DUPL-01

DOKUMEN UJI PERANGKAT LUNAK

Sistem Informasi Apoteker

untuk:

Instalasi Farmasi RS

Dipersiapkan oleh:

Mar Ayu Fotina/ 1301174013

Fithroh Hito Naruhodo/ 1301174437

Putri Apriyanti Windya/ 1301174169

Ardhia Nanda Pramusti/ 1301174108

Daftar Isi

Daftar Gambar

Daftar Tabel

Daftar Lampiran

Pendahuluan

1.1 Tujuan Pembuatan Dokumen

Dokumen ini merupakan dokumen yang berisi deskripsi pengujian (testing) perangkat lunak yang ditulis berdasarkan dokumen yang telah disusun sebelumnya yaitu Dokumen Perancangan Perangkat Lunak (DPPL). Tujuan dari DUPL ini yaitu ingin mewujudkan sistem apotek yang baik dan memecahkan permasalahan yang terjadi pada sistem apotek itu sendiri.

1.2 Ruang Lingkup Pengujian

Untuk ruang lingkup pengujian pada aplikasi ini, kamu menggambi masih dalam lingkup kampus, yang mana dalam pengujian ini memastikan kelas-kelas atau fungsi di dalam aplikasi ini sudah berjalan dengan baik sesuai dokumen yang sudah di buat di SKPL maupun DPPL.

1.3 Referensi

Referensi yang digunakan dalam pembuatan dokumen ini merujuk pada dokumen SKPL dan DPPL sistem informasi apoteker yang berkaitan dengan kebutuhan user dalam proses pembuatan aplikasi.

1.4 Overview Sistem & Fitur Utamanya

Sistem dan fitur utama dari aplikasi ini, layaknya sebuah datasbase, menyimpan data-data ataupun trasnsaksi-transksi apa saja yang ada di apoteker. Sehingga dapat memudahkan apotek tersebut.

1.5 Overview Pengujian

1.5.1 Perangkat Keras Pengujian

Perangkat keras yang dibutuhkan :

- Sistem Operasi : Microsoft server 2003 atau linux server
- *Harddisk :* Min. 1 TB
- *Memory* : Min. 8 *GB RAM*
- *Database* : *MySQL Database*

1.5.2 Sumber Daya Manusia

Menjelaskan sumber daya manusia yang terlibat dalam pengujian perangkat lunak sumber daya yang terlibat dalam pengujian perangkat lunak ini diantara lain adalah

a. Developer pembuat perangkat lunak sistem apotek
b. Admin sistem apotek
c. User system apotek

1.5.3 Perangkat Lunak Pengujian

Perangkat Lunak sisi *client* yang dibutuhkan :

- Sistem Operasi : *Microsoft Windows 7/8/10*

- RAM : Min. 2 GB

1.5.4 Strategi dan Metode Pengujian

Metode yang digunakan untuk integration testing adalah dengan pengujian Black Box dan White Box. Black box adalah jenis pengujuan perangkat lunak yang berfokus pada semua persyaratan fungsional perangkat lunak. White Box adalah pengujian yang didasarkan pada pengecekkan terhadap detail perancangan menggunakan struktur kontrol dari desain program secara procedural untuk membagi pengujian ke dalam beberapa kasus pengujian.

Pelaksanaan Pengujian

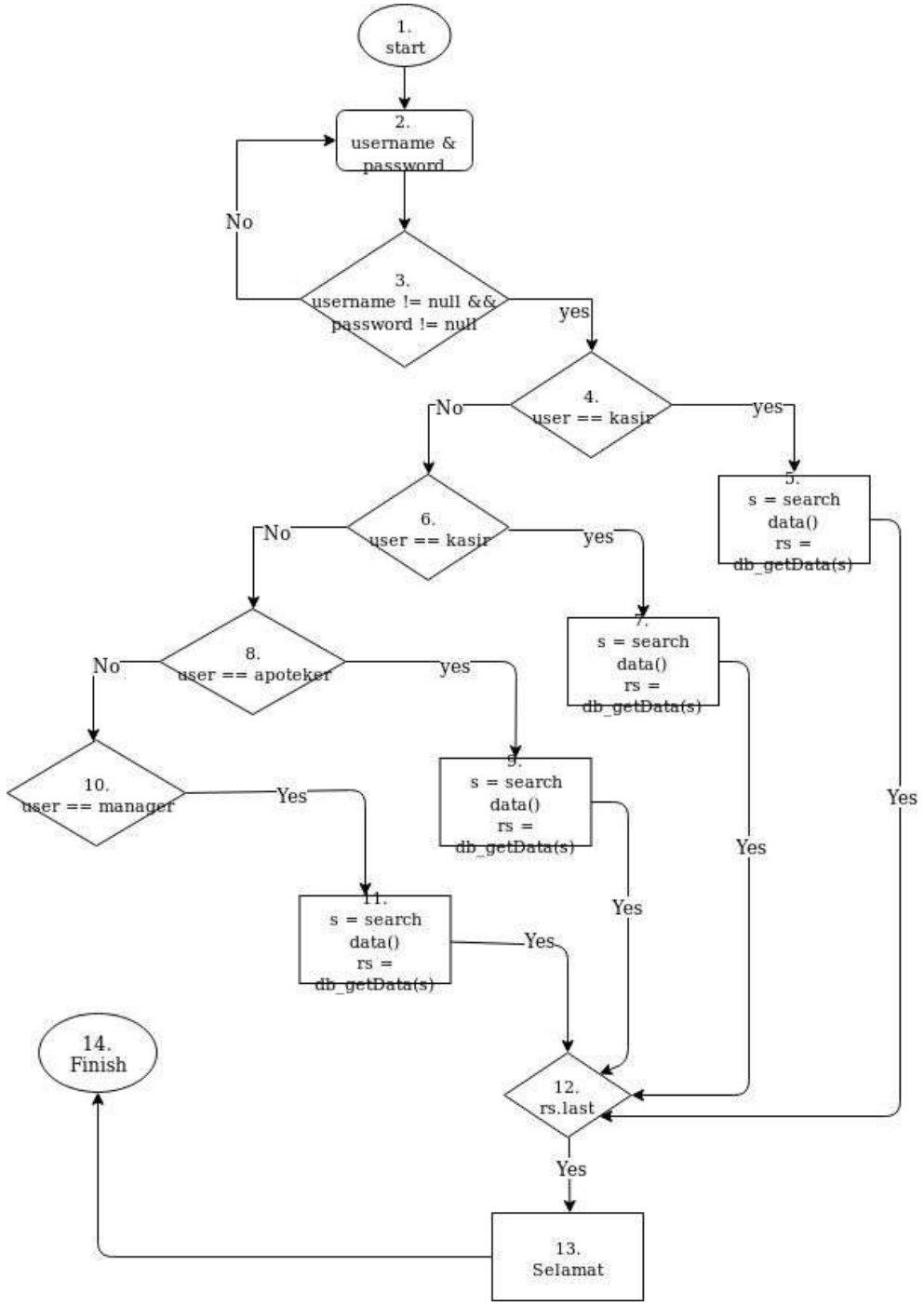
Pengujian dilaksanakan pada tanggal 10 Desember dengan metode black box dan metode white box

2.1 Pengujian UNIT

2.1.1 Pengujian White Box Method

a. Class yang akan di uji yaitu : Database

b. Flowchart/Flowgraphnya



c. Menghitung Cyclomatic Complexity

1. Method Database()

Cyclomatic Complexity = 2 + 1 = 3

2. Method getData(String SQLString)

Cyclomatic Complexity = 1 + 1 = 2

3. Method query(String SQLString)

Cyclomatic Complexity = 1 + 1 =2

4. Method loginUser(LoginModel lg, Login log)

Cyclomatic Complexity = (6 + 1) + (1+1) = 9

5. Method saveDataPemesanan(PemesananModel p)

Cyclomatic Complexity = 1 + 1 = 2

6. Method saveDataPembayaran(PembayaranModel p)

Cyclomatic Complexity = 1 + 1 = 2

7. Method viewDataPemesanan(KasirDataPemesanan k)

Cyclomatic Complexity = 1 + 1 = 2

8. Method viewDataPembayaran(KasirDataPembayaran k)

Cyclomatic Complexity = 1 + 1 = 2

9. Method cariDataPemesanan(KasirDataPemesanan k)

Cyclomatic Complexity = 1 + 1 = 2

10. Method cariDataPembayaran(KasirDataPembayaran k)

Cyclomatic Complexity = 1 + 1 = 2

d. Path yang diuji

Daftar path yang perlu diuji dengan Cyclomatic Complexity tertinggi yaitu Method loginUser(LoginModel lg, Login log)

e. Siapkan data uji untuk setiap path
Data yang akan di uji yaitu berdasarkan jalur yang ada di Method loginUser(LoginModel lg, Login log)

- 1-2-3-4-5-12-13 -14
- 1-2-3-4-5-12-14
- 1-2-3-4-6-7-12-13-14
- 1-2-3-4-6-7-12-14
- 1-2-3-4-6-8-9-12-13-14
- 1-2-3-4-6-8-9-12-14
- 1-2-3-4-6-8-10-11-12-13-14
- 1-2-3-4-6-8-10-12-14

- 1-2-3-2

f. Hasil pengujian

- Def (x): manager, 123456, kasir, siapaya, apoteker, wastOnme, pendata, apaleu, supplier
- Use(x): kasir, siapayah, supplier, apaleu
- Username, password = kasir, siapayah (valid)(jalur 1)
- Username, password = supplier, wastOnme (tidak valid)(jalur 8)

2.1.2 Pengujian Class dengan JUnit/PhPUnit

Jelaskan di sini contoh pengujian sebuah class.
Setiap methodnya perlu diuji, dengan data uji yang membuat VALID atau yang membuat FAIL.

Tabel 2 Pengujian Class

CLASS	Method	Kasus dan Hasil Uji (Data normal)			
		Data Masukan	**Yang diharapkan**	**Pengamatan***	**Kesimpulan**
Login	testGetUsername	User ID: Universitas Nama : xyz password: rahasia konfirmasi password:rahasia(hasil pada lampiran B(Gambar B.2 dan B.4))	Form menampilkan data user baru untuk *user* peneliti dan responden	Dapat melakukan pengisian data *user* baru Sesuai yang diharapkan	[X] diterima [ ] ditolak
		Klik tombol simpan (hasil pada lampiran B(Gambar B.3 dan B.6))	Data tersimpan di file *User* peneliti dan *user* Responden	Data pengisian *user* Responden dan peneliti tersimpan Sesuai yang diharapkan	[X] diterima [ ] ditolak
		Klik tombol Reset (hasil pada lampiran B(Gambar B.7 dan B.8)	Data yang telah terisi telah dihapus	Data telah terhapus sesuai yang diharapkan	[X] diterima [ ] ditolak

* Contoh pengujian dengan JUnit/PhpUnit dilampirkan

A. JUnit/PhpUnit untuk pengujian Class : (Login)

```
package Model;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author PUTRI
 */
public class LoginModelTest {

    public LoginModelTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }
```

- Test Get Username

```
    @Test
    public void testGetUsername() {
        System.out.println("getUsername");
        LoginModel instance = new LoginModel("PutriCan","mercyOnMe");
        String expResult = "PutriCan";
        String result = instance.getUsername();
        assertEquals(expResult, result);
    }
```

- Test Set Username

```
@Test
public void testSetUsername() {
    System.out.println("setUsername");
    String username = "";
    LoginModel instance = new LoginModel("PutriCan","mercyOnMe");
    instance.setUsername(username);
}
```

- Test Get Username

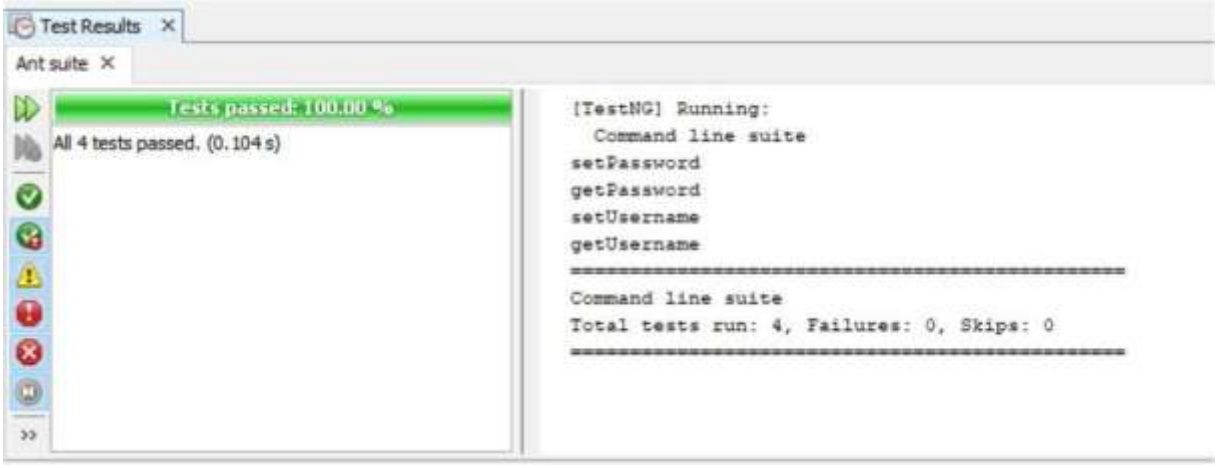
```
@Test
public void testGetPassword() {
    System.out.println("getPassword");
    LoginModel instance = new LoginModel("PutriCan","mercyOnMe");
    String expResult = "mercyOnMe";
    String result = instance.getPassword();
    assertEquals(expResult, result);

}
```

- Test Set Username

```
@Test
public void testSetPassword() {
    System.out.println("setPassword");
    String password = "";
    LoginModel instance = new LoginModel("PutriCan","mercyOnMe");
    instance.setPassword(password);
}
```

B. Screenshoot hasil pengujian JUnit



A. JUnit/PhpUnit untuk pengujian Class : (Pembayaran)

```
@Test
public void testSetIdPembayaran() {
    System.out.println("setIdPembayaran");
    String idPembayaran = "PMB0014";
    PembayaranModel instance = new PembayaranModel("PMB0014","PMS0014","KSR0003","2019-11-19",200000
    instance.setIdPembayaran(idPembayaran);
    // TODO review the generated test code and remove the default call to fail.
//      fail("The test case is a prototype.");
}
```

```
package Model;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author PUTRI
 */
public class PembayaranModelTest {

    public PembayaranModelTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }
```

- Test Set id_pembayaran
- Test Set total

```
    @Test
    public void testSetTotal() {
        System.out.println("setTotal");
        int total = 200000;
        PembayaranModel instance = new PembayaranModel("PMB0014","PMS0014","KSR0003","2019-11-19",200000);
        instance.setTotal(total);
        // TODO review the generated test code and remove the default call to fail.
//        fail("The test case is a prototype.");
    }
```

- Test Set pembayaran

```
     * Test of setTglPmbyr method, of class PembayaranModel.
     */
    @Test
    public void testSetTglPmbyr() {
        System.out.println("setTglPmbyr");
        String tglPmbyr = "2019-11-19";
        PembayaranModel instance = new PembayaranModel("PMB0014","PMS0014","KSR0003","2019-11-19",200000);
        instance.setTglPmbyr(tglPmbyr);
        // TODO review the generated test code and remove the default call to fail.
//        fail("The test case is a prototype.");
    }
```

- Test Set tanggal_pembayaran

```
     * Test of setTglPmbyr method, of class PembayaranModel.
     */
    @Test
    public void testSetTglPmbyr() {
        System.out.println("setTglPmbyr");
        String tglPmbyr = "2019-11-19";
        PembayaranModel instance = new PembayaranModel("PMB0014","PMS0014","KSR0003","2019-11-19",200000);
        instance.setTglPmbyr(tglPmbyr);
        // TODO review the generated test code and remove the default call to fail.
//        fail("The test case is a prototype.");
    }
```

- Test Set id_pemesanan

```
82      @Test
83      public void testSetIdPemesanan() {
84          System.out.println("setIdPemesanan");
85          String idPemesanan = "PMS0014";
86          PembayaranModel instance = new PembayaranModel("PMB0014","PMS0014","KSR0003","2019-11-19",200000);
87          instance.setIdPemesanan(idPemesanan);
88          // TODO review the generated test code and remove the default call to fail.
89  //        fail("The test case is a prototype.");
90      }
91
```

- Test Set id_kasir

```
    @Test
    public void testSetIdKasir() {
        System.out.println("setIdKasir");
        String idKasir = "KSR0003";
        PembayaranModel instance = new PembayaranModel("PMB0014","PMS0014","KSR0003","2019-11-19",200000);
        instance.setIdKasir(idKasir);
        // TODO review the generated test code and remove the default call to fail.
/        fail("The test case is a prototype.");
    }
```

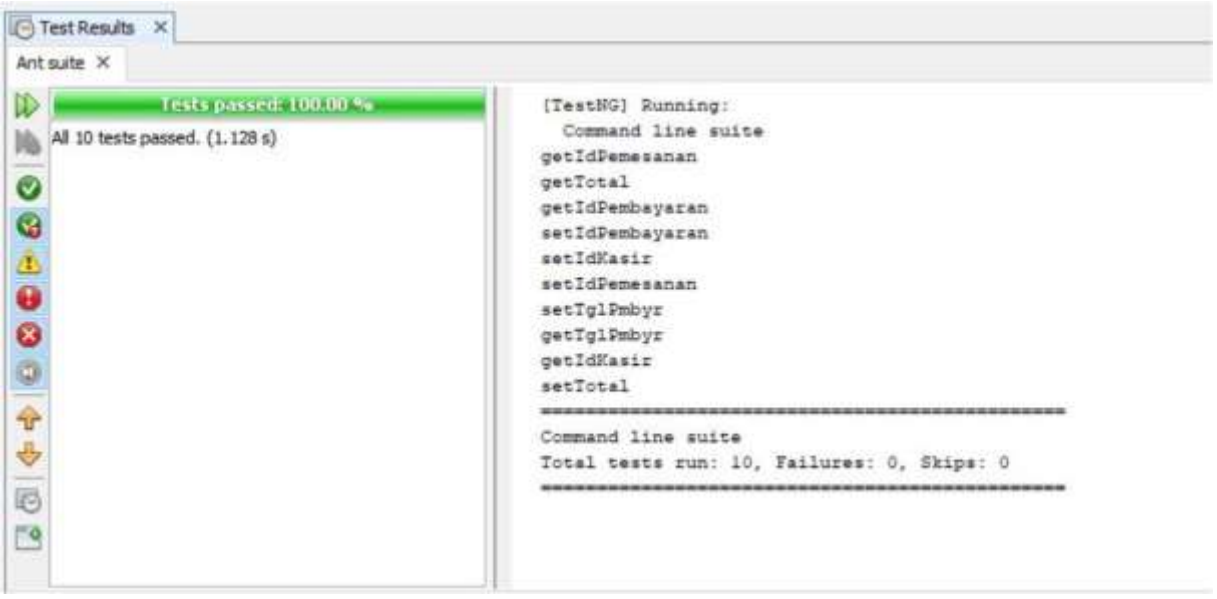
- Test Set get_total

```
@Test
public void testGetTotal() {
    System.out.println("getTotal");
    PembayaranModel instance = new PembayaranModel("PMB0014","PMS0014","KSR0003","2019-11-19",200000);
    int expResult = 200000;
    int result = instance.getTotal();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
      fail("The test case is a prototype.");
}
```

- Test get id

```
@Test
public void testGetTglPmbyr() {
    System.out.println("getTglPmbyr");
    PembayaranModel instance = new PembayaranModel("PMB0014","PMS0014","KSR0003","2019-11-19",200000);
    String expResult = "2019-11-19";
    String result = instance.getTglPmbyr();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
//      fail("The test case is a prototype.");
}
```

B. Screenshoot hasil pengujian JUnit



A. JUnit/PhpUnit untuk pengujian Class : (Pemesanan)

```java
package Model;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author PUTRI
 */
public class PemesananModelTest {

    public PemesananModelTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }
```

- Test total

```java
    @Test
    public void testSetTotal() {
        System.out.println("setTotal");
        int total = 2;
        PemesananModel instance = new PemesananModel("PMS0015",2,"2019-11-19");
        instance.setTotal(total);
        // TODO review the generated test code and remove the default call to fail.
/         fail("The test case is a prototype.");
    }
```

- Test tanggal_pemesanan

```
@Test
public void testSetTglPmsn() {
    System.out.println("setTglPmsn");
    String tglPmsn = "2019-11-19";
    PemesananModel instance = new PemesananModel("PMS0015",2,"2019-11-19");
    instance.setTglPmsn(tglPmsn);
    // TODO review the generated test code and remove the default call to fail.
      fail("The test case is a prototype.");
}
```

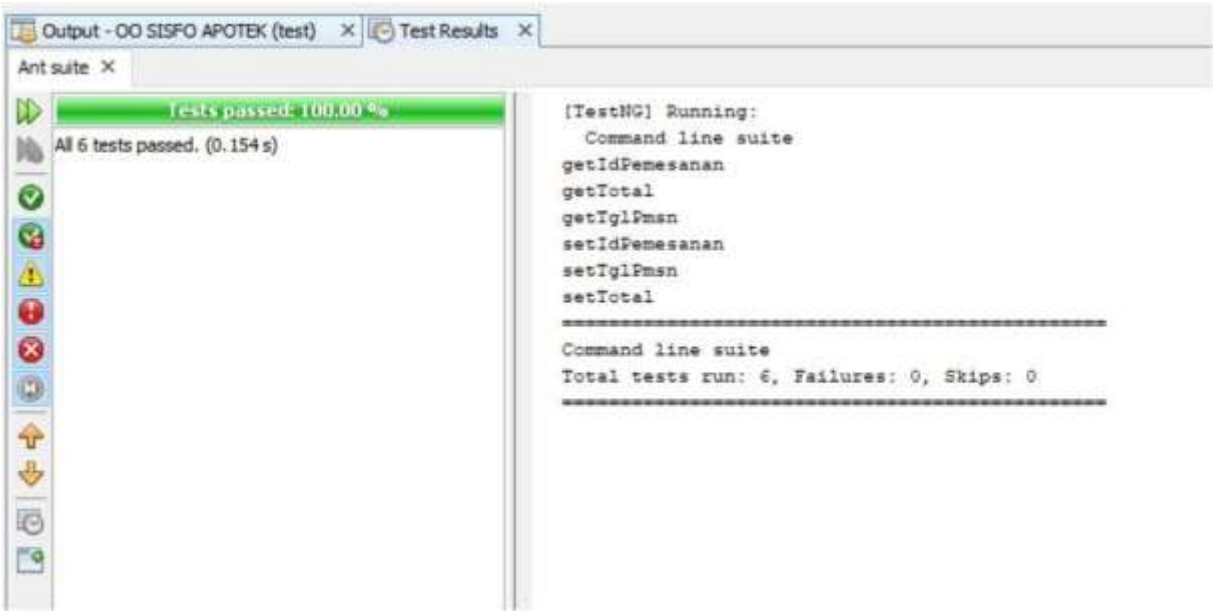
- Test id_pemesanan

```
     */
    @Test
    public void testGetIdPemesanan() {
        System.out.println("getIdPemesanan");
        PemesananModel instance = new PemesananModel("PMS0015",2,"2019-11-19");
        String expResult = "PMS0015";
        String result = instance.getIdPemesanan();
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
//        fail("The test case is a prototype.");
    }
```

- Test get_total

```
     */
    @Test
    public void testGetTotal() {
        System.out.println("getTotal");
        PemesananModel instance = new PemesananModel("PMS0015",2,"2019-11-19");
        int expResult = 2;
        int result = instance.getTotal();
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
//        fail("The test case is a prototype.");
    }
```

B. Screenshoot hasil pengujian JUnit



2.2 Pengujian USE CASE

Di bagian ini dijelaskan pengujian terhadap setiap use case.

2.2.1 Pengujian DUPL-01 Login User

Pengujian *login* terbagi menjadi dua bagian diantaranya pendaftaran *user* baru dan penggecekan user yang telah terdaftar sebagai berikut :

Tabel 1 Pengujian Login

USE CASE	Kasus dan Hasil Uji (Data normal)			
	Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Login	User ID: password: konfirmasi password:rahasia	Form menampilkan data user baru untuk *user* peneliti dan responden	Dapat melakukan pengisian data *user* baru Sesuai yang diharapkan	[X] diterima [ ] ditolak
	Klik tombol Login	Data tersimpan di file *User* peneliti dan *user* Responden	Data pengisian *user* Responden dan peneliti tersimpan Sesuai yang diharapkan	[X] diterima [ ] ditolak
	Klik tombol Reset	Data yang telah terisi telah dihapus	Data telah terhapus	[X] diterima

			sesuai yang diharapkan	[ ] ditolak

2.2.1.1 Pengujian DUPL-01_01 Pendaftaran *User* Baru

Berikut ini adalah tabel pengujian *login* untuk pendaftaran *user* baru :

Tabel 2 Pengujian Pendaftaran *User* Baru

USE CASE	Kasus dan Hasil Uji (Data normal)			
	Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Registrasi	User ID: Universitas Nama : xyz password: rahasia konfirmasi password:rahasia	Form menampilkan data user baru untuk *user* peneliti dan responden	Dapat melakukan pengisian data *user* baru Sesuai yang diharapkan	[X] diterima [ ] ditolak
	Klik tombol simpan	Data tersimpan di file *User* peneliti dan *user* Responden	Data pengisian *user* Responden dan peneliti tersimpan Sesuai yang diharapkan	[X] diterima [ ] ditolak
	Klik tombol Reset	Data yang telah terisi telah dihapus	Data telah terhapus sesuai yang diharapkan	[X] diterima [ ] ditolak

Use CASE	Kasus dan Hasil Uji (Data salah:)			
	Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Login	Data Login *User* Peneliti dan Responden tidak terdaftar. User Id : aaa Password : aaa	Tidak Dapat *login* dan Menampilkan pesan “Login Gagal”	*User* tidak dapat *login* dan memberikan pesan ”login gagal” sesuai yang diharapkan	[X] diterima [ ] ditolak
	Data Login *User* Peneliti dan	Tidak Dapat melakukan pendaftaran karena	Mengeluarkan pesan User	[X] diterima

	Responden telah terdaftar : User Id : universitas Nama : xyz password : rahasia Konfirmasi password : rahasia	userid telah terdaftar	Id telah terdaftar	[ ] ditolak

2.2.1.2 Pengujian DUPL- Input Data Pemesanan Kasir

Berikut ini adalah tabel pengujian input data pemesanan kasir:

Tabel 3 Input Data Pemesanan kasir

USE CASE	Kasus dan Hasil Uji (Data normal)			
	Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Input	klik tombol/icon tambah	menampilkan form dan user melakukan inputan pada form tersebut	System menampilkan form dan user dapat menginputkannya	[X] diterima [ ] ditolak
	Klik tombol simpan	Data tersimpan di database yang sudah disedikan	Data pemesanan kasir Responden dan peneliti tersimpan Sesuai yang diharapkan	[X] diterima [ ] ditolak

2.2.1.3 Pengujian DUPL- Edit Data Pemesanan Kasir

Berikut ini adalah tabel pengujian input data pemesanan kasir:

Tabel 3 Input Data Pemesanan kasir

USE CASE	Kasus dan Hasil Uji (Data normal)			
	Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Edit	Menekan tombol/icon edit	menampilkan form dan user melakukan inputan editan pada form tersebut	System menampilkan form dan user dapat editan menginputkannya	[X] diterima [ ] ditolak

	Klik tombol simpan	Data tersimpan di database yang sudah disedikan	Data pemesanan kasir yang diedit Responden dan peneliti tersimpan Sesuai yang diharapkan	[X] diterima [ ] ditolak

2.2.1.4 Pengujian DUPL- Hapus Data Pemesanan Kasir

Berikut ini adalah tabel pengujian input data pemesanan kasir:

Tabel 3 Input Data Pemesanan kasir

USE CASE	**Kasus dan Hasil Uji (Data normal)**			
	Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Hapus	Menekan tombol/icon hapus	menampilkan form dan user melakukan inputan id pada data yang ingin di hapus	System menampilkan form penghapusan data	[X] diterima [ ] ditolak
	Klik tombol hapus	Data terhapus dari database yang sudah disedikan	Data pemesanan kasir yang dihapus Responden dan peneliti tersimpan Sesuai yang diharapkan	[X] diterima [ ] ditolak

2.3 Kesimpulan Pengujian

(Berisi laporan dari pengujian yang telah dilakukan, dengan menyampaikan informasi status dari setiap fungsional yang diuji apakah telah berhasil/tidak)

Kelas Uji	**Butir Uji**	**Kesimpulan pengujian**
Login User	Pendaftaran *User* baru (kasus Uji : Data Normal)	Diterima
	Pendaftaran *User* baru (kasus Uji : Data Salah)	Diterima
	Pengecekan *User* yang telah terdaftar (kasus Uji : Data Normal)	Diterima

	Pengecekan *User* yang telah terdaftar (kasus Uji : Data Salah)	Diterima

Lampiran

A. Hasil pengukuran OOMetric aplikasi yang telah berhasil dibangun dengan software (tool) pengukuran OOMetric (lihat http://www.virtualmachinery.com/jhawkmetricsclass.htm)