Automation Testing

Chapter 5 - Topic 3

Selamat datang di Chapter 5 Topic 3 online course

React Native dari Binar Academy!

Yuhuuuuuu~

Chapter 5 Topic 2 sudah kamu lewati. Sekarang saatnya masuk di topik 3 nih. By the way, kalau di topik sebelumnya kamu belajar tentang TDD alias Test Driven Development, sekarang kamu akan belajar tentang **Automation Testing** yang mencakup Automation Testing End-to-End dan Detox.

Sudah siap berpusing-pusing ria lagi? Mantap! Karena kamu pasti bisa!

Yaudah, yuklah langsung kita bahas~

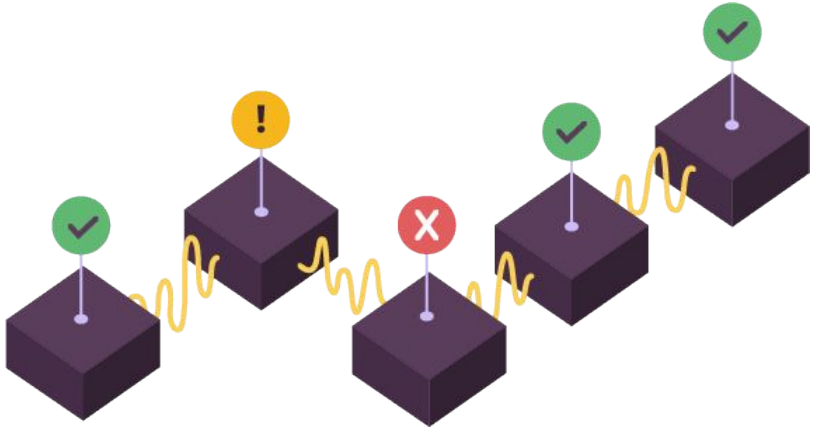
Detailnya, kita bakal bahas hal-hal berikut ini:

1. Memahami Konsep TDD (Test Driven Development)
2. Memahami cara kerja pengujian (testing)
3. Memahami cara penulisan code uji dengan Jest

Guys, untuk meminimalisir bug dan memastikan aplikasi berjalan sesuai harapan, kamu perlu melakukan testing (pengujian) sebelum aplikasi diluncurkan.

Bagaimana caranya? Yuk langsung kita pelajari :)

Setelah mempelajari dasar pengujian (testing) pada aplikasi, yang sebelumnya kamu lakukan pengujian dari sisi code, sekarang kamu akan mempelajari jenis testing yang disebut **End-to-End testing (E2E).**



Dalam setiap tim Software Development, ada QA (Quality Assurance) khusus yang bertanggung jawab untuk menguji fitur sebelum dirilis ke Production stage.

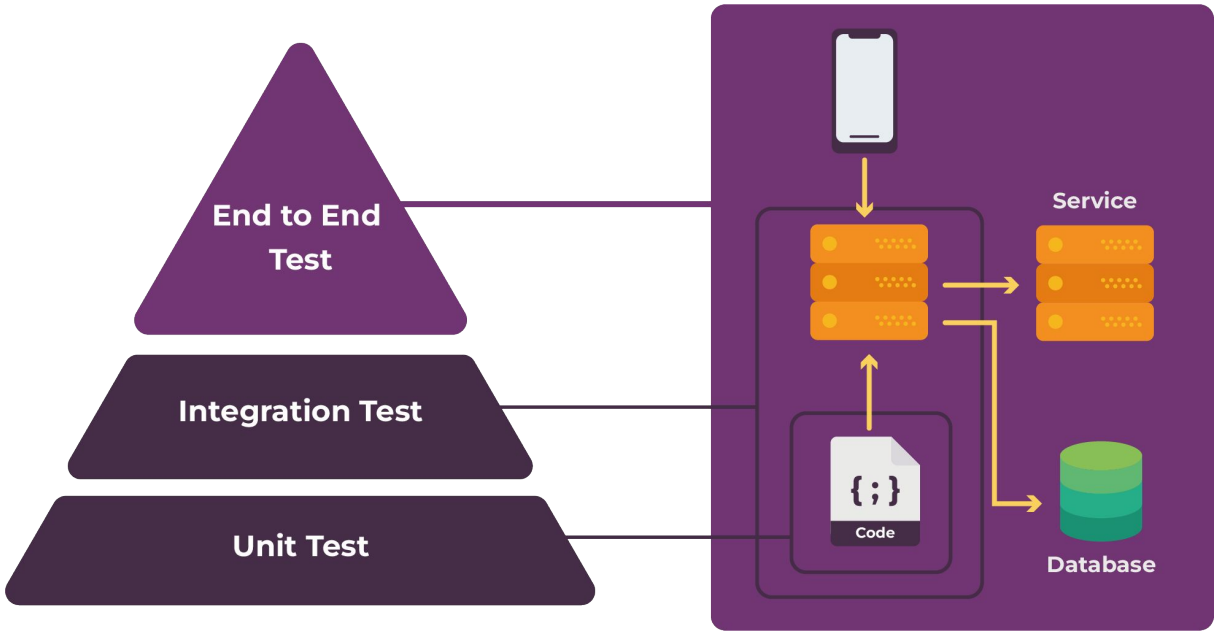
Ini bukan hal yang sulit untuk QA jika aplikasinya nggak terlalu rumit. Tapi bagaimana ya kalau aplikasi yang dibangun sudah terlalu besar??

Nah, untuk menjawab pertanyaan itu, kamu perlu mempelajari dan mengetahui dulu tentang **End-to-End testing** dan **Test Case.**

Apa itu End-to-End testing?

End-to-End testing adalah teknik untuk menguji seluruh produk perangkat lunak dari awal hingga akhir, untuk memastikan aliran aplikasi berperilaku seperti yang diharapkan.

Intinya, teknik ini bertujuan untuk sistem produk dan memastikan semua bagian terintegrasi bekerja sama seperti yang diharapkan.



Tujuan utama pengujian End-to-End adalah untuk menguji dari experience user dengan mensimulasikan skenario pengguna sebenarnya, serta memvalidasi sistem yang diuji dan komponennya untuk integrasi dan integritas data.

Nah, skenario pengguna inilah yang disebut dengan Test Case.

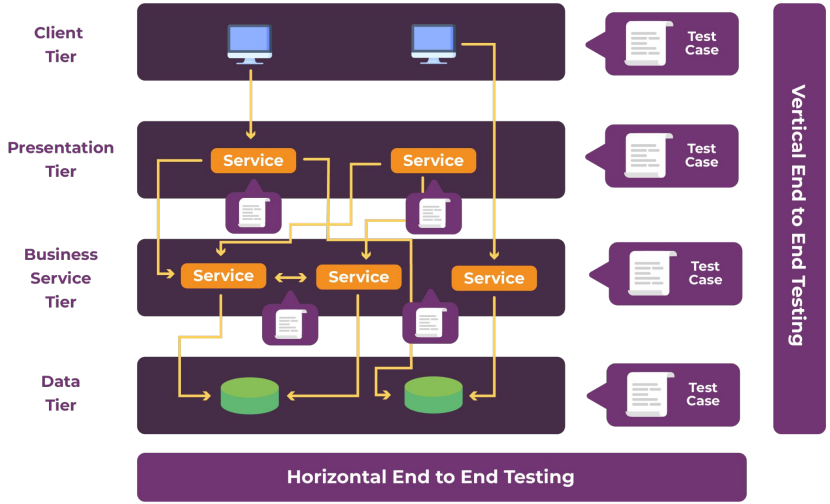
Project Name	Bank Webiste Functionality								
Module Name	Login Functionality								
Created By	Archana								
Created Date	yyyy-mm-xx								
Executed By	XYZ								
Executed Date	YYYY-MM-DD								
Test Case ID	Test Case Description	Pre Steps	Test Step	Preconditions	Test Data	Expected Result	Actual Result	Status	Comments
Test the Login Functionality in Banking	Verify login functionality with valid username & password		Navigate to login page			Able to see the login page	As expected	Pass	
			Enter valid username	Valid Username	username: choudaryac97@gmail.com	Credential can be entered	As expected	Pass	
			Enter valid password	Valid password	password: XXXXXXX@1	Credential can be entered	As expected	Pass	
			Click on login button			User logged	User logged successfully	Pass	
Test the Login Functionality in Banking	Verify login functionality with valid username & invalid password		Navigate to login page			Able to see the login page	As expected	Pass	
			Enter valid username	Valid Username	username: choudaryac97@gmail.com	Credential can be entered	As expected	Pass	
			Enter valid password	Invalid password	password: XXXXXXX@2	Credential can be entered	As expected	Pass	
			Click on login button			User logged	Unsuccessful login	Fail	

Contoh dari Test Case login dapat kamu lihat pada gambar di atas.

Gambar di atas menunjukkan, aplikasi memiliki halaman login, informasi harus valid username dan password, dll. Jika salah satu kondisi nggak terpenuhi maka akan diberi label fail dan jika berhasil akan diberi label pass.

Sudah mulai terbayang belum dengan End-to-End testing??

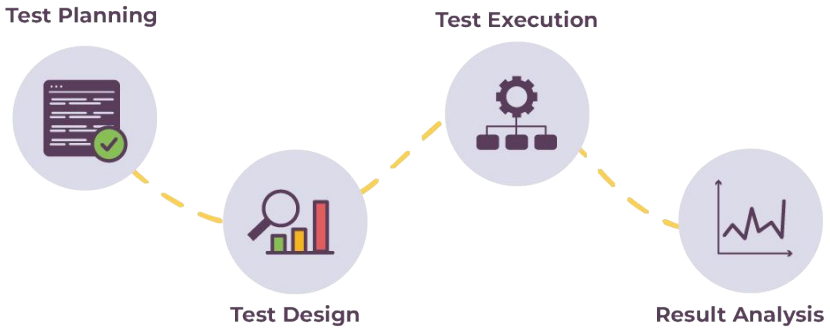
Hmm.. jadi singkatnya ada team QA yang akan menilai aplikasi yang dibangun oleh developer. Untuk menguji kelangsungan aplikasi, QA tersebut telah membuat skenario Test Case berdasarkan requirement yang diterima oleh developer dari product management.



Umumnya seorang QA akan menerapakan alur seperti pada gambar di samping.

Setiap ada fitur baru yang akan dirilis, mereka akan menyiapkan skenario test. Setelah mendiskusikan bagaimana cara mengujinya, mereka akan melakukan tahap pengujian mulai dari tes desain hingga tes eksekusi.

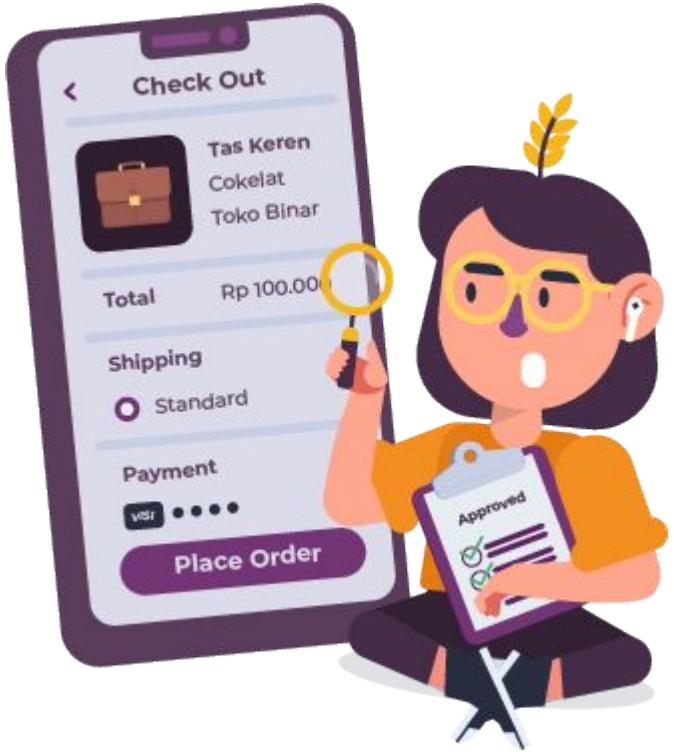
Langkah tersebut diterapkan hingga pada akhirnya mereka bisa mengukur dan menilai apakah aplikasi yang telah dibangun siap untuk dipakai atau belum.



Lalu bagaimana jika aplikasimu semakin besar?

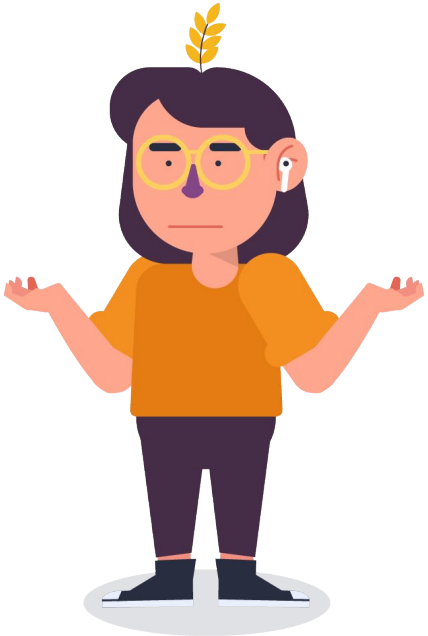
Misalnya kamu punya aplikasi ecommerce yang sudah banyak fiturnya, lalu kamu mau mengembangkan fitur baru. Saat mendevelop fitur baru tersebut, tentu kamu nggak mau ada efek bug terhadap fitur checkout.

Nah, di sinilah tugas QA. Meskipun ada tambahan fitur lain, tetapi QA harus melakukan pengujian ke core system dari aplikasi, dalam hal ini fitur checkout.

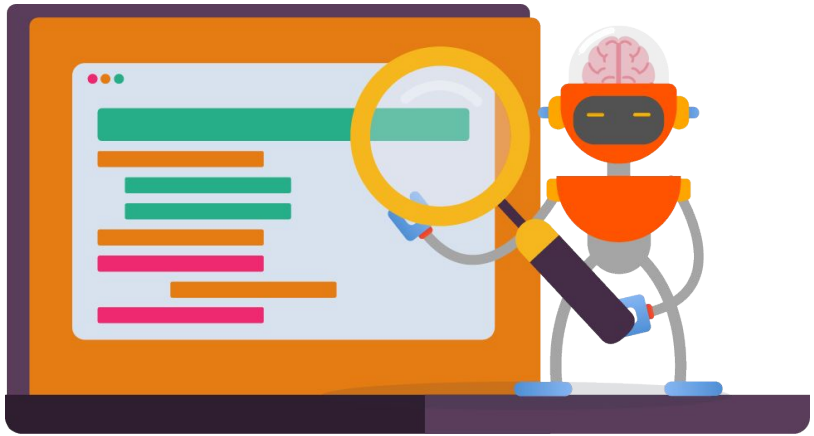


Tentu saja penambahan fitur bisa dan boleh dilakukan, tapi kalau nggak teliti dalam pengerjaannya, hal tersebut malah akan menimbulkan kerugian, terutama dari segi waktu dan efisiensi.

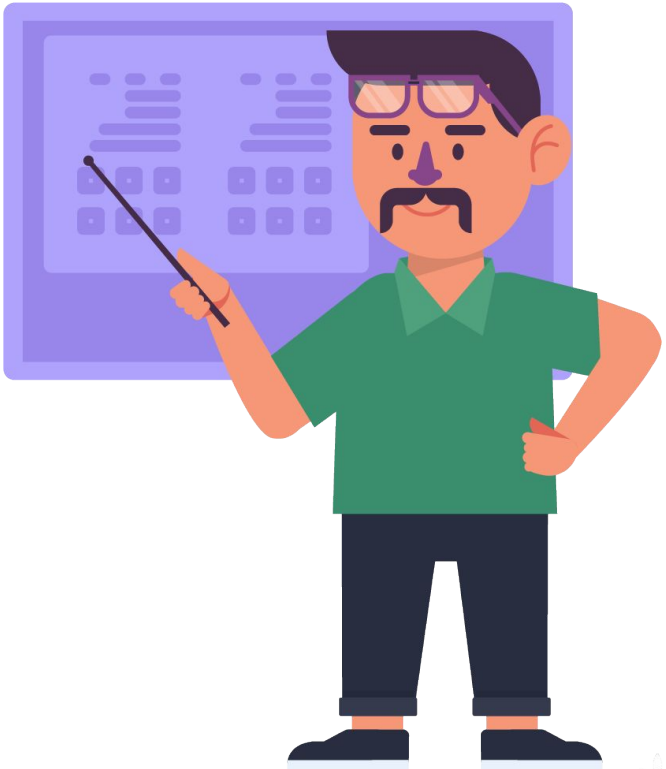
Lalu bagaimana ya solusinya?



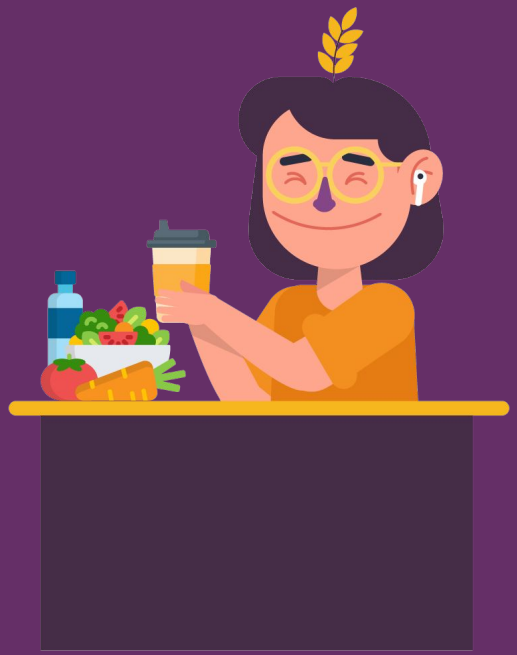
Solusinya adalah dengan menerapkan automated testing. Selain agar kamu bisa percaya diri dengan aplikasimu, automated testing akan mencegah code fitur nggak menyenggol code fitur lain.



Naaaahh, salah satu software automated testing yang akan kita pelajari kali ini adalah **detox**.



Kalau fungsi detox bagi kesehatan adalah membersihkan tubuh dari racun sehingga kamu tetap sehat dan kuat, maka **Detox** pada aplikasi mengeksekusi testing dengan cepat, sehingga aplikasimu akan lebih kokoh.



Jadi, apa itu Detox Testing Framework?

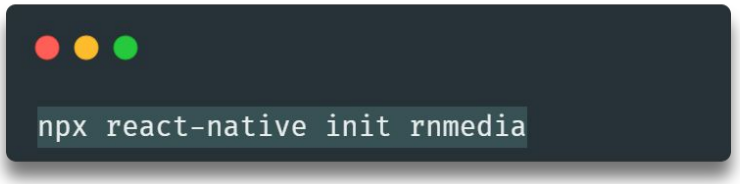
Detox adalah framework End-to-End testing untuk mobile apps yang dikembangkan oleh Wix, salah satu kontributor teratas komunitas React Native.

Wix juga mengelola proyek luar biasa seperti **react-native-navigation** dan **react-native-ui-lib.**

Oke biar paham, silakan langsung coba install Wix ke dalam project-mu. Jalankan react-native init nama_project contoh: **`testDetoxRN`**

Pastikan kamu juga sambil membaca dokumentasi, karena sebagai developer yang baik kemampuan membaca dokumentasi yang baik juga diperlukan.

https://wix.github.io/Detox/docs/introduction



2. Install Detox Command Line Tools (detox -cli)

Karena nantinya kamu akan menggunakan command yang diawali dengan detox, maka kamu harus menginstall detox cli secara global, .

Jalankan command di bawah ini pada terminalmu:

```
npm install -g detox-cli
```

3. Pindah ke React Native project, jalankan dan install detox package

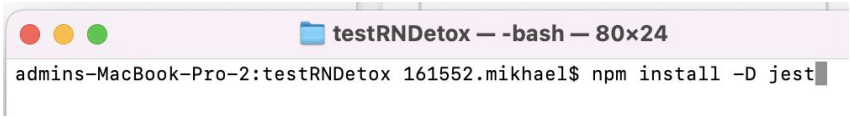
Karena nantinya kita akan menggunakan command yang di awal dengan detox, maka kita harus menginstall detox cli secara global, jalankan command dibawah ini di terminal kalian

```
npm install detox --save-dev
```

4. Setup Test Runner

Sekarang project-mu sudah memiliki nodemodules/detox, nah langkah selanjutnya adalah setup test runner.

Test runner yang akan kamu gunakan adalah Jest. Jalankan npm install -D jest di terminal.

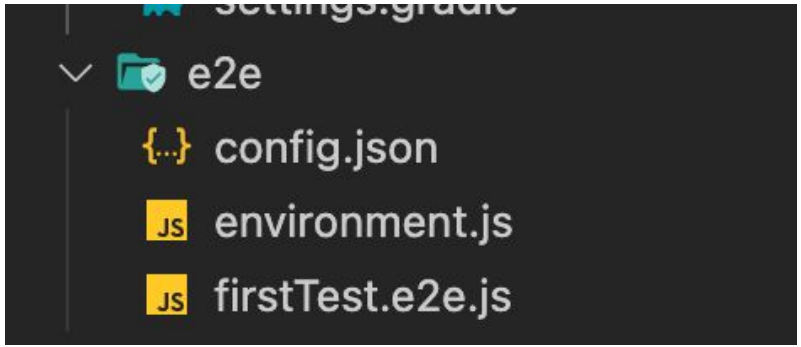


5. Setup Test Runner Scaffold

Selanjutnya, jalankan detox init -r Jest.

Jika semua berjalan lancar, maka di project-mu akan bertambah 1 folder baru bernama E2E.

Dan jika sebelumnya untuk unit test kamu menggunakan __tests__, maka untuk End-to-End testing kamu memakai folder E2E.



6. Setup iOS development dan testing environment

Untuk iOS, pastikan kamu sudah instal Xcode, kemudian jalankanlah xcode-select --install.

Selanjutnya, jalankan perintah di bawah ini:

```
brew tap wix/brew

brew install applesimutils
```

7. Setup android development dan testing environment

Untuk Android, kamu harus melakukan setup untuk Java terlebih dulu. Jadi pastikan kamu menginstal Java minimal jdk version 11 ke atas ya.

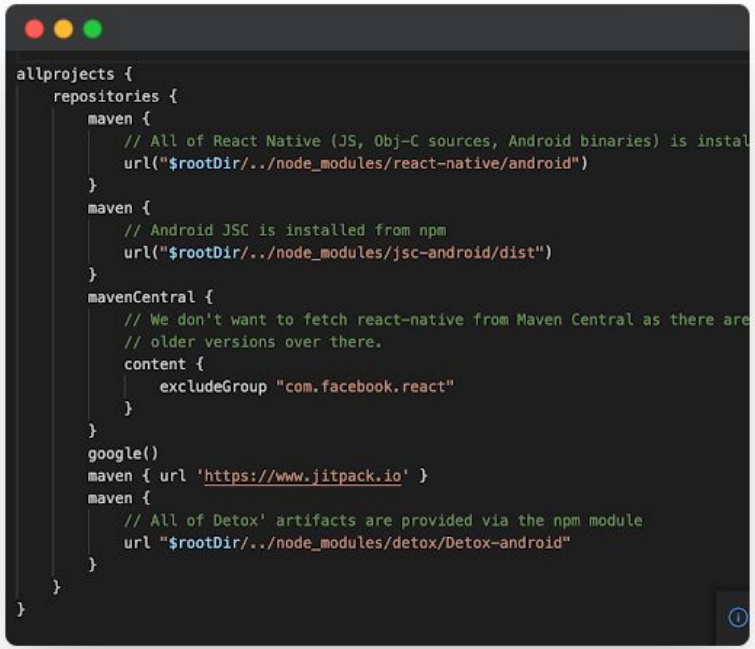
Coba ketik command di bawah ini untuk memastikan versi berapa yang telah terinstal:

```
java -version
```

8. Setup detox for android

Tambahkan native detox dependency, lalu tambahkan juga code di bawah ini pada android/build.gradle:

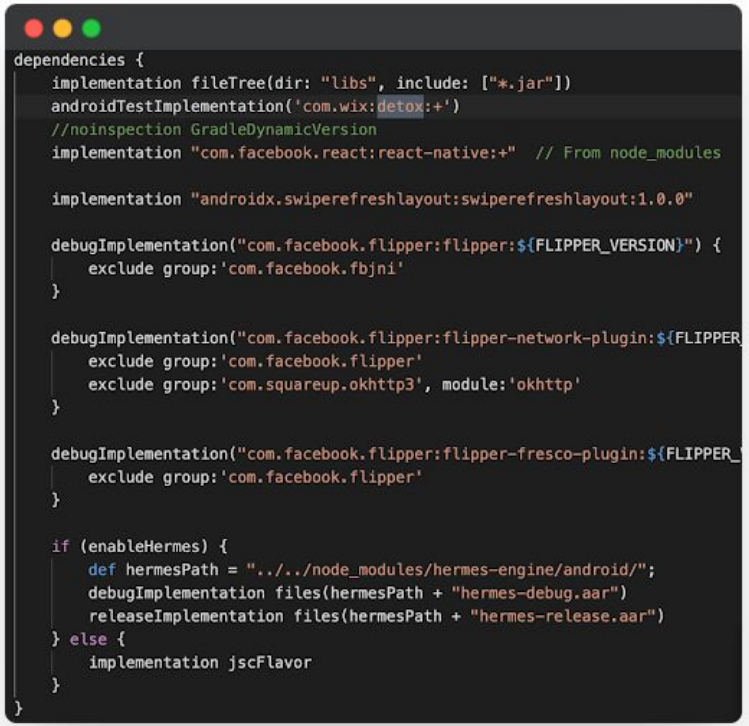
```
// Note: add the 'allproject' section if it
doesn't exist
allprojects {
    repositories {
        // ...
        google()
        maven {
            // All of Detox' artifacts are
provided via the npm module
            url
"$rootDir/../node_modules/detox/Detox-android"
        }
    }
}
```



9. Add native detox dependency (2)

Setelah itu, tambahkan code di bawah ini pada ba android/app/build.gradle:

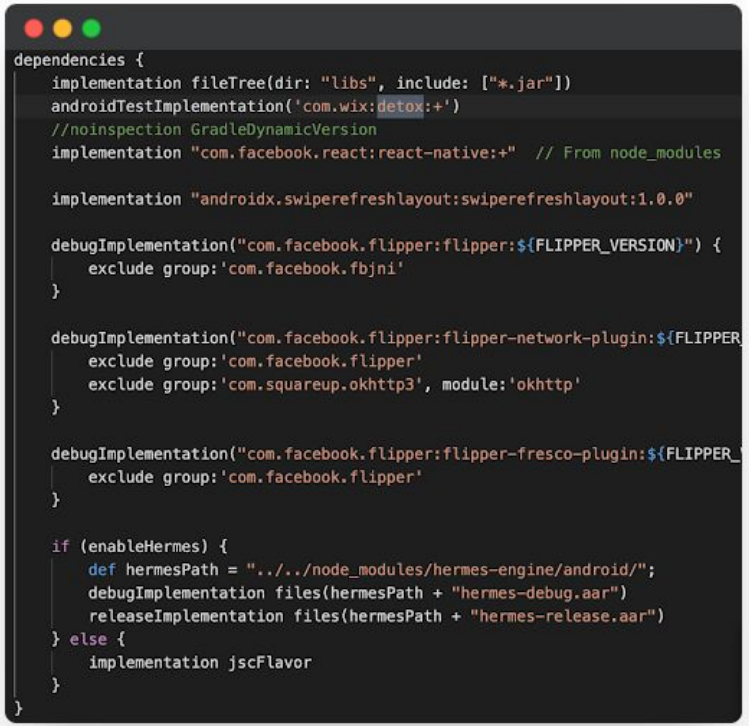
```
dependencies {
    // ...
    androidTestImplementation('com.wix:detox:+')
}
```



9. Add native detox dependency (3)

Tambahkan juga code di bawah ini pada bagian android/app/build.gradle, subsection default config:

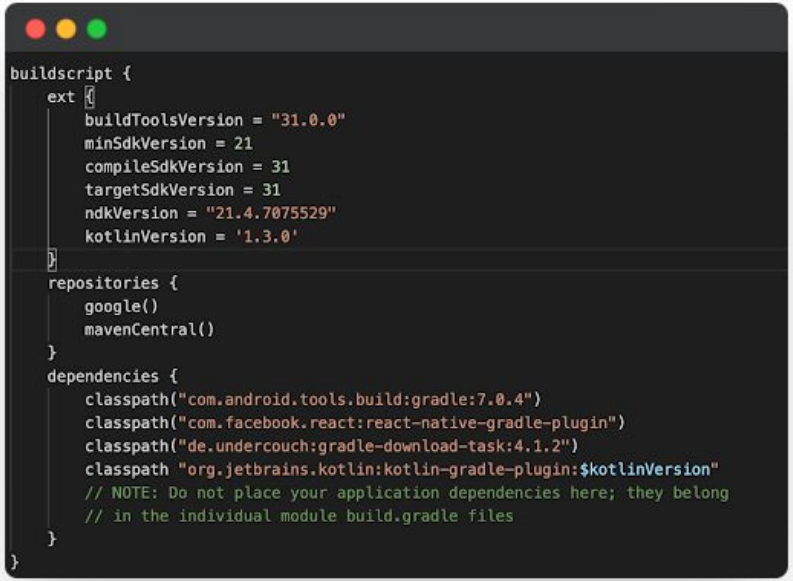
```
android {
  // ...

  defaultConfig {
      // ...
      testBuildType System.getProperty('testBuildType',
'debug')  // This will later be used to control the test
apk build type
      testInstrumentationRunner
'androidx.test.runner.AndroidJUnitRunner'
  }
}
```



10. Tambahkan kotlin

Tambahkan juga code di bawah ini pada bagian android/build.gradle:

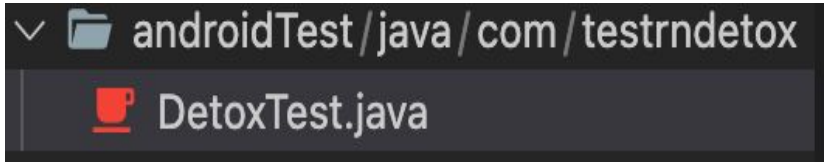
```
buildscript {
    // ...
    ext.kotlinVersion = '1.3.0' // (check what
the latest version is!)
    dependencies {
        // ...
        classpath
"org.jetbrains.kotlin:kotlin-gradle-plugin:$kotli
nVersion"
    }
}
```



11. Buat Detox Text Class

Buatlah folder file baru seperti di bawah ini:
`android/app/src/androidTest/java/com/[your.package]/DetoxTest.java`

Cek 2 kali lokasi folder dan nama file, pastikan untuk membuat folder dengan nama file project.

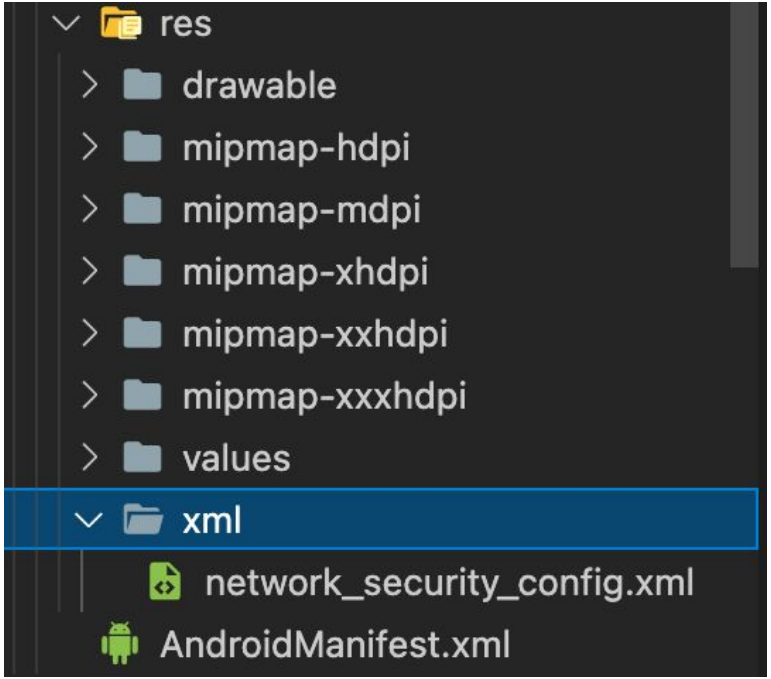


Copy dan paste isi code di dari link di bawah ini ke file DetoxTest.java kamu:

https://github.com/wix/Detox/blob/master/examples/demo-react-native/android/app/src/androidTest/java/com/example/DetoxTest.java

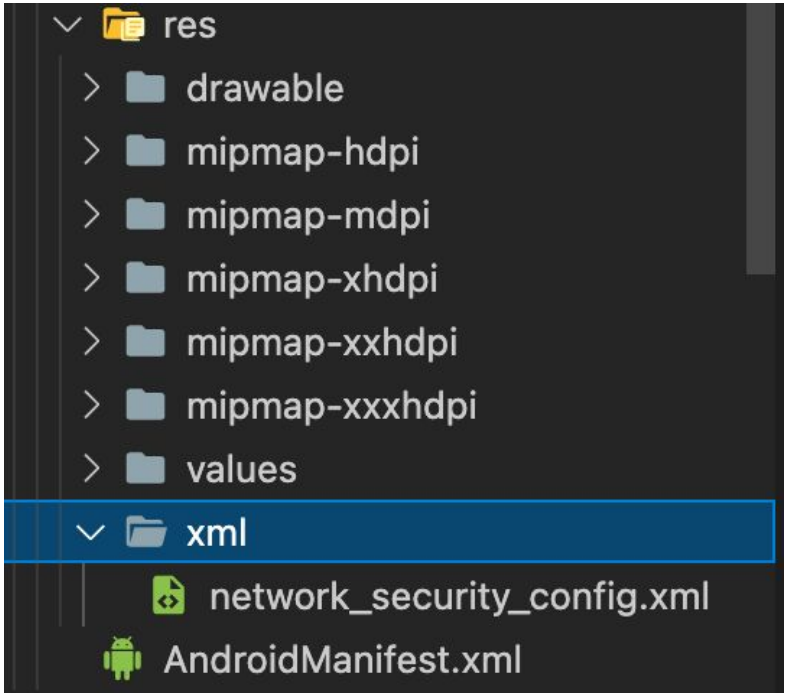
12. Izinkan clear-text (unencrypted) traffic untuk Detox

Buatlah file baru di folder dengan root seperti dib awah ini:
`android/app/src/main/res/xml/network_security_config.xml`



Lalu tambahkan code di bawah ini :

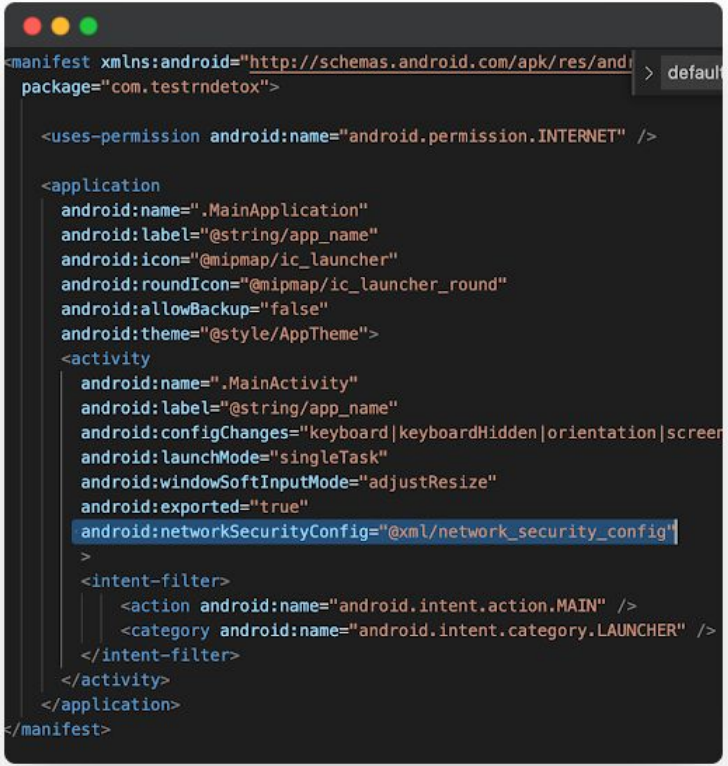
```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config
cleartextTrafficPermitted="true">
        <domain
includeSubdomains="true">10.0.2.2</domain>
        <domain
includeSubdomains="true">localhost</domain>
    </domain-config>
</network-security-config>
```



13. Izinkan clear-text (unencrypted) traffic untuk Detox(2)

Setelah membuat file network_security_config.xml, kamu perlu memanggil file tersebut di androidManifest.xml. Tambahkan code berikut :

```
<manifest>
  <application
        ...

android:networkSecurityConfig="@xml/network_securit
y_config">
  </application>
</manifest>
```



14. ProGuard (Minification, Obfuscation)

Tambahkan code berikut di app/build.gradle :

```
    buildTypes {
        // 'release' is typically the default proguard-enabled
build-type
        release {
            minifyEnabled true

            // Typical pro-guard definitions
            proguardFiles
getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
            // Detox-specific additions to pro-guard
            proguardFile
"${rootProject.projectDir}/../node_modules/detox/android/detox/p
roguard-rules-app.pro"
        }
    }
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
  package="com.testrndetox">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
      android:name=".MainApplication"
      android:label="@string/app_name"
      android:icon="@mipmap/ic_launcher"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:allowBackup="false"
      android:theme="@style/AppTheme">
      <activity
        android:name=".MainActivity"
        android:label="@string/app_name"
        android:configChanges="keyboard|keyboardHidden|orientation|screen
        android:launchMode="singleTask"
        android:windowSoftInputMode="adjustResize"
        android:exported="true"
        android:networkSecurityConfig="@xml/network_security_config"
        >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
      </activity>
    </application>
</manifest>
```

Sampai di sini, apakah kalian sudah semakin paham? Apakah sudah berhasil setup untuk android? Double check kembali semua konfigurasimu di dokumentasi resmi dari wix/detox, karena kita akan lanjut setup untuk ios:

https://wix.github.io/Detox/docs/introduction/android

Psst, kalau bingung, silakan tanya ke fasilmu ya. Fasil akan dengan senang hati membantu.

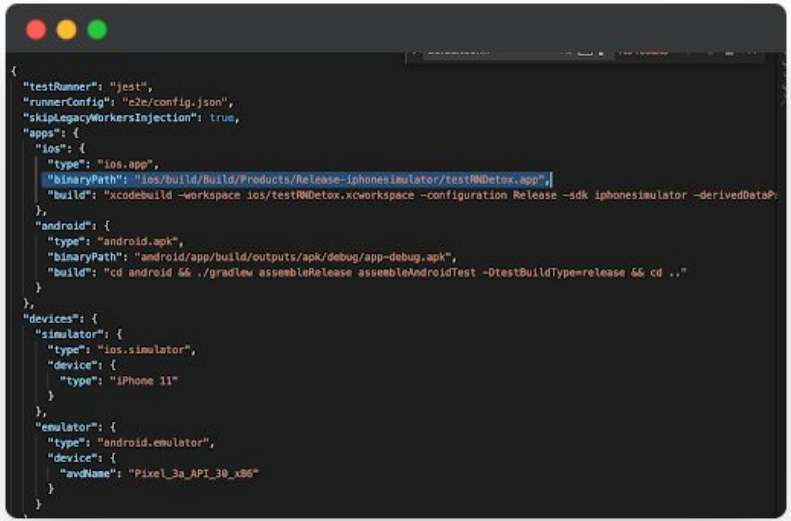
Oke, kamu sudah melewati setup untuk Android, selanjutnya kamu akan belajar setup iOS.

Jangan khawatir, setup iOS kali ini lebih simpel dan gampang kok, guys.

15. Setup IOS - Konfigurasi Detox

Cari dan buka file detoxrc.json kemudian tambahkar
folder pada bagian binary path untuk build file app iOS,
gambar di sar

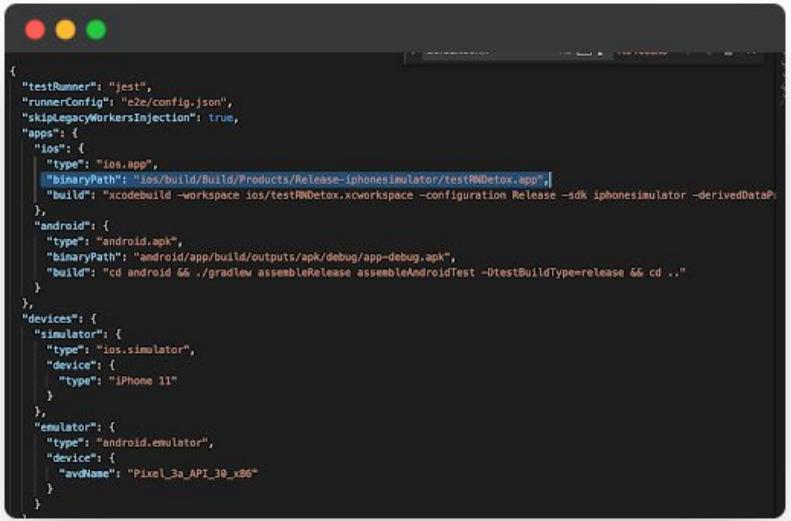
Selesai deh setup untuk IOS



15. Apply detox configuration

Cari dan buka file detoxrc.json kemudian pada bagian binary path tambahkan juga untuk Android path folder untuk build file app android, seperti gambar di samping.

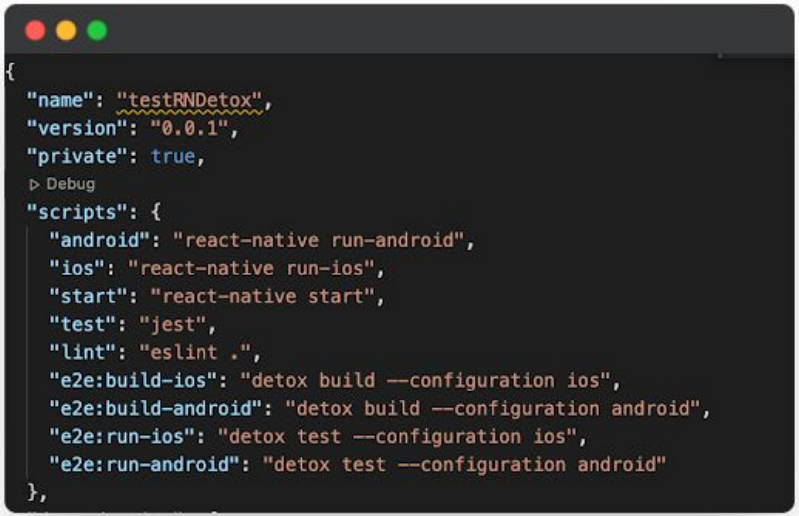
Pastikan kamu nggak salah mengetik app-nya yaa.



16. Menambahkan File command pada package.json

Tambahkan code di bawah ini pada package.json:

```
"e2e:build-ios": "detox build --configuration ios",
"e2e:build-android": "detox build --configuration android",
"e2e:run-ios": "detox test --configuration ios",
"e2e:run-android": "detox test --configuration android"
```



Tujuannya adalah untuk menjalankan dengan command saat kamu mulai menguji. Nah, selesai deh semua step untuk instalasi detox.

Code Detox

Bukalah file firstTest.e2e.js di dalam folder E2E, maka secara default detox akan menampilkan file dengan isi code seperti di samping.

FYI, detox memanfaatkan asynchronous function dalam menjalankan codenya.

```
describe('Example', () => {
  beforeAll(async () => {
    await device.launchApp();
  });

  beforeEach(async () => {
    await device.reloadReactNative();
  });

  it('should have welcome screen', async () => {
    await expect(element(by.id('welcome'))).toBeVisible();
  });

  it('should show hello screen after tap', async () => {
    await element(by.id('hello_button')).tap();
    await expect(element(by.text('Hello!!!'))).toBeVisible();
  });

  it('should show world screen after tap', async () => {
    await element(by.id('world_button')).tap();
    await expect(element(by.text('World!!!'))).toBeVisible();
  });
});
```

Penulisan code Detox

Detox akan mengenali setiap elemen yang ada pada code-mu berdasarkan id, text dll dari yang telah kamu berikan dan akan menyamakan dengan function matchers yang tersedia.

Menuliskan contoh dari testId, kamu dapat menandai komponen apa yang akan kamu tes.

Seperti contoh di samping, kamu memberikan testId terhadap komponen touchableopacity. Kamu juga bisa menambahkan di komponen lain seperti:

View, Text, TextInput, Switch, ScrollView

```
<View>
  <TouchableOpacity testID='MyUniqueId123'>
    <Text>Some button</Text>
  </TouchableOpacity>
</View>
```

Yuk, buat tes pertamamu!

Kamu akan coba untuk melakukan pengujian pertama kali. Coba edit file firstTest.e2e.js menjadi seperti gambar di samping, di project yang baru kamu buat di awal.

```
describe('Example', () => {
  beforeEach(async () => {
    await device.launchApp();
  });

  it('should have "Step One" section', async () => {
    await expect(element(by.text('Step One'))).toBeVisible();
  });

  it('should have "See Your Changes" section', async () => {
    await expect(element(by.text('See Your Changes'))).toBeVisible();
  });
});
```

Kemudian jalankan npm run e2e build:android atau npm run e2e build:ios.

Command ini akan membangun aplikasi kita, tunggu hingga ada informasi build succeeded.

Selanjutya, jalankan npm run e2e test:android atau npm run e2e test:ios.

Command ini akan menjalankan proses automation testing, dan jika semua test pass, maka akan muncul informasi di seperti di samping.

Ini nih penjelasannya, guys

Code di samping akan menguji secara automatis apakah setelah app launch di device, ada element text yg memiliki text Step One dan see your changes.

Karena kamu bikin project baru, maka file tersebut sudah tersedia secara default.

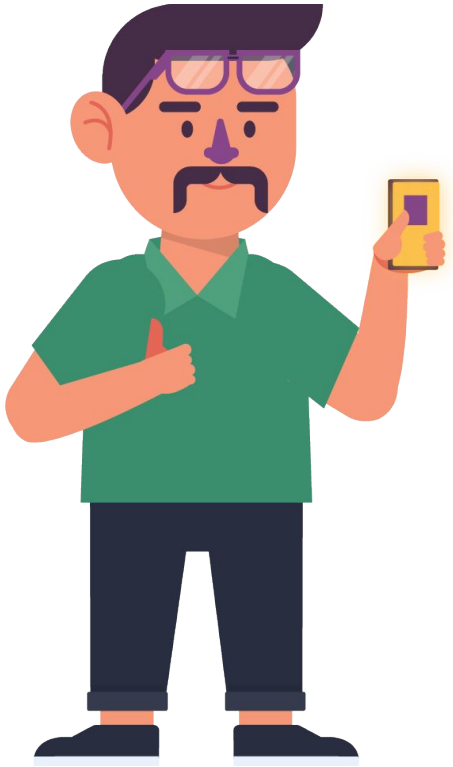
```
describe('Example', () => {
  beforeEach(async () => {
    await device.launchApp();
  });

  it('should have "Step One" section', async () => {
    await expect(element(by.text('Step One'))).toBeVisible();
  });

  it('should have "See Your Changes" section', async () => {
    await expect(element(by.text('See Your Changes'))).toBeVisible();
  });
});
```

Selanjutnya, bagaimana ya untuk cara menjalankan Automation Testing pada case Login?

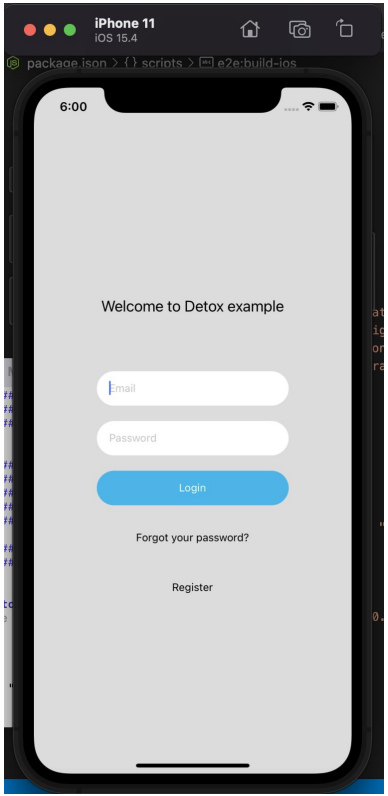
Kita coba yuk!

Oke, sekarang kamu akan menjalankan automation testing terhadap app yang memiliki halaman login.

Jadi buatlah dulu halaman login sesuai yang kamu inginkan yaa.

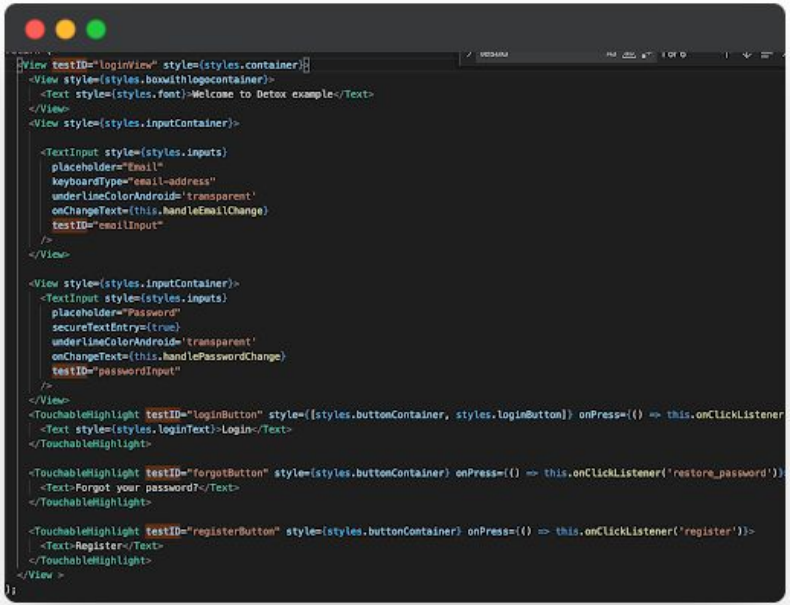


Gambar di samping adalah contoh pembuatan UI login component.



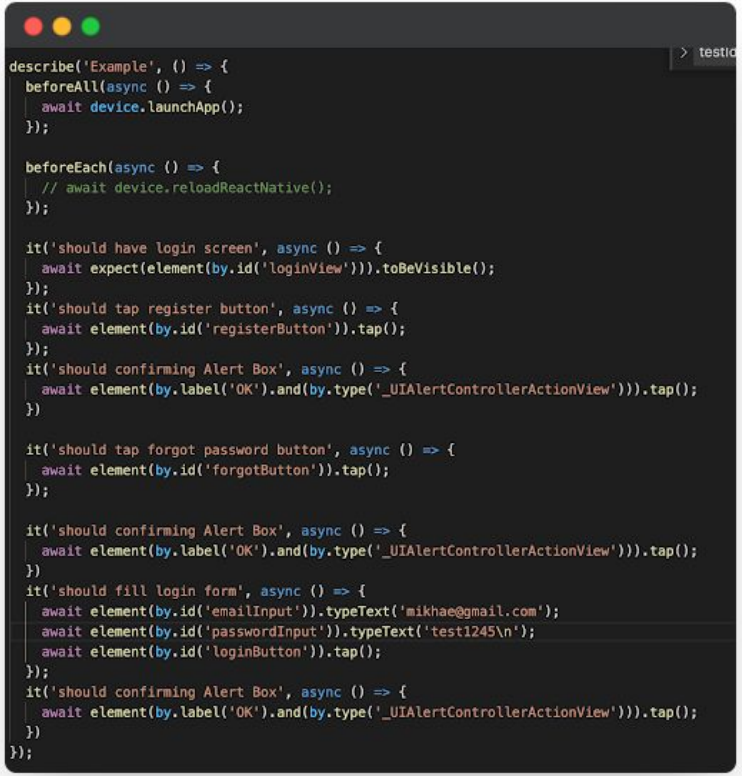
1. Custom UI login-mu

Jangan lupa untuk menambahkan properties testId di setiap komponen yang akan digunakan pada test case yang akan kamu buat.



2. Buat test case code login case

Selanjutnya, buat code test case seperti pada gambar di samping.



```
describe('Example', () => {
  beforeAll(async () => {
    await device.launchApp();
  });

  beforeEach(async () => {
    // await device.reloadReactNative();
  });

  it('should have login screen', async () => {
    await expect(element(by.id('loginView'))).toBeVisible();
  });
  it('should tap register button', async () => {
    await element(by.id('registerButton')).tap();
  });
  it('should confirming Alert Box', async () => {
    await element(by.label('OK').and(by.type('_UIAlertControllerActionView'))).tap();
  })

  it('should tap forgot password button', async () => {
    await element(by.id('forgotButton')).tap();
  });

  it('should confirming Alert Box', async () => {
    await element(by.label('OK').and(by.type('_UIAlertControllerActionView'))).tap();
  })
  it('should fill login form', async () => {
    await element(by.id('emailInput')).typeText('mikhae@gmail.com');
    await element(by.id('passwordInput')).typeText('test1245\n');
    await element(by.id('loginButton')).tap();
  });
  it('should confirming Alert Box', async () => {
    await element(by.label('OK').and(by.type('_UIAlertControllerActionView'))).tap();
  })
});
```

3. Running Test case dengan detox

Perlu diingat setiap kamu menambahkan perubahan kepada component yang akan kamu tes, selalu lakukan build ulang terhadap project-mu dengan command:

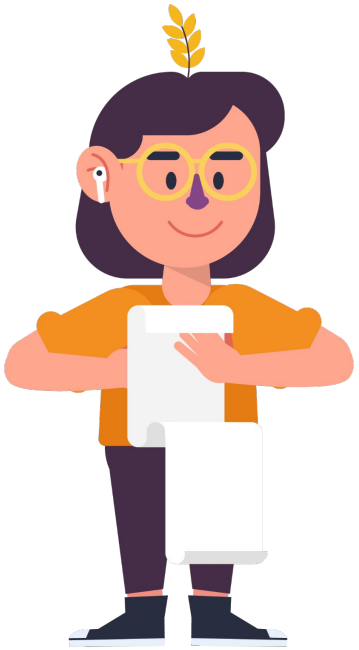
npm run e2e:build-android / ios, kemudian setelah itu jalan npm run e2e:run-android/ios.

Dengan begitu, device akan otomatis melakukan test case login .

Naah, sekarang coba kamu tulis test case automation sendiri yuk!

Kamu dapat melanjutkan automation test terhadap register screen ataupun test case-mu masing-masing.

Selamat mencoba!



Saatnya kita

Quiz!

1

Apa Keuntungan kita melakukan Automation Testing End to End?..

A. Akan menambah Area pengujian.

B. akan Mempercepat proses Development.

C. Akan meminimalisir bug yang terjadi di production.

2

Pernyataan dibawah ini yang benar terkait automation testing adalah, kecuali..

A reduce cost

B detect bugs

C simulating test case

3

Berikut ini yang bukan bagian dari End To End Testing lifecycle adalah?

A Test Planning

B Test Case

C Test Execution

Terima Kasih!

Next Topic

loading...

BINAR