

Laporan

From Manual to Automation Testing & BDD

Soal From Manual to Automation Testing

- 1. Jelaskan jenis-jenis dari performance testing (minimal: 3).**

Load Testing: jenis pengujian yang bertujuan untuk menguji kinerja sistem atau aplikasi dalam kondisi beban tinggi. Dalam pengujian ini, sistem diberikan beban yang lebih besar dari kapasitas normalnya, sehingga dapat diukur bagaimana sistem merespons beban tersebut.

Stress Testing: jenis pengujian yang bertujuan untuk menguji batas maksimum kinerja sistem atau aplikasi. Dalam pengujian ini, sistem diberikan beban yang jauh melebihi kapasitas normalnya, sehingga dapat diukur bagaimana sistem merespons beban yang ekstrim tersebut. Tujuan dari pengujian ini adalah untuk menemukan titik lemah sistem sehingga dapat diperbaiki sebelum terjadi kegagalan atau kerusakan.

Soak testing : pengujian yang melakukan tes dengan load yang normal, namun dalam waktu yang panjang. Tujuannya untuk melakukan analisa dan evaluasi behavior sistem kita. Digunakan untuk mengetahui apakah sistem akan mengalami memory leak pada penggunaan jangka panjang.

<https://sysctl.id/jenis-performance-test/>

- 2. Sebutkan dan jelaskan tools yang dapat digunakan untuk melakukan performance testing (minimal: 3)**

jMeter atau The Apache JMeter™ merupakan aplikasi open source berbasis Java yang bisa digunakan untuk performance test. jMeter juga bisa digunakan untuk melakukan load/stress testing Web Application, FTP Application dan Database server test.

Postman merupakan sebuah aplikasi (berupa plugin) untuk browser chrome, fungsinya adalah sebagai REST Client atau istilahnya adalah aplikasi yang digunakan untuk melakukan uji coba REST API yang telah kita buat.

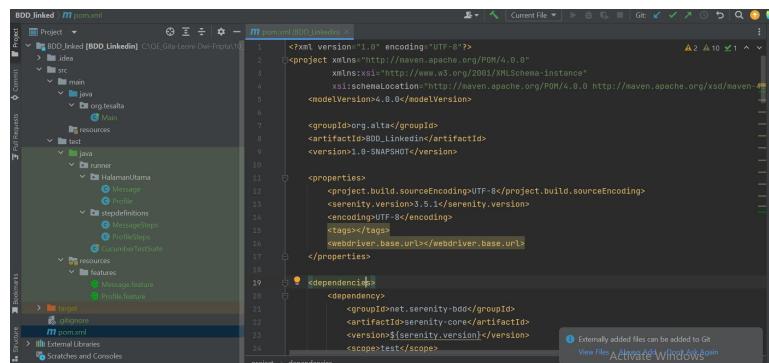
Katalon Studio berfungsi untuk melakukan pengujian API, Web, seluler, dan pengujian aplikasi desktop. Ini juga memiliki serangkaian fitur yang kaya untuk jenis pengujian ini dan mendukung banyak platform termasuk Windows, macOS, dan Linux

<https://toghru.com/10-tools-testing-terbaik/#:~:text=jMeter%20atau%20The%20Apache%20JMeterTM,bisa%20digunakan%20untuk%20performance%20test.>

Soal Behavior Driven Development (BDD)

Soal Prioritas 1 (80)

1. Lakukan testing pada aplikasi <https://www.linkedin.com/> dengan langkah-langkah sebagai berikut:
 1. Buatlah scenario test sebanyak mungkin menggunakan format BDD pada fitur-fitur halaman utama <https://www.linkedin.com/> .
 2. Buatlah project baru menggunakan intelliJ.
 3. Buatlah file feature Cucumber
 4. Buatlah class runner dan step
 5. Buatlah class steps yang berisi method-method yang dijalankan dalam scenario test.
 6. Jalankan class runner.
 7. Tampilkan report HTML dari Cucumber.
1. Pom.xml (project object model) adalah file konfigurasi pada proyek java yang digunakan untuk mengelola dependensi dan konfigurasi proyek. Pada intelliJ, POM dapat digunakan untuk proyek BDD(Behavior Driven Development) dengan menggunakan bahasa gherkin.ini adalah contoh gambar code POM pada projek BDD Linkedin.



The screenshot shows the IntelliJ IDEA interface with the project 'BDD_Linked' selected. The left sidebar displays the project structure with packages like 'java', 'org.thesulta', and 'test'. The right side shows the 'pom.xml' file in the editor. The code is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
    modelVersion="4.0.0">
    <groupId>org.altis</groupId>
    <artifactId>BDD_Linkedin</artifactId>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <serenity.version>3.5.1</serenity.version>
        <encoding>UTF-8</encoding>
        <tags></tags>
        <webdriver.base.url>/webdriver.base.url</webdriver.base.url>
    </properties>
    <dependencies>
        <dependency>
            <groupId>net.serenitybdd</groupId>
            <artifactId>serenity-core</artifactId>
            <version>${serenity.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

```

2. Setelah itu kita langsung membuat folder feature yang di dalamnya terdapat file profil.feature dan message.feature. Profil.feature adalah file yang digunakan dalam BDD untuk menulis skenario pengujian menggunakan bahasa gherkin

Profile.feature

```

Feature: user profiles
  Scenario: Search for User Profiles
    Given the user goes to their profile home page
    When a enters name or keyword in search box
    And a user clicks search
    Then a should get a list of relevant LinkedIn profiles
  
```

Message.feature

```

Feature: Message
  Scenario: Sending Messages to Other Users
    Given a user goes to their profile home page
    When a user opens another user's profile
    And a user clicks the Send Message button and enter a message
    Then a user should be able to send messages to that other user
  
```

3. Setelah membuat file feature selanjutnya adalah membuat java class profile dan message pada package runner halaman utama. folder runner dapat digunakan untuk menyimpan skrip pengujian yang digunakan untuk menjalankan pengujian BDD. Skrip pengujian dapat dijalankan secara otomatis oleh alat pengujian atau oleh pengguna secara manual.

profile

```

package runner.HalamanUtama;

import net.thucydides.core.annotations.Step;

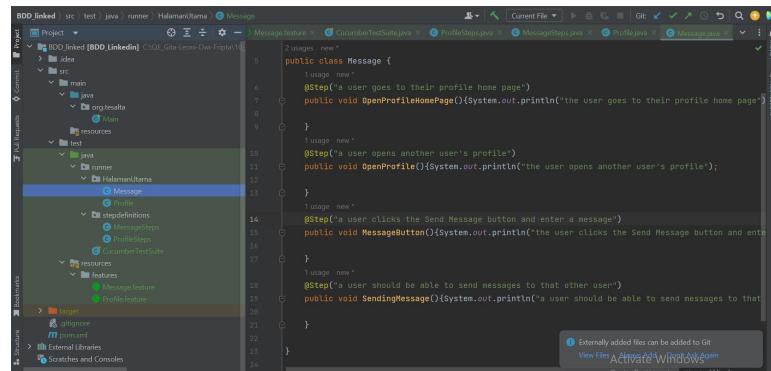
public class Profile {
  @Step("a user goes to their profile home page")
  public void OpenProfile(){System.out.println("a user goes to their profile home page");}

  @Step("a enters name or keyword in search box")
  public void EnterName(){System.out.println("a enters name or keyword in search box");}

  @Step("a user clicks search")
  public void Click(){System.out.println("a user clicks search");}

  @Step("a should get a list of relevant LinkedIn profiles")
  public void LinkedInProfile(){System.out.println("a should get a list of relevant LinkedIn profiles");}
}
  
```

Message



The screenshot shows the IntelliJ IDEA interface with the 'Message' feature file open in the editor. The code defines a step class 'Message' with various scenarios for opening profiles and sending messages.

```
public class Message {
    @Step("a user goes to their profile home page")
    public void OpenProfileHomePage(){System.out.println("the user goes to their profile home page")}

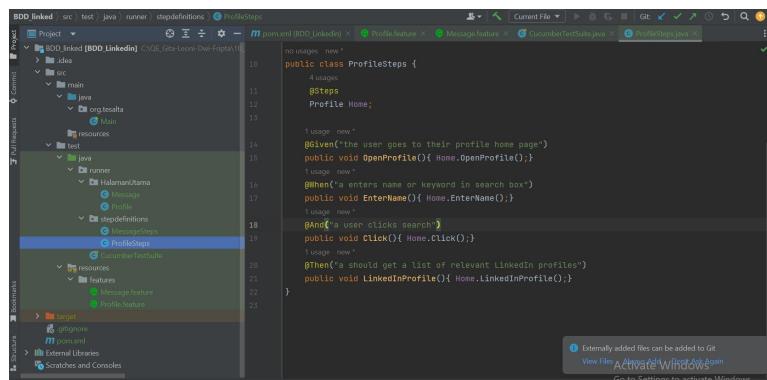
    @Step("a user opens another user's profile")
    public void OpenProfile(){System.out.println("the user opens another user's profile")}

    @Step("a user clicks the Send Message button and enter a message")
    public void MessageButton(){System.out.println("the user clicks the Send Message button and enters a message")}

    @Step("a user should be able to send messages to that other user")
    public void SendingMessage(){System.out.println("a user should be able to send messages to that other user")}
}
```

- Setelah itu membuat folder stepdefinition yaitu untuk tempat scenario yang sama dengan feature dan memanggil file step yang tadi sudah ada pada code di file runner halaman utama.

profileSteps



The screenshot shows the 'ProfileSteps' step definition file. It contains steps for opening profiles and entering search terms.

```
public class ProfileSteps {
    @Steps
    Profile Home;

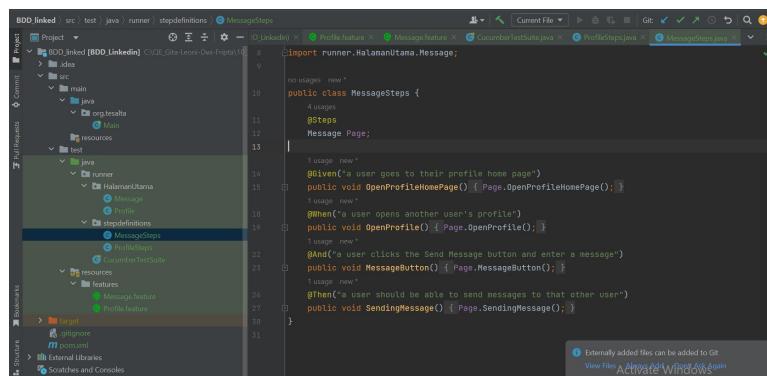
    @Given("the user goes to their profile home page")
    public void OpenProfile(){ Home.OpenProfile();}

    @When("a user enters name or keyword in search box")
    public void EnterName(@Home.EnterName){ Home.EnterName();}

    @Then("a user clicks search")
    public void Click(@Home.Click){ Home.Click();}

    @Then("a user should get a list of relevant LinkedIn profiles")
    public void LinkedInProfile(@Home.LinkedInProfile){ Home.LinkedInProfile(); }
}
```

Message Step



The screenshot shows the 'MessageSteps' step definition file. It contains steps for opening profiles and sending messages.

```
import runner.HalamanUtama.Message;

public class MessageSteps {
    @Steps
    Message Page;

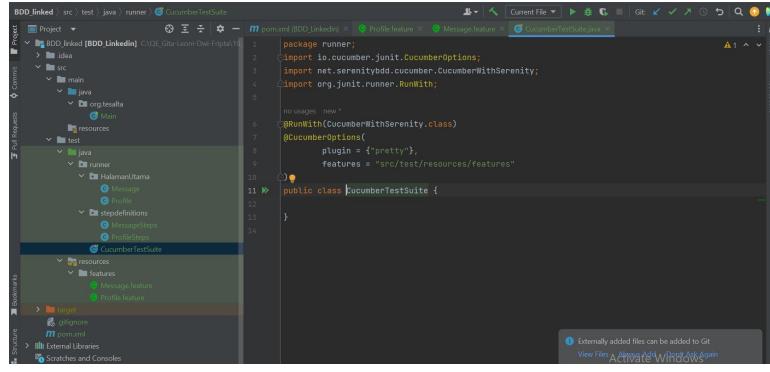
    @Given("a user goes to their profile home page")
    public void OpenProfileHomePage() { Page.OpenProfileHomePage(); }

    @When("a user opens another user's profile")
    public void OpenProfile() { Page.OpenProfile(); }

    @And("a user clicks the Send Message button and enter a message")
    public void MessageButton() { Page.MessageButton(); }

    @Then("a user should be able to send messages to that other user")
    public void SendingMessage() { Page.SendingMessage(); }
}
```

- Selanjutnya Membuat java class CucumberTestSuit yaitu java class untuk menjalankan featur yang ada pada project.



```

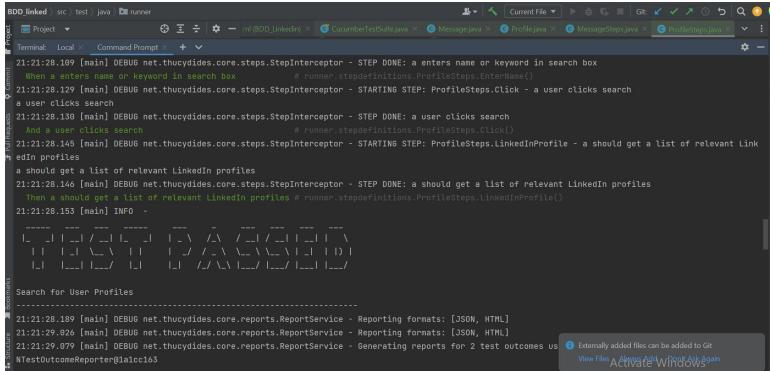
package runner;
import io.cucumber.junit.CucumberOptions;
import net.serenitybdd.cucumber.CucumberWithSerenity;
import org.junit.runner.RunWith;

no usage: new
@RunWith(CucumberWithSerenity.class)
@CucumberOptions(
    plugin = {"pretty"},
    features = "src/test/resources/features"
)
public class CucumberTestSuite {
}

```

6. Setelah selesai semua tahap scenario BDD selanjutnya yaitu run project menggunakan terminal dengan mengetik mvn clean verify. Setelah itu akan muncul test stared dan tes passed seperti gambar dibawah ini.

Test passed profile

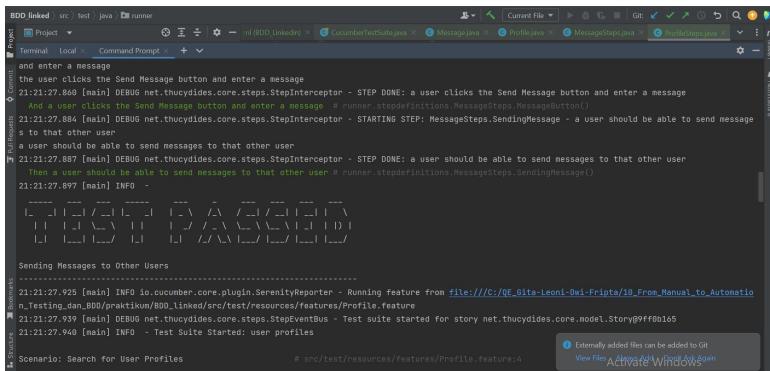


```

21:21:28.109 [main] DEBUG net.thucydides.core.steps.StepInterceptor - STEP DONE: a enters name or keyword in search box
when a enters name or keyword in search box # runner.stepDefinitions.ProfileSteps.enterName()
21:21:28.129 [main] DEBUG net.thucydides.core.steps.StepInterceptor - STARTING STEP: ProfileSteps.click - a user clicks search
a user clicks search
21:21:28.138 [main] DEBUG net.thucydides.core.steps.StepInterceptor - STEP DONE: a user clicks search
and a user clicks search # runner.stepDefinitions.ProfileSteps.click()
21:21:28.145 [main] DEBUG net.thucydides.core.steps.StepInterceptor - STARTING STEP: ProfileSteps.LinkedInProfile - a should get a list of relevant Linkedin profiles
a should get a list of relevant linkedin profiles
21:21:28.146 [main] DEBUG net.thucydides.core.steps.StepInterceptor - STEP DONE: a should get a list of relevant LinkedIn profiles
then a should get a list of relevant LinkedIn profiles # runner.stepDefinitions.ProfileSteps.LinkedInProfile()
21:21:28.153 [main] INFO -
-----
[...]
-----
Search for User Profiles
-----
21:21:28.189 [main] DEBUG net.thucydides.core.reports.ReportService - Reporting formats: [JSON, HTML]
21:21:29.026 [main] DEBUG net.thucydides.core.reports.ReportService - Reporting formats: [JSON, HTML]
21:21:29.079 [main] DEBUG net.thucydides.core.reports.ReportService - Generating reports for 2 test outcomes us

```

Tes passed message



```

and enter a message
the user clicks the Send Message button and enter a message
21:21:27.808 [main] DEBUG net.thucydides.core.steps.StepInterceptor - STEP DONE: a user clicks the Send Message button and enter a message
the user clicks the Send Message button and enter a message # runner.stepDefinitions.MessageSteps.sendMessage()
21:21:27.884 [main] DEBUG net.thucydides.core.steps.StepInterceptor - STARTING STEP: MessageSteps.SendMessage - a user should be able to send message
s to that other user
a user should be able to send messages to that other user
21:21:27.887 [main] DEBUG net.thucydides.core.steps.StepInterceptor - STEP DONE: a user should be able to send messages to that other user
then a user should be able to send messages to that other user # runner.stepDefinitions.MessageSteps.sendMessage()
21:21:27.897 [main] INFO -
-----
[...]
-----
Sending Messages to Other Users
-----
21:21:27.925 [main] INFO io.cucumber.core.plugin.SerenityReporter - Running feature from file:///C:/QE_Gits-Leoni-Dwi-Fripta/10_Framwork_Manual_to_Automatic_n_Testing_dan_BDD/praktikum/BDD_linked/src/test/resources/features/Profile.feature
21:21:27.939 [main] DEBUG net.thucydides.core.steps.StepEventBus - Test suite started for story net.thucydides.core.model.Story@fffb0165
21:21:27.948 [main] INFO - test Suite Started: user profiles
Scenario: Search for User Profiles

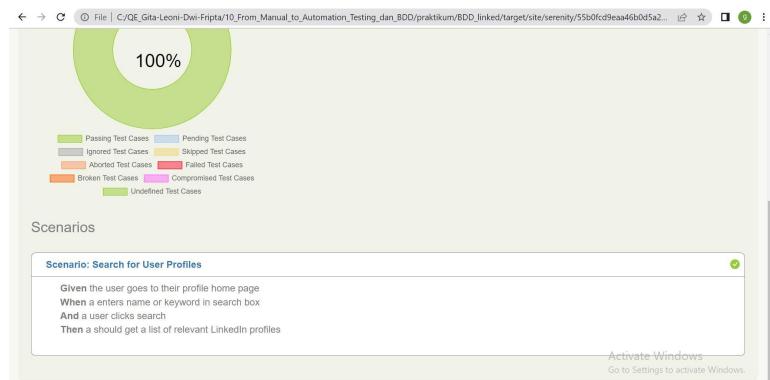
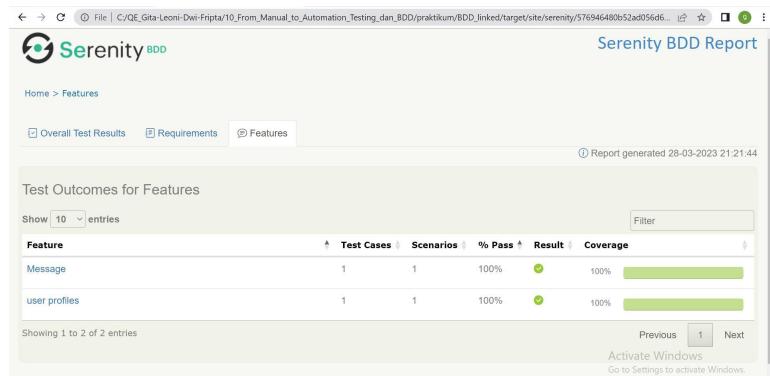
```

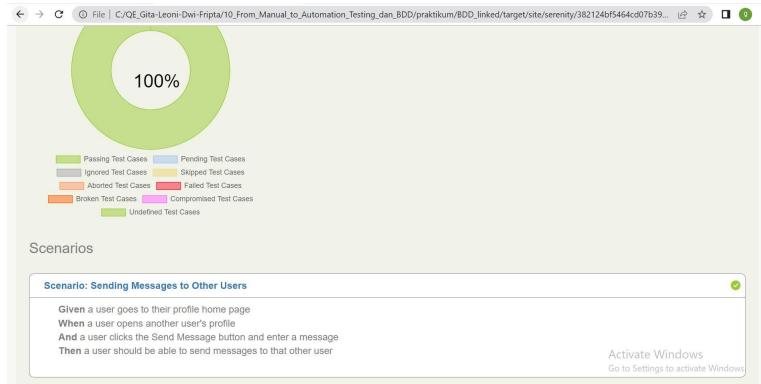
Link report

```
[INFO] | Tests with errors | 0
[INFO] | Tests compromised | 0
[INFO] | Tests aborted | 0
[INFO] | Tests pending | 0
[INFO] | Tests ignored/skipped | 0
[INFO] -----
[INFO] | Total Duration| is 236ms
[INFO] | Fastest test took| 179ms
[INFO] | Slowest test took| is 057ms
[INFO] -----
[INFO] SERENITY REPORTS
[INFO] Full Report: file:///C:/QE_Gita-Leoni-Dwi-Fripta/10_From_Manual_to_Automation_Testing_dan_BDD/praktikum/BDD_linked/target/site/serenity/index.html
[INFO] -----
[INFO] ... maven-failsafe-plugin:3.0.0-M5:verify (default) @ BDD_Linked ...
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 42.722 s
[INFO] Finished at: 2023-03-28T21:21:45+07:00
[INFO] -----
PS C:\QE_Gita-Leoni-Dwi-Fripta\10_From_Manual_to_Automation_Testing_dan_BDD\praktikum\BDD_linked>
```

Hasil report menjelaskan jika scenario yang dibuat pada feature setelan di run akan berberstatus passed

file:///C:/QE_Gita-Leoni-Dwi-Fripta/10_From_Manual_to_Automation_Testing_dan_BDD/praktikum/BDD_linked/target/site/serenity/576946480b52ad056d6f5bddf874399c83582ecf90963cc074a14c70580e7d9f.html





Soal Prioritas 2 (20)

1. Lakukan testing pada aplikasi <https://www.sepulsa.com/> dengan langkah-langkah sebagai berikut:
 1. Buatlah scenario test menggunakan format BDD pada fitur login, pilih produk dan pilih metode pembayaran.
 2. Buatlah project baru menggunakan intelliJ.
 3. Buatlah file feature Cucumber
 4. Buatlah class runner dan step
 5. Buatlah class steps yang berisi method-method yang dijalankan dalam scenario test.
 6. Jalankan class runner.
 7. Tampilkan report HTML dari Cucumber.
1. Pom.xml (project object model) adalah file konfigurasi pada proyek java yang digunakan untuk mengelola dependensi dan konfigurasi proyek. Pada intellij, POM dapat digunakan untuk proyek BDD(Behavior Driven Development) dengan menggunakan bahasa gherkin.ini adalah contoh gambar code POM pada projek BDD sepulsa.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
    modelVersion="4.0.0">

    <groupId>rg.alta</groupId>
    <artifactId>BDD_Sepulsa</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <serenity.version>3.5.1</serenity.version>
        <encoding>UTF-8</encoding>
        <tags></tags>
        <webdriver.base.url>http://www.sepulsa.com</webdriver.base.url>
    </properties>

    <dependencies>
        <dependency>
            <groupId>net.serenity-bdd</groupId>
            <artifactId>serenity-core</artifactId>
            <version>$(serenity.version)</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

```

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
    modelVersion="4.0.0">

    <dependencies>
        <dependency>
            <groupId>net.serenity-bdd</groupId>
            <artifactId>serenity-core</artifactId>
            <version>$(serenity.version)</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>net.serenity-bdd</groupId>
            <artifactId>serenity-cucumber</artifactId>
            <version>$(serenity.version)</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>net.serenity-bdd</groupId>
            <artifactId>serenity-playwright</artifactId>
            <version>$(serenity.version)</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>net.serenity-bdd</groupId>
            <artifactId>serenity-webdriver</artifactId>
            <version>$(serenity.version)</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

```

2. Setelah itu kita langsung membuat folder feature yang di dalamnya terdapat file login.feature, payment.feature dan product.feature. Login.feature dkk adalah file yang digunakan dalam BDD untuk menulis skenario pengujian menggunakan bahasa gherkin
- payment.feature

```

Feature: product sepulsa
  Scenario: Select a product
    Given I am logged in
    When I navigate to the products page
    And select a product
    Then I should see the product details
  
```

Login.feature

```

Feature: login ke sepulsa
  As a user
  I want to buy products
  I want to login, select a product, and choose a payment method

  Scenario: Login with valid credentials
    Given I am on the login page
    When I enter my valid email and password
    And click on the login button
    Then I should be redirected to the home page
  
```

Product.feature

```

Feature: product sepulsa
  Scenario: Select a product
    Given I am logged in
    When I navigate to the products page
    And select a product
    Then I should see the product details
  
```

3. Setelah membuat file feature selanjutnya adalah membuat java class login, product dan payment pada package runner sepulsa. folder runner dapat digunakan untuk menyimpan skrip pengujian yang digunakan untuk menjalankan pengujian BDD. Skrip pengujian dapat dijalankan secara otomatis oleh alat pengujian atau oleh pengguna secara manual.

login

```

package Runner.sepulsa;
import net.thucydides.core.annotations.Step;

public class Login {
    @Step("I am on the login page")
    public void LoginPage(){System.out.println("I am on the login page");}

    @Step("I enter my valid email and password")
    public void NameAndPassword(){System.out.println("I enter my valid email and password");}

    @Step("Click on the login button")
    public void ClickButton(){System.out.println("click on the login button");}

    @Step("I should be redirected to the home page")
    public void HomeSepulsa(){System.out.println("I should be redirected to the home page");}
}

```

Product

```

import net.thucydides.core.annotations.Step;

public class product {
    @Step("I logged in")
    public void Login(){System.out.println("I am logged in");}

    @Step("I navigate to the products page")
    public void Navigate(){System.out.println("I navigate to the products page");}

    @Step("select a product")
    public void SelectProduct(){System.out.println("select a product");}

    @Step("I should see the product details")
    public void ProductSepulsa(){System.out.println("I should see the product details");}
}

```

Payment

```

import net.thucydides.core.annotations.Step;

public class payment {
    @Step("Considering I have added product then go to history page")
    public void ProductRiwayat(){System.out.println("Considering I have added product then go to history page");}

    @Step("I navigate to the checkout page")
    public void NavigatedCheckout(){System.out.println("I navigate to the checkout page");}

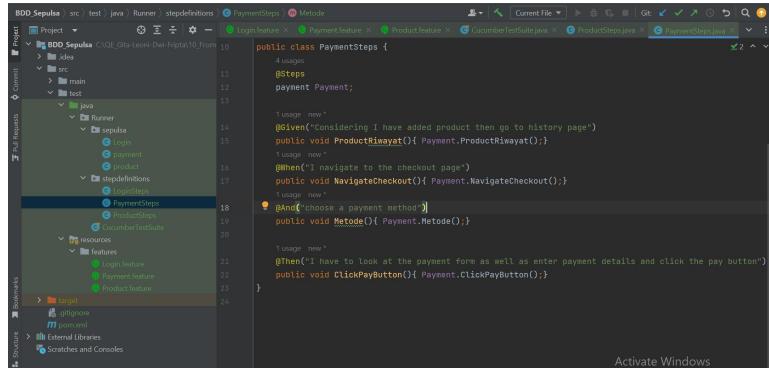
    @Step("choose a payment method")
    public void Metode(){System.out.println("choose a payment method");}

    @Step("I have to look at the payment form as well as enter payment details and click the pay button")
    public void ClickPayButton(){System.out.println("I have to look at the payment form as well as enter payment details and click the pay button");}
}

```

4. Setelah itu membuat folder stepdefinition yaitu untuk tempat scenario yang sama dengan feature dan memanggil file step yang tadi sudah ada pada code di file runner sepulsa.

payment.Steps



```

public class PaymentSteps {
    @Steps
    payment Payment;

    @Usage("new")
    @Given("Considering I have added product then go to history page")
    public void ProductRiwayat(){ Payment.ProductRiwayat(); }

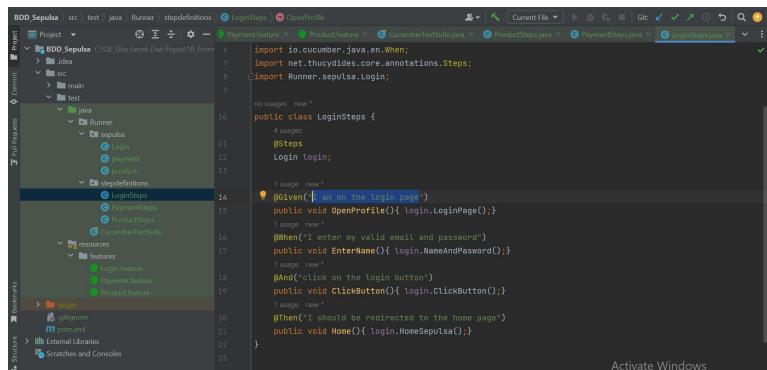
    @Usage("now")
    @When("I navigate to the checkout page")
    public void NavigateCheckout(){ Payment.NavigateCheckout(); }

    @Usage("now")
    @And("choose a payment method")
    public void Metode(){ Payment.Metode(); }

    @Usage("now")
    @Then("I have to look at the payment form as well as enter payment details and click the pay button")
    public void ClickPayButton(){ Payment.ClickPayButton(); }
}

```

login.Steps



```

import io.cucumber.java.en.When;
import net.thucydides.core.annotations.Steps;
import Runner.sepulsa.Login;

no usages: new

public class LoginSteps {
    @Steps
    Login login;

    @Usage("new")
    @Given("I am on the login page")
    public void OpenProfile(){ login.LoginPage(); }

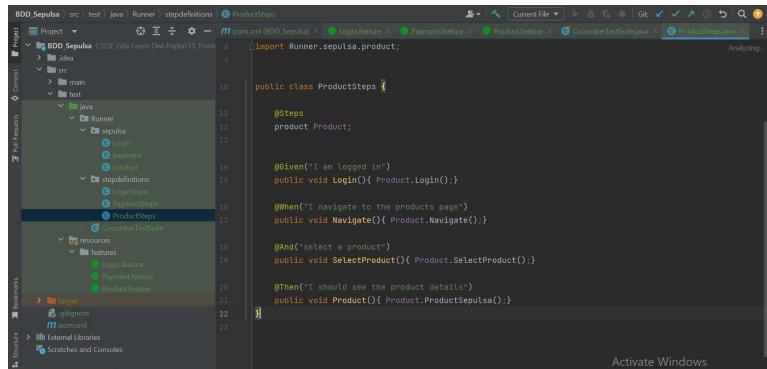
    @Usage("new")
    @When("I enter my valid email and password")
    public void EnterName(@Login.NameAndPassword()); }

    @Usage("now")
    @And("click on the login button")
    public void ClickButton(){ login.ClickButton(); }

    @Usage("now")
    @Then("I should be redirected to the home page")
    public void Home(){ login.HomeSepulsa(); }
}

```

ProductSteps



```

import Runner.sepulsa.product;

public class ProductSteps {
    @Steps
    product Product;

    @Usage("now")
    @Given("I am logged in")
    public void Login(){ Product.Login(); }

    @Usage("now")
    @When("I navigate to the products page")
    public void Navigate(){ Product.Navigate(); }

    @Usage("now")
    @And("select a product")
    public void SelectProduct(){ Product.SelectProduct(); }

    @Usage("now")
    @Then("I should see the product details")
    public void Product(){ Product.ProductSepulsa(); }
}

```

5. Selanjutnya Membuat java class CucumberTestSuit yaitu java class untuk menjalankan featur yang ada pada project.

```

package Runner;
import io.cucumber.junit.CucumberOptions;
import net.serenitybdd.cucumber.CucumberWithSerenity;
import org.junit.runner.RunWith;

public class Runner {
    @RunWith(CucumberJunitRunner.class)
    @CucumberOptions(
            plugin = {"pretty"},
            features = "src/test/resources/features"
    )
    public class CucumberTestSuite {
        public CucumberTestSuite(FeatureLoader featureLoader) {
        }
    }
}

```

6. Setelah selesai semua tahap scenario BDD selanjutnya yaitu run project menggunakan terminal dengan mengetik mvn clean verify. Setelah itu akan muncul test started dan tes passed seperti gambar dibawah ini.

Product passed

```

When I navigate to the products page # starter.stepDefinitions.ProductSteps.Navigate()
I should see the product details
Then I should see the product details
1 of 1 scenarios passed

```

Payment passed

```

And choose a payment method # starter.stepDefinitions.PaymentSteps.Method()
I have to look at the payment form as well as enter payment details and click the pay button
Then I have to look at the payment form as well as enter payment details and click the pay button
1 of 1 scenarios passed

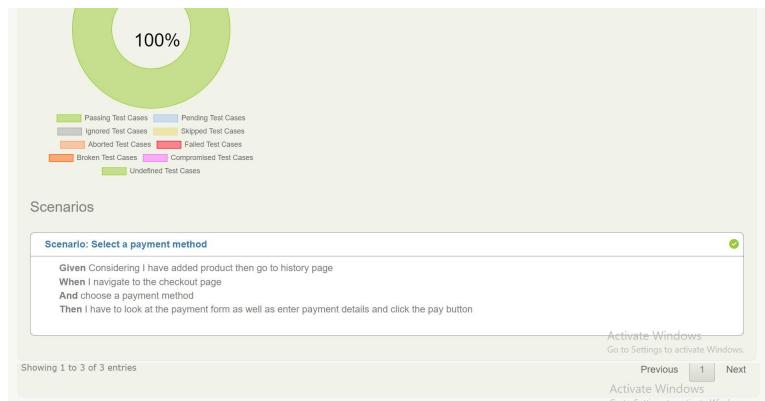
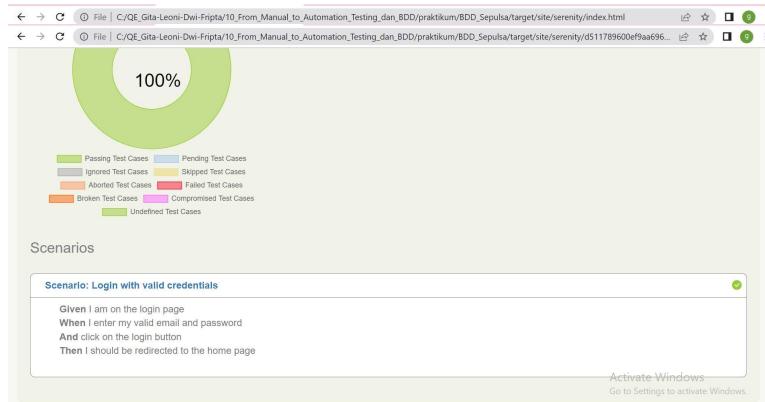
```

Login passed

```
20:59:26.488 [main] DEBUG net.thucydides.core.steps.StepInterceptor - STARTING STEP: LoginSteps.ClickButton - click on the login button
20:59:26.689 [main] DEBUG net.thucydides.core.steps.StepInterceptor - STEP DONE: click on the login button
20:59:26.710 [main] DEBUG net.thucydides.core.steps.StepInterceptor - STARTING STEP: LoginSteps.Home - I should be redirected to the home page
20:59:26.713 [main] DEBUG net.thucydides.core.steps.StepInterceptor - STEP DONE: I should be redirected to the home page
20:59:26.725 [main] INFO -
Then I should be redirected to the home page # stepDefinitions.LoginSteps.Home()
20:59:26.729 [main] INFO -
-----  
Login with valid credentials-----  
20:59:26.747 [main] INFO io.cucumber.core.plugin.SerenityReporter - Running Feature from file:///C:/QE_Gita-Leoni-Dwi-Fripta/10_From_Manual_to_Automation_Testing_dan_BDD/praktikum/BDD_Sepulsa/src/test/resources/features/Payment.feature  
20:59:26.748 [main] DEBUG net.thucydides.core.steps.StepEventBus - Test suite started for story net.thucydides.core.model.Story@537a1de8  
20:59:26.767 [main] INFO - Test Suite Started: Payment  
Scenario: Select a payment method # src/test/resources/features/Payment.feature:3  
20:59:26.845 [main] INFO io.cucumber.core.plugin.SerenityReporter - Running Feature from file:///C:/QE_Gita-Leoni-Dwi-Fripta/10_From_Manual_to_Automation_Testing_dan_BDD/praktikum/BDD_Sepulsa/target/site/serenity/576946480b52ad056d6f5bddf874399c83582ecf90963cc074a14c70580e7d9f.html
```

Hasil report menjelaskan jika scenario yang dibuat pada feature setelan di run akan berberstatus passed

file:///C:/QE_Gita-Leoni-Dwi-Fripta/10_From_Manual_to_Automation_Testing_dan_BDD/praktikum/BDD_Sepulsa/target/site/serenity/576946480b52ad056d6f5bddf874399c83582ecf90963cc074a14c70580e7d9f.html



Feature: Login Ke Sepulsa

As a user
In order to buy products
I want to login, select a product, and choose a payment method

Specifications Test Results

Feature Coverage By Scenario

100%

Legend: Passing Test Cases (Green), Pending Test Cases (Light Blue), Ignored Test Cases (Grey), Skipped Test Cases (Yellow), Aborted Test Cases (Orange), Failed Test Cases (Red), Broken Test Cases (Pink), Compromised Test Cases (Purple), Undefined Test Cases (Dark Green)

Overview

Scenario: Login with valid credentials

Activate Windows
Go to Settings to activate Windows.

File | C:/QE_Gita-Leoni-Dwi-Fripta/10_From_Manual_to_Automation_Testing_dan_BDD/praktikum/BDD_Sepulsa/target/site/serenity/8743079cae419b74...

100%

Legend: Passing Test Cases (Green), Pending Test Cases (Light Blue), Ignored Test Cases (Grey), Skipped Test Cases (Yellow), Aborted Test Cases (Orange), Failed Test Cases (Red), Broken Test Cases (Pink), Compromised Test Cases (Purple), Undefined Test Cases (Dark Green)

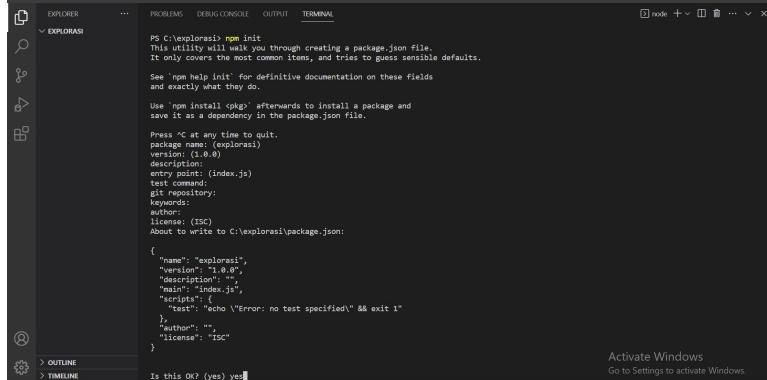
Scenarios

Scenario: Select a product

Given I am logged in
When I navigate to the products page
And select a product
Then I should see the product details

Activate Windows
Go to Settings to activate Windows.

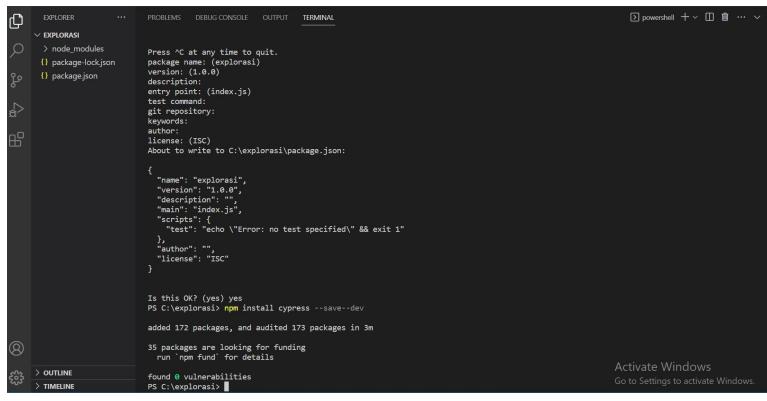
Soal Eksplorasi (20)

1. Lakukan testing pada aplikasi <https://www.sepulta.com/> dengan langkah-langkah sebagai berikut:
 1. Menggunakan **cypress** sebagai tools untuk melakukan testing. Referensi penggunaan BDD dengan cypress dapat dilihat [disini](#).
 2. Buatlah scenario test menggunakan format BDD pada fitur login, pilih produk dan pilih metode pembayaran.
 3. Tampilkan hasil testing dengan menggunakan cypress.
1. Membuat folder baru di file explore setelah itu membuka visual studio code dan open folder. Setelah itu kita lalu ketik di terminal npm init fungsi tersebut adalah perintah yang digunakan untuk membuat proyek Node.js baru dan membuat file package.json.

```
PS C:\explorasi> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.
See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Press  at any time to quit.
package name: (explorasi)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\explorasi\package.json:

{
  "name": "explorasi",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\"Warning: no test specified!\\& exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes) yes
```
2. Setelah itu kita mengetik di terminal npm install cypress --save-dev perintah tersebut digunakan untuk Cypress sebagai dependensi pengembangan untuk proyek node.js. Setelah menjalankan perintah tersebut cypress akan di install di direktori node_modules proyek dan informasi baru akan ditambahkan ke dalam file package.json.

```
PS C:\explorasi> npm install cypress --save-dev
added 172 packages, and audited 173 packages in 3s
35 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
PS C:\explorasi>
```

- Setelah itu kita mengetik lagi perintah `npm install --save-dev cypress-cucumber-preprocessor`. Perintah tersebut berfungsi untuk menginstal dan menyimpan cypress cucumber sebagai dependensi pengembangan di proyek.js. Dan perintah `npx cypress open` adalah untuk membuka cypresnya.

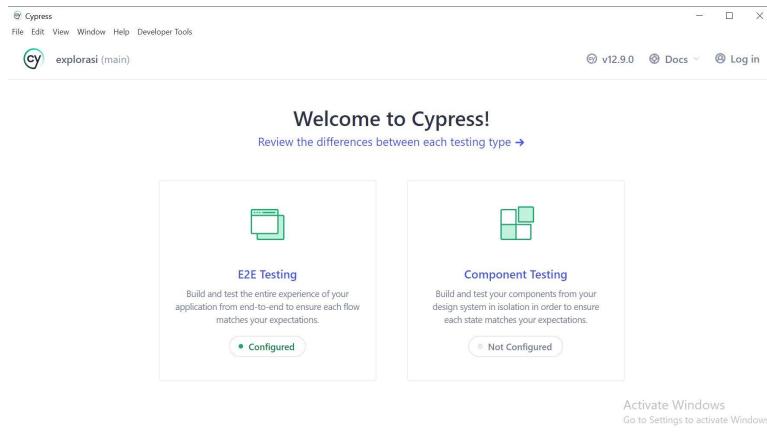
```

    run "npm fund" for details
    found 8 vulnerabilities
    PS C:\explorasi> npm install --save-dev cypress-cucumber-preprocessor
    npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy, new code should use the URLSearchParams API instead.
    npm WARN deprecated gherkin@5.1.0: This package is transitioning under @cucumber/gherkin
    npm WARN deprecated gherkin@5.0.0: This package is transitioning under @cucumber/gherkin
    npm WARN deprecated cucumber-expressions@6.6.2: This package is now published under @cucumber/cucumber-expressions
    npm WARN deprecated cucumber@9.2.1: Cucumber is publishing new releases under @cucumber/cucumber
    npm WARN deprecated core-js@3.16.4: core-js has been deprecated. Please use esm or another modern alternative for usage due to the number of issues. Because use of the V8 engine whims, feature detection in old core-js versions could cause a slowdown up to 10x even if nothing is polyfilled. Some versions have well compatibility issues. Please, upgrade your dependencies to the actual version of core-js.

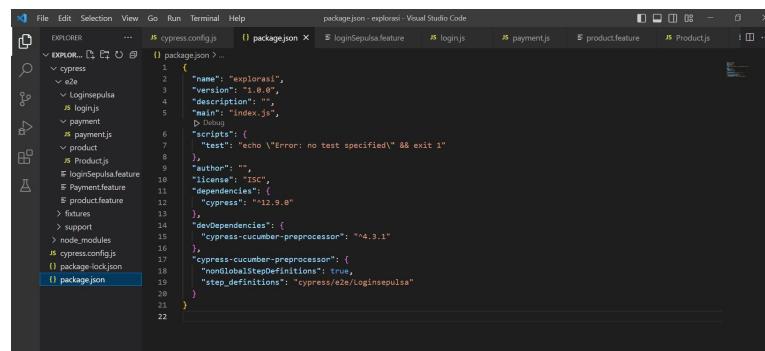
    added 404 packages, and audited 577 packages in 57s

    56 packages are looking for funding
    run "npm fund" for details
    found 8 vulnerabilities
    PS C:\explorasi> npx cypress open
    It looks like this is your first time using Cypress: 12.9.0
    ✓ Verified Cypress! C:\Users\GITA\AppData\Local\Cypress\Cache\12.9.0\Cypress
    Opening Cypress...
    GET /_/ 200 84.934 ms -
    GET /_/listAccount 200 1ms
    GET /_/resourceContext 200 1ms
    GET /_/resourceContexts 200 1ms
    GET /_/resourceContexts@nonstandard 200 893.389 ms -
    GET /_/resourceContexts@nonstandard? 200 1163.533 ms -
    GET /_/index-48d1c9ff.css 200 212.879 ms
    GET /_/_/polyfills-50ed1593.js 200 368.562 ms -
    GET /_/_/main-27e0345f.js 200 1139.278 ms -
    GET /_/_/cypressrunner.js 200 28.488 ms -
    GET /_/_/specs-4f97a79.js 200 139.707 ms - 485
    GET /_/_/assets/2d6c1cd4735... 200 124.257 ms
    GET /_/_/resources/404.html 200 117.904 ms - 489
  
```

- Setelah itu akan muncul tampilan cypress dan langsung pilih e2e. End to End (E2E) testing pada cypress bertujuan untuk menguji bagian aplikasi secara keseluruhan dari permulaan sampai akhir melalui pengujian jalur yang diambil oleh pengguna dan interaksi.

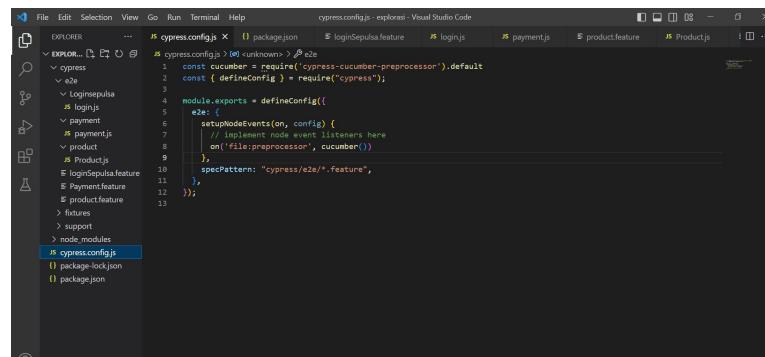


5. Setelah itu kita menambahkan code dari line 17 sampai 20 di file package.json



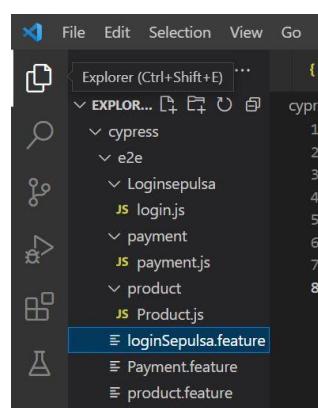
```
1   "name": "Explorasi",
2   "version": "1.0.0",
3   "description": "",
4   "main": "index.js",
5   "bin": {},
6   "scripts": {
7     "test": "echo \"Error: no test specified\" & exit 1"
8   },
9   "author": "",
10  "license": "ISC",
11  "dependencies": {
12    "cypress": "12.9.0"
13  },
14  "devDependencies": {
15    "cypress-cucumber-preprocessor": "^4.3.1"
16  },
17  "cypress-cucumber-preprocessor": {
18    "nonGlobalStepDefinitions": true,
19    "step_definitions": "cypress/e2e/Loginsepulta"
20  }
21}
22}
```

6. Setelah itu menambahkan code pada line 8 sampai 10



```
1 const cucumber = require('cypress-cucumber-preprocessor').default
2 const { defineConfig } = require('cypress')
3
4 module.exports = defineConfig({
5   e2e: {
6     setupBeforeEvents(on, config) {
7       // Implement mode event listeners here
8       on('file:preprocessor', cucumber())
9     },
10    specPattern: 'cypress/e2e/*.feature',
11  },
12});
13
```

7. Setelah itu langsung membuat folder loginsepulta, payment, dan product di e2e.



8. Setelah membuat foldernya sekarang kita membuat file feature dalam folder yang telah dibuat di e2e.

The image consists of three vertically stacked screenshots of the Visual Studio Code interface, each showing a different feature file being edited:

- Screenshot 1 (Top):** Shows the `product.feature` file open in the editor. The code defines a feature named "Select Product" with one scenario: "Given I am logged in When I navigate to the products page And select a product Then I should see the product details".
- Screenshot 2 (Middle):** Shows the `loginSepulsa.feature` file open in the editor. The code defines a feature named "Login" with one scenario: "As a user Given I am on the login page When I enter my valid email and password And click on the login button Then I should be redirected to the home page".
- Screenshot 3 (Bottom):** Shows the `Payment.feature` file open in the editor. The code defines a feature named "Select Payment Method" with one scenario: "Select After I have added the product Then go to the history page When I navigate to the checkout page And choose a payment method Then I have to look at the payment form as well as enter payment details click the pay button".

In all three screenshots, the left sidebar shows the project structure with folders like `e2e`, `cypress`, `node_modules`, and `support`, along with files like `package.json`, `cypress.config.js`, and `package-lock.json`.

9. Setelah itu membuat file json pada setiap folder loginsepulta, product, payment yang telah dibuat.

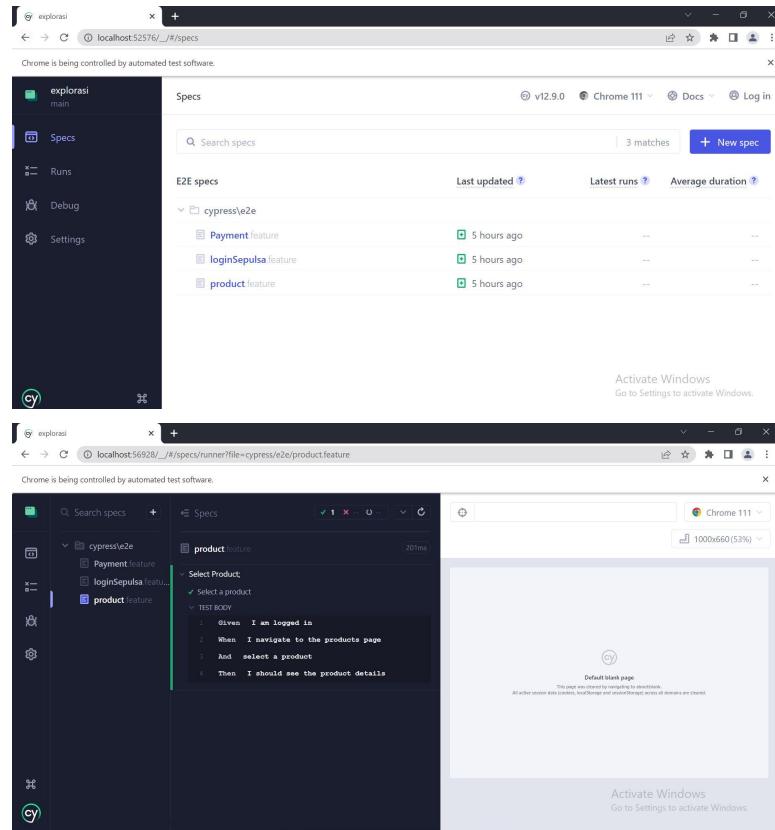
```
Product.js - explorasi - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER ... js Product.js
cypress > e2e > product > js Product.js > ...
1 import { And, Given, Then, When } from "cypress-cucumber-preprocessor/steps";
2
3 Given("I am logged in",function(){
4   console.log("I am logged in")
5 });
6 When("I navigate to the products page",function(){
7   console.log("I navigate to the products page")
8 });
9 And("select a product",function(){
10   console.log("select a product")
11 });
12 Then("I should see the product details",function(){
13   console.log("I should see the product details")
14 });

login.js - explorasi - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER ... js login.js
cypress > e2e > Loginsepulta > js login.js > ...
1 import { And, Given, Then, When } from "cypress-cucumber-preprocessor/steps";
2
3 Given("I am on the login page", function(){
4   console.log("I am on the login page")
5 });
6 When("I enter my valid email and password", function(){
7   console.log("I enter my valid email and password")
8 });
9 And("click on the login button",function(){
10   console.log("click on the login button")
11 });
12 Then("I should be redirected to the home page",function(){
13   console.log("I should be redirected to the home page")
14 });

payment.js - explorasi - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER ... js payment.js
cypress > e2e > payment > js payment.js > ...
1 import { And, Given, Then, When } from "cypress-cucumber-preprocessor/steps";
2
3 Given("After I have added the product then go to the history page",function(){
4   console.log("After I have added the product then go to the history page")
5 });
6 When("I navigate to the checkout page",function(){
7   console.log("I navigate to the checkout page")
8 });
9 And("choose a payment method",function(){
10   console.log("choose a payment method")
11 });
12 Then("I have to look at the payment form as well as enter payment details click the pay button",function(){
13   console.log("I have to look at the payment form as well as enter payment details click the pay button")
14 });


```

10. Setelah itu di run di terminal menggunakan cmd dengan mengetik npx Cypress open



The screenshot displays the Cypress Explorasi interface with two browser tabs open in Chrome 111.

Top Tab: Localhost:56928/#/specs/runner?file=cypress/e2e/loginSepulsa.feature

Chrome is being controlled by automated test software.

Specs:

- cypress/e2e
- Payment feature
- loginSepulsa feature
- product.feature

loginSepulsa feature (208ms)

Log in:

✓ Login with valid credentials

TEST BODY

- Given I am on the login page
- When I enter my valid email and password
- And click on the login button
- Then I should be redirected to the home page

Bottom Tab: Localhost:56928/#/specs/runner?file=cypress/e2e/Payment.feature

Chrome is being controlled by automated test software.

Specs:

- cypress/e2e
- Payment feature
- loginSepulsa feature
- product.feature

Payment feature (316ms)

Select Payment Method:

✓ Choose a payment method

TEST BODY

- Given After I have added the product then go to the history page
- When I navigate to the checkout page
- And choose a payment method
- Then I have to look at the payment form as well as enter payment details click the pay button

Activate Windows
Go to Settings to activate Windows.