

**PENGUJIAN WEBSITE COFFESHOP MENGGUNAKAN METODE  
BLACK BOX DAN WHITE BOX**



**Disusun Oleh :**

Zaki Fauzan Rabbani

2109116028

Muhammad Nauvaldi Caesar

2109116040

**PRODI STUDI SISTEM INFORMASI FAKULTAS TEKNIK  
UNIVERSITAS MULAWARMAN**

**2024**

# DAFTAR ISI

<b>DAFTAR ISI.....</b>	<b>2</b>
<b>DAFTAR GAMBAR.....</b>	<b>3</b>
<b>DAFTAR TABLE .....</b>	<b>4</b>
<b>BAB I PEMBAHASAN TEORI .....</b>	<b>5</b>
1.1. Black Box Testing.....	5
A. Pengertian Black Box Testing.....	5
B. Sejarah Metode Black Box.....	5
C. Konsep Utama Pengujian Metode Black Box.....	6
1.2. White Box Testing .....	6
A. Pengertian Whitebox Testing .....	6
B. Cyclomatic Complexity.....	7
C. Sejarah Whitebox Testing .....	7
D. Konsep Utama Pengujian Metode White Box .....	8
<b>BAB II UNIT TESTING .....</b>	<b>10</b>
2.1. Analisis Data .....	10
2.2. Hasil Testing .....	11
A. Pengujian Whitebox .....	11
B. Pengujian Blackbox.....	27
<b>BAB III PENUTUP.....</b>	<b>32</b>
3.1. Kesimpulan.....	32

## DAFTAR GAMBAR

Gambar 1 Source Code Regis Form .....	11
Gambar 2 Source Code Regis Form .....	11
Gambar 3 Flow Chart Regis .....	12
Gambar 4 Flowgraph Regis .....	13
Gambar 5 Source Code Login Form .....	14
Gambar 6 Source Code Login Form .....	15
Gambar 7 Source Code Login Form .....	15
Gambar 8 Flowchart Login Form .....	16
Gambar 9 Flowgraph Login Form .....	17
Gambar 10 Source Code Keranjang .....	18
Gambar 11 Source Code Keranjang .....	19
Gambar 12 Source Code Keranjang .....	19
Gambar 13 Flowchart Keranjang.....	20
Gambar 14 Flowgraph Keranjang.....	21
Gambar 15 Source Code Kasir .....	22
Gambar 16 Source Code Kasir .....	23
Gambar 17 Flowchart Kasir.....	24
Gambar 18 Flowgraph Kasir.....	25
Gambar 19 Tampilan Regis Form.....	27
Gambar 20 Tampilan Login Form .....	27
Gambar 21 Tampilan Jika Akun belum terdaftar .....	28
Gambar 22 Tampilan Halaman Home .....	28
Gambar 23 Tampilan Halaman Kasir .....	29
Gambar 24 Tampilan Halaman Kasir jika pembayaran gagal .....	29
Gambar 25 Halaman Kasir jika pembayaran berhasil .....	30

## DAFTAR TABLE

Table 1 Table Blackbox Testing .....	31
--------------------------------------	----

# **BAB I**

## **PEMBAHASAN TEORI**

### **1.1. Black Box Testing**

#### **A. Pengertian Black Box Testing**

Metode Black Box adalah teknik pengujian perangkat lunak yang menilai fungsi dari suatu aplikasi tanpa melihat struktur internal atau kode sumbernya. Dalam pendekatan ini, penguji hanya berinteraksi dengan antarmuka pengguna, memberikan input, dan memeriksa output yang dihasilkan oleh sistem untuk memastikan bahwa fungsi berjalan sesuai dengan yang diharapkan. Pengujian ini didasarkan pada spesifikasi perangkat lunak dan persyaratan fungsional tanpa pengetahuan mendalam tentang bagaimana sistem tersebut diimplementasikan.

#### **B. Sejarah Metode Black Box**

Metode Black Box telah berkembang seiring dengan perkembangan ilmu komputer dan rekayasa perangkat lunak. Berikut adalah garis waktu yang menjelaskan evolusi metode ini:

##### **1. Awal Komputer (1950-an hingga 1960-an)**

Pada awal pengembangan komputer, pengujian perangkat lunak dilakukan secara ad hoc dan tidak terstruktur. Metode Black Box muncul sebagai salah satu teknik formal pertama dalam pengujian perangkat lunak. Pada era ini, komputer digunakan untuk tujuan komputasi sederhana dan pengujian berfokus pada memastikan bahwa fungsi-fungsi dasar bekerja dengan benar.

##### **2. Pengembangan Metodologi Pengujian (1970-an)**

Pada tahun 1970-an, kebutuhan untuk metodologi pengujian yang lebih sistematis mulai dirasakan. Istilah "Black Box Testing" mulai digunakan secara lebih formal. Glenford J. Myers mempopulerkan istilah ini dalam bukunya "The Art of Software Testing" yang diterbitkan pada tahun 1979. Myers menekankan pentingnya pengujian dari perspektif pengguna akhir dan memperkenalkan teknik-teknik seperti pengujian partisi ekuivalen dan analisis nilai batas.

##### **3. Standar dan Praktik Terbaik (1980-an hingga 1990-an)**

Pada dekade-dekade ini, standar industri seperti IEEE (Institute of Electrical and Electronics Engineers) mulai memperkenalkan pedoman dan praktik terbaik untuk pengujian perangkat lunak. Metode Black Box menjadi komponen utama dari siklus pengembangan perangkat lunak yang lebih formal. Organisasi mulai mengakui pentingnya pengujian sistematis untuk memastikan kualitas perangkat lunak.

#### 4. Rekayasa Perangkat Lunak Modern (2000-an hingga sekarang)

Dalam era ini, metode Black Box berkembang seiring dengan metodologi pengembangan perangkat lunak seperti Agile dan DevOps. Pengujian otomatisasi menjadi lebih umum, dan alat serta kerangka kerja pengujian yang mendukung metode Black Box, seperti Selenium dan JUnit, mulai digunakan secara luas. Selain itu, munculnya konsep seperti continuous integration/continuous deployment (CI/CD) membuat pengujian Black Box menjadi bagian integral dari pipeline pengembangan perangkat lunak.

### C. Konsep Utama Pengujian Metode Black Box

#### 1. Pengujian Fungsi (Functional Testing):

Fokus pada pengujian fungsi spesifik dari aplikasi berdasarkan spesifikasi yang diberikan. Teknik ini termasuk pengujian unit, pengujian integrasi, dan pengujian sistem.

#### 2. Pengujian Non-Fungsional (Non-Functional Testing):

Menguji aspek-aspek non-fungsional dari aplikasi, seperti performa, keandalan, dan keamanan.

#### 3. Teknik Pengujian:

- Partisi Ekuivalen (Equivalence Partitioning): Membagi input data ke dalam partisi-partisi yang diharapkan diperlakukan sama oleh sistem.
- Analisis Nilai Batas (Boundary Value Analysis): Menguji batas-batas dari partisi ekuivalen.
- Pengujian Berdasarkan Kasus Penggunaan (Use Case Testing): Menguji skenario spesifik dari perspektif pengguna.

### 1.2. White Box Testing

#### A. Pengertian Whitebox Testing

White box testing adalah metode pengujian perangkat lunak yang melibatkan pemeriksaan langsung dari kode sumber dan struktur internal sistem yang diuji. Ini bertujuan

untuk memastikan bahwa semua jalur logika di dalam program berfungsi sebagaimana mestinya. Sejarahnya dimulai pada awal pengembangan perangkat lunak dan terus berkembang seiring dengan kompleksitas teknologi.

## **B. Cyclomatic Complexity**

Siklomatik adalah bagian penting dari white box testing yang berkaitan dengan pengujian semua jalur logika dalam kode program. Ini memastikan bahwa setiap jalur instruksi diuji untuk mengidentifikasi kegagalan potensial atau kesalahan logika. Dengan demikian, siklomatik membantu meningkatkan keandalan dan kualitas perangkat lunak.

## **C. Sejarah Whitebox Testing**

White Box Testing telah berkembang seiring dengan perkembangan perangkat lunak dan kebutuhan akan pengujian yang lebih efektif. Beberapa poin penting dalam sejarah White Box Testing meliputi:

### **1. Awal Perkembangan:**

Pada tahun-tahun awal pengembangan perangkat lunak, pengujian dilakukan secara manual oleh pengembang yang sama yang menulis kode. Mereka menggunakan pengetahuan mereka tentang struktur kode untuk menemukan dan memperbaiki kesalahan.

### **2. Evolusi Metodologi:**

Seiring berkembangnya metode pengembangan perangkat lunak, metodologi pengujian juga berevolusi. Pada tahun 1970-an dan 1980-an, konsep dan teknik formal mulai diperkenalkan, seperti pengujian berbasis alur kontrol (control flow testing) dan alur data (data flow testing).

### **3. Pengujian Berbasis Komputer:**

Dengan munculnya alat bantu komputer dan bahasa pemrograman yang lebih canggih, White Box Testing mulai menggunakan perangkat lunak khusus untuk otomatisasi pengujian dan analisis kode.

### **4. Standarisasi dan Praktik Terbaik:**

Seiring dengan semakin pentingnya kualitas perangkat lunak, standar industri dan praktik terbaik untuk White Box Testing dikembangkan. Ini termasuk penggunaan metrik seperti cakupan kode (code coverage) dan siklomatik kompleksitas.

#### **D. Konsep Utama Pengujian Metode White Box**

Pengujian metode white box, juga dikenal sebagai pengujian struktural atau pengujian kode terbuka, adalah pendekatan pengujian perangkat lunak di mana penguji memiliki pengetahuan penuh tentang struktur internal atau implementasi kode dari aplikasi yang diuji. Berikut adalah konsep utama dari pengujian metode white box:

1. **Pengetahuan Kode Internal:**

Penguji memiliki akses dan pengetahuan tentang kode sumber, algoritma, dan struktur data yang digunakan dalam aplikasi. Hal ini memungkinkan penguji untuk merancang tes yang lebih mendalam dan spesifik.

2. **Pengujian Aliran Kendali:**

Melibatkan pengujian jalur eksekusi dalam kode, termasuk semua cabang, loop, dan kondisi. Tujuannya adalah untuk memastikan bahwa setiap jalur eksekusi dalam program telah diuji setidaknya sekali.

3. **Pengujian Kondisi dan Cabang:**

Memastikan bahwa setiap kondisi logis (misalnya, if-else statements) dan setiap cabang dalam kode telah diuji. Ini membantu mengidentifikasi kesalahan yang mungkin terjadi dalam kondisi tertentu.

4. **Pengujian Jalur:**

Melibatkan identifikasi semua jalur yang mungkin diambil oleh program dari awal hingga akhir dan membuat kasus uji untuk setiap jalur tersebut. Pengujian ini bertujuan untuk mencakup semua kombinasi jalur eksekusi yang mungkin.

5. **Pengujian Loop:**

Fokus pada pengujian semua loop dalam kode. Ini termasuk pengujian loop dengan nol iterasi, satu iterasi, beberapa iterasi, dan batas maksimum iterasi untuk memastikan loop berfungsi dengan benar dalam semua skenario.

6. **Pengujian Pernyataan (Statement Coverage):**

Memastikan bahwa setiap baris kode dalam program dieksekusi setidaknya sekali selama pengujian. Ini membantu menemukan bagian kode yang tidak pernah dieksekusi dan mungkin mengandung bug.

7. **Pengujian Kondisi/Gabungan (Condition/Decision Coverage):**



Kombinasi dari pengujian kondisi dan pengujian keputusan untuk memastikan bahwa setiap kondisi dalam keputusan logis dan setiap cabang dalam keputusan telah diuji.

8. Pengujian Aliran Data:

Fokus pada variabel dan nilai data, memastikan bahwa setiap variabel diinisialisasi sebelum digunakan, dan nilai data diperbarui serta digunakan dengan benar di seluruh program.

9. Otomatisasi Pengujian:

Karena pengujian white box seringkali membutuhkan pengetahuan mendalam tentang kode dan dapat menjadi sangat kompleks, alat otomatisasi pengujian sering digunakan untuk menjalankan pengujian dengan efisien.

10. Deteksi Kesalahan Lebih Awal:

Pengujian white box memungkinkan deteksi kesalahan atau bug di tahap awal siklus pengembangan perangkat lunak karena fokusnya pada struktur internal program.

## **BAB II**

### **UNIT TESTING**

#### **2.1. Analisis Data**

Penelitian ini menggunakan data yang berasal dari kode sumber (source code) yang digunakan dalam pembuatan sebuah website untuk coffeeshop sederhana. Terdapat 11 file utama yang membentuk keseluruhan sistem website ini. Berikut adalah rincian file-file tersebut:

1. index.php: Berfungsi sebagai file utama yang menjadi pintu masuk (entry point) dari website. File ini memuat login form dan ketika berhasil akan mengarahkan pengguna ke halaman-halaman lain dalam sistem.
2. data.php: Berperan dalam mengelola data yang diperlukan oleh aplikasi, termasuk pengambilan dan penyimpanan data dari dan ke array session.
3. home.php: Menyediakan halaman beranda (home) yang dilihat pertama kali oleh pengguna saat mengakses website.
4. signup.php: Digunakan untuk menangani proses pendaftaran pengguna baru ke dalam sistem.
5. info.php: Menampilkan informasi terkait coffeeshop.
6. kasir.php: Merupakan halaman untuk kasir dalam mengelola transaksi penjualan di coffeeshop.
7. logout.php: Mengatur proses keluar (logout) dari sistem, sehingga sesi pengguna diakhiri dengan aman.
8. menu.php: Menampilkan daftar menu yang tersedia di coffeeshop.
9. style.css: Berisi kode CSS yang digunakan untuk mengatur tampilan dan tata letak website agar terlihat menarik dan konsisten.
10. keranjang.js: Mengelola fungsi-fungsi JavaScript yang berkaitan dengan keranjang belanja pengguna.
11. krj.js: Mengatur fungsi-fungsi JavaScript untuk keperluan kasir.

Pengujian dilakukan dengan menguji fungsi yang terdapat pada website ini dengan index.php sebagai file utama.

## 2.2. Hasil Testing

### A. Pengujian Whitebox

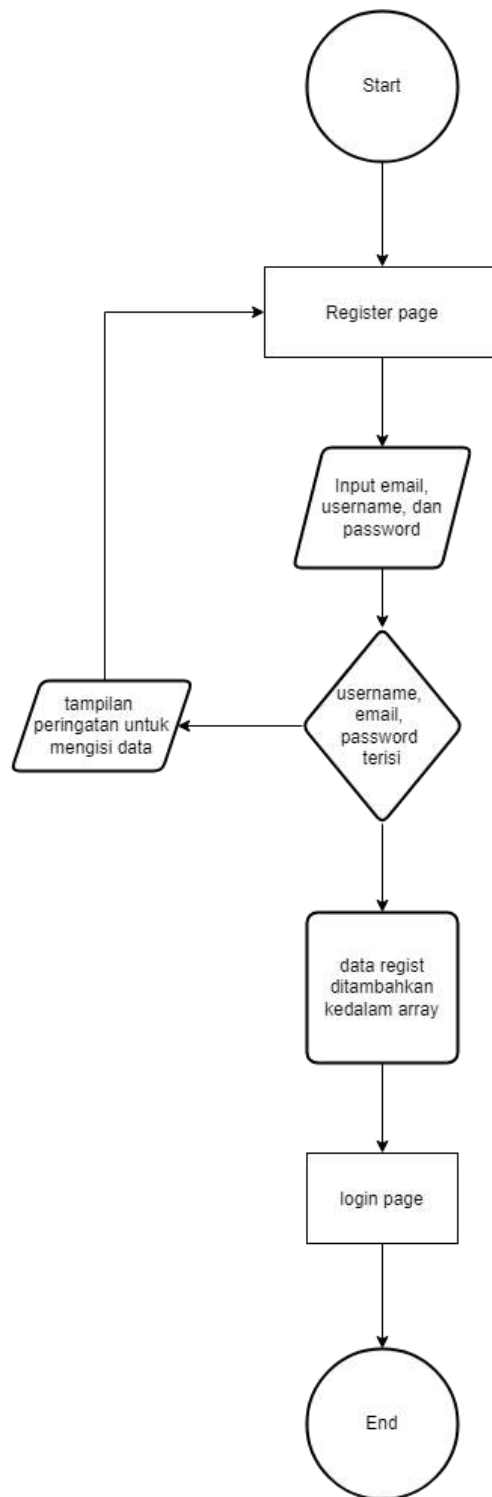
#### 1. Regist Form

```
signap.php > html > body > div > div > div.box2 > form
2  <html lang="en">
13 <body>
14 <header>
33 </header>
34 <?php
35     session_start();
36     $_SESSION['email'] = array();
37     $_SESSION['nama'] = array();
38     $_SESSION['pw'] = array();
39     if(isset($_POST['submit'])){
40         $email = $_POST['email'];
41         $username = $_POST['nama'];
42         $password = $_POST['password'];
43         array_push($_SESSION['email'], $email);
44         array_push($_SESSION['nama'], $username);
45         array_push($_SESSION['pw'], $password);
46         header("Location: index.php");
47     }
48 }
49 <div style="min-height: 90.5vh; background-image: url('assets/image/bg3.jpg'); background-size: cover;">
50 <div style="padding-top: 7%; display: flex; height: 100%; justify-content: center; align-items: center;">
51 <div class="box2">
52 <form method="POST">
53 <h1>Sign Up</h1>
54 <table class="table">
55 <tbody>
56 <tr>
57 <th scope="row">Email</th>
58 <td></td>
59 <td><input type="email" name="email" class="email"/></td>
60 </tr>
61 </tbody>
```

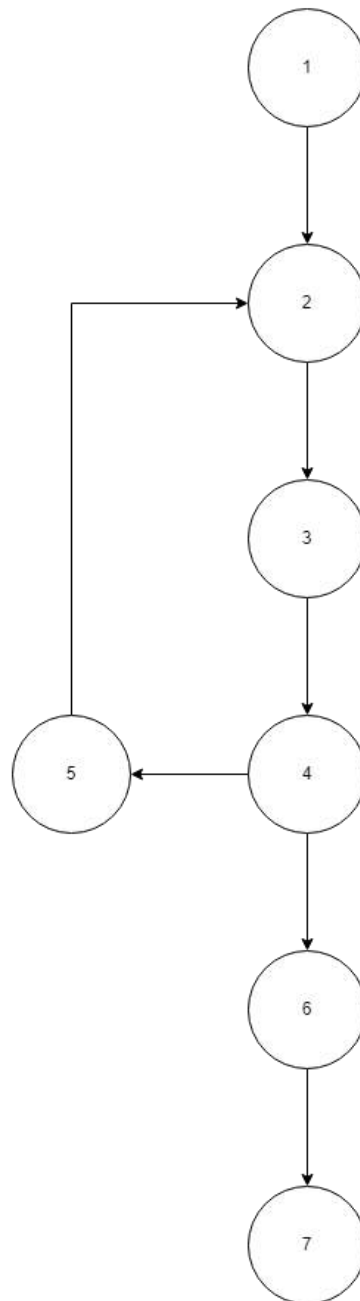
Gambar 1 Source Code Regist Form

```
signap.php > html > body > div > div > div.box2 > form > table
2  <html lang="en">
13 <body>
49 <div style="min-height: 90.5vh; background-image: url('assets/image/bg3.jpg'); background-size: cover;">
50 <div style="padding-top: 7%; display: flex; height: 100%; justify-content: center; align-items: center;">
51 <div class="box2">
52 <form method="POST">
53 <h1>Sign Up</h1>
54 <table class="table">
55 <tbody>
56 <tr>
57 <th scope="row">Email</th>
58 <td></td>
59 <td><input type="email" name="email" class="email"/></td>
60 </tr>
61 <tr>
62 <th scope="row">Username</th>
63 <td></td>
64 <td><input type="text" name="nama" class="email"/></td>
65 </tr>
66 <tr>
67 <th scope="row">Password</th>
68 <td></td>
69 <td><input type="password" name="password" class="email" /></td>
70 </tr>
71 </tbody>
72 </table>
73 <button style="margin-top: 0.5%; margin-bottom: 2%; width: auto;" type="submit" name="submit" class="btn bi">
74     Sign Up
75 </button>
76 </form>
77 </div>
78 </div>
```

Gambar 2 Source Code Regist Form



Gambar 3 Flow Chart Regis



Gambar 4 Flowgraph Regis

Perhitungan *flowgraph* yang dilakukan pada halaman regis sesuai dari flowgraph pada gambar 4 yang bersumber dari flowchart pada gambar 3 proses regis adalah sebagai berikut:

$$V(G) = (E - N) + 2$$

$V(G) = \text{Jumlah Region}$

$E = \text{Jumlah edge yang ditentukan dengan gambar panah}$

$N = \text{Jumlah simpul grafik (node) dengan gambar lingkaran}$

$V(G) = (7-7) + 2 \quad V(G) = 2$

Sehingga didapatkan hasil path atau jalur yang digunakan pada regis adalah sebagai berikut:

- 1-2-3-4-6-7.
- 1-2-3-4-5-2-3-4-6-7

## 2. Login Form

```
Index.php > <html> <body> <div> <div> formLogin > <div> <div> container-fluid > <table> <table>
12 <html lang="en">
13 <body>
14 <header>
15 <?php
16 session_start();
17 $kopi = array(
18     "nama_kopi" => ["affogato","cappuccino","americano","cortado","espresso","flat-white","latte","macchiato","moca
19     "stok_kopi" => [10,15,20,10,15,20,10,15,20,15],
20     "harga_kopi" => [15000,15000,13000,15000,15000,13000,15000,15000,13000,15000],
21 );
22 $panjang = count($kopi['nama_kopi']);
23 $i = 0;
24 $_SESSION['nama_kopi'] = array();
25 $_SESSION['stok_kopi'] = array();
26 $_SESSION['harga_kopi'] = array();
27 while ($i < $panjang) {
28     array_push($_SESSION['nama_kopi'], $kopi["nama_kopi"][$i]);
29     array_push($_SESSION['stok_kopi'], $kopi["stok_kopi"][$i]);
30     array_push($_SESSION['harga_kopi'], $kopi["harga_kopi"][$i]);
31     $i++;
32 }
33 $nama_kopi = json_encode($_SESSION['nama_kopi']);
34 $stok_kopi = json_encode($_SESSION['stok_kopi']);
35 $harga_kopi = json_encode($_SESSION['harga_kopi']);
36 $nama_ses1 = $_SESSION['nama'] ?? null;
37 if(isset($_POST['submit'])) {
38     $username = $_POST['nama'];
39     $password = $_POST['password'];
40     if(isset($_POST['role'])) {
41         $selectedOption = $_POST['role'];
42         $_SESSION['role'] = $selectedOption;
```

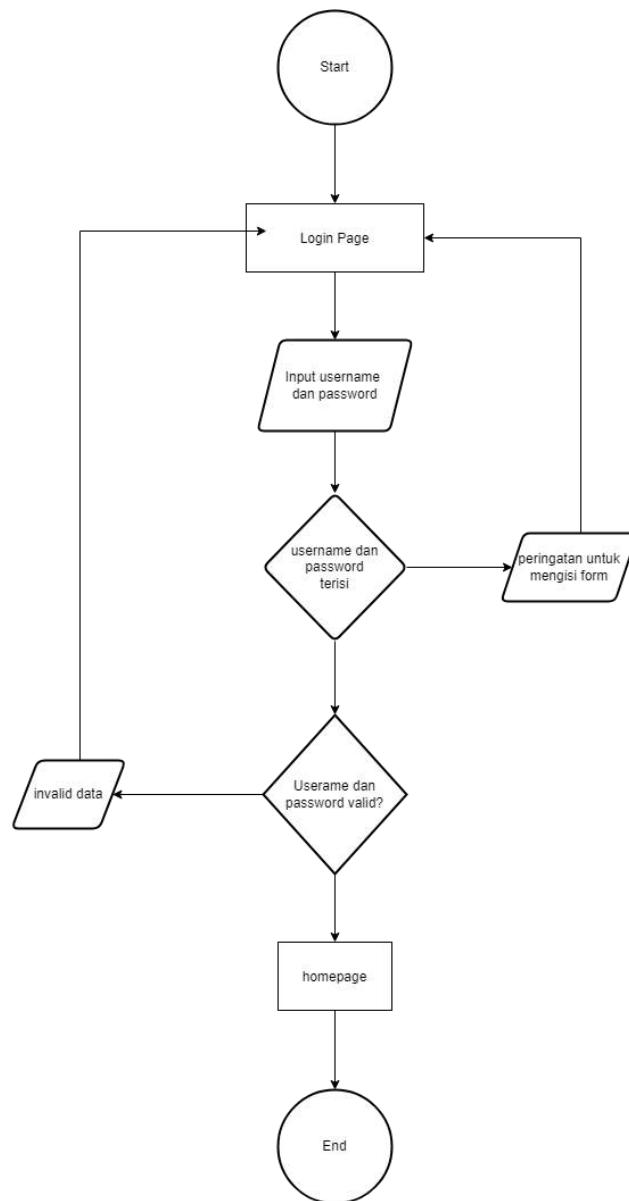
Gambar 5 Source Code Login Form

```
index.php M X JS keranjang.js M signup.php M info.php M home.php M # style.css M
index.php > html > body > div > div > form#login > div.box1.container-fluid > table.table
2 <html Lang="en">
13 <body>
14 <header>
33 $nama_kopi = json_encode($_SESSION['nama_kopi']);
34 $stok_kopi = json_encode($_SESSION['stok_kopi']);
35 $harga_kopi = json_encode($_SESSION['harga_kopi']);
36 $nama_sesi = $_SESSION['nama'] ?? null;
37 if(isset($_POST['submit'])) {
38     $username = $_POST['nama'];
39     $password = $_POST['password'];
40     if(isset($_POST['role'])) {
41         $selectedOption = $_POST['role'];
42         $_SESSION['role'] = $selectedOption;
43     }
44     if($nama_sesi != null){
45         foreach($_SESSION['nama'] as $nama){
46             foreach($_SESSION['pw'] as $pw){
47                 if($username == $nama && $password == $pw){
48                     $_SESSION['username'] = $username;
49                     echo "<script type = 'text/JavaScript'>";
50                     echo "alert('Berhasil masuk')";
51                     echo "</script>";
52                     header('Location: home.php');
53                 }
54             }
55         }
56     }else{
57         echo "<script type = 'text/JavaScript'>";
58         echo "alert('Tidak ada akun, silahkan regist')";
59         echo "</script>";
60     }
61 }
```

Gambar 6 Source Code Login Form

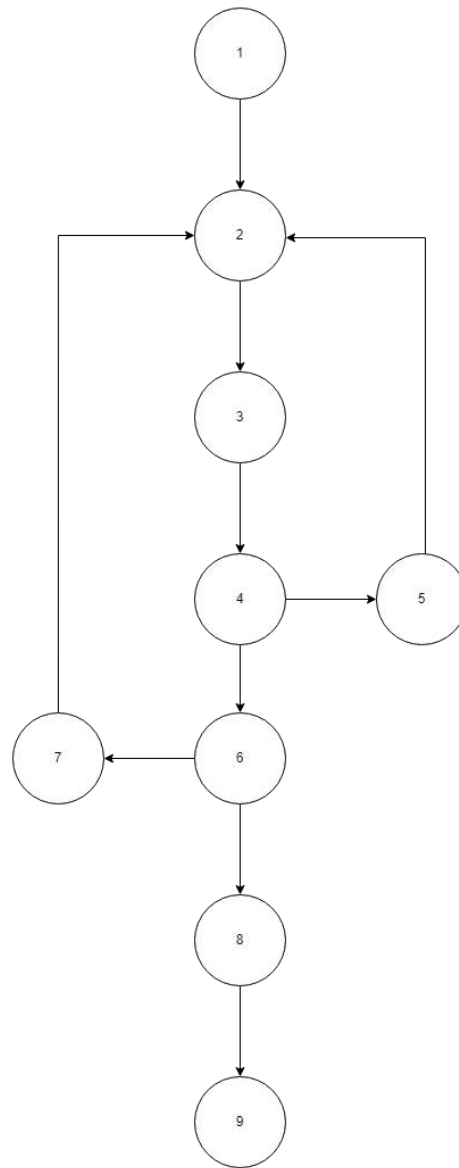
```
index.php M X JS keranjang.js M signup.php M info.php M home.php M # style.css M
index.php > html > body > div > div > form#login > div.box1.container-fluid > table.table
2 <html Lang="en">
13 <body>
82 <div style="min-height: 90.5vh; background-image: url(assets/image/bg3.jpg); background-size: cover;">
83 <div style="padding-top: 8%; display: flex; height: 100%; justify-content: center; align-items: center;">
84 <form method="POST" id="login">
85 <div class="box1 container-fluid">
86 <h1>Sign In</h1>
87 <table class="table">
88 <tbody>
89 <tr>
90 <th scope="row">Username</th>
91 <td></td>
92 <td><input type="text" name="nama" class="email"/></td>
93 </tr>
94 <tr>
95 <th scope="row">Password</th>
96 <td></td>
97 <td><input type="password" name="password" class="email"/></td>
98 </tr>
99 </tbody>
100 </table>
101 <button style="margin-bottom: 2%; type="submit" name="submit"
102 class="btn btn-success">
103 Sign In
104 </button>
105 </div>
106 </form>
107 </div>
108 <div style="color: white; padding-top: 1%; text-align: center;">Dont have an account?
109 <a href="signup.php" style="text-decoration: none; color: #3498db;">Sign Up</a>
110 </div>
111 </body>
```

Gambar 7 Source Code Login Form



Gambar 8 Flowchart Login Form





Gambar 9 Flowgraph Login Form

Perhitungan *flowgraph* yang dilakukan pada halaman login sesuai dari flowgraph pada gambar 4 yang bersumber dari flowchart pada gambar 3 proses login adalah sebagai berikut:

$$V(G) = (E - N) + 2$$

$V(G)$  = Jumlah Region

$E$  = Jumlah edge yang ditentukan dengan gambar panah

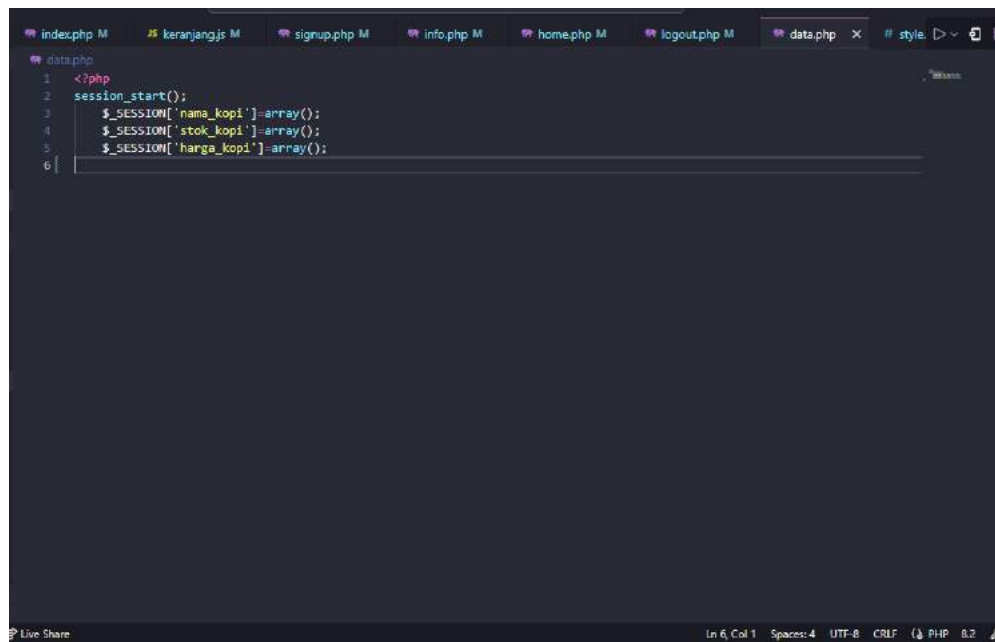
$N$  = Jumlah simpul grafik (node) dengan gambar lingkaran

$$V(G) = (10-9) + 2 \quad V(G) = 3$$

Sehingga didapatkan hasil path atau jalur yang digunakan pada login adalah sebagai berikut:

- a. Jalur 1-2-3-4-6-8-9.
- b. Jalur 1-2-3-4-5-2-3-4-6-8-9.
- c. Jalur 1-2-3-4-6-7-2-3-4-6-8-9.

### 3. Keranjang



```
data.php
1 <?php
2 session_start();
3 $_SESSION['nama_kopi']=array();
4 $_SESSION['stok_kopi']=array();
5 $_SESSION['harga_kopi']=array();
6
```

Gambar 10 Source Code Keranjang

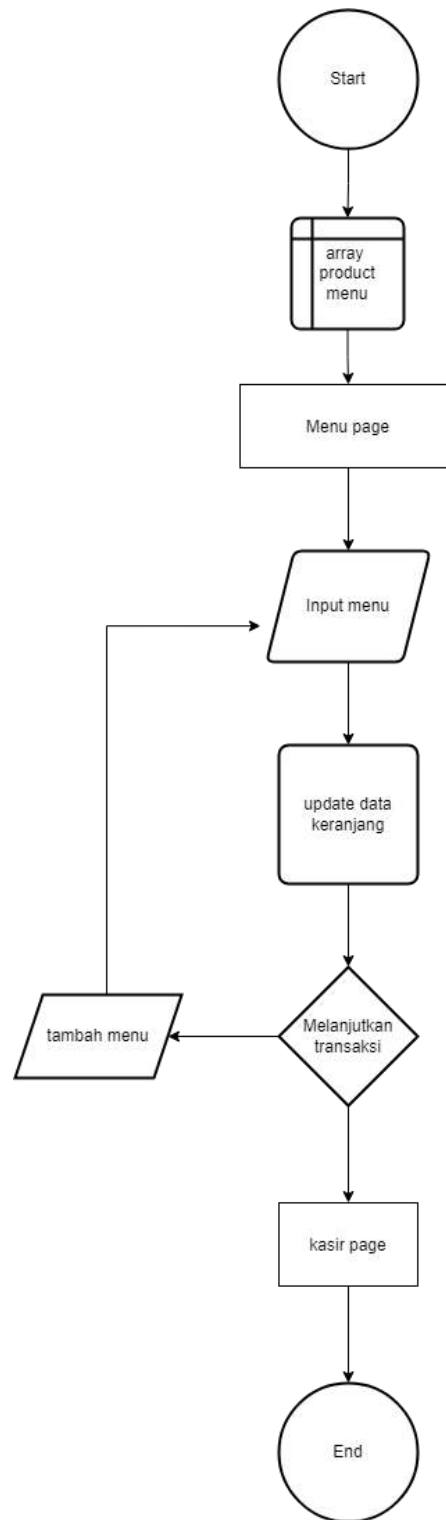
```
index.php M JS keranjang.js M X signup.php M info.php M home.php M # style.css M
js > JS keranjang.js > beli
1 const button = document.getElementById("buy");
2 let angka = 0;
3 let satuan=1;
4 const data_pesanan = localStorage.getItem('pesanan');
5 const data =JSON.parse(data_pesanan) || [];
6 const keranjang = document.getElementById("krj");
7 let total = data.reduce((total, item) => total + item.jumlah, 0);
8 console.log(total);
9 function krj(){
10     alert("Berhasil")
11     window.location.href='profile.php';
12 }
13 function beli(i){
14     const nama = document.getElementById("nama"+i);
15     const harga = document.getElementById("harga"+i);
16     const isi = nama.textContent;
17     const hrg = harga.textContent;
18     console.log(isi);
19     console.log(hrg);
20     let itemFound = false;
21     for (let a = 0; a < data.length; a++) {
22         if(data[a].nama === isi){
23             data[a].jumlah++;
24             itemFound = true;
25             break;
26         }
27     }
28     if(!itemFound) {
29         const pesanan = {
30             nama:isi,
31             jumlah:1,
32             harga:hrg

```

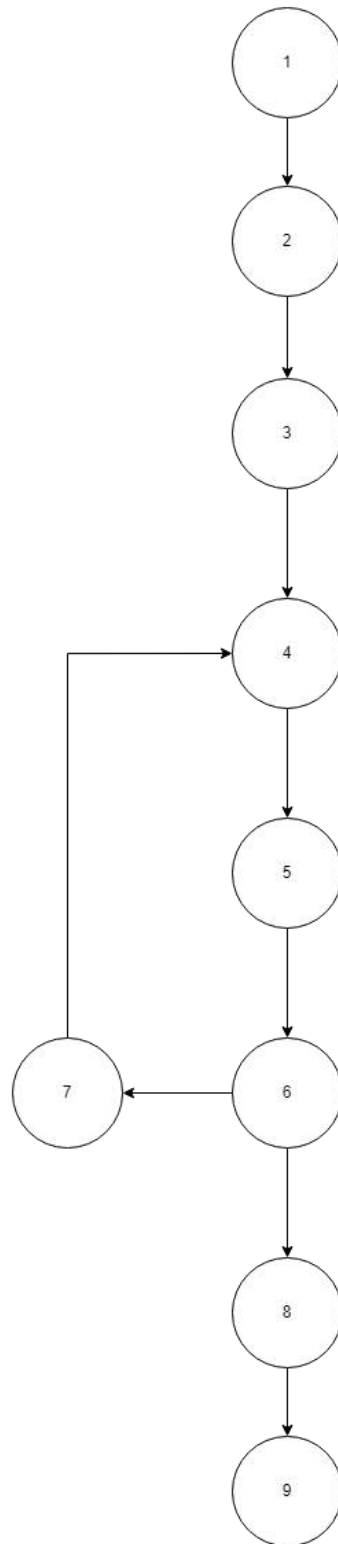
Gambar 11 Source Code Keranjang

```
index.php M JS keranjang.js M X signup.php M info.php M home.php M # style.css M
js > JS keranjang.js > beli
12 }
13 function beli(i){
14     const nama = document.getElementById("nama"+i);
15     const harga = document.getElementById("harga"+i);
16     const isi = nama.textContent;
17     const hrg = harga.textContent;
18     console.log(isi);
19     console.log(hrg);
20     let itemFound = false;
21     for (let a = 0; a < data.length; a++) {
22         if(data[a].nama === isi){
23             data[a].jumlah++;
24             itemFound = true;
25             break;
26         }
27     }
28     if(!itemFound) {
29         const pesanan = {
30             nama:isi,
31             jumlah:1,
32             harga:hrg
33         };
34         data.push(pesanan);
35     }
36     angka++;
37     localStorage.setItem('pesanan', JSON.stringify(data));
38     keranjang.innerHTML = total+angka;
39 }
40 keranjang.innerHTML = total;
```

Gambar 12 Source Code Keranjang



Gambar 13 Flowchart Keranjang



Gambar 14 Flowgraph Keranjang

Perhitungan *flowgraph* yang dilakukan pada halaman keranjang sesuai dari flowgraph pada gambar 4 yang bersumber dari flowchart pada gambar 3 adalah sebagai berikut:

$$V(G) = (E - N) + 2$$

$$V(G) = \text{Jumlah Region}$$

E= Jumlah edge yang ditentukan dengan gambar panah

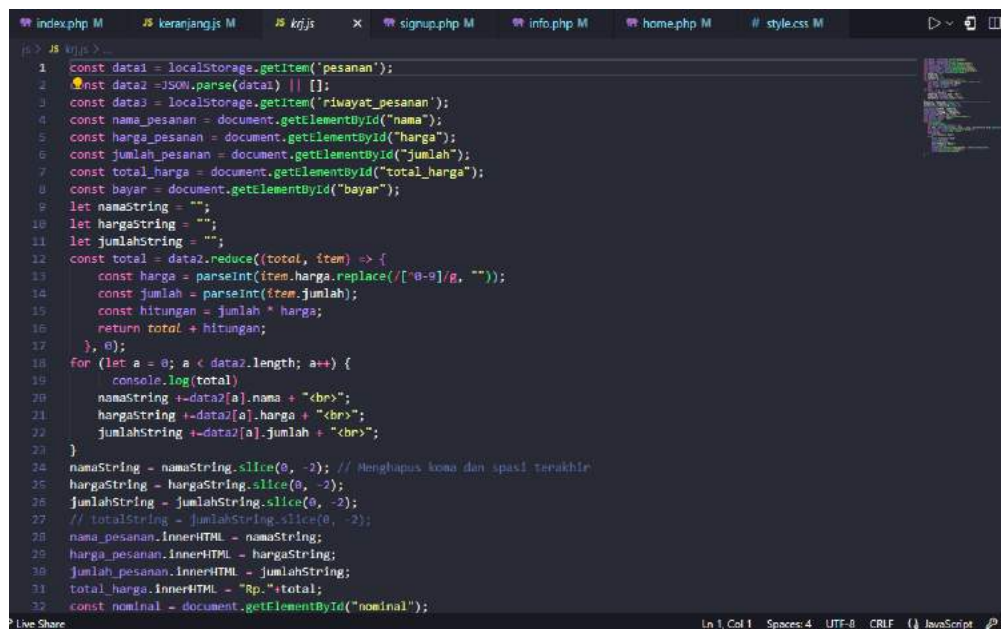
N= Jumlah simpul grafik (node) dengan gambar lingkaran

$$V(G) = (9 - 9) + 2 \quad V(G) = 2$$

Sehingga didapatkan hasil path atau jalur yang digunakan pada keranjang adalah sebagai berikut:

- a. jalur 1-2-3-4-5-6-8-9
- b. jalur 1-2-3-4-5-6-7-4-5-6-8-9

#### 4. Kasir

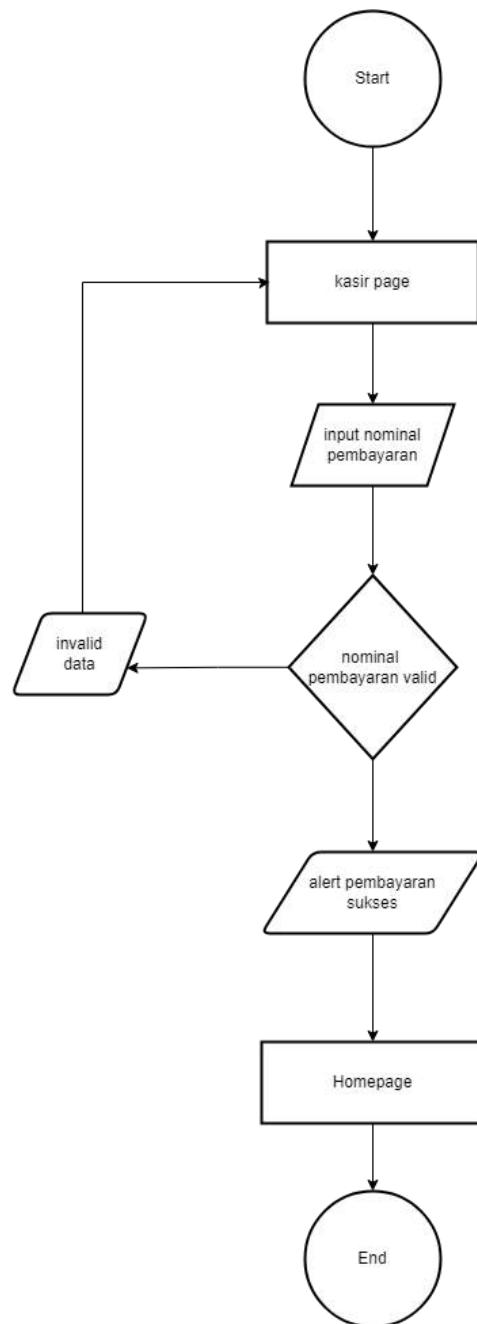
The image shows a code editor with a dark theme. The top bar shows several open files: 'index.php M', 'keranjangjs M', 'krjs', 'signup.php M', 'info.php M', 'home.php M', and 'style.css M'. The active file is 'krjs'. The code is JavaScript, starting with a comment 'js > krjs >'. It defines several constants for retrieving data from localStorage and document elements. It then uses the reduce method to calculate the total price and quantity. Finally, it uses the for loop to iterate over the data and update the innerHTML of the 'nama\_pesanan', 'harga\_pesanan', 'jumlah\_pesanan', and 'total\_harga' elements. The code ends with a console.log statement and a return statement for the 'nominal' element.

```
1 const data1 = localStorage.getItem('pesanan');
2 const data2 = JSON.parse(data1) || [];
3 const data3 = localStorage.getItem('riwayat_pesanan');
4 const nama_pesanan = document.getElementById("nama");
5 const harga_pesanan = document.getElementById("harga");
6 const jumlah_pesanan = document.getElementById("jumlah");
7 const total_harga = document.getElementById("total_harga");
8 const bayar = document.getElementById("bayar");
9 let namaString = "";
10 let hargaString = "";
11 let jumlahString = "";
12 const total = data2.reduce((total, item) => {
13   const harga = parseInt(item.harga.replace(/[^\d]/g, ""));
14   const jumlah = parseInt(item.jumlah);
15   const hitungan = jumlah * harga;
16   return total + hitungan;
17 }, 0);
18 for (let a = 0; a < data2.length; a++) {
19   console.log(total)
20   namaString += data2[a].nama + "<br>";
21   hargaString += data2[a].harga + "<br>";
22   jumlahString += data2[a].jumlah + "<br>";
23 }
24 namaString = namaString.slice(0, -2); // Menghapus koma dan spasi terakhir
25 hargaString = hargaString.slice(0, -2);
26 jumlahString = jumlahString.slice(0, -2);
27 // totalString = jumlahString.slice(0, -2);
28 nama_pesanan.innerHTML = namaString;
29 harga_pesanan.innerHTML = hargaString;
30 jumlah_pesanan.innerHTML = jumlahString;
31 total_harga.innerHTML = "Rp." + total;
32 const nominal = document.getElementById("nominal");
```

Gambar 15 Source Code Kasir

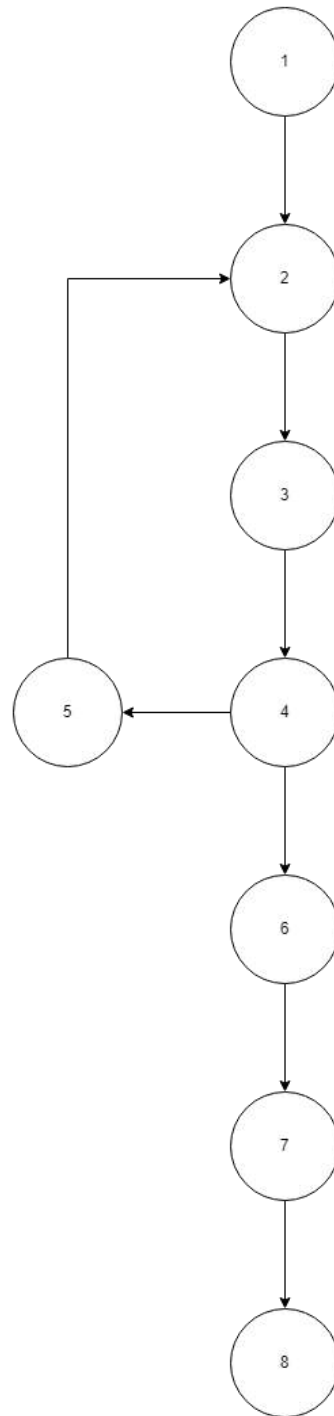
```
index.php M JS keranjang.js M JS kjs X signup.php M info.php M home.php M # style.css M
js kjs
25 hargaString = hargaString.slice(0, -2);
26 jumlahString = jumlahString.slice(0, -2);
27 // totalString = jumlahString.slice(0, -2);
28 nama_pesanan.innerHTML = namaString;
29 harga_pesanan.innerHTML = hargaString;
30 jumlah_pesanan.innerHTML = jumlahString;
31 total_harga.innerHTML = "Rp." + total;
32 const nominal = document.getElementById("nominal");
33 const hitung = nominal - total;
34 bayar.onclick=trans;
35 function trans(e){
36     e.preventDefault();
37     const data4 = JSON.parse(data3) || [];
38     let total = data2.reduce((total, item) => total + parseInt(item.harga.replace(/[^\d-]/g, "")), 0);
39     const nominal = document.getElementById("nominal");
40     const hitung = nominal.value - total;
41     if(hitung < 0){
42         alert("Pembayaran Gagal")
43     }else{
44         console.log(nominal.value)
45         console.log(total)
46         console.log(hitung);
47         alert("Pembayaran Sukses")
48         data4.push(data2)
49         localStorage.setItem('riwayat_pesanan', JSON.stringify(data4));
50         localStorage.removeItem('pesanan');
51         window.location.href="home.php";
52     }
53 }
```

Gambar 16 Source Code Kasir



Gambar 17 Flowchart Kasir





Gambar 18 Flowgraph Kasir

Perhitungan *flowgraph* yang dilakukan pada halaman kasir sesuai dari flowgraph pada gambar 4 yang bersumber dari flowchart pada gambar 3 proses kasir adalah sebagai berikut:

$$V(G) = (E - N) + 2$$

$V(G)$  = Jumlah Region

$E$  = Jumlah edge yang ditentukan dengan gambar panah

$N$  = Jumlah simpul grafik (node) dengan gambar lingkaran

$$V(G) = (8 - 8) + 2 \quad V(G) = 2$$

Sehingga didapatkan hasil path atau jalur yang digunakan pada regis adalah sebagai berikut:

- a. jalur 1-2-3-4-6-7-8
- b. jalur 1-2-3-4-5-2-3-4-6-7-8

## 5. Cyclomatic complexity

Pengujian	Jumlah path	Hasil
Regis	2	Valid
Login	3	Valid
Keranjang	2	Valid
Kasir	2	Valid

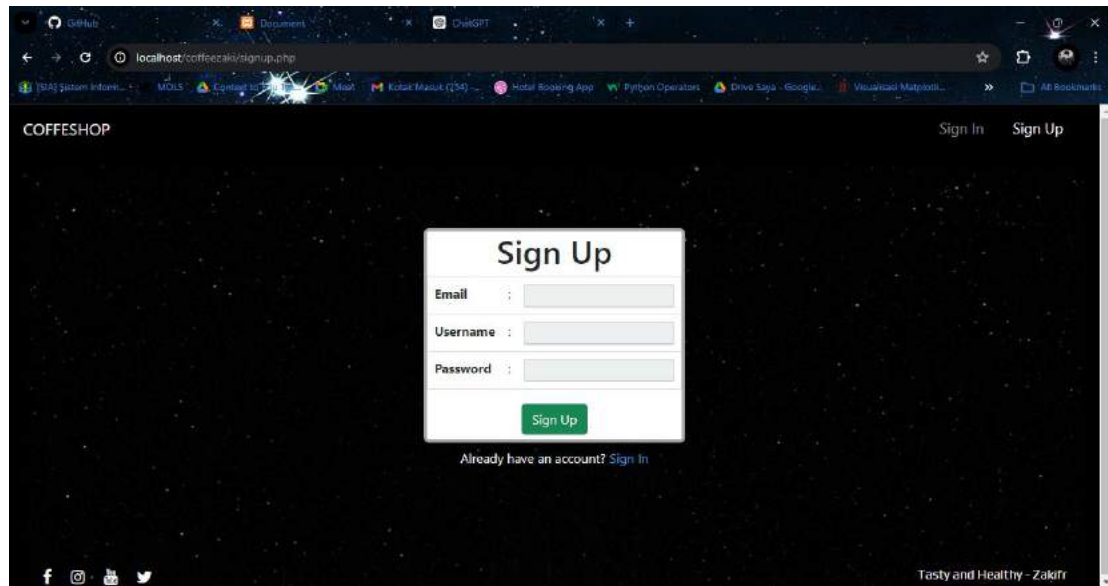
Setelah diketahui jumlah jalur independennya, maka akan dilakukan perbandingan menggunakan tabel hubungan antara cyclomatic complexity dan resiko pada tabel berikut:

Nilai	Tipe Prosedur	Tingkat Resiko
1-4	Prosedur Sederhana	Rendah
5-10	Prosedur yang terstruktur dengan baik dan stabil	Rendah
11-20	Prosedur yang lebih kompleks	Menengah
21-50	Prosedur yang kompleks dan kritis	Menengah
>50	Rentan kesalahan, sangat mengganggu, prosedur tidak dapat diuji	Sangat Tinggi

Sehingga menurut tabel di atas dapat dikelompokkan setiap pengujian whitebox termasuk kedalam kategori prosedur tingkat resiko yang rendah.

## B. Pengujian Blackbox

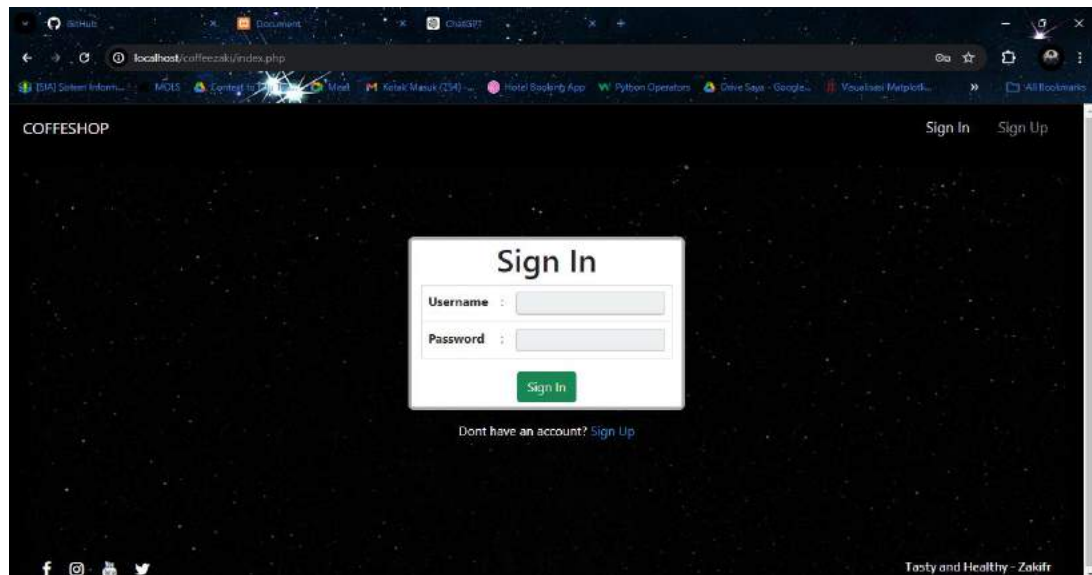
### 1. Regist Form



The screenshot shows a web browser window with the URL `localhost/coffeezaku/signup.php`. The page has a dark, starry background. At the top left is the text "COFFESHOP" and at the top right are links for "Sign In" and "Sign Up". In the center is a white "Sign Up" form with three input fields: "Email", "Username", and "Password". Below these fields is a green "Sign Up" button. Under the button, it says "Already have an account? Sign In". At the bottom left are social media icons for Facebook, Instagram, YouTube, and Twitter. At the bottom right is the text "Tasty and Healthy - Zakir".

Gambar 19 Tampilan Regis Form

### 2. Login Form

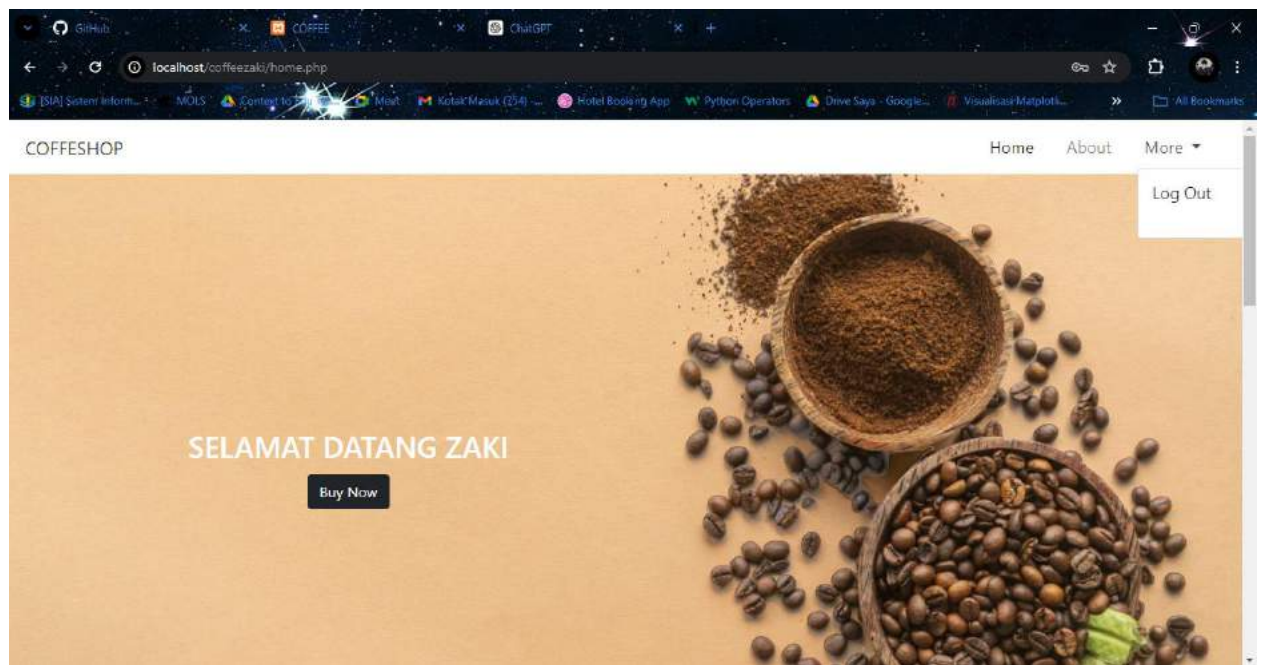


The screenshot shows a web browser window with the URL `localhost/coffeezaku/index.php`. The page has a dark, starry background. At the top left is the text "COFFESHOP" and at the top right are links for "Sign In" and "Sign Up". In the center is a white "Sign In" form with two input fields: "Username" and "Password". Below these fields is a green "Sign In" button. Under the button, it says "Dont have an account? Sign Up". At the bottom left are social media icons for Facebook, Instagram, YouTube, and Twitter. At the bottom right is the text "Tasty and Healthy - Zakir".

Gambar 20 Tampilan Login Form

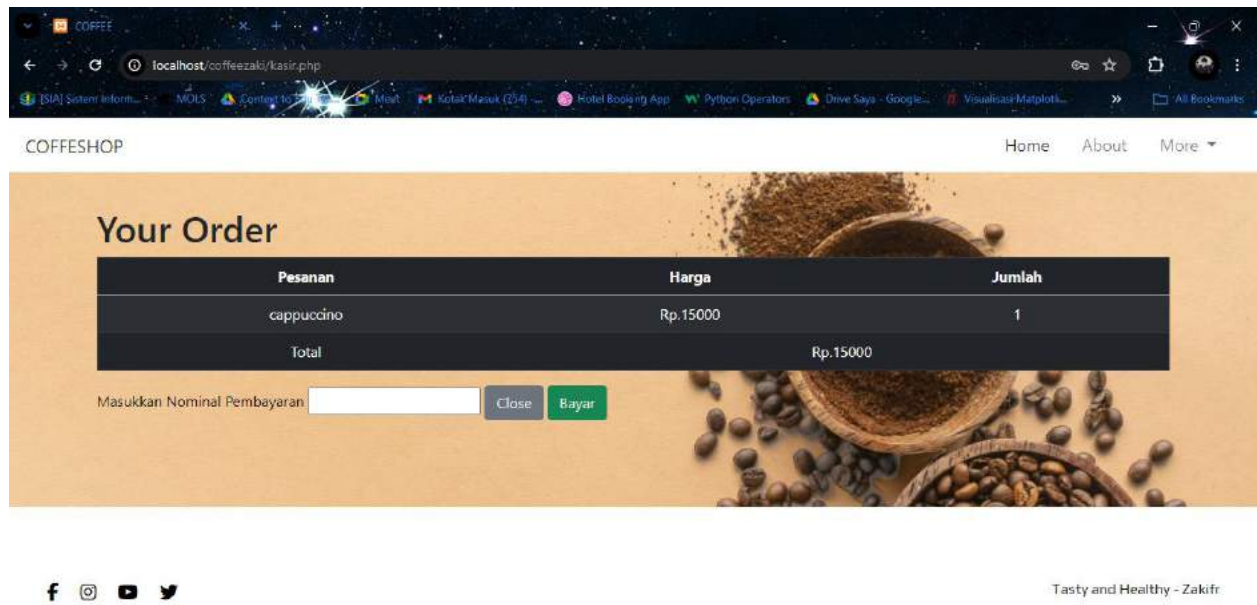


Gambar 21 Tampilan Jika Akun belum terdaftar

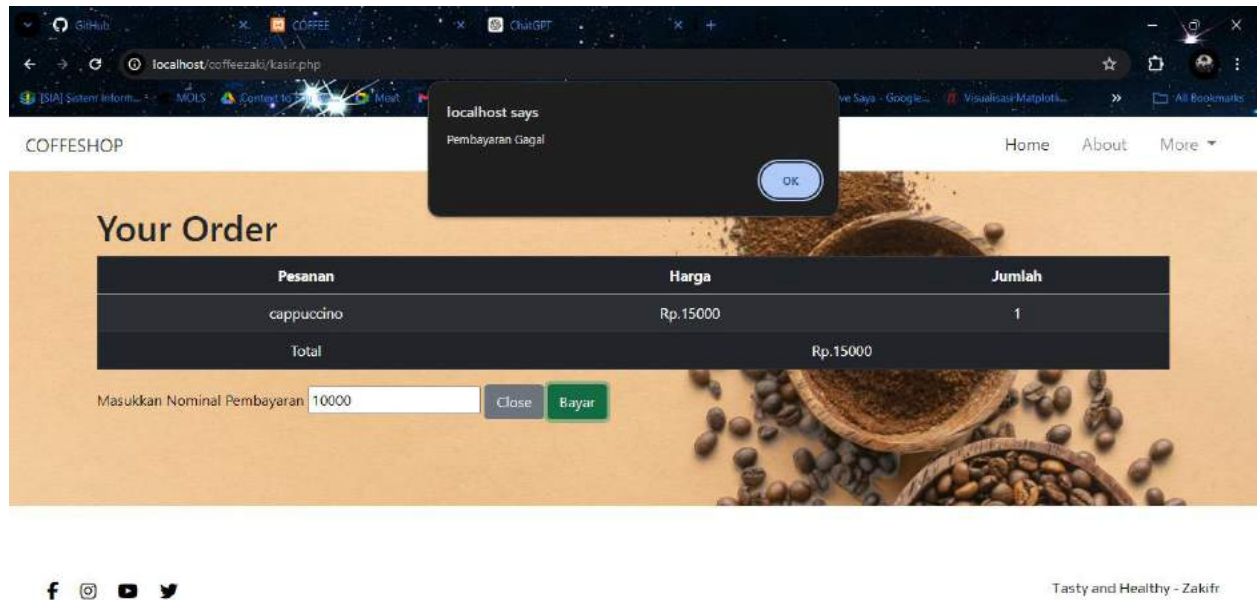


Gambar 22 Tampilan Halaman Home

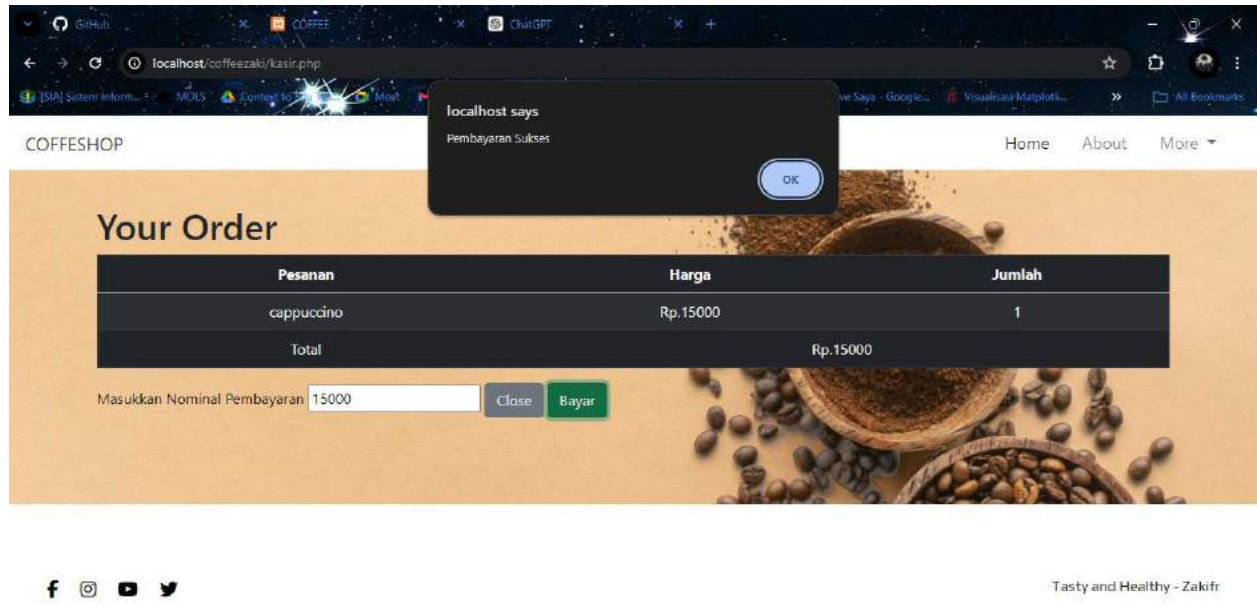
### 3. Kasir



Gambar 23 Tampilan Halaman Kasir



Gambar 24 Tampilan Halaman Kasir jika pembayaran gagal



Gambar 25 Halaman Kasir jika pembayaran berhasil

#### 4. Tabel Blackbox Testing

ID	Skenario	Test Case	Hasil diharapkan	Kesimpulan
CF01	Regist	Input email kosong	Sistem menampilkan peringatan untuk mengisi form regist email.	VALID
CF02	Regist	Input Username Kosong	Sistem menampilkan peringatan untuk mengisi form regist username.	VALID
CF03	Regist	Input Password Kosong	Sistem menampilkan peringatan untuk mengisi form regist password.	VALID
CF04	Regist	Input Email, Username dan Password terisi	Sistem akan mendaftarkan anda ke array penyimpanan.	VALID
CF05	Login	Input Username Kosong	Sistem menampilkan untuk mengisi form username.	VALID

CF06	Login	Input Password Kosong	Sistem menampilkan untuk mengisi form password	VALID
CF07	Login	Input Username dan Password terisi tetapi salah	Sistem menampilkan alert akun belum terdaftar	VALID
CF08	Login	Input Username dan Password terisi dan benar	Sistem akan memindahkan pengguna ke halaman beranda dan usernamenya akan ditampilkian diberanda	VALID
CF09	Kasir	Input Nilai nominal pembayaran jika kurang dari total pembayaram	Sistem akan menampilkan alert bertulisan pembayaran gagal	VALID
CF10	Kasir	Input Nilai nominal pembayaran jikan benar	Sistem akan menampilkan alert bertulisan Pembayaran Sukses	VALID

Table 1 Table Blackbox Testing

## **BAB III**

### **PENUTUP**

#### **3.1. Kesimpulan**

Dari hasil pengujian melalui whitebox dan juga blackbox didapatkan hasil sebagai berikut:

1. Pengujian whitebox dinyatakan valid dengan terjalannya semua path dengan baik sesuai dengan sourcecode yang ada serta menghasilkan tingkat resiko yang rendah dikarenakan setiap fungsi kode hanya memiliki 2-3 jalur independen saja.
2. Pengujian blackbox dinyatakan valid sebanyak 10 dari 10 tes yang di uji coba kepada setiap form input yang ada pada website coffeeshop sederhana.

Sehingga melalui pengujian ini web coffeeshop sederhana telah dinyatakan lolos tanpa adanya error.