BAB III

ANALISA DAN PERANCANGAN SOFTWARE

3.1 Analisa Kebutuhan Software

Pada bab ini penulis akan menganalisa kebutuhan perangkat lunak pembuatan aplikasi perhitungan komponen SMD *(surface mount device)* dari identifikasi masalah sampai dengan analisa kebutuhannya.

3.1.1 Identifikasi Masalah

identifikasi permasalahan pada pembuatan aplikasi ini sebagai berikut:.

1. Sangat sulit untuk menghitung nilai suatu komponen SMD secara manual, karena teknisi harus menghafal suatu tabel komponen SMD beserta rumusnya.
2. Banyak masyarakat, pelajar, mahasiswa yang belum mengerti akan pengetahuan komponen SMD.
3. Banyak kendala yang dialami teknisi reparasi perangkat *handphone* dan komputer ketika mendapatkan kasus *motherboard* yang mengalami kerusakan komponen SMD, yaitu sulitnya membedakan tipe antar komponen SMD.

3.1.2 Analisa Kebutuhan

Dalam perancangan aplikasi perhitungan komponen SMD, penulis menganalisa kebutuhan *hardware* dan *software* yang diperlukan yaitu sebagai berikut :.

1. Komponen Hardware.

   Komputer yang digunakan penulis mempunyai spesifikasi sebagai berikut.

   1) Tipe : Intel pentium Core i3 3217u.
   2) HDD : 500 GB
   3) RAM : 4 GB
   4) *Proccesor* : *Inte1 Core i3 1.8 GHz*
   5) *Graphic* : *Nvidia Geforce* 740m

2. Komponen Software

   Software yang digunakan penulis pada laptop sebagai berikut.

   1) Sistem Operasi Windows 10

      Adalah sistem operasi komputer yang dikembangkan oleh Microsoft.

   2) Android Studio

      Merupakan sebuah *tools* untuk pengerjaan *project* aplikasi yang sudah tersedia didalamnya berupa android SDK.

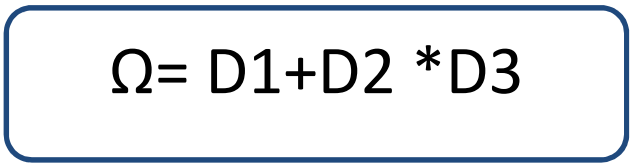
   3) Java JDK

      Java Development Kit (JDK) digunakan untuk *plugin* bahasa pemrograman java.

3.2 Desain

3.2.1 Rancangan Algoritma pada kasus

Rancangan algoritma pada resisitor 3 digit adalah sebagai berikut,



Sumber: Pribadi

Gambar III. 1. Algortima Menghitung Resistor SMD 3 digit

Keterangan :

Ω = Nilai Hambatan (Ω)
D1 = Nilai Pada Digit 1 (menunjukan angka pertama)
D2 = Nilai Pada Digit 2 (menujukan angka kedua)
D3 = Nilai Pada Digit 3 (menunjukan angka multiplier)

Kasus:
Diketahui jika resistor SMD 3 Digit nilai sebagai berikut
D1 = 1, D2 = 1, D3=10^3

Ω=(10+1)*10^3

Ω=(11)*10^3

Ω=11*1000

Ω=11000 Ω / 11 k Ω

3.2.2 Software Architecture

1. Pseudocode Algoritma

Pseudocode algoritma aplikasi komponen SMD dapat ditunjukkan sebagai berikut:

```
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.aksi_menghapus) {

            // add your code here

        if ( a==3)

        {
            ResistorSMD3Digit.NilaiResistansi.setText("");
            ResistorSMD3Digit.kodedigitsatu.setText("1");
            ResistorSMD3Digit.kodedigitdua.setText("0");
            ResistorSMD3Digit.kodedigittiga.setText("0");
            ResistorSMD3Digit.k1=10;
            ResistorSMD3Digit.k2=0;
            ResistorSMD3Digit.k3=1;
            ResistorSMD3Digit.NilaiResistansi.setText("");
        }

        else if (a==5)
        {
            ResistorSMD4Digit.kodegelangsatu.setText("1");
            ResistorSMD4Digit.kodegelangdua.setText("0");
            ResistorSMD4Digit.kodegelangtiga.setText("0");
            ResistorSMD4Digit.kodegelangempat.setText("0");
            ResistorSMD4Digit.k1 = 100;
            ResistorSMD4Digit.k2 = 0;
            ResistorSMD4Digit.k3 = 0;
            ResistorSMD4Digit.k4 = 1;
            ResistorSMD4Digit.NilaiResistansi.setText("");
        }

        else if (a==7)
        {
            ResistorSMDEIA96.kodegelangsatu.setText("01");
            ResistorSMDEIA96.kodegelangdua.setText("A");
            ResistorSMDEIA96.k1=100;
            ResistorSMDEIA96.k2=1;
            ResistorSMDEIA96.NilaiResistansi.setText("");
        }

        else if (a==9)
        {
            KapasitorSMDVentax.kodehurufkapven.setText("1");
            KapasitorSMDVentax.kodeangkakapven.setText("0");
            KapasitorSMDVentax.n1=1.0;
            KapasitorSMDVentax.n2=2;
            KapasitorSMDVentax.NilaiKapasitansi.setText("");
        }
```

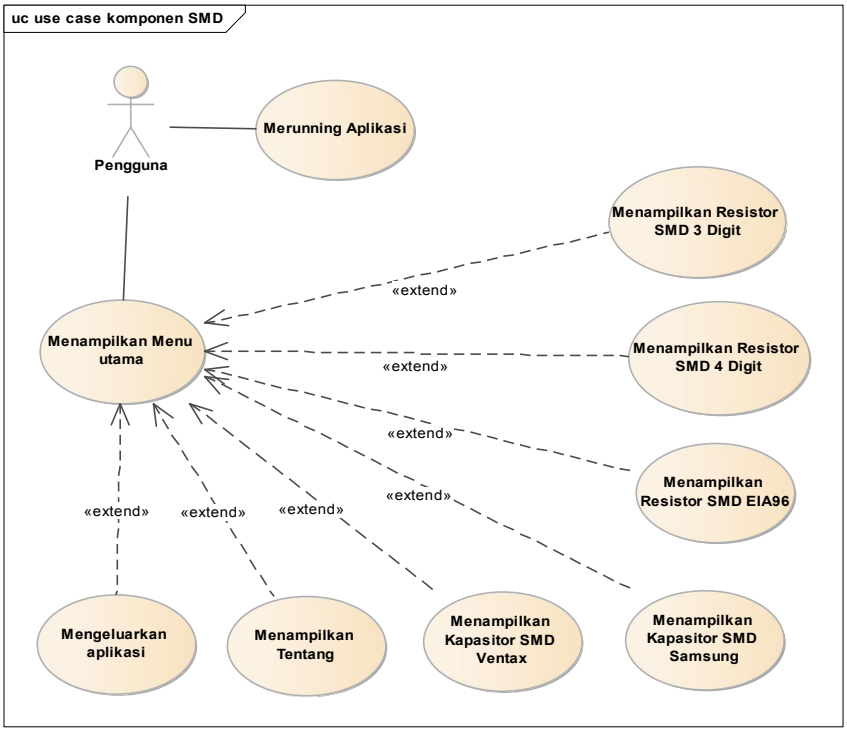
```
            else if (a==11)
            {
                KapasitorSMDSamsung.kodedigitkapsam.setText("1");
                KapasitorSMDSamsung.kodedigit2kapsam.setText("0");
                KapasitorSMDSamsung.n1=0;
                KapasitorSMDSamsung.n2=0;
                KapasitorSMDSamsung.NilaiKapasitansi.setText("");
            }

            return true;
        }

    if (id == R.id.aksi_menghitung)
            {

            // add your code here

        if ( a==3)

           {

            ResistorSMD3Digit.knhasil=(ResistorSMD3Digit.k1+ResistorS
            MD3Digit.k2)*ResistorSMD3Digit.k3;

              if (ResistorSMD3Digit.knhasil > 999 &&
                  ResistorSMD3Digit.knhasil < 999999    )
                {

                double nfinal;

                    nfinal = ResistorSMD3Digit.knhasil/1000;

                    ResistorSMD3Digit.NilaiResistansi.setText(nfinal+"
                    "+"K");
                }

                else if (ResistorSMD3Digit.knhasil > 999999    )

                {
                    double nfinal;

                    nfinal = ResistorSMD3Digit.knhasil/1000000;

                    ResistorSMD3Digit.NilaiResistansi.setText(nfinal+"
                    "+"M");
                }

                else (ResistorSMD3Digit.knhasil < 999    )

                {

                 ResistorSMD3Digit.NilaiResistansi.setText(String.
                 valueOf(ResistorSMD3Digit.knhasil)+" " )
                }
```

```
else if (a==5)

     {

     ResistorSMD4Digit.knhasil=(ResistorSMD4Digit.k1+ResistorS
     MD4Digit.k2+ResistorSMD4Digit.k3)*ResistorSMD4Digit.k4;

     if (ResistorSMD4Digit.knhasil > 999 &&
         ResistorSMD4Digit.knhasil < 999999    )
        {
          double nfinal;

          nfinal = ResistorSMD4Digit.knhasil/1000;

          ResistorSMD4Digit.NilaiResistansi.setText(nfinal+"
          "+"K");
        }

     else if (ResistorSMD4Digit.knhasil > 999999    )
        {
          double nfinal;

          nfinal = ResistorSMD4Digit.knhasil/1000000;


          ResistorSMD4Digit.NilaiResistansi.setText(nfinal+"
          "+"M");
         }

     else if (ResistorSMD4Digit.knhasil < 999    )
         {

           ResistorSMD4Digit.NilaiResistansi.setText(String.va
           lueOf(ResistorSMD4Digit.knhasil)+" " );

          }

     }


  else if (a==7)
     {

       ResistorSMDEIA96.knhasil = ResistorSMDEIA96.k1 *
       ResistorSMDEIA96.k2;

      if (ResistorSMDEIA96.knhasil > 999 &&
          ResistorSMDEIA96.knhasil < 999999    )
          {
             double nfinal;

             nfinal = ResistorSMDEIA96.knhasil/1000;
            ResistorSMDEIA96.NilaiResistansi.setText(nfinal+"
            "+"K");
           }
```

```
        else if (ResistorSMDEIA96.knhasil > 999999     )
            {

             double nfinal;

             nfinal = ResistorSMDEIA96.knhasil/1000000;

             ResistorSMDEIA96.NilaiResistansi.setText(nfinal+"
             "+"M");
            }

         else (ResistorSMDEIA96.knhasil < 999     )
            {

              ResistorSMDEIA96.NilaiResistansi.setText(String.v
              alueOf(ResistorSMDEIA96.knhasil)+"" );

             }
    }
```

2. Pemodelan UML

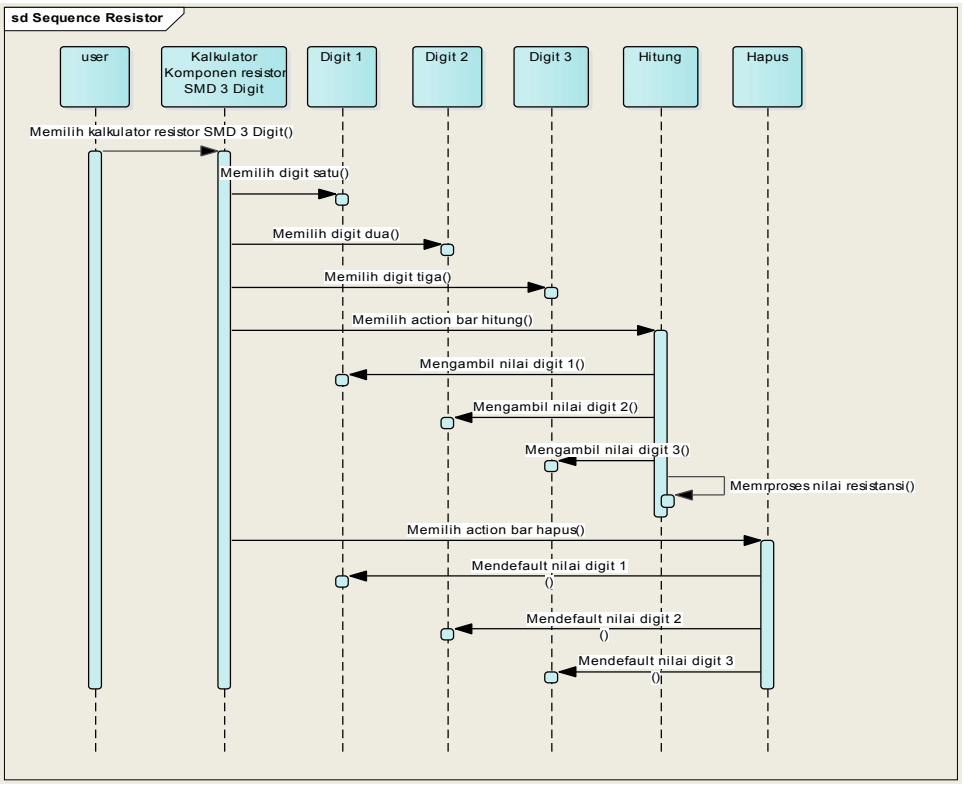
a. Diagram Use Case Aplikasi Komponen SMD.



Sumber: Pribadi

Gambar III. 2. Diagam Use Case Aplikasi Komponen SMD

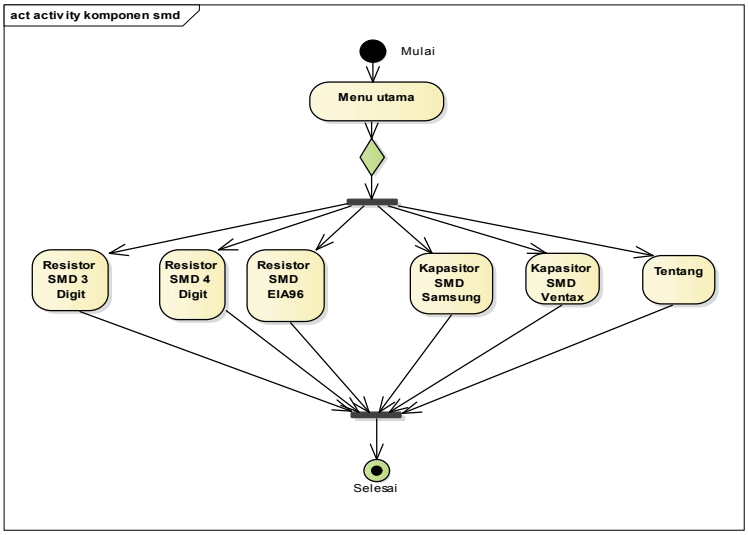
b. Diagram *Sequence* memilih kalkulator komponen resistor SMD 3 digit



Sumber: Pribadi

Gambar III. 3. Diagam Sequence memilih kalkulator komponen resistor SMD 3 digit

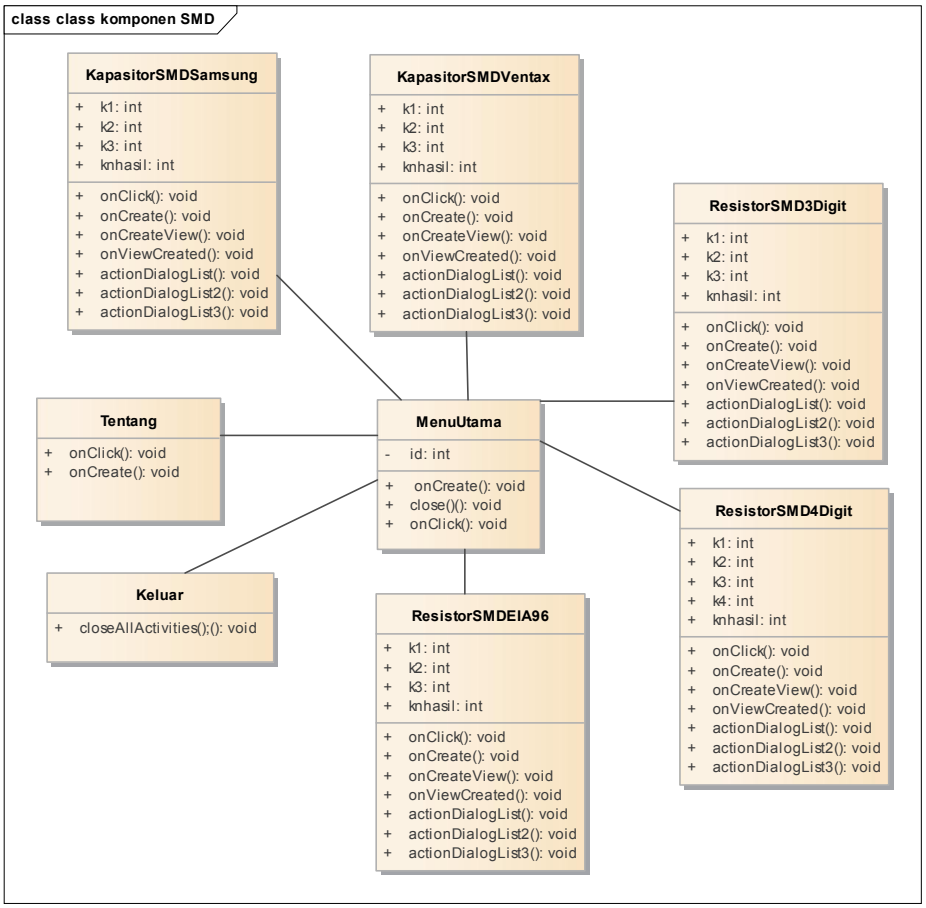
c. Diagram *Activity* Aplikasi Komponen SMD



Sumber: Pribadi

Gambar III. 4. Diagram Activity Aplikasi Kalkulator Komponen SMD

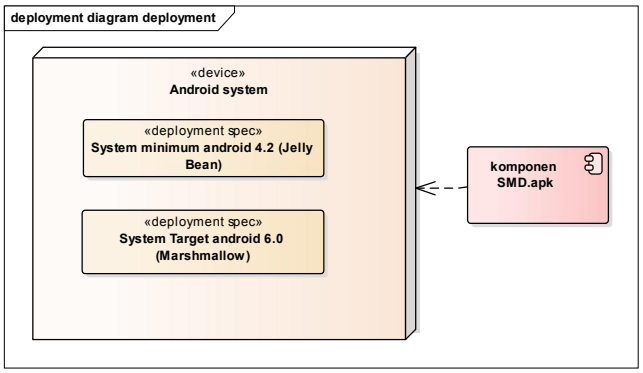
d. Diagram Class Aplikasi Kalkulator Komponen SMD



Sumber: Pribadi

Gambar III. 5. Diagram Class Aplikasi Kalkulator Komponen SMD

e. Diagram Deployment



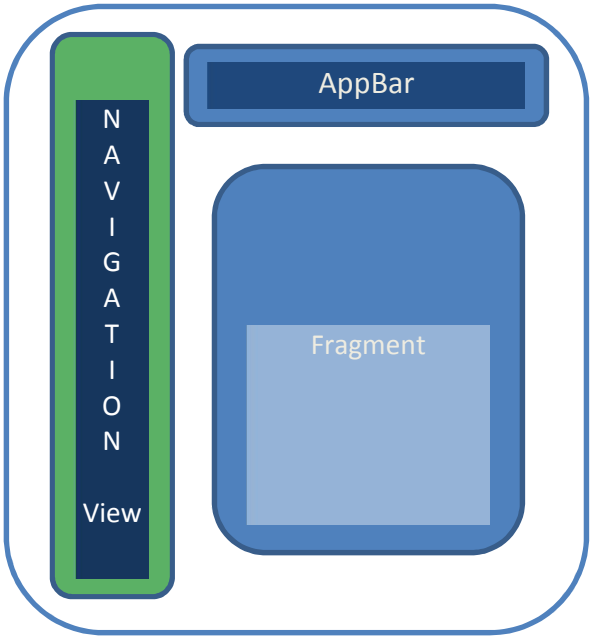
Sumber: Pribadi

Gambar III. 6. Diagram Deployment Aplikasi Komponen SMD

3.2.3 User Interface

User Interface harus dibuat sedemikian rupa. Aplikasi dengan user interface menarik, interaktif dan tentunya user friendly akan memudahkan pengguna aplikasi dalam penggunaan aplikasinya. Dalam perancangan *User Interface* pada aplikasi komponen SMD penulis membuat rancangan sebagai berikut:

1. Menu Utama



Sumber: Pribadi

Gambar III. 7. *User Interface* Menu Utama

a. AppBar

Untuk tools bar yang terdapat tools untuk menghitung dan menghapus.

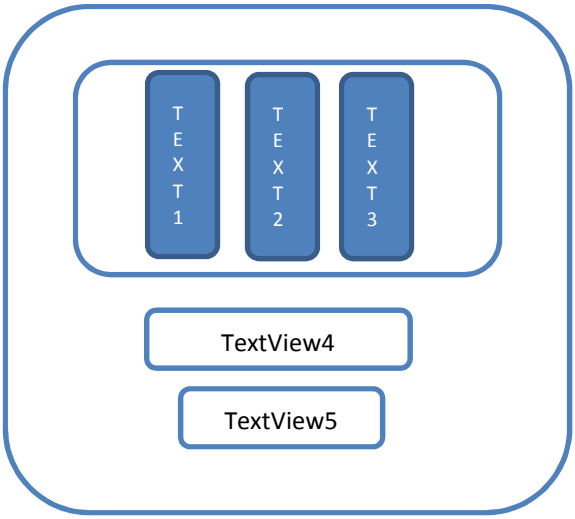
b. NavigationView

Untuk menampilkan macam-macam menu seperti kalkulator *resistor* SMD, kalkulator *kapasitor* SMD , menu tentang serta pilihan untuk keluar dari aplikasi.

c. Fragment

Berisi tempat untuk menampung class yang dipanggil seperti class ResistorSMD3Digit, ResistorSMDEIA96 dan class yang lainnya.

2. Menu Kalkulator Resistor SMD 3 Digit



Sumber: Pribadi

Gambar III. 8. Menu Kalkulator Resistor SMD 3 Digit

a. TextView1

TextView yang berfungsi untuk menampilkan deretan nilai kode resistor SMD pada digit 1.

b. TextView2

TextView yang berfungsi untuk menampilkan deretan nilai kode resistor SMD pada digit 2.

c. TextView3

TextView yang berfungsi untuk menampilkan deretan nilai kode resistor SMD pada digit 3.

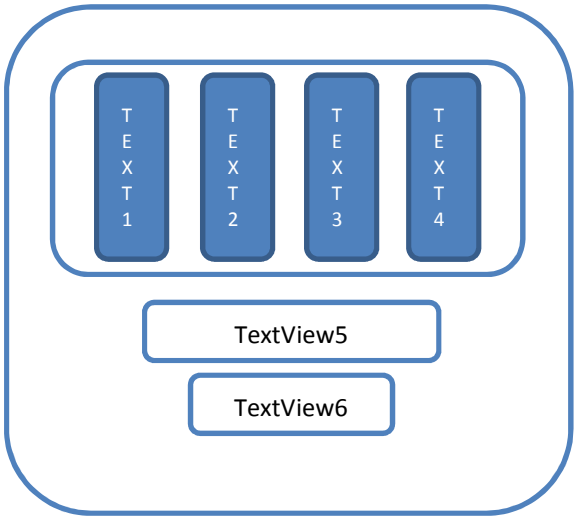
d. TextView4

TextView yang berisi text “nilai resistansi”

e. TextView5

TextView yang berfungsi untuk menampilkan hasil dari suatu proses perhitungan resistor SMD 3 digit.

3. Menu Kalkulator Resistor SMD 4 Digit



Sumber: pribadi

Gambar III. 9. User Interface Resistor SMD 4 Digit

a. TextView1

   TextView yang berfungsi untuk menampilkan deretan nilai kode resistor SMD pada digit 1

b. TextView2

   TextView yang berfungsi untuk menampilkan deretan nilai kode resistor SMD pada digit 2.

c. TextView3

   TextView yang berfungsi untuk menampilkan deretan nilai kode resistor SMD pada digit 3.

d. TextView4

   TextView yang berfungsi untuk menampilkan deretan nilai kode resistor SMD pada digit 4.

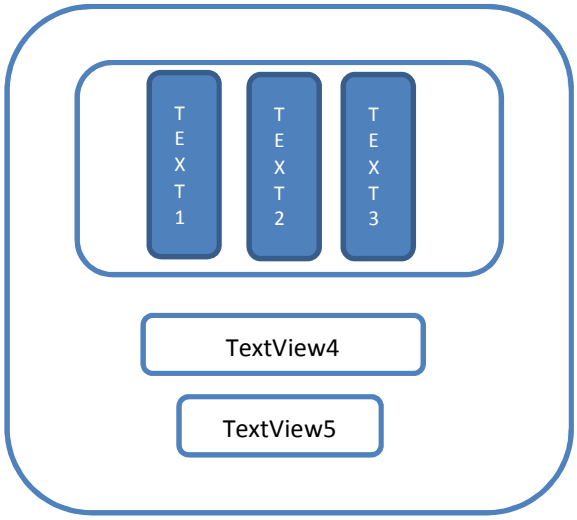
e. TextView5

   TextView yang berisi text “nilai resistansi”.

f. TextView6

   TextView yang berfungsi untuk menampilkan hasil dari suatu proses perhitungan resistor SMD 4 digit.

4. Menu Kalkulator Resistor SMD EIA96



Sumber: Pribadi

Gambar III. 10. User Interface Resistor SMD 4 Digit

a. TextView1

   TextView yang berfungsi untuk menampilkan deretan nilai kode resistor SMD pada digit 1

b. TextView2

   TextView yang berfungsi untuk menampilkan deretan nilai kode resistor SMD pada digit 2.

c. TextView3

   TextView yang berfungsi untuk menampilkan deretan nilai kode resistor SMD pada digit 3.

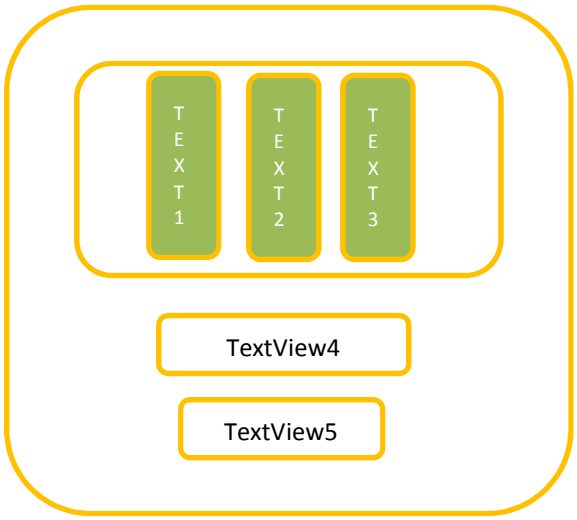
d. TextView4

   TextView yang berisi text “nilai resistansi”

e. TextView5

TextView yang berfungsi untuk menampilkan hasil dari suatu proses perhitungan resistor SMD EIA96.

5. Menu Kalkulator Kapasitor SMD Samsung



Sumber: Pribadi

Gambar III. 11. User Interface Kapasitor SMD Samsung

a. TextView1

TextView yang berfungsi untuk menampilkan deretan nilai kode kapasitor SMD samsung pada digit 1

b. TextView2

TextView yang berfungsi untuk menampilkan deretan nilai kode kapasitor SMD samsung pada digit 2.

c. TextView3

TextView yang berfungsi untuk menampilkan deretan nilai kode kapasitor SMDsamsung pada digit 3.

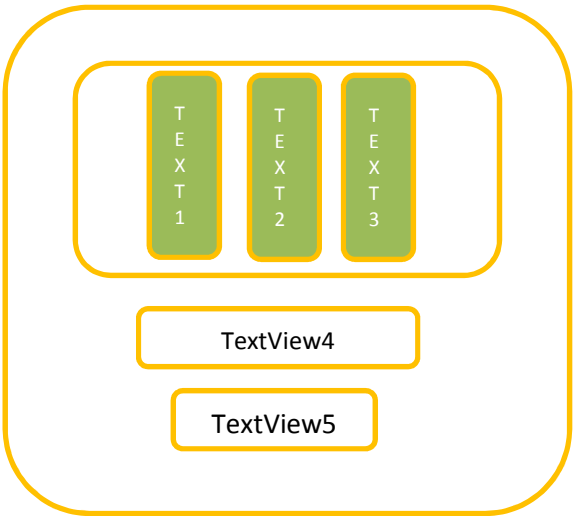
d. TextView4

TextView yang berisi text “nilai kapasitasni”

e. TextView5

TextView yang berfungsi untuk menampilkan hasil dari suatu proses perhitungan kapasitor SMD samsung 3 digit.

6. Menu Kalkulator Kapasitor SMD Ventax



Sumber: Pribadi

Gambar III. 12. User Interface Kapasitor SMD Ventax

a. TextView1

TextView yang berfungsi untuk menampilkan deretan nilai kode kapasitor SMD Ventax pada digit 1

b. TextView2

TextView yang berfungsi untuk menampilkan deretan nilai kode kapasitor SMD Ventax pada digit 2.

c. TextView3

TextView yang berfungsi untuk menampilkan deretan nilai kode kapasitor SMD Ventax pada digit 3.

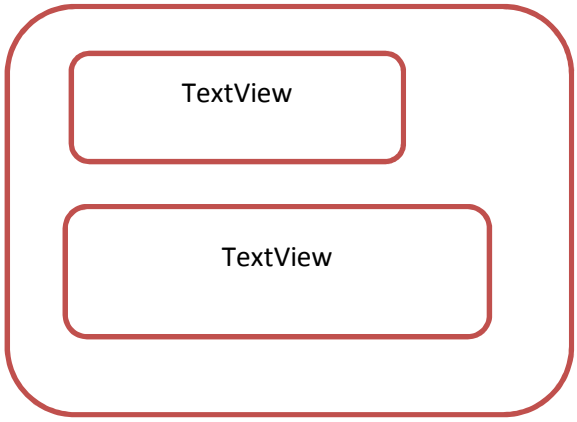
d. TextView4

TextView yang berisi text “nilai kapasitasni”

e. TextView5

TextView yang berfungsi untuk menampilkan hasil dari suatu proses perhitungan kapasitor SMD Ventax 3 digit.

7. Menu Tentang



Sumber: Pribadi

Gambar III. 13. Menu Tentang

a. TextView1

TextView yang berisi “Tentang Aplikasi”.

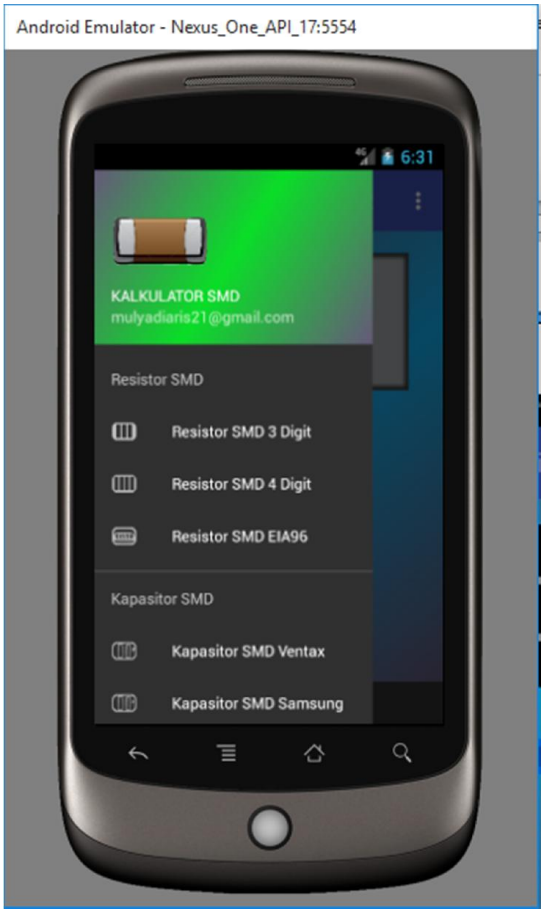
b. TextView2

TextView yang berisi deskripsi aplikasi

3.3 Implementasi

Setelah melakukan perancangan *user interface*, penulis menerapkan UI yang dibuat ke android studio tools dengan membuat desain sesuai dengan perancangan *user interface.* Berikut tampilan aplikasinya:

1. Desain Menu



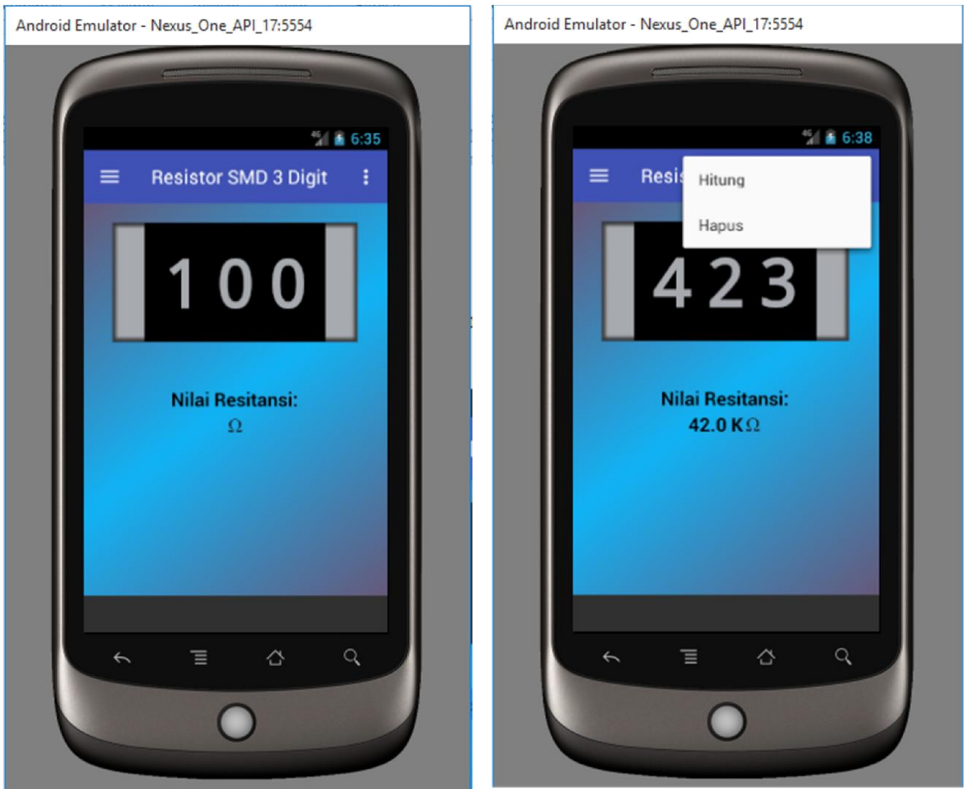
Sumber: Pribadi

Gambar III. 14.Desain Menu Utama

Dengan listing inti *Java* sebagai berikut:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.layout_menu_utama);
```

2. Desain Kalkulator resistor SMD 3 Digit



Sumber: Pribadi
Gambar III. 15. Desain Kalkulator

Dengan listing inti *Java* sebagai berikut:

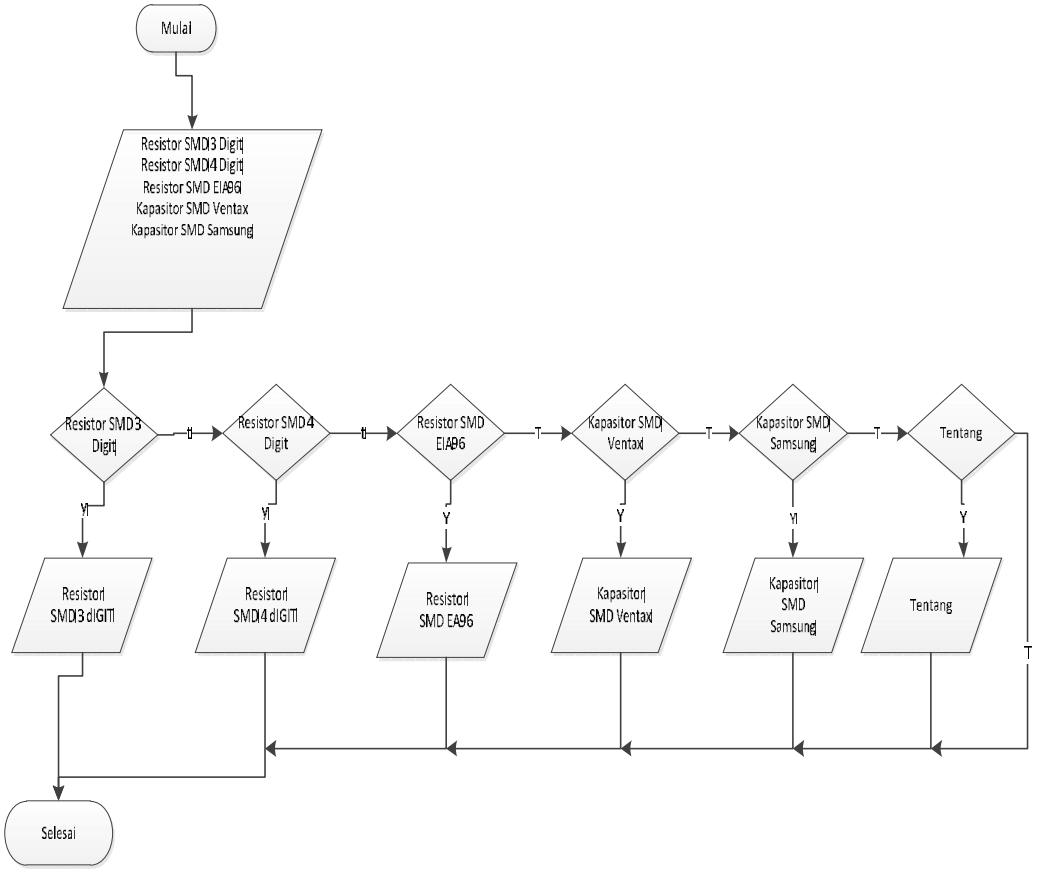
```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState)
{
  //returning our layout file
  //change R.layout.yourlayoutfilename for each of your
    fragments

return inflater.inflate(R.layout.layout_resistorsmd3digit,
container, false);

}
```

3.4 Testing

Dalam pengujian aplikasi ini penulis menggunakan *white box* dan *black box*. Adapun untuk pengujian *white box* dengan menggunakan skema diagram alir.

3.4.1. Flowchart

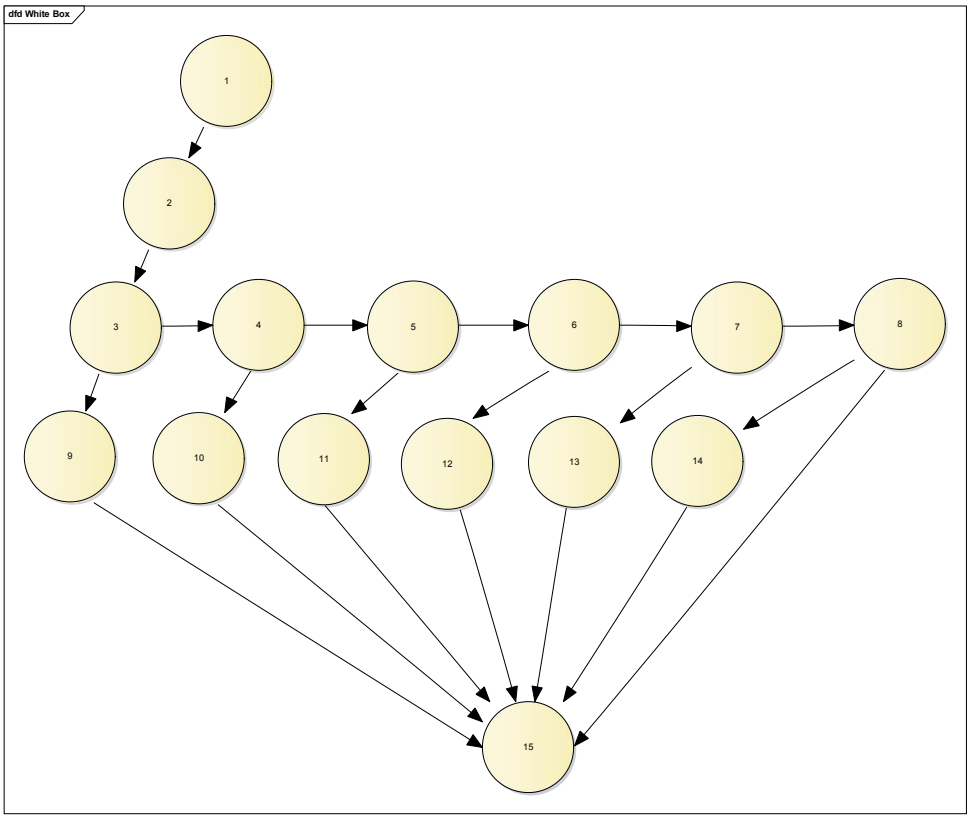
Berikut *adalah Flowchart* atau diagram alir kalkulator komponen SMD dibuat untuk pengujian *white box.*



Gambar III. 16. *Flowchart* kalkulator komponen SMD.

3.4.2. Pengujian White Box

White Box alir Komponen SMD ditunjukan pada gambar berikut:



Gambar III. 17. *White Box* Alir Kalkulator Komponen SMD

Sedangkan *script pada white box* alir kalkulator komponen SMD adalah sebagai berikut:

1

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
    setContentView(R.layout.layout_menu_utama);
toolbar = (Toolbar) findViewById(R.id.toolbar);
toolbar.setTitle("Resistor SMD 3 Digit");

    setSupportActionBar(toolbar);
```

```
@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
// Handle navigation view item clicks here.

displaySelectedScreen(item.getItemId());


return true;
}



private void displaySelectedScreen(int itemId) {

//creating fragment object
Fragment fragment = null;
    Fragment fragment2 = null;

//initializing the fragment object which is selected
switch (itemId) {


case R.id.id_resistorsmd3:
            fragment = new ResistorSMD3Digit();
a=3;

toolbar.setTitle("Resistor SMD 3 Digit");


break;


case R.id.id_resistorsmd4:
            fragment = new ResistorSMD4Digit();

a=5;


toolbar.setTitle("Resistor SMD 4 Digit");

break;

case R.id.id_resistorsmdeia96:
            fragment = new ResistorSMDEIA96();

a=7;


toolbar.setTitle("Resistor SMD EIA96");

break;

case R.id.id_kapsmdventax:
```

```
            fragment = new KapasitorSMDVentax();

a=9;

toolbar.setTitle("Kapasitor SMD Ventax");

break;

case R.id.id_kapsmdsamsung:
            fragment = new KapasitorSMDSamsung();

a=11;

toolbar.setTitle("Kapasitor SMD Samsung");

break;

case R.id.id_tentang:
            fragment = new Tentang();


toolbar.setTitle("Tentang Aplikasi");

break;

case R.id.id_keluar:


            finish();

break;
```

```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
//returning our layout file
    //change R.layout.yourlayoutfilename for each of your
```

```
fragments


return inflater.inflate(R.layout.layout_resistorsmd3digit,
container, false);
```


```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
//returning our layout file
    //change R.layout.yourlayoutfilename for each of your
fragments
return inflater.inflate(R.layout.layout_resistorsmd4digit,
container, false);


}
```


```
public View onCreateView(LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
//returning our layout file
    //change R.layout.yourlayoutfilename for each of your
fragments

    // = (TextView) rootView.findViewById(R.id.NilaiResistansi);

return inflater.inflate(R.layout.layout_resistorsmdeia96,
container, false);


}
```


```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
//returning our layout file
```

```
    //change R.layout.yourlayoutfilename for each of your
fragments

    // = (TextView) rootView.findViewById(R.id.NilaiResistansi);

return inflater.inflate(R.layout.layout_kapasitorsmdsamsung,
container, false);


}
```

```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
//returning our layout file
    //change R.layout.yourlayoutfilename for each of your
fragments

    // = (TextView) rootView.findViewById(R.id.NilaiResistansi);

return inflater.inflate(R.layout.layout_kapasitorsmdventax,
container, false);


}
```

13

```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
//returning our layout file
    //change R.layout.yourlayoutfilename for each of your
fragments


return inflater.inflate(R.layout.layout_tentang, container,
false);


}
```

14

Setelah dijabarkan diatas, penulis menghitung kompleksitas siklomatis grafik alir *White Box* dapat diperoleh dengan cara sebagai berikut:

$V(G) = E – N + 2$

Dimana:

E = Jumlah Edge yang ditentukan gambar panah

N = Jumlah simpul grafik alir ditentukan dengan gambar lingkaran

$V(G) = 20- 15 + 2 = 7$

$V(G) < 10$ berarti memenuhi syarat kekomplesitasi siklomatisnya.

Baris set yang dihasilkan dari jalur independent adalah sebagai berikut

1. 1-2-3-9-15
2. 1-2-3-4-10-15
3. 1-2-3-4-5-11-15
4. 1-2-3-4-5-6-12-15
5. 1-2-3-4-5-6-7-13-15
6. 1-2-3-4-5-6-7-8-14-15
7. 1-2-3-4-5-6-7-8-15

Ketika aplikasi dijalankan, maka terlihat bahwa satu set baris yang dihasilkan adalah 1-2-3-9-15-1-2-3-4-10-15-1-2-3-4-5-11-15-1-2-3-4-5-6-12-15-1-2-3-4-5-6-7-13-15-1-2-3-4-5-6-7-8-14-15-1-2-3-4-5-6-7-8-15 dan terlibat bahwa simpul telah dieksekusi satu kali.

3.4.3. Pengujian Black Box

Adapun pengujian Black Box pada kalkulator *resistor* SMD 3 Digit sebagai berikut.

Tabel III. 1. Pengujian Black Box Kalkulator Resistor SMD 3 Digit

No.	Skenario Uji	Test Case	Hasil yang diharapkan	Valid
1	Memilih kode digit1	ActionDialogList1()	Menampilkan list digit1	Valid
2	Memilih kode digit2	ActionDialogList2()	Menampilkan list digit2	Valid
3	Memilih kode digit3	ActionDialogList3()	Menampilkan list digit3	Valid
5	Mengklik toolbar "Hitung"	Klik ToolBar Hitung	Menampilkan nilai resistansi	Valid
6	Mengklik toolbar "Hapus"	ToolBar Hapus	Menghapus ActionDialogList1() sampai dengan ActionDialogList4() dan menghapus TextView1 dan TextView2	Valid

Sumber: Pribadi

3.5. Support

Adapun spesifikasi *hardware* dan *software* yang memadai untuk pembuatan aplikasi kalkulator komponen SMD adalah sebagai berikut:

1. Hardware
    a. Spesifikasi Komputer
        1). Processor minimum Core i3
        2). RAM 2 GB
        3). VGA Card 512 MB
    b. Spesifikasi Smartphone
        1) Processor dual core
        2) RAM 1024 MB
2. Software
    a. Spesifikasi Komputer
        1). Windows 7 32 bit atau 64bit
        2). Windows 8 32 bit atau 64bit
        3). Windows 10 32 bit atau 64bit
    b. Spesifikasi Smartphone

        Aplikasi bisa dijalankan minimum pada OS A*ndroid Jelly Bean 4.2.2*