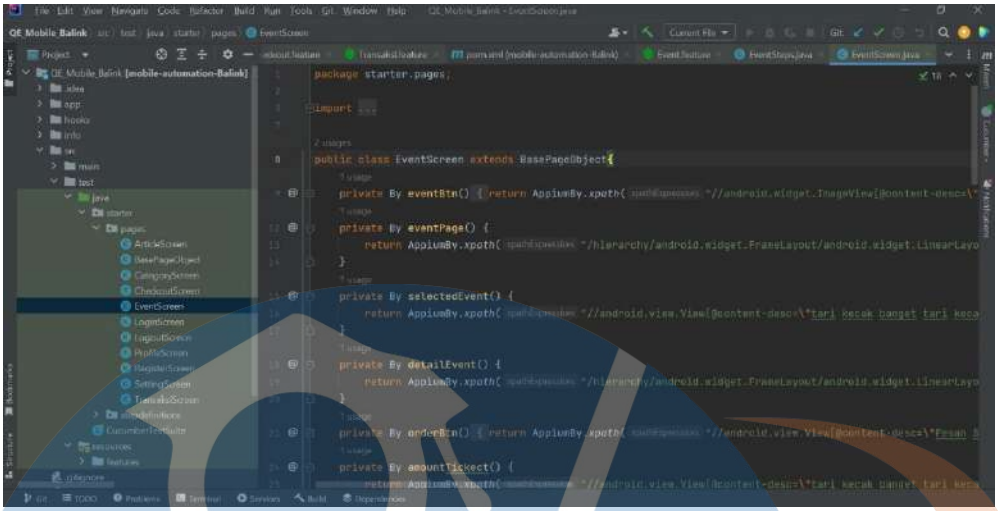


Gambar 4.16 Tampilan Stepdefinitions

Class berikut dari penulisan kode *automation testing* yaitu *stepdefinitions*, *stepdefinitions* ini digunakan untuk mengghubungkan beberapa langkah atau *step Gherkin* menjadi satu. *Step* ini akan membantu *cucumber* untuk mencari *step* yang cocok dalam menjalankan langkah *Gherkin* dalam sebuah *scenario*. Dalam *scenario* yang telah dibuat *class* ini memuat *keyword-keyword* seperti *@Given*, *@When*, *@And*, dan *@Then* yang bertujuan untuk membantu pemetaan dalam running kode program pengujian.

Stepdefinitions mengandung *class step* yang berperan penting dalam *running* kode program agar mudah dibaca dan dipahami oleh *tools* pen *testing*nya. *Class step* ini memberitahu *serenity* bahwa variabel yang terdapat dalam *class* ini adalah *step library*. Pada step ini terdapat definisi-definisi seperti "apa" yang diinginkan dan "bagaimana" cara menghasilkan hasil pengujian. Dengan adanya *class step* ini pengujian akan lebih mudah dipahami dan dilakukan karena tidak terlalu teknis dan dapat membatasai perulangan kode program saat *running*.

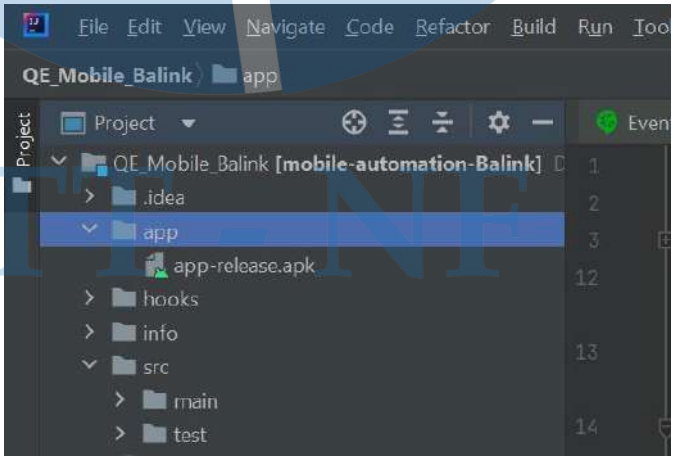
3. *Pages*



Gambar 4.17 Tampilan Class Page Object

Class berikutnya dalam penulisan kode program *automation testing* yaitu *page object*. *Page object* adalah cara kode program untuk mengirim detail implementasi dari halaman *mobile* ke dalam suatu *class* dengan menggunakan *inspect element*. Biasanya *programmer* mencari *element* aplikasi yang akan di uji bisa menggunakan *syntax @FindBy* dengan menggunakan *id* atau *Xpath* dari *element* ketika tidak ditemukan *id* pada *element* tersebut. Dalam *class* ini akan mendefinisikan fitur-fitur yang akan diuji dengan terdapat perintah-perintah program.

4. *App*

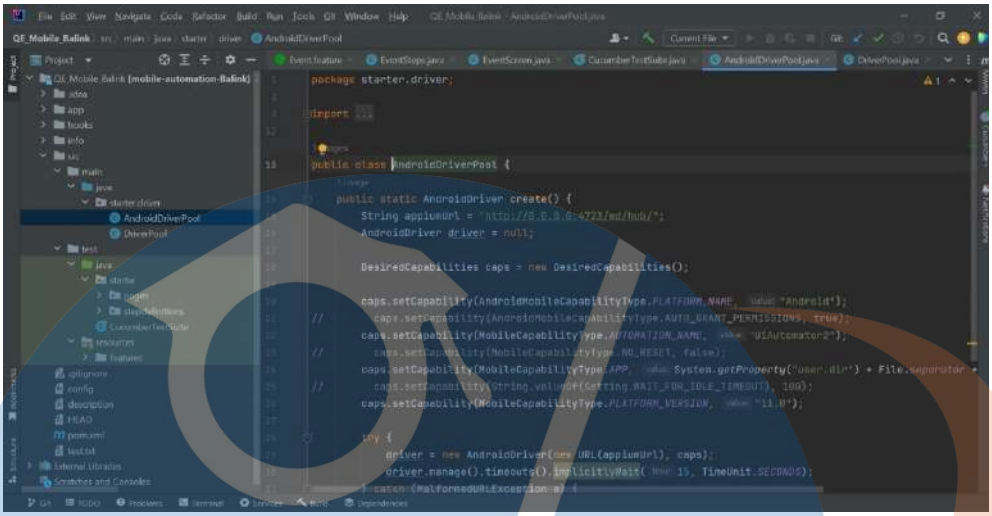


Gambar 4.18 Tampilan File App

File App ini tidak termasuk dalam *class-class java* tetapi file ini berperan penting dalam pengujian aplikasi *Android Balink*. Didalam file ini

terdapat file aplikasi *Andoid Balink* yang sudah *ready* untuk pengujian, file ini biasanya berbentuk *.apk*.

5. Driver Android



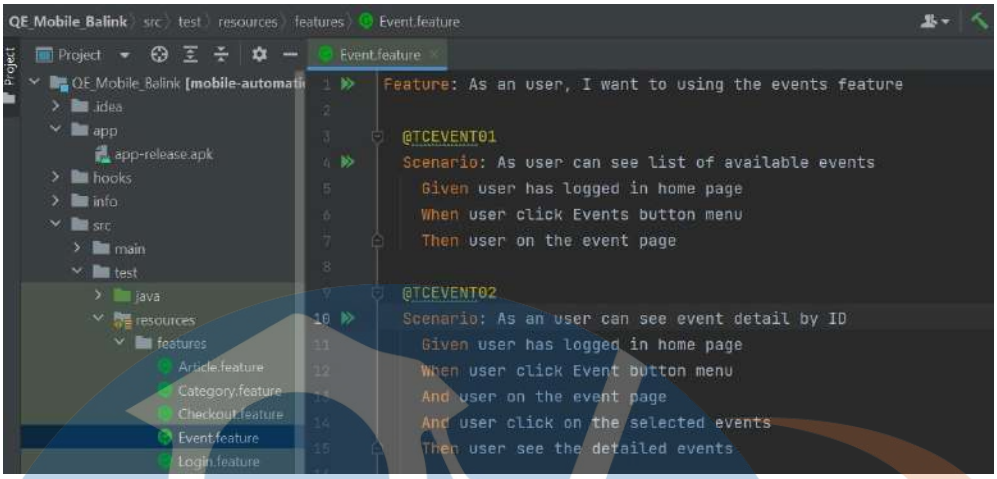
Gambar 4.19 Tampilan Driver Andorid

Andorid Driver ini juga bukan termasuk dari *class java*, *android driver* ini digunakan untuk menghubungkan antara apliksi *Balink* yang akan diuji dengan *tools testing*. Pada *android driver* ini akan di*setting* untuk *Url appium* yang digunakan, *UiAutomator* dan *logic* yang diperlukan untuk pengujian.

4.2.2 Implementasi *Test Case* kedalam *Script Automation Testing*

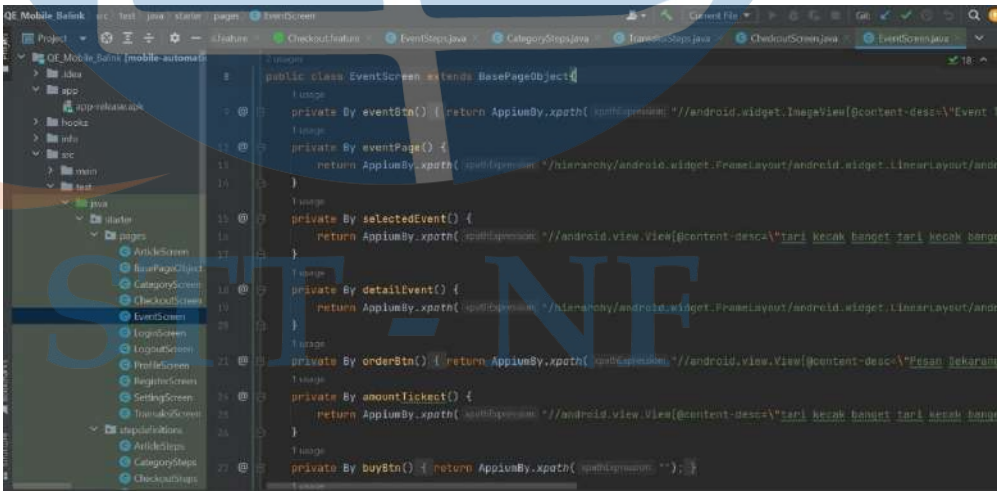
Dalam pengujian *automation testing* juga diperlukan *script automation* untuk mempermudah ketika melakukan *testing*, pada pengujian aplikasi *Balink* ini ada beberapa fitur yang akan diuji. *Script* pengujian aplikasi *Balink* akan dijelaskan sebagai berikut:

1. **Script Automation Fitur Event**



Gambar 4.20 Feature Event

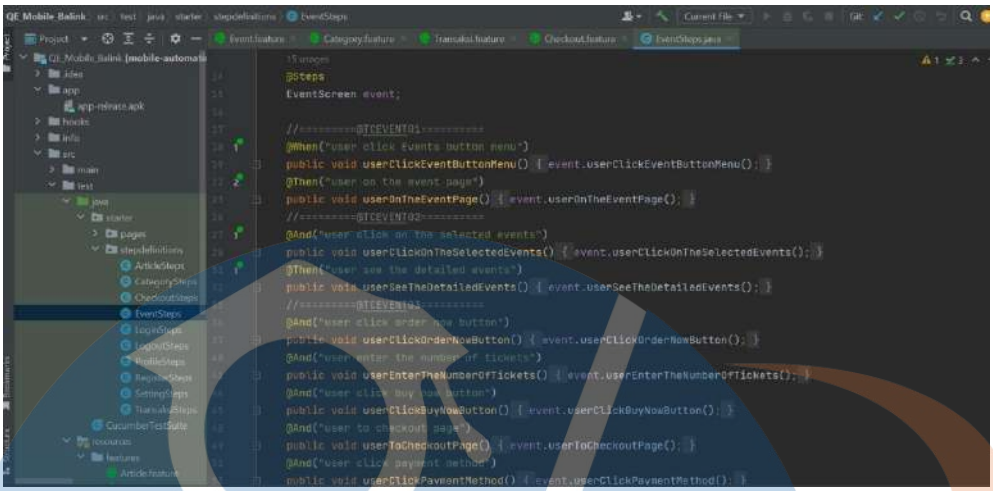
Penulisan *test case* pada kode program *automation testing* disimpan pada folder *feature*, pada file ini didefinisikan mengenai *scenario* pengujian yang akan dijalankan, terdapat penjelasan tentang apa, bagaimana dan seperti apa hasil yang diharapkan pada saat pengujian itu dilakukan. Pada gambar diatas ditulis kode program *@TCEVENT01* yang dimana *test case* tersebut ditulis pengguna ketika ingin melihat semua *list event* yang ada pada aplikasi *Balink*. Sedangkan *@TCEVENT02* digunakan pengguna untuk melihat detail dari salah satu *event* yang tersedia pada aplikasi *Balink*.



Gambar 4.21 Page Object Event

Setelah *feature* dibuat, tahap selanjutnya yaitu pembuatan *page Object* atau *object* apa yang akan digunakan untuk mengirimkan detail implementasi kode program yang sudah dibuat ke halaman *mobile Balink*. Dengan cara

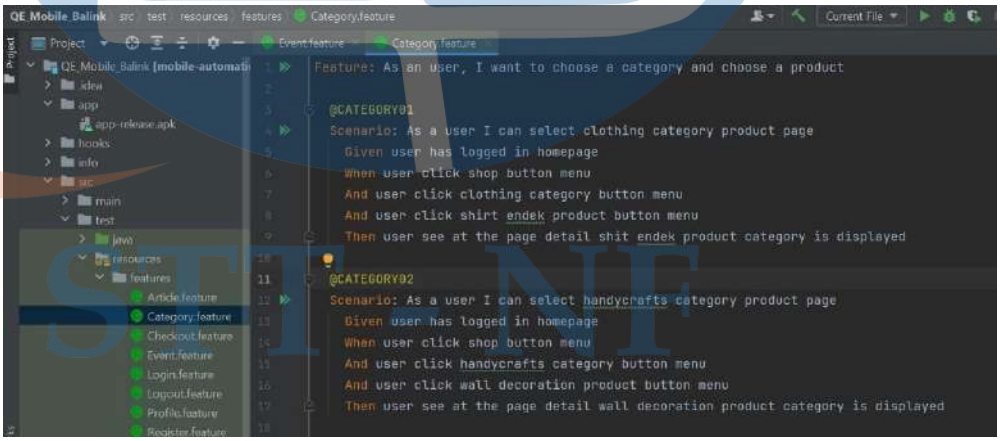
inpect element pada *icon* atau pada tombol tertentu sehingga bisa diketahui pada elemen tersebut terdapa *id* atau *class* yang bisa digunakan.



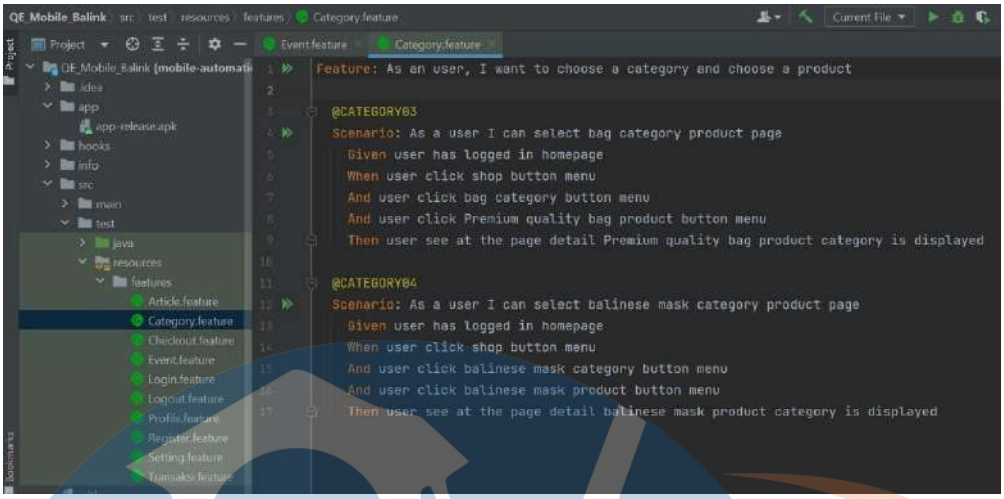
Gambar 4.22 Stepdefinition Event

Gambar diatas adalah *stepdefinition* dari fitur *event*, pada file ini step-step pengujian yang akan dilakukan didetailkan satu persatu seperti apa yang diinginkan pada pengujian fitur event ini, bagaimana caranya step ini dijalanya, dan hasil akhir apa yang akan diharapkan dengan didefinisikan menggunakan *@When*, *@Then*, *@and*. Pada file ini juga digunakan untuk memanggil kode program yang sudah dibuat pada file pages.

2. *Script Automation* Fitur Kategori



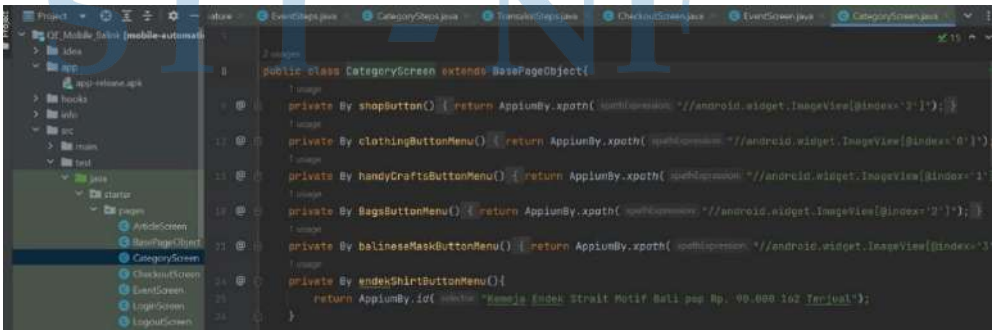
Gambar 4.23 Feature kategori 1



Gambar 4.24 Feature Kategori 2

Pada gambar diatas terdapat kode program dari implementasi *test case* fitur kategori yang dimana terdapat kata-kata *Feature* yang mendeskripsikan seluruh *test case* yang ada pada fitur yang akan diuji. Dalam *test case* tersebut menjelaskan keinginan pengguna untuk memilih kategori produk dan kemudian memilih produk tersebut dalam kategori pada suatu aplikasi.

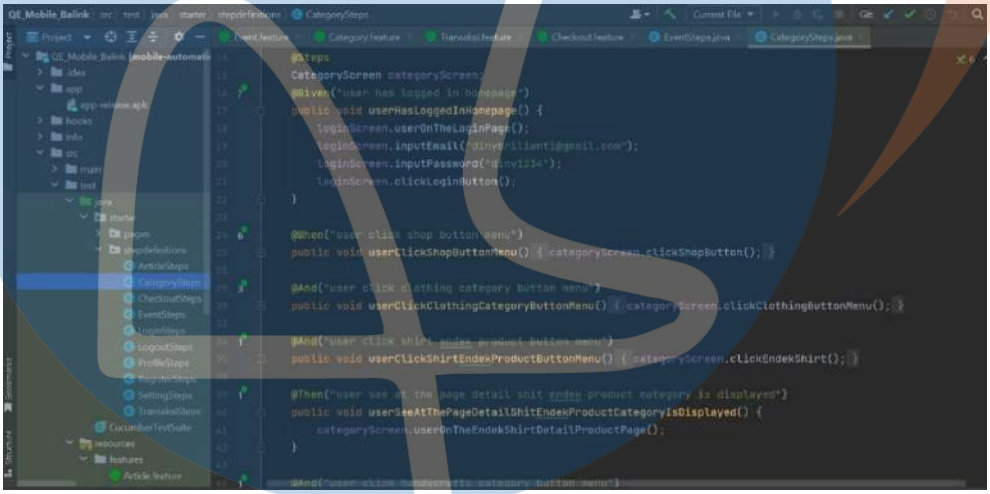
@Category1, *@Category2*, *@Category3* dan *@Category* 4 ini adalah *tag* yang digunakan untuk mengelompokkan atau menyaring tes. Mereka dapat digunakan untuk menjalankan hanya *scenario* tertentu yang cocok dengan *tag*. *Scenario* ini digunakan untuk menguraikan situasi atau case tertentu yang ingin dilakukan oleh pengguna. Pada *case* pertama pengguna ingin penguji fungsionalitas memilih kategori pakaian dari menu toko, *case* kedua penguji memilih kategori kerajinan tangan dari menu toko, *case* ketiga penguji memilih kategori kerajinan tas dari menu toko dan *case* keempat penguji memilih kategori masker bali pada menu toko.



Gambar 4.25 Page Object Kategori

Script diatas adalah *page object* dari fitur kategori. Pada *script* tersebut mengandung beberapa metode yang digunakan untuk mengembalikan *widget*, yang akan digunakan pada antarmuka pengguna aplikasi *Balink*. Setiap metode memiliki kode program yang menunjukkan fungsinya, seperti *shopButton(), HandcraftButtonMenu()* dan *businessKitsButtonMenu()*. Secara keseluruhan *script* ini dirancang untuk memudahkan navigasi dan interaksi pengguna dengan elemen-elemen *UI* dalam aplikasi *Balink*.

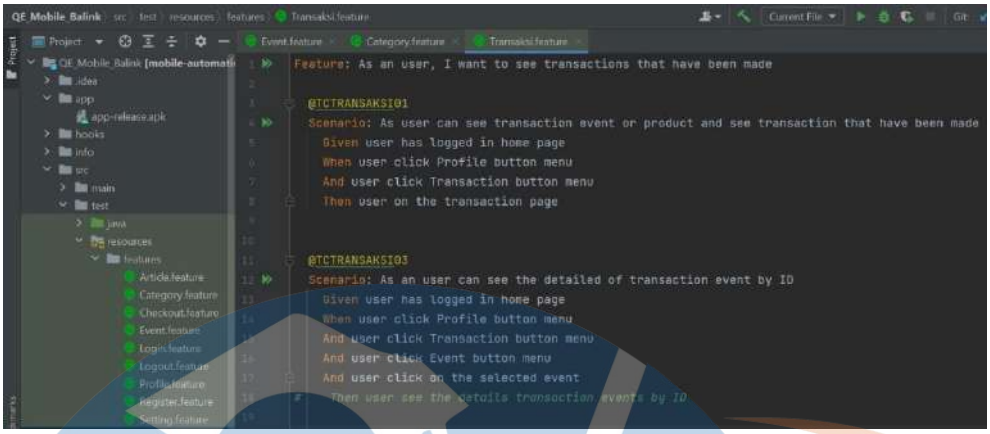
Setiap metode di atas menggunakan *AppiumBy.xpath* atau *AppiumBy.id* untuk menemukan elemen *UI* berdasarkan *Xpath* atau *ID*. Misalnya, metode *shoButton()* dibuat untuk mengembalikan sebuah *widget* yang dirancang sebagai tombol untuk pengguna berbelanja dalam aplikasi *Balink*. Metode ini menggunakan *AppiumBy.xpath* untuk menemukan elemen *UI* berdasarkan struktur *XML* aplikasi *Android Balink*.



Gambar 4.26 Stepdefinition kategori

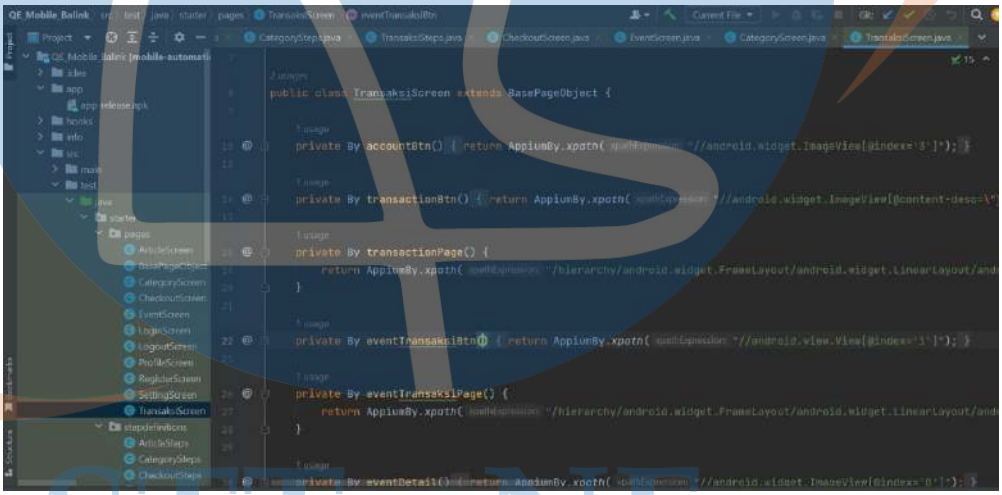
Script diatas dirancang untuk meniru interaksi pengguna dengan elemen *UI* aplikasi *Android* dalam *automation testing*. Anatosi *@Given*, *@When*, dan *@Then* dapat membantu mendefinisikan langkah-langkah dalam *scenario* pengujian yang menggambarkan kondisi awal, *action* yang dilakukan dan hasil yang diharapkan. Seperti kode program yang ingin melihat suatu halaman *shop* dengan *action* menekan tombol shopnya sehingga hasil akhir yang didapatkan yaitu pengguna dapat melihat keseluruhan halaman *shop*. Script ini juga digunakan untuk memanggil kode program yang sebelumnya sudah dibuat pada file *page object*.

3. *Script Automation* Fitur Transaksi



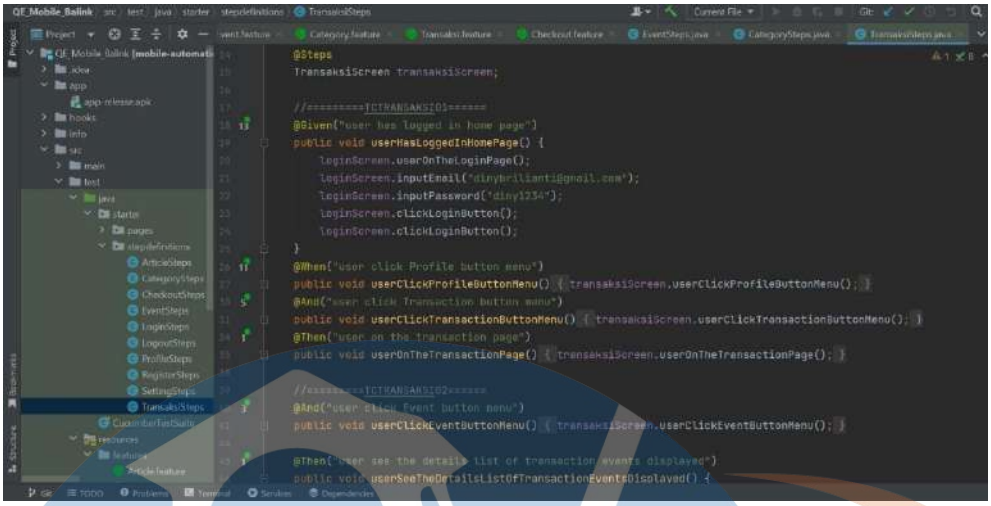
Gambar 4.27 Feature Transaksi

Script di atas dibuat untuk menjelaskan dan menggambarkan difitur transaksi yang akan diuji dengan beberapa test case yang ada. Dengan case pertama pengguna ingin melihat history transaksi yang pernah dilakukan dan case kedua yaitu pengguna ingin melihat salah satu detail transaksi dari event yang pernah dibeli.



Gambar 4.28 Page Object Transaksi

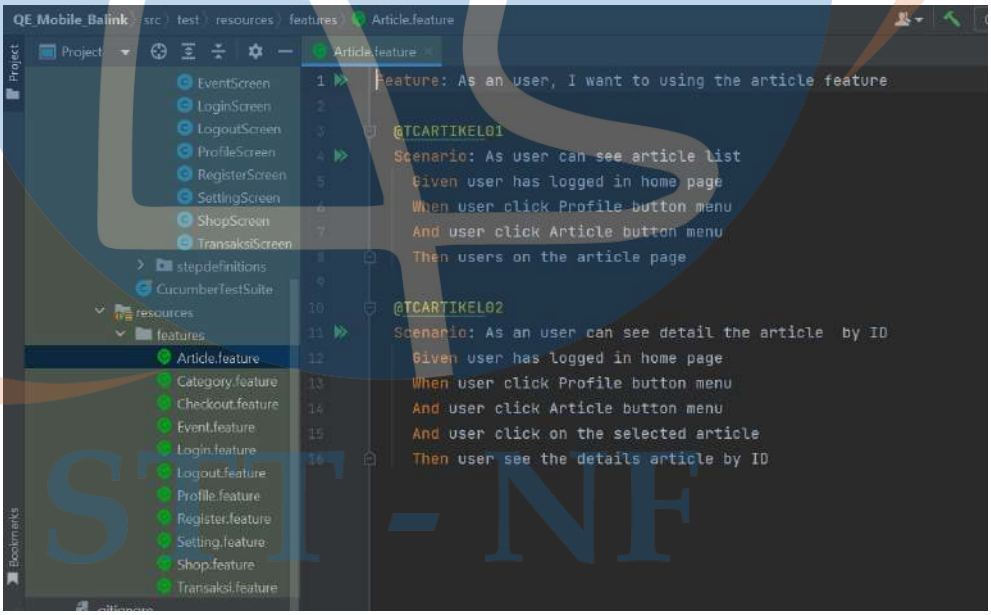
Gambar berikutnya yaitu menjelaskan mengenai *page object* yang digunakan untuk pengujian fitur transaksi. Sama dengan fitur-fitur sebelumnya pada kode program ini juga menggunakan *AppiumBy.Xpath* dan *AppiumBy.ID* untuk mengambil atau menemuka elemen *UI* seperti *icon*, tombol atau *text* yang akan digunakan sebagai perantara pengujian.



Gambar 4.29 Stepdefinition Transaksi

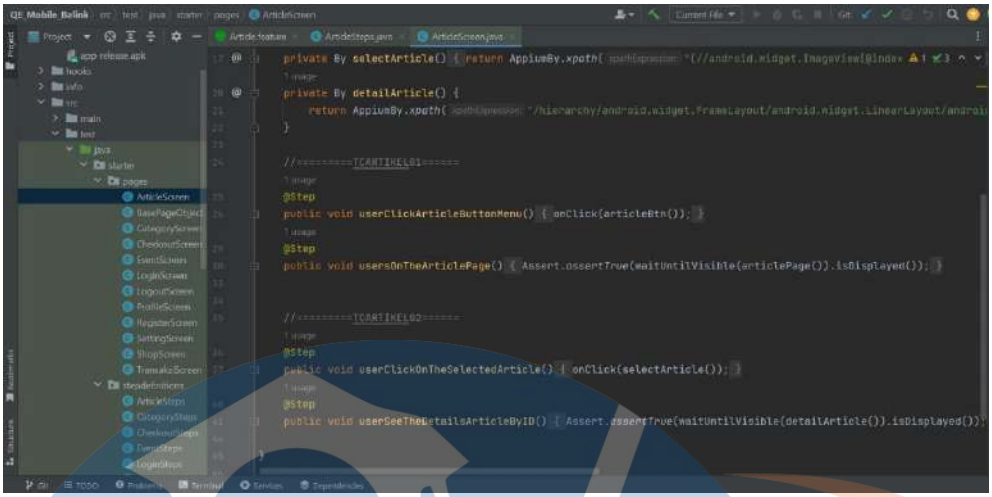
Script diatas dirancang untuk membantu *programmer* dalam mendefinisikan langkah-langkah *scenario* pengujian fitur transaksi yang telah dibuat. *Script* diatas menjelaskan kondisi awal yang terjadi, *action* yang akan dilakukan dan hasil akhir yang diharapkan.

4. *Script Automation* Fitur Artikel



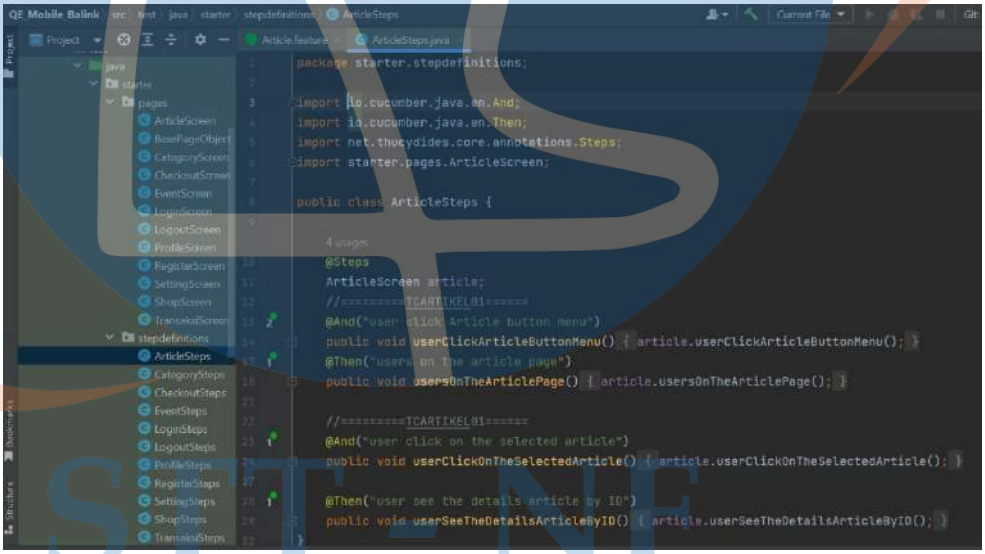
Gambar 4.30 Feature Artikel

Script diatas merupakan *scenario* pengujian pada fitur artikel, yang dibuat untuk menguji fitur artikel dengan *scenario* pengujian pertama yaitu pengguna ingin melihat seluruh data artikel, dan *scenario* yang kedua yaitu pengguna ingin membaca salah satau detail atau deskripsi dari artikel yang dipilih.



Gambar 4.31 Page Object Artikel

Script berikutnya merupakan *page object* dari fitur artikel yang dimana pada *script* ini terdapat kode program yang digunakan untuk mengambil elemen-elemen aplikasi seperti *icon* atau *text* dengan cara *AppimBy.Xpath* atau *AppiumBy.ID* sehingga mempermudah pada saat *execution* kode program



Gambar 4.32 Stepdefinitions Artikel

Gambar di 4.13 merupakan kode program yang dirancang untuk memudahkan *programme*r dalam mengeksekusi pengujian untuk memanggil step-step *scenario* pengujian yang telah dibuat[14]. Dengan menjelaskan kondisi awal yang terjadi, *action* yang dilakukan dan hasil yang diharapkan[15].

4.2.3 Hasil Implementasi *Automation Testing*

Hasil pengujian otomatis untuk aplikasi *Balink* akan ditampilkan dalam *Serenity Report*. Laporan *Serenity* ini mencatat hasil pengujian, termasuk informasi tentang *scenario* yang lulus atau tidak. Selain itu, laporan ini juga mendokumentasikan bagaimana fitur-fitur diuji dan apa yang dilakukan oleh aplikasi.

Tabel 4.6 Hasil Automation Testing Aplikasi Balink

ID	Pengujian	Deskripsi	Hasil
TCEVENT01	Pengguna dapat melihat semua daftar event.	Melihat seluruh daftar data event pada halaman event dengan mengklik menu event.	Berhasil dieksekusi
TCEVENT02	Pengguna dapat melihat lebih detail data dari salah satu event	Melihat data event lebih detail dengan mengklik salah satu event yang tersedia	Berhasil dieksekusi
TCCATEGORY01	Pengguna dapat melihat seluruh kategori produk.	Melihat seluruh daftar kategori yang tersedia.	Berhasil dieksekusi
TCCATEGORY02	Pengguna dapat melihat detail kategori produk pakaian.	Melihat data produk pakaian lebih detail dengan mengklik salah satu produknya.	Berhasil dieksekusi
TCCATEGORY03	Pengguna dapay melihat detail kategori produk tas.	Melihat detail produk tas dengan mengklik salah satu tas yang ada.	Berhasil dieksekusi
TCCATEGORY04	Pengguna dapat melihat detail dari	Melihat detail produk kerajinan tangan dengan	Berhasil dieksekusi

	kategori produk kerajinan tangan.	mengklik salah satu kerajinan.	
TCTRANSAKSI01	Pengguna dapat melihat seluruh histori transaksi.	Melihat seluruh data transaksi dengan mengklik fitur transaksi.	Berhasil dieksekusi
TCTRANSAKSI02	Pengguna dapat melihat histori transaksi event.	Melihat lebih detail salah satu transaksi event yang telah dilakukan.	Berhasil dieksekusi
TCARTIKEL01	Pengguna dapat melihat seluruh artikel pada halaman artikel.	Melihat seluruh daftar data artikel pada halaman artikel dengan mengklik menu artikel.	Berhasil dieksekusi
TCARTIKEL02	Pengguna dapat melihat lebih detail data dari salah satu artikel.	Melihat lebih detail artikel dengan mengklik salah satu artikel yang tersedia.	Berhasil dieksekusi

Tabel 4.6 menunjukkan hasil pengujian *automation* aplikasi *Balink* menggunakan *tools Appium*. Dari hasil pengujian 10 *test case*, tidak ditemukan *bug*, dan semuanya berjalan sesuai dengan rancangan atau harapan. Laporan tersebut menunjukkan bahwa 10 *test case* berstatus "passing" dengan persentase 100% dalam pengujian.

Dari hasil diatas menunjukkan pengujian 10 *test cas*e telah dijalankan tanpa adanya *bug*, dan semuanya berjalan sesuai dengan rancangan atau harapan. Hal ini dapat diartikan bahwa semua *scenario* pengujian berjalan sesuai dengan yang diinginkan dan semua *script automation* berhasil diimplementasikan pada pengujian aplikasi *Balink*.

Dari hasil pengujian automation performa aplikasi *Balink* yang tercatat, dapat disimpulkan bahwa seluruh pengujian berjalan lancar tanpa menemukan

bug. Selain itu, aplikasi *Balink* juga mampu mempertahankan performanya dengan menyelesaikan setiap *test* dalam rentang waktu yang sesuai. Hal ini memberikan keyakinan bahwa aplikasi tersebut dapat beroperasi secara optimal dalam lingkungan produksi dengan waktu *respons* yang cepat.

4.2.4 Evaluasi Hasil Pengujian

Setelah menyelesaikan tahap implementasi pengujian otomatis pada antarmuka pengguna aplikasi Balink dan mengevaluasi hasilnya, langkah selanjutnya adalah melakukan evaluasi keseluruhan hasil pengujian. Evaluasi ini bertujuan untuk menilai sejauh mana aplikasi yang telah dikembangkan sesuai dengan harapan. Dalam evaluasi ini, setiap fitur yang telah direncanakan akan diuji sesuai dengan jadwal pengujian yang telah ditentukan.

Dari hasil pengujian yang telah dilakukan menunjukkan semua *scenario* berhasil dijalakan dengan persentase 100% dan dapat menguji beberapa fitur yang ada. Dapat diartikan bahwa pengujian aplikasi *Balink* ini telah menguji semua test case dari total test case 10 yang telah dibuat, 10/10 x 100% = 100% dengan tidak ada masalah yang terdeteksi selama pengujian berhasil di execution. Kualitas aplikasi *Balink* dapat diandalkan sesuai harapan dengan efisiensi waktu yang lebih cepat karena menggunakan *automation testing* dan aplikasi Balink dapat beroperasi secara optimal tanpa khawatin akan masalah dan *bug*.

BAB V
KESIMPULAN DAN SARAN

Pada bab kelima ini, akan diuraikan kesimpulan-kesimpulan berdasarkan penelitian yang telah dilakukan mulai dari metodologi penelitian pengujian (bab ketiga) hingga hasil implementasi dan evaluasi pengujian (bab keempat). Selain itu, bab ini juga berisi saran-saran terkait kekurangan penelitian yang telah dilakukan serta rekomendasi untuk penelitian selanjutnya yang perlu dianalisis atau diteliti lebih lanjut.

5.1 Kesimpulan

Dari penelitian yang telah dilakukan dapat diambil beberapa kesimpulan yaitu:

1. Dalam penelitian ini, penulis menggali cara membuat *test script* pada pengujian aplikasi *Balink*. *Test script* ditulis terstruktur dalam beberapa folder seperti *feature* untuk menyimpan *test case* pengujian, *page object* untuk menyimpan elemen-elemen aplikasi yang akan dilakukan pengujian dan *stepdefinitions* yang digunakan untuk memanggil setiap step pengujian serta *androidDriverPool* untuk menghubungkan antara kode program dengan *emulator* aplikasi. Hasilnya menunjukkan bahwa implementasi *test script* dapat dilakukan dengan baik.
2. Proses *execution test script* pada pengujian ini berjalan lancar dan sesuai harapan dengan tidak ada bug pada aplikasi. Dengan menggunakan *Appium* sebagai *tools* pengujiannya dan *Appium Inspector* untuk inspeksi elemen UI pada aplikasi *Balink* dan membantu dalam mengidentifikasi elemen yang akan diotomatisasi.
3. Berdasarkan hasil pengujian, pengujian test script *automation* terhadap implementasi pengujian aplikasi mobile Android mencapai tingkat keberhasilan 100%. Ini menandakan bahwa semua test case yang dibuat berhasil dieksekusi.

5.2 Saran

Dalam pengujian automation aplikasi *Balink* ini masih terdapat kekurangan yang perlu diperbaiki untuk pengujian aplikasi selanjutnya:

1. Penulis berharap penelitian selanjutnya dapat mengembangkan kode program automation testing menjadi lebih mudah digunakan tanpa harus menulis kodenya kembali.
2. Melakukan analisis lebih mendalam terkait performa dan efisiensi pengujian aplikasi dengan menggunakan *tools Appium*. Seperti analisis waktu pembuatan *test script automation* untuk setiap skenario pengujian. Bandingkan antara pengujian manual tanpa *test script* dengan pengujian otomatis menggunakan *test script automation*.

DAFTAR PUSTAKA

[1] W. Nur Cholifah and S. Melati Sagita, "Pengujian Black Box Testing Pada Aplikasi Action & Strategy Berbasis Android Dengan Teknologi Phonegap," *Jurnal String*, vol. 3, no. 2, 2018.

[2] A. Fahrezi, F. N. Salam, G. M. Ibrahim, R. R. Syaiful, and A. Saifudin, "Pengujian Black Box Testing pada Aplikasi Inventori Barang Berbasis Web di PT. AINO Indonesia," *Jurnal Ilmu Komputer dan Pendidikan*, vol. 1, no. 1, pp. 1–5, 2022.

[3] J. Wang and J. Wu, "Research on mobile application automation testing technology based on appium," in *Proceedings - 2019 International Conference on Virtual Reality and Intelligent Systems, ICVRIS 2019*, Institute of Electrical and Electronics Engineers Inc., Sep. 2019, pp. 247–250. doi: 10.1109/ICVRIS.2019.00068.

[4] A. R. Rambe and H. Prihantoro, "Pengujian Otomatis Aplikasi Mobile dengan Teknik Black-box Menggunakan Appium (Studi Kasus: Pengembangan Aplikasi Jala Mobile)," 2022.

[5] M. Anugraha, A. E. Tjandra, and R. P. Kristianto, "Pengujian Kualitas Perangkat Lunak menggunakan Pendekatan Manual dan Automation Testing pada Aplikasi 'Nuzantara' Menggunakan Katalon Software Quality Testing through Manual and Automation Testing Approaches on the 'Nuzantara' Application with Katalon," 2024.

[6] J. Lian Min, A. Istiqomah, A. Rahmani, P. Negeri Bandung, and P. P. Tester Padepokan Tujuh Sembilan-Bandung, "Evaluasi Penggunaan Manual Dan Automated Software Testing Pada Pelaksanaan End-To-End Testing," *Jurnal Teknologi Terapan)* |, vol. 6, no. 1, 2020.

[7] L. Maximillian, R. P. Kristianto, and D. Cendika, "Pengujian Aplikasi Mobile Web Wisata Bandung Menggunakan Blackbox Testing dan Katalon Berbasis Manual dan Automation Testing Testing the Bandung Tourism Mobile Web Application Using Blackbox Testing and Katalon Based on Manual and Automation Testing," 2024.

[8] B. D. Saputra and A. Stefanie, "Automation Testing Api, Android, dan Website Menggunakan Serenity Bdd Pada Software Sistem Manajemen Rumah Sakit," *Jurnal Ilmiah Wahana Pendidikan, Mei*, vol. 9, no. 10, pp. 114–126, 2023, doi: 10.5281/zenodo.7983405.

[9] U. Sa'adah *et al.*, "Framework Testing Otomatis Berbasis Serenity Jenkins Automated Build," *Jurnal Ilmiah Teknologi Informasi*, vol. 19, no. 2, pp. 102–110, 2021.

[10] M. M. Panjaitan and I. Mantra, *Pembangunan Framework Web Automation Testing Menggunakan Serenity Bdd Pada Studi Kasus Aplikasi Supply Chain*, vol. 28, no. 2020. 2020.

[11] M. Nur Ichsanudin, M. Yusuf, S. Jurusan Rekayasa Sistem Komputer, J. Teknik Industri, I. AKPRIND Yogyakarta, and R. Artikel, "Pengujian Fungsional Perangkat Lunak Sistem Informasi Perpustakaan Dengan Metode Black Box Testing Bagi Pemula," vol. 1, no. 2, pp. 1–8, 2022, doi: 10.55123.

[12] S. Kantun, "Penelitian Evaluatif Sebagai Salah Satu Model Penelitian Dalam Bidang Pendidikan (Suatu Kajian Konseptual)," *Jurnal Ilmiah Ilmu Pendidikan, Ilmu Ekonomi dan Ilmu Sosial*, vol. 10, no. 2, 2017.

[13] Supriyono, "Software Testing with the approach of Blackbox Testing on the Academic Information System," *International Journal of Information System & Technology*, vol. 3, no. 2, pp. 227–233, 2020.

[14] F. Hudoyo and A. Andrianingsih, "Implementasi Web Service Pada Sistem Informasi Geografis (SIG) Peta Sebaran Data Covid-19 Berbasis Mobile Apps," vol. 8, no. 3, pp. 1278–1293, 2021, [Online]. Available: http://jurnal.mdp.ac.id

[15] R. P. Adi, Y. Koswara, J. Tashika, Y. Devi, and A. Saifudin, "Pengujian Black Box pada Aplikasi Pertokoan Minimarket Menggunakan Metode Equivalence Partitioning," *Jurnal Teknologi Sistem Informasi dan Aplikasi*, vol. 3, no. 2, p. 100, Apr. 2020, doi: 10.32493/jtsi.v3i2.4695.

STT - NF