BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Seperti yang sudah dijelaskan sebelumnya, berlangsungnya magang ini ditempatkan di direktorat *Information Technology* tepatnya di departemen *Quality Assurance Engineer* untuk posisi *Quality Assurance Internship*. Melakukan magang sebagai *Quality Assurance* di perusahaan ini dibimbing oleh Pak Gustav Sri Raharjo yang menjabat sebagai *Full Stack Developer* di perusahaan ini. Selama melaksanakan magang di perusahaan ini, semua tugas didapat dari Pak Gustav dan *update* mengenai apa yang sudah dikerjakan juga kembali ke Pak Gustav sehingga Pak Gustav bertanggung jawab penuh selama keberlangsungan magang ini. Tugas yang diberikan oleh Pak Gustav adalah mengenai projek *UI Automation Testing* yang menjadi projek selama keberlangsungan magang di perusahaan ini.

3.2 Tugas yang Dilakukan

Berikut adalah tugas-tugas yang dilakukan selama pekerjaan yang dilakukan selama melaksanakan kerja magang sebagai *Quality Assurance* di PT Keindahan Sejahtera Hutama yang dilakukan selama 3 bulan.

- Membuat test case
- Melakukan *testing* manual
- Mengubah tampilan website admin UNNIS
- Membuat page influencer untuk website admin
- Membuat program testing otomatis

3.3 Uraian Pelaksanaan Magang

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama magang

Minggu Ke-	**Pekerjaan yang dilakukan**
1	• Melakukan manual testing aplikasi UNNIS • Membuat laporan terhadap *bugs* yang ditemukan
2	• Melakukan manual testing aplikasi UNNIS • Membuat laporan terhadap *bugs* yang ditemukan • Melaporkan *bugs* yang ditemukan ke *supervisor*
3	• Mengubah Tampilan (Front-End) Website Admin • Membuat page baru pada website admin
4	• Mengubah Tampilan (Front-End) Website Admin • Membuat page baru pada website admin

Lanjut pada halaman berikutnya

Tabel 3.1 Pekerjaan yang dilakukan tiap minggu selama magang (lanjutan)

Minggu Ke-	**Pekerjaan yang dilakukan**
5	• Melakukan manual testing aplikasi UNNIS • Membuat laporan terhadap *bugs* yang ditemukan • Melaporkan *bugs* yang ditemukan ke *supervisor*
6	• Mempelajari cara menggunakan Framework Appium menggunakan bahasa pemrograman Python • Mensetup enviroment Appium
7	• Mempelajari cara menggunakan Framework Appium menggunakan bahasa pemrograman Python • Mensetup enviroment Appium • Membuat *priority list* untuk function yang harus memiliki testing otomatis • Membagung program UI automation testing untuk aplikasi UNNIS
Lanjut pada halaman berikutnya	

Tabel 3.1 Pekerjaan yang dilakukan tiap minggu selama magang (lanjutan)

Minggu Ke-	**Pekerjaan yang dilakukan**
8	• Membuat *priority list* untuk function yang harus memiliki testing otomatis • Membagung program UI automation testing untuk aplikasi UNNIS
9	• Membuat *priority list* untuk function yang harus memiliki testing otomatis • Membagung program UI automation testing untuk aplikasi UNNIS
10	• Membuat *priority list* untuk function yang harus memiliki testing otomatis • Membagung program UI automation testing untuk aplikasi UNNIS
Lanjut pada halaman berikutnya	

Tabel 3.1 Pekerjaan yang dilakukan tiap minggu selama magang (lanjutan)

Minggu Ke-	Pekerjaan yang dilakukan
11	• Membuat *priority list* untuk function yang harus memiliki testing otomatis • Membagung program UI automation testing untuk aplikasi UNNIS • Melakukan manual testing aplikasi UNNIS • Membuat laporan terhadap *bugs* yang ditemukan • Melaporkan *bugs* yang ditemukan ke *supervisor*
12	• Membuat *priority list* untuk function yang harus memiliki testing otomatis • Membagung program UI automation testing untuk aplikasi UNNIS • Melakukan manual testing aplikasi UNNIS • Membuat laporan terhadap *bugs* yang ditemukan • Melaporkan *bugs* yang ditemukan ke *supervisor*

Berikut adalah uraian pekerjaan yang dilakukan selama melaksanakan praktik kerja magang sebagai *Quality Assurance* di PT Keindahan Sejahtera Hutama yang dilakukan selama 3 bulan yang diuraikan seperti pada Tabel 3.1.

Pada minggu pertama, tugas saya adalah melakukan pengujian manual terhadap aplikasi UNNIS dan membuat laporan bugs yang ditemukan. Selanjutnya, pada minggu kedua, saya melanjutkan tugas pengujian manual aplikasi UNNIS dan

melaporkan bugs kepada supervisor.

Pada minggu ketiga dan keempat, fokus saya beralih ke pekerjaan merancang ulang tampilan (Front-End) Website Admin dan membuat beberapa halaman baru pada website tersebut.

Minggu kelima kembali menuntut saya untuk melakukan pengujian manual aplikasi UNNIS, membuat laporan bugs, dan melaporkannya kepada supervisor.

Pada minggu keenam, saya mulai mempelajari penggunaan Framework Appium dengan bahasa pemrograman Python dan melakukan setup environment Appium.

Selanjutnya, pada minggu ketujuh hingga kesembilan, tugas utama saya adalah membuat daftar prioritas untuk fungsi-fungsi yang perlu diuji secara otomatis dan mengembangkan program UI automation testing untuk aplikasi UNNIS. Saya terus melakukan perbaikan dan penyempurnaan program tersebut.

Pada minggu kesepuluh hingga kedua belas, fokus saya adalah mengembangkan program UI automation testing untuk aplikasi UNNIS dan melengkapi laporan magang. Saya juga melakukan beberapa manual testing dan finalisasi kode program untuk memastikan semuanya berjalan dengan baik.

Secara keseluruhan, selama periode magang 12 minggu ini, saya telah melakukan pengujian manual, pengembangan tampilan website, pembuatan halaman baru, dan pengembangan program UI automation testing. Tugas-tugas ini membantu saya memperluas pemahaman saya tentang pengembangan aplikasi dan memberikan pengalaman berharga dalam pengujian otomatis menggunakan Framework Appium.

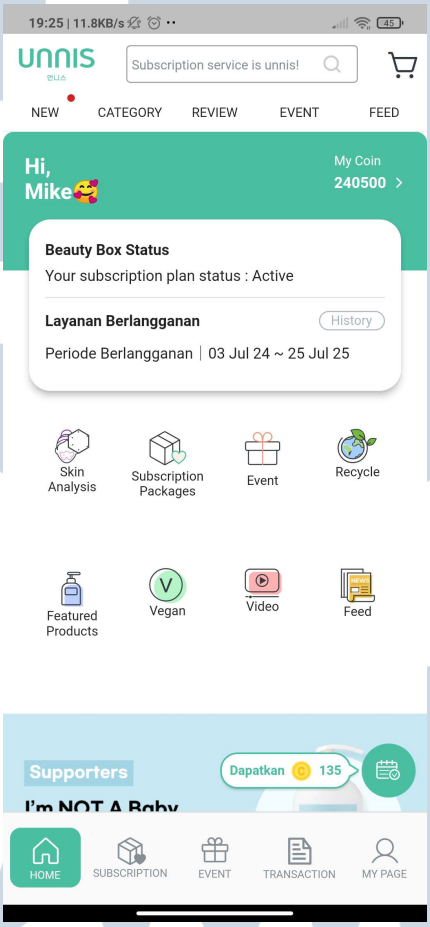
3.3.1 Gambaran umum aplikasi UNNIS

UNNIS adalah aplikasi mobile yang bertujuan untuk membantu pengguna menemukan produk *skin care* yang tepat untuk jenis kulit pengguna. Aplikasi ini memungkinkan pengguna untuk memeriksa jenis kulit pengguna dan menerima rekomendasi pribadi untuk produk perawatan kulit yang cocok untuk kulit pengguna. Ini bisa sangat membantu bagi individu yang tidak yakin dengan produk apa yang harus digunakan atau mencari produk baru untuk dicoba.

Pengguna dapat dengan mudah memasukkan informasi pengguna dan menerima rekomendasi secara langsung. Selain itu, aplikasi ini juga memungkinkan pengguna untuk membeli produk yang direkomendasikan langsung melalui aplikasi, membuat proses menemukan dan membeli produk perawatan kulit menjadi

lebih mudah.

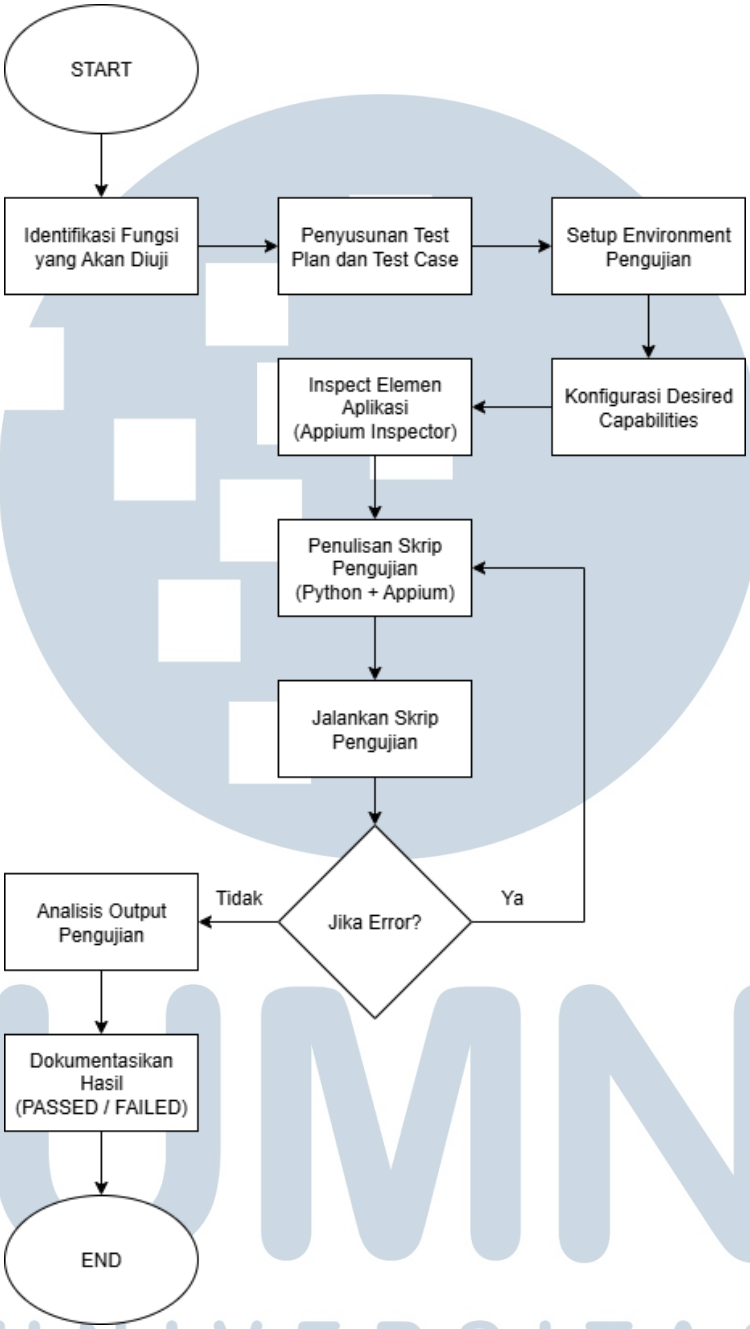
Secara keseluruhan, UNNIS adalah aplikasi yang dapat digunakan oleh siapa saja yang ingin meningkatkan rutinitas perawatan kulit pengguna. Dengan memberikan rekomendasi pribadi dan platform yang mudah digunakan, UNNIS membuatnya sederhana bagi pengguna untuk menemukan produk yang tepat untuk jenis kulit pengguna.



Gambar 3.1. Halaman utama UNNIS

3.3.2 Proses Perancangan Sistem Pengujian Otomatis

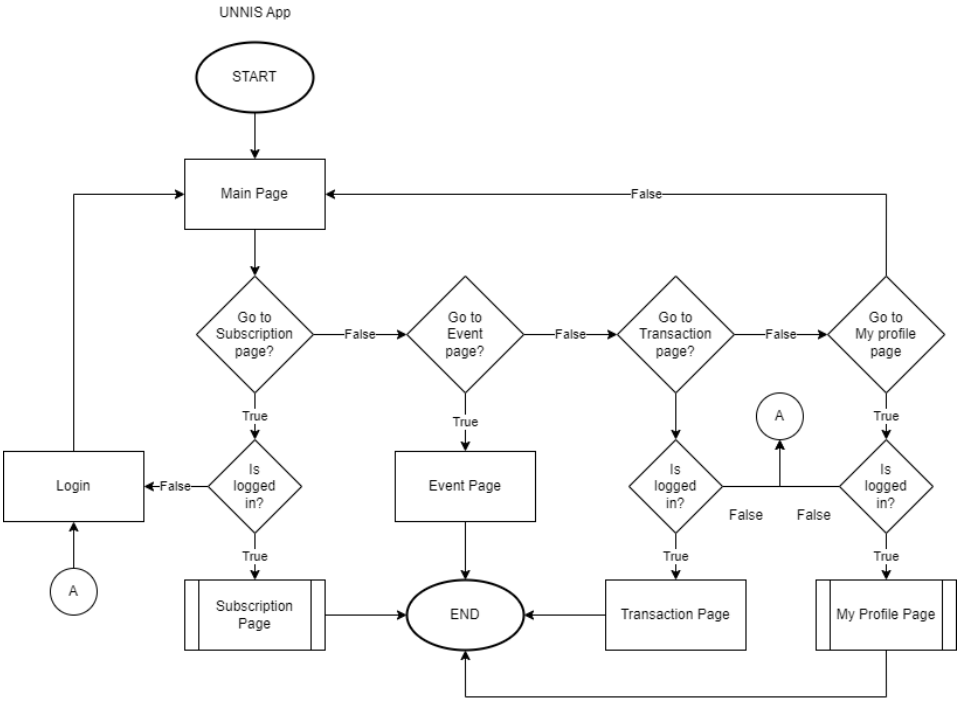
Dalam mengembangkan sistem pengujian otomatis untuk aplikasi UNNIS, proses perancangan dilakukan secara bertahap dimulai dari identifikasi kebutuhan pengujian hingga implementasi skrip Appium. Tahapan ini dirancang agar pengujian berjalan efisien dan sesuai dengan standar QA yang berlaku di industri perangkat lunak. Berikut tahapan lengkap yang dilakukan:



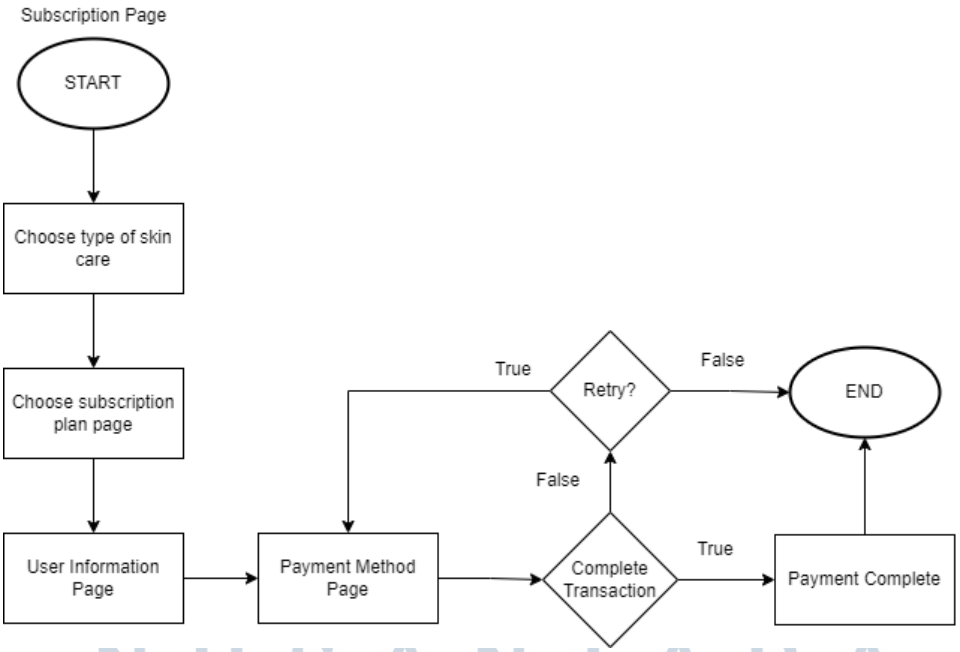
Gambar 3.2. Flowchart Proses Pengujian Otomatis Umum Menggunakan Appium

3.3.3 Flowchart function subscribe

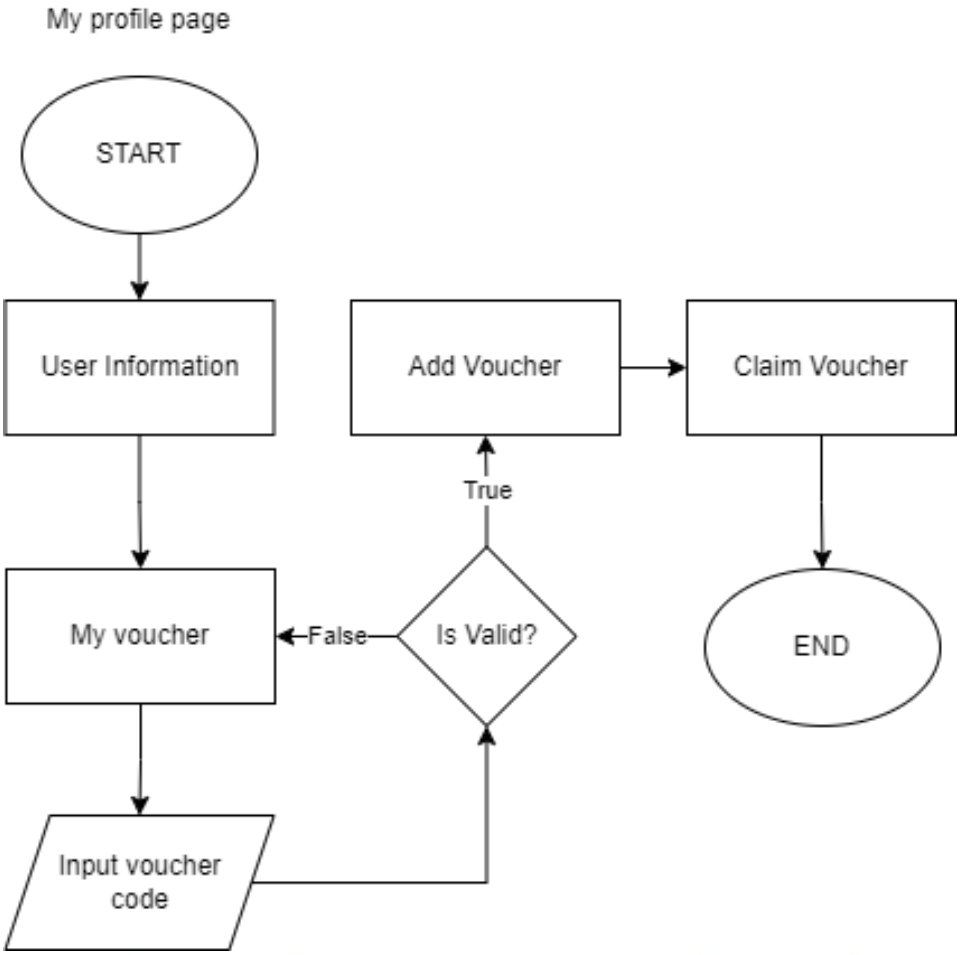
Hal pertama yang harus dilakukan adalah membuat flow testingnya, berikut adalah flowchart aplikasi UNNIS dan flowchart *subscription function* dan *claim voucher function*.



Gambar 3.3. Flowchart UNNIS



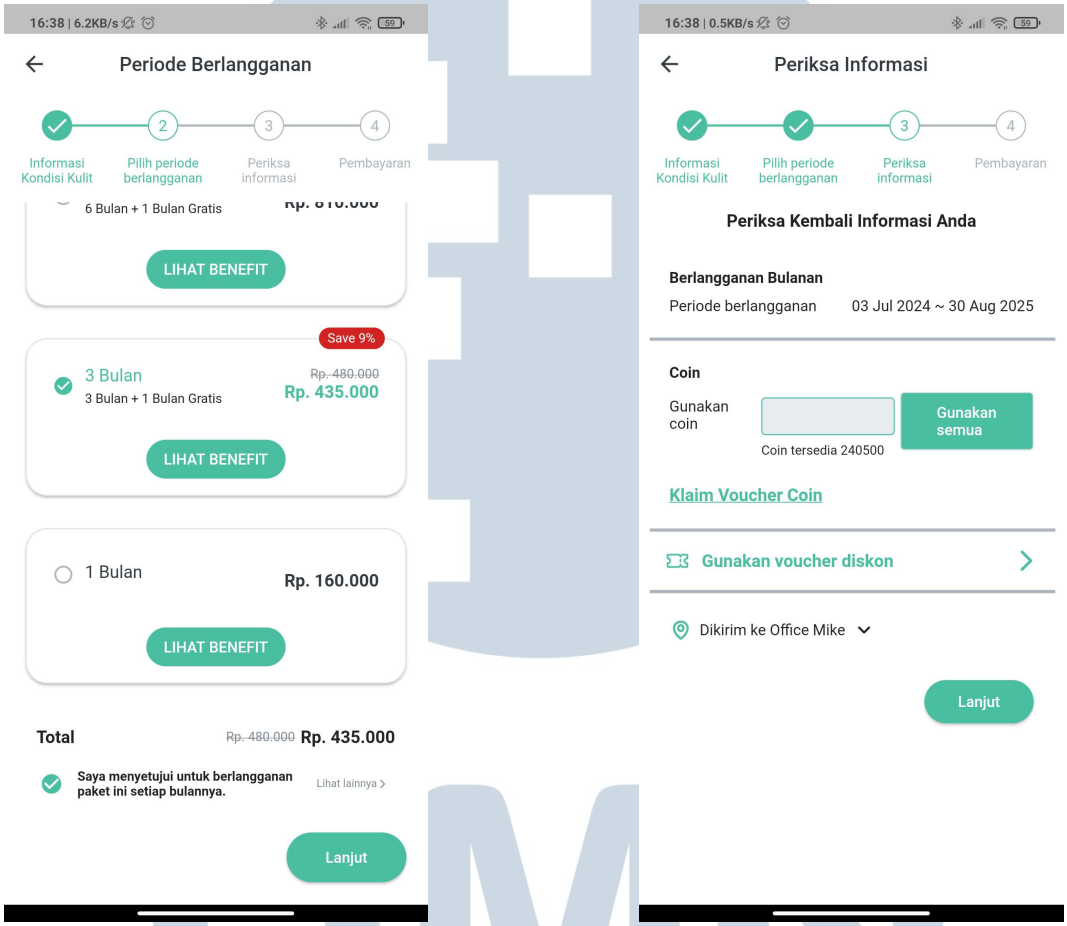
Gambar 3.4. Flowchart Subscribe



Gambar 3.5. Flowchart Voucher

3.3.4 Screenshot test

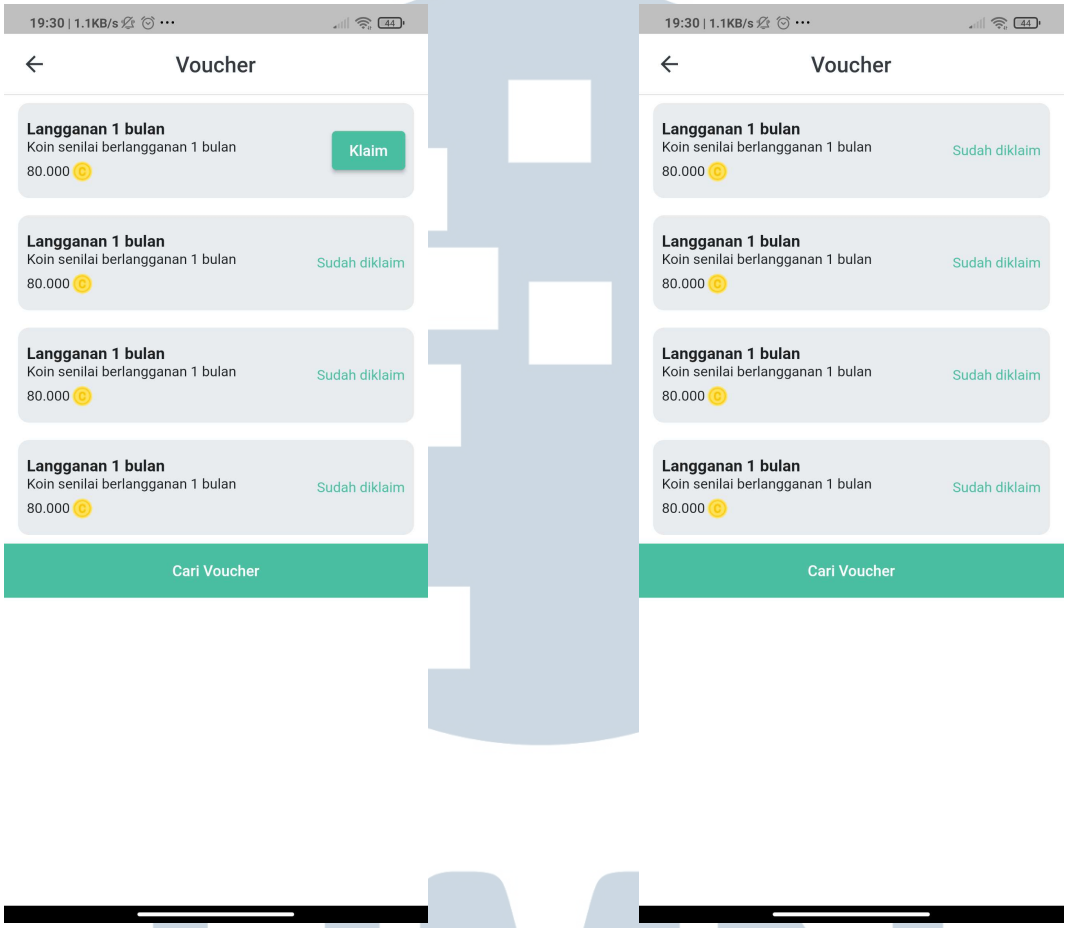
A Subscription



(a) Subsciption plan

(b) User Information

B Voucher



(a) Unclaimed Voucher (b) Claimed voucher

3.3.5 Proses pembuatan kode otomatis

A Appium Server

Sebelum membuat program testing, perlu dijalankan server appium agar dapat berkomunikasi dengan *android device*.

```
C:\Users\xaver>Appium
[Appium] Welcome to Appium v1.22.3
[Appium] Appium REST http interface listener started on 0.0.0.0:4723
```

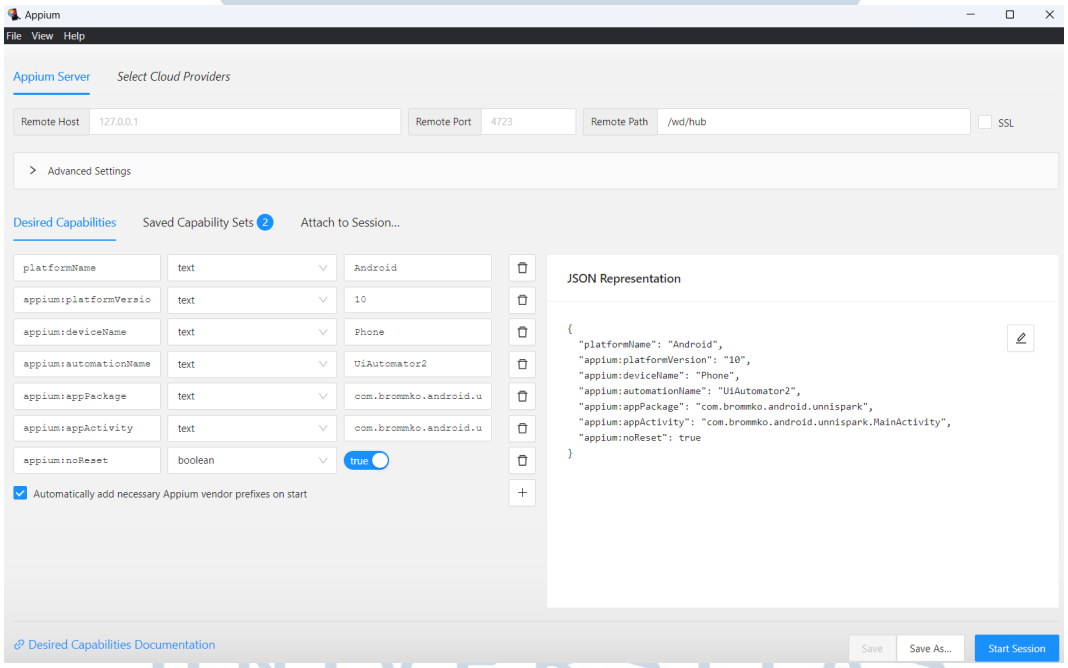
Gambar 3.8. Gambar Menjalankan Server

Setelah menjalankan server seperti yang terlihat pada Gambar 3.8, langkah selanjutnya adalah membuka Appium Inspector untuk mengatur dan menyesuaikan

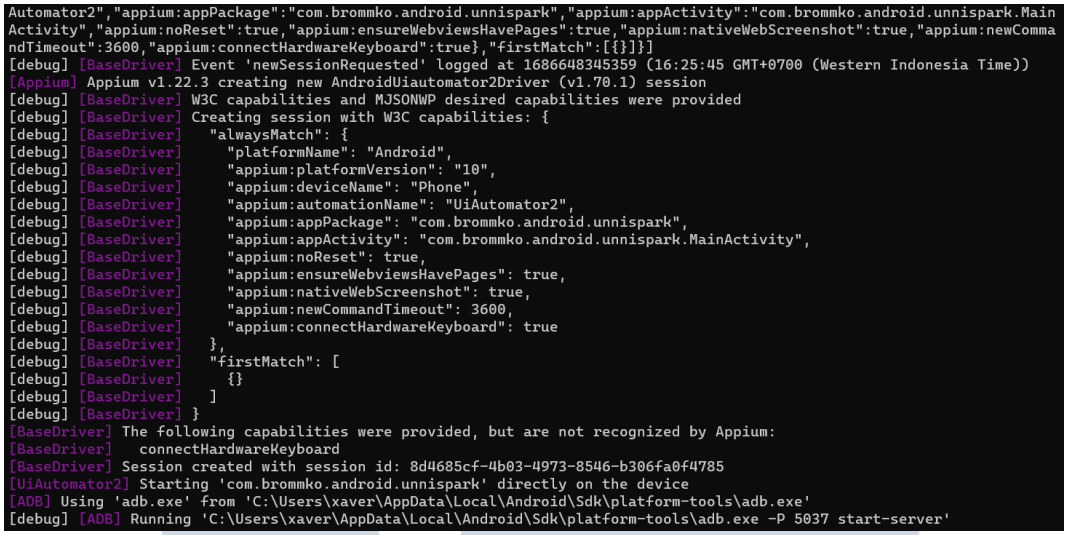
desired capabilities sesuai dengan perangkat yang digunakan dan kebutuhan pengujian. Pada Appium Inspector, pengguna dapat mengisi nilai-nilai yang diinginkan untuk *desired capabilities* seperti ”platformName”, ”deviceName”, ”platformVersion”, ”automationName”, dan lain-lain. Pengisian informasi ini penting agar program dapat berkomunikasi dengan perangkat yang akan diuji.

B Inspecting elements

Setelah menjalankan server seperti yang terlihat pada Gambar 3.8, langkah selanjutnya adalah membuka Appium Inspector untuk melakukan pengaturan dan penyesuaian pada *desired capabilities* sesuai kebutuhan. Appium Inspector merupakan alat yang memungkinkan pengguna untuk dengan mudah mengatur konfigurasi pengujian dengan memasukkan nilai-nilai yang diinginkan pada *desired capabilities*.



Gambar 3.9. *Appium Inspector*



Gambar 3.10. *Connecting To Device*

Proses penggunaan *Appium Inspector* dimulai dengan langkah-langkah berikut. Pertama, pastikan telah menginstal dan mengonfigurasi lingkungan *Appium* pada perangkat yang digunakan. Selanjutnya, jalankan *Appium Server* dan pastikan server berjalan dengan baik. Setelah itu, buka *Appium Inspector*.

Appium Inspector adalah alat yang berguna dalam pengujian aplikasi mobile otomatis. Dalam konteks ini, *Appium Inspector* memungkinkan pengembang dan pengujian untuk menelusuri struktur elemen dalam aplikasi yang sedang diuji. Hal ini memungkinkan untuk mengeksplorasi, menganalisis, dan memodifikasi elemen-elemen ini.

Ketika *Appium Inspector* terbuka, pengaturan dasar seperti alamat IP server *Appium*, nomor port, dan pilihan perangkat atau emulator yang digunakan untuk pengujian perlu diatur. Setelah itu, sesi pengujian dapat dimulai dengan mengklik tombol *"Start Inspector Session"*.

Appium Inspector akan membuka aplikasi yang ingin diuji pada perangkat yang telah dipilih. Ini akan menampilkan antarmuka pengguna aplikasi secara real-time bersama dengan panel *"Element Tree"*. Panel ini menampilkan hierarki elemen-elemen dalam aplikasi, mulai dari elemen utama hingga elemen anak.

Untuk melakukan inspeksi elemen, berbagai alat yang disediakan oleh *Appium Inspector* dapat digunakan. Elemen tertentu pada antarmuka aplikasi dapat disorot untuk melihat properti dan atributnya di panel samping. Tombol *"Inspect"* juga dapat diklik untuk memilih elemen secara langsung pada antarmuka aplikasi, dan panel akan menyoroti elemen yang dipilih serta menampilkan informasi terkait.

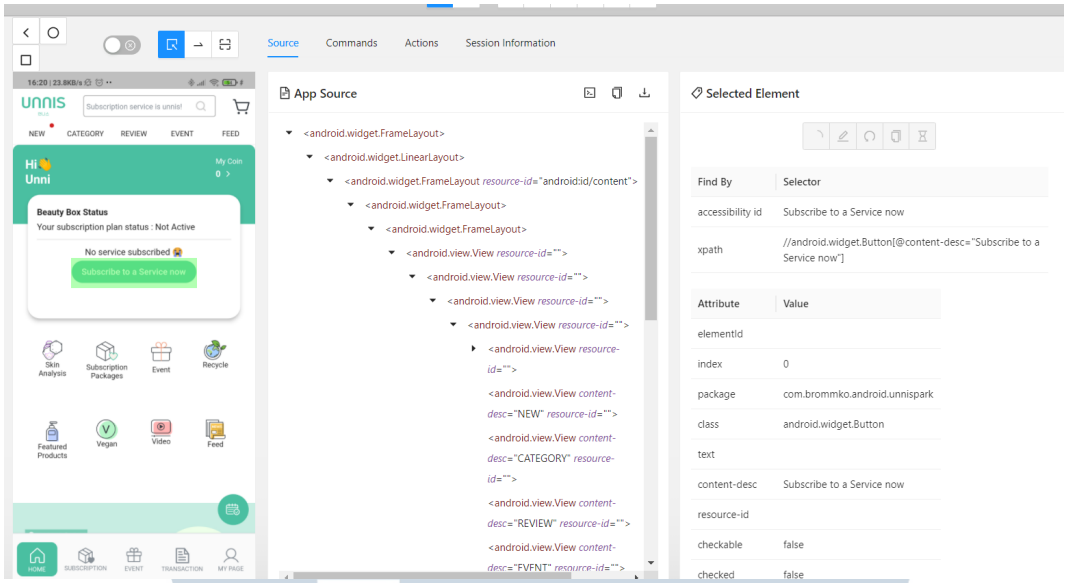
Dalam proses identifikasi elemen, Appium Inspector memberikan informasi tentang atribut-atribut yang unik untuk setiap elemen pada aplikasi mobile. Atribut-atribut ini dapat digunakan sebagai referensi untuk mengenali elemen yang ingin diinteraksikan atau diuji, seperti tombol ”Login” atau bidang masukan untuk mengisi formulir.

Sebagai contoh, seseorang dapat menyoroti tombol ”Login” pada layar aplikasi menggunakan Appium Inspector. Setelah itu, Inspector akan menampilkan informasi mengenai atribut-atribut tombol tersebut, seperti ID, nama kelas, dan nilai. Atribut-atribut ini dapat digunakan dalam skrip pengujian untuk mengidentifikasi dan berinteraksi dengan elemen tersebut. Misalnya, ID atau nama kelas tombol ”Login” dapat digunakan sebagai selektor dalam skrip pengujian untuk mengklik tombol tersebut.

Dengan menggunakan Appium Inspector, pengguna dapat dengan mudah menemukan atribut-atribut yang unik dari setiap elemen pada aplikasi mobile. Ini memudahkan dalam proses identifikasi dan interaksi dengan elemen-elemen tersebut dalam skenario pengujian atau dalam skrip pengujian otomatis.

Kegunaan dari *Appium Inspector* adalah mempermudah proses pengujian aplikasi mobile secara otomatis. *Appium Inspector* memungkinkan pengembang dan pengujian untuk dengan mudah mengidentifikasi elemen dalam aplikasi, mengakses, memodifikasi, serta melakukan tindakan seperti klik atau pengisian formulir secara otomatis.

Dalam pengembangan dan pengujian aplikasi mobile yang efisien, *Appium Inspector* menjadi alat yang penting. Ia menyediakan kemampuan untuk menganalisis elemen-elemen dalam aplikasi secara visual, mengurangi waktu yang dibutuhkan untuk menemukan dan memahami struktur elemen, serta mempercepat proses pembuatan dan debugging skrip pengujian otomatis.



Gambar 3.11. *Inspecting Elements*

C Import Libraries

Kode di bawah merupakan contoh penggunaan library dan modul dalam bahasa Python untuk mengautomasi pengujian aplikasi mobile menggunakan Appium dan Selenium WebDriver. Tujuan dari kode tersebut adalah untuk mengimplementasikan tindakan-tindakan pengujian otomatis pada aplikasi mobile.

```
from appium import webdriver
from appium.webdriver.common.mobileby import MobileBy as AppiumBy
import time
from selenium.webdriver import ActionChains
from selenium.webdriver.common.actions import interaction
from selenium.webdriver.common.actions.action_builder import ActionBuilder
from selenium.webdriver.common.actions.pointer_input import PointerInput
```

Gambar 3.12. *Import*

Pada baris pertama, dilakukan impor library ”webdriver” dari modul ”appium” dan ”MobileBy” dari modul ”appium.webdriver.common.mobileby as AppiumBy”. Hal ini dilakukan untuk mengakses kelas-kelas dan fungsi-fungsi yang diperlukan dari library Appium.

Kemudian, pada baris kedua, dilakukan impor modul ”time”. Modul ini digunakan untuk mengatur waktu jeda antara tindakan-tindakan dalam pengujian.

Selanjutnya, pada baris ketiga, dilakukan impor kelas ”ActionChains” dari modul ”selenium.webdriver”. Kelas ini menyediakan metode-metode untuk melakukan tindakan-tindakan seperti klik, geser, dan sebagainya pada elemen-elemen aplikasi.

Baris keempat dan kelima merupakan impor modul-modul dan kelas-kelas tambahan yang digunakan dalam pengujian, seperti ”interaction”, ”ActionBuilder”, dan ”PointerInput”. Modul-modul dan kelas-kelas ini menyediakan fungsionalitas tambahan untuk melakukan tindakan-tindakan kompleks dalam pengujian.

Secara keseluruhan, kode di atas menggabungkan library Appium dan Selenium WebDriver, serta beberapa modul tambahan, untuk melakukan pengujian otomatis pada aplikasi mobile. Dengan menggunakan kode ini, pengguna dapat mengimplementasikan serangkaian tindakan pengujian seperti klik, geser, atau interaksi lainnya pada elemen-elemen aplikasi. Kode ini menjadi salah satu alat yang berguna dalam melaksanakan pengujian otomatis secara efisien dan terstruktur pada aplikasi mobile.

D Desired Capabilities

Kode di bawah adalah contoh pengaturan ”desired capabilities” untuk server Appium. ”Desired capabilities” adalah kumpulan pengaturan yang digunakan untuk mengkonfigurasi pengujian pada aplikasi mobile dengan menggunakan Appium. Dalam contoh ini, pengaturan tersebut ditentukan dalam bentuk kamus (dictionary) yang disebut *desired cap*.

```
# Set up the desired capabilities for the Appium server
desired_cap = {
    "platformName": "Android",
    "appium:platformVersion": "10",
    "appium:deviceName": "Phone",
    "appium:automationName": "UiAutomator2",
    "appium:appPackage": "com.brommko.android.unnispark",
    "appium:appActivity": "com.brommko.android.unnispark.MainActivity",
    "noReset": True
}
```

Gambar 3.13. *Desired Cap*

Beberapa pengaturan yang ditentukan dalam `desired_cap` meliputi:

- `"platformName": "Android"`: Menentukan bahwa platform yang akan diuji adalah Android.
- `"appium:platformVersion": "10"`: Menentukan versi Android yang digunakan pada perangkat yang akan diuji.
- `"appium:deviceName": "Phone"`: Menentukan nama perangkat yang akan diuji.
- `"appium:automationName": "UiAutomator2"`: Menentukan alat otomasi yang akan digunakan, dalam hal ini adalah UiAutomator2.
- `"appium:appPackage": "com.brommko.android.unnispark"`: Menentukan paket aplikasi yang akan diuji.
- `"appium:appActivity": "com.brommko.android.unnispark.MainActivity"`: Menentukan aktivitas (activity) utama dari aplikasi yang akan diuji.
- `"noReset": True`: Menentukan bahwa data aplikasi tidak akan di-reset sebelum setiap pengujian.

Pengaturan ”desired capabilities” ini digunakan untuk memberikan informasi penting kepada server Appium mengenai perangkat, aplikasi, dan pengaturan pengujian yang akan dilakukan. Dengan menyediakan pengaturan yang akurat, server Appium dapat mempersiapkan lingkungan yang sesuai untuk menjalankan pengujian pada aplikasi mobile yang ditargetkan.

Kode ini menunjukkan pemahaman dan implementasi pengaturan ”desired capabilities” dalam pengembangan dan pengujian aplikasi mobile menggunakan Appium. Pengaturan ini membantu dalam menghubungkan dan mengkonfigurasi server Appium dengan perangkat dan aplikasi yang akan diuji, memastikan bahwa pengujian dapat dilakukan dengan benar dan konsisten.

E *Connecting To Server*

Kode di bawah digunakan untuk melakukan koneksi ke server Appium. Pada baris tersebut, kita membuat objek `driver` yang merupakan instance dari kelas `webdriver.Remote`. Koneksi ke server Appium dilakukan dengan menggunakan

URL `http://localhost:4723/wd/hub` dan pengaturan ”desired capabilities” yang telah ditentukan sebelumnya dalam variabel `desired_cap`.

```
# Connect to the Appium server
driver = webdriver.Remote("http://localhost:4723/wd/hub", desired_cap)
```

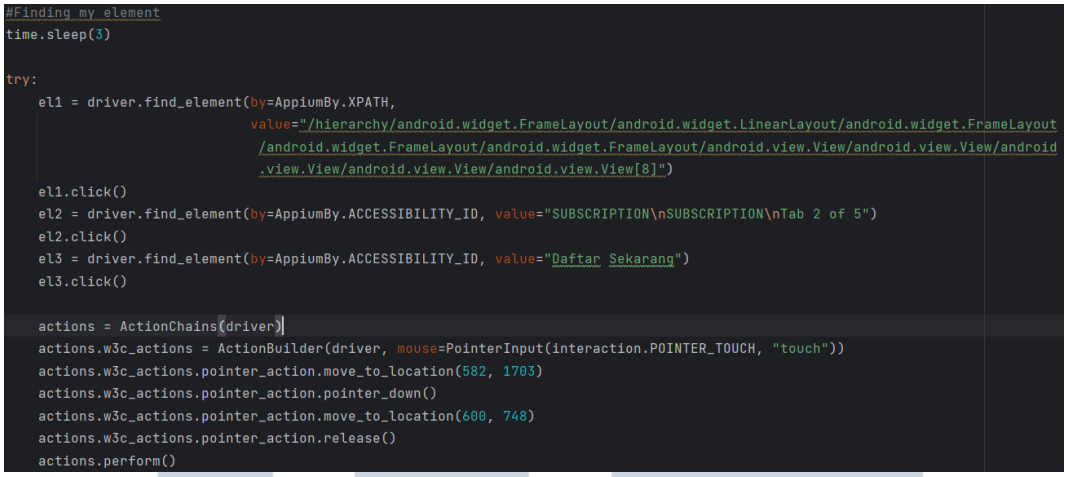
Gambar 3.14. *Connect to server*

Melalui koneksi ini, kita dapat mengendalikan perangkat mobile yang akan diuji melalui server Appium. Objek `driver` akan menjadi titik akses kita untuk melakukan berbagai tindakan dan pengujian pada aplikasi mobile.

Dalam konteks laporan magang, penggunaan kode ini menunjukkan langkah awal untuk menghubungkan dan memulai sesi dengan server Appium. Dengan melakukan koneksi ke server, kita dapat mengontrol perangkat mobile yang akan diuji dan menjalankan tindakan-tindakan pengujian pada aplikasi mobile yang ditargetkan. Kode ini merupakan langkah penting dalam proses pengujian otomatis aplikasi mobile menggunakan Appium dan akan menjadi dasar untuk langkah-langkah pengujian yang lebih lanjut.

F Implementasi Kode

Kode di bawah merupakan contoh dari serangkaian tindakan untuk menemukan dan berinteraksi dengan elemen-elemen dalam aplikasi mobile menggunakan driver Appium. Pada awalnya, dilakukan penundaan waktu selama 3 detik menggunakan `time.sleep(3)` untuk memberikan waktu bagi elemen-elemen dalam aplikasi untuk dimuat sepenuhnya.



```
#Finding my element
time.sleep(3)

try:
    el1 = driver.find_element(by=AppiumBy.XPATH,
                              value="/hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout
                               /android.widget.FrameLayout/android.widget.FrameLayout/android.view.View/android.view.View/android
                               .view.View/android.view.View/android.view.View[8]")
    el1.click()
    el2 = driver.find_element(by=AppiumBy.ACCESSIBILITY_ID, value="SUBSCRIPTION\nSUBSCRIPTION\nTab 2 of 5")
    el2.click()
    el3 = driver.find_element(by=AppiumBy.ACCESSIBILITY_ID, value="Daftar Sekarang")
    el3.click()

    actions = ActionChains(driver)
    actions.w3c_actions = ActionBuilder(driver, mouse=PointerInput(interaction.POINTER_TOUCH, "touch"))
    actions.w3c_actions.pointer_action.move_to_location(582, 1703)
    actions.w3c_actions.pointer_action.pointer_down()
    actions.w3c_actions.pointer_action.move_to_location(600, 748)
    actions.w3c_actions.pointer_action.release()
    actions.perform()
```

Gambar 3.15. *Subscription 1*

Selanjutnya, dilakukan pencarian elemen menggunakan metode `driver.find_element()` dengan menggunakan berbagai strategi pencarian, seperti XPath dan Accessibility ID. Setelah elemen ditemukan, dilakukan tindakan seperti klik pada elemen menggunakan metode `click()`.

Selanjutnya, digunakan objek `actions` dari kelas `ActionChains` untuk melakukan tindakan penggeseran (swipe) pada elemen. Penggeseran ini dilakukan dengan menggunakan metode `move_to_location()`, `pointer_down()`, `move_to_location()`, dan `release()` dari objek `actions`.

Kemudian, dilakukan pencarian elemen lagi dan tindakan klik pada elemen tersebut. Setelah itu, dilakukan tindakan penggeseran (swipe) lagi dengan menggunakan objek `actions`.

```
el4 = driver.find_element(by=AppiumBy.ACCESSIBILITY_ID, value="Lanjut")
el4.click()
el5 = driver.find_element(by=AppiumBy.ACCESSIBILITY_ID,
                          value="12 Month\n12 Month + 2 Month Free\nRp. 1.920.000\nRp. 1.188.000")
el5.click()

actions = ActionChains(driver)
actions.w3c_actions = ActionBuilder(driver, mouse=PointerInput(interaction.POINTER_TOUCH, "touch"))
actions.w3c_actions.pointer_action.move_to_location(582, 1703)
actions.w3c_actions.pointer_action.pointer_down()
actions.w3c_actions.pointer_action.move_to_location(600, 748)
actions.w3c_actions.pointer_action.release()
actions.perform()

el6 = driver.find_element(by=AppiumBy.XPATH,
                          value="//android.view.View[@content-desc=\"Saya menyetujui untuk berlangganan paket ini setiap
                           bulannya.\"]/android.widget.CheckBox")
el6.click()
el7 = driver.find_element(by=AppiumBy.ACCESSIBILITY_ID, value="Lanjut")
el7.click()
el8 = driver.find_element(by=AppiumBy.ACCESSIBILITY_ID, value="Lanjut")
el8.click()
```

Gambar 3.16. *Subscription 2*

Pada bagian kode di atas, terdapat serangkaian tindakan lanjutan untuk menemukan dan berinteraksi dengan elemen-elemen dalam aplikasi mobile menggunakan driver Appium.

Pertama, dilakukan pencarian elemen dengan menggunakan metode `find_element()` dan menggunakan strategi pencarian berdasarkan Accessibility ID atau XPath. Setelah elemen ditemukan, dilakukan tindakan klik pada elemen tersebut menggunakan metode `click()`.

Selanjutnya, digunakan objek `actions` dari kelas `ActionChains` untuk melakukan tindakan penggeseran (swipe) pada elemen. Tindakan penggeseran ini dilakukan dengan menggunakan metode-metode seperti `move_to_location()`, `pointer_down()`, `move_to_location()`, dan `release()` dari objek `actions`. Tujuannya adalah untuk menggerakkan pointer pada perangkat mobile ke posisi yang diinginkan.

Kemudian, dilakukan pencarian elemen lagi menggunakan metode `find_element()`. Beberapa elemen yang ditemukan akan diklik, sedangkan elemen lainnya seperti `CheckBox` akan diklik untuk memilihnya.

Kode di atas digunakan untuk melakukan interaksi lebih lanjut dengan elemen-elemen dalam aplikasi mobile yang dituju. Tindakan-tindakan tersebut mencakup klik, penggeseran, dan pemilihan `CheckBox` sesuai dengan kebutuhan aplikasi yang sedang diuji.

```
actions = ActionChains(driver)
actions.w3c_actions = ActionBuilder(driver, mouse=PointerInput(interaction.POINTER_TOUCH, "touch"))
actions.w3c_actions.pointer_action.move_to_location(582, 1703)
actions.w3c_actions.pointer_action.pointer_down()
actions.w3c_actions.pointer_action.move_to_location(600, 748)
actions.w3c_actions.pointer_action.release()
actions.perform()

el10 = driver.find_element(by=AppiumBy.XPATH,
                           value="//android.widget.ImageView[@content-desc=\"BCA\"]/android.widget.RadioButton")
el10.click()
el11 = driver.find_element(by=AppiumBy.ACCESSIBILITY_ID, value="Ya")
el11.click()

actions = ActionChains(driver)
actions.w3c_actions = ActionBuilder(driver, mouse=PointerInput(interaction.POINTER_TOUCH, "touch"))
actions.w3c_actions.pointer_action.move_to_location(548, 521)
actions.w3c_actions.pointer_action.pointer_down()
actions.w3c_actions.pointer_action.pause(0.1)
actions.w3c_actions.pointer_action.release()
actions.perform()

el13 = driver.find_element(by=AppiumBy.ACCESSIBILITY_ID, value="Cek Status Transaksi")
el13.click()
el14 = driver.find_element(by=AppiumBy.ACCESSIBILITY_ID, value="Tutup")
el14.click()
print(b + "PASSED")
driver.quit()
```

Gambar 3.17. *Subscription 3*

```
except:
    driver.quit()
    print(b + "FAILED")
```

Gambar 3.18. *Failed*

Pada bagian kode di atas, terdapat serangkaian tindakan lanjutan untuk menemukan dan berinteraksi dengan elemen-elemen dalam aplikasi mobile menggunakan driver Appium.

Pertama, objek `actions` dari kelas `ActionChains` digunakan untuk melakukan tindakan yang melibatkan penggunaan pointer pada perangkat mobile. Dalam contoh ini, digunakan objek `ActionBuilder` dan `PointerInput` untuk mengkonfigurasi tindakan pointer. Tindakan tersebut mencakup pergerakan pointer ke lokasi yang diinginkan dengan menggunakan metode `move_to_location()`, menekan pointer dengan metode `pointer_down()`, menggeser pointer ke lokasi lain dengan metode `move_to_location()`, dan melepaskan pointer dengan metode `release()`. Setelah konfigurasi tindakan selesai, tindakan tersebut dieksekusi menggunakan metode `perform()`.

Selanjutnya, dilakukan pencarian elemen menggunakan metode

`find_element()` dengan menggunakan strategi pencarian berdasarkan XPath atau Accessibility ID. Setelah elemen ditemukan, dilakukan tindakan klik pada elemen tersebut menggunakan metode `click()`.

Kode di atas juga menunjukkan penggunaan objek `actions` dari kelas `ActionChains` untuk melakukan tindakan penggeseran (swipe) pada elemen. Tindakan ini melibatkan pergerakan pointer ke lokasi tertentu dengan metode `move_to_location()`, menekan pointer dengan metode `pointer_down()`, menghentikan sementara tindakan dengan metode `pause()`, dan melepaskan pointer dengan metode `release()`. Setelah konfigurasi tindakan selesai, tindakan tersebut dieksekusi menggunakan metode `perform()`.

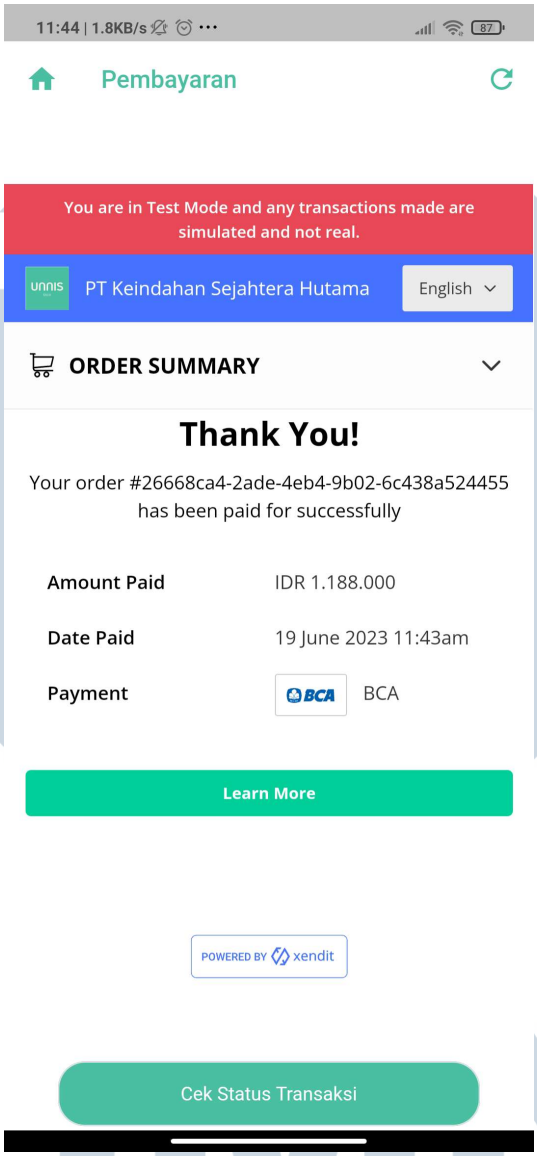
Setelah itu, dilakukan pencarian elemen lagi dan dilakukan tindakan klik pada elemen tersebut. Kemudian, dilakukan tindakan klik pada elemen lainnya. Terakhir, kode mencetak ”PASSED” dan menjalankan perintah `driver.quit()` untuk mengakhiri sesi pengujian dan jika terjadi pengecualian dalam blok kode sebelumnya, maka dilakukan penutupan (quit) driver dengan menggunakan metode `driver.quit()`. Selain itu, juga dicetak pesan ”FAILED” untuk menandakan bahwa proses tersebut tidak berhasil..

Dalam konteks laporan magang, penggunaan kode ini menunjukkan kemampuan dalam melakukan tindakan interaktif yang lebih kompleks pada aplikasi mobile menggunakan driver Appium. Hal ini berguna dalam melakukan pengujian otomatis yang melibatkan pergerakan pointer, penggeseran, dan pemilihan elemen sesuai dengan kebutuhan aplikasi yang sedang diuji.

```
C:\Users\xaver\PycharmProjects\autoTest\venv\Scripts\python.exe C:\Users\xaver\PycharmProjects\autoTest\Controller.py
Subscription Test: PASSED
Event Sort By Test: PASSED
Feed Page Test: PASSED
----------------------------------
My Profile Functions
----------------------------------
Voucher Claim Test: PASSED
Subscription Info Page Test: PASSED
History Page Test: FAILED
FAQ Page Test: PASSED

Process finished with exit code 0
```

Gambar 3.19. *Output*



Gambar 3.20. *Payment Complete*

Pada output yang diberikan, terdapat hasil pengujian beberapa fungsi pada profil pengguna. Berikut adalah penjelasan mengenai hasil pengujian tersebut:

```
Subscription Test: PASSED
Event Sort By Test: PASSED
Feed Page Test: PASSED
Voucher Claim Test: PASSED
Subscription Info Page Test: PASSED
History Page Test: FAILED
```

```
FAQ Page Test: PASSED
```

Output di atas mencerminkan hasil pengujian beberapa fungsi pada profil pengguna. Fungsi-fungsi yang berhasil (PASSED) ditandai dengan tulisan **PASSED**, sementara fungsi-fungsi yang tidak berhasil (FAILED) ditandai dengan tulisan **FAILED**. Hal ini memberikan gambaran tentang keberhasilan atau kegagalan fungsi-fungsi yang telah diuji. Informasi ini berguna dalam melaporkan hasil pengujian dan mengidentifikasi masalah yang perlu diperbaiki dalam pengembangan perangkat lunak.

3.4 Kendala dan Solusi yang Ditemukan

3.4.1 Kendala yang ditemukan

Berikut merupakan kendala-kendala yang dihadapi dalam membuat *testing* otomatis ini.

1. Beberapa fungsi menghasilkan output ”FAILED” karena terjadi perubahan antarmuka pengguna pada *activity* yang sedang diuji.
2. Elemen-elemen pada aplikasi tidak memiliki *accessibility id*, sehingga menyebabkan proses pengujian menjadi lebih lambat.

3.4.2 Solusi yang ditemukan

Adapun solusi yang diterapkan terhadap kendala-kendala yang disebutkan sebelumnya.

1. Berkonsultasi dengan *supervisor* untuk mengecek *priority list* yang dibuat
2. Berkonsultasi dengan pengembang (*developer*) untuk menambahkan *accessibility id* untuk mempermudah dan mempercepat proses *automation*.