BAB 5 IMPLEMENTASI

Bab ini akan menjelaskan bagaimana implementasi sistem berdasarkan metode yang digunakan pada penelitian ini. Implementasi pada penelitian ini bertujuan untuk membuat sistem yang lebih dinamis agar dapat mengolah data sesuai dengan metode yang digunakan. Pada bab ini akan dibahas secara detail bagaimana sistem dibuat berserta tampilan antarmuka sistem.

1.1 Spesifiikasi Kebutuhan Sistem

Dalam pembuatan sistem diperlukan hardware maupun software yang kompatibel agar proses pembuatan sistem berkerja sesuai harapan. Berikut spesifikasi hardware dan software yang digunakan selama proses perancangan dan pembangunan sistem.

1.1.1 Spesifikasi Kebutuhan *Hardware*

Dalam pembangunan sistem akan digunakan sebuah laptop dengan spesifikasi yang ditunjukan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Kebutuhan Hardware Laptop

Nama Komponen	**Spesifikasi**
Processor	Core i-5
Memory(RAM)	2Gb
Hardisk	500Gb
Graphic Card	ATI Radeon 5000

1.1.2 Spesifikasi Kebutuhan *Software*

Dalam proses pengerjaan skripsi ini juga akan dibutuhkan software guna menyusun laporan, merancang sistem, membangun Aplikasi atau program dan sistem operasi. Adapun spesifikasi dari software yang dibutuhkan ditunjukan pada Tabel 5.2.

Tabel 5.2 Spesifikasi Kebutuhan Software

Nama Komponen	**Spesifikasi**
Sistem Operasi	Windows 10 Pro
Tools Dokumentasi	Microsoft Office Word 2016
Tools Diagram	Ms Visio
Bahasa Pemograman	Java, Android
Tools Pemograman	NetBeans, Android Studio 2.2

1.2 Implementasi AHP-TOPSIS

Implementasi AHP-TOPSIS akan dibahas bagaimana proses yang dilalui sistem dalam mengolah data sesuai metode yang digunakan pada penelitian ini. Agar mempermudah

pembahasan, maka proses tesebut akan dibagi menjadi 2bagian yaitu implementasi proses AHP dan Impementasi proses TOPSIS.

1.2.1 Implementasi Proses AHP

Impelementasi proses AHP akan dimulai dari normalisasi matriks perbandingan hingga ditemukan rata-rata setiap baris pada proses normalisasi yang akan digunakan pada proses TOPSIS. Selain dari rata-rata tersebut, proses ini akan dilanjutkan untuk mencari nilai dari konsistensi berupa lamda maksimum, ci dan cr dalam memeriksa kelakayan bobot yang digunakan. Berikan adalah pembahasan implementasi proses AHP yang dielaskan melalui source code atau kode program.

Kode program 5.1 adalah proses untuk menjumlahkan data dari masung-masing kolom matrik perbandingan berpasangan. Hasil penjumlahan masing-masing kolom ini akan dimasukan kedalam *array* 1 dimensi dengan nama a*rray* jumMper.

```
1  public void MatrikPerbandingan () {
2        jumMper = new double [data.isi [0].length];
3        for (int i = 0;i < data.isi[0].length;i++) {
4              jumMper[i]=0;
5              for(int j = 0; j < data.isi.length; j++) {
6                          JumMper[i] += data.isi[j][i];
7  }}}
```

Kode Program 5.1 Proses penjumlahan kolom

Penjelasan :

- Pada bari 2 menyatakan inisialisasi variable jumMper
- Pada baris 3 menyatakan looping dimana batasnya adalah kurang dari data.isi[0].length
- Pada baris 4 diguakan sebagai membuat angka default awal yakni dengan nilai 0
- Pada baris 5 menyatakan looping pada lapis kedua dengan batasannya adalah kurang dari data.isi.length
- Pada baris 6 berisi bagaimana jumMper akan diisi terus dengan data yang telah sesuai pada kolomnya.

Kode Program 5.2 adalah proses untuk menjalankan normalisasi data matrik perbandingan berpasangan. Pada proses ini akan dilakukan pembagian masing-masing data dengan jumlah dari setiap kolom yang tersimpan pada *array* jumMper. Hasil dari pembagian kemudian disimpan dalam *array* 2 dimensi dengan nama *array* normalisasi sesuai dari indeks matrik perbandingan berpasangan.

```
1  jumMper = new double[data.isi[0].length];
2  for (int i = 0; i < data.isi[0].length; i++) {
3  jumMper[i] = 0;
4          for (int j = 0; j < data.isi.length; j++) {
```

```
5  jumMper[i] += data.isi[j][i];
6  }}}
```

Kode Program 5.2 Proses Normalisasi

Penjelasan :

- Pada baris 2 adalah inisialisasi variable normalisasi
- Pada baris 3 adalah looping dengan batas kurang dari normalisasi.length
- Pada baris 4 adalah looping dari lapis kedua dengan batas normalisasi[0].length
- Pada baris 5 menyatakan pemberian nilai pada nomalisasi[i][j] dengan rummus data.isi[i][j] dibagi dengan jumMper[j]

Kode program 5.3 merupakan proses yang digunakan untuk mencari rata-rata setiap baris pada *array* normalisasi. Proses ini dilakukan dengan menjumlahkan setiap baris *array* normalisasi untuk dibagi dengan jumlah baris *array* tersebut. Hasil pencarian pada rata-rata baris akan disimpan kedalam sebuah *array* 1 dimensi dengan nama *array average.*

```
1  public void Average() {
2        average=new double[normalisasi.length];
3        For(int i = 0; i < normalisasi.lengthh; i++) {
4            average[i]=0;
5            for(int j = 0; i < normalisasi[0].length; j++) {
6            average[i] += normalisasi[i][j];
7            }
8            average[i]=average[i] / normalisasi[0].length;
9  }}
```

Kode Progarm 5.3 Proses pencarian rata-rata baris

Penjelasan :

- Pada baris 2 menyatakan inisialisasi dari variable average
- Pada baris 3 menyatakan looping dimana batasannya adalah kurang dari normalisasi.length
- Pada baris 4 digunakan untuk membuat angka default awal yakni dengan nilai 0
- Pada baris 5 menyatakan looping lapis ke 2 dengan batas pada normalisasi[0].length
- Pada baris 6 berisi bagaimana average akan dijumlah terus dengan normalisasi sesuai dengan kolomnya
- Pada baris 8 adalah isi rumus average

Kode Program 5.4 merupakan proses perhitungan untuk mengukur nilai konsistensi matrik perbandingan berpasangan. Proses ini dilakukan dengan mengalikan masing-masing data matrik perbandingan berpasangan dengan rata-rata setiap baris *array* normalisasi yang sudah tersimpan pada *array* average, kemudian dijumlahkan pada setiap baris. Hasil penjumlahan stiap bbaris kemudian disimpan didalam *array* 1 diemnsi dengan nama *array* konsistensi.

```
1  public void Konsistensi () {
2        konsistensi = new double[normalisasi.length];
```

```
3       for (int i = 0; i< normalisasi.length; i++ ) {
4                   konsistensi[i] = 0;
5                   for(int j = 0; j < normalisasi[0].length;
    j++)  {
6       konsistensi[i] += data.isi[i][j] – average[j];
7   }}}
```

Kode Program 5.4 Proses perhitungan nilai konsistensi

Penjelasan :

- Pada baris 2 merupakan inisialisasi variable konsistensi
- Pada baris 3 menyatakan looping pada batasannya dengan batas kuranng dari normalisasi.length
- Pada baris 4 digunakan untuk membuat angka default awal dengan nilai 0
- Pada baris 5 menyatakan looping lapis ke 2 dengan batas normalisasi[0].length
- Pada baris 6 berisi konsistensi merupakan

Kode Program 5.5 merupakan proses untuk mencari jumlah vektor bobot. Dimana proses ini dilakukan dengan membagi setiap indeks *array* konsistensi dengan indeks yang sama pada *array* average, kemudian dijumlhkan. Hasil penjumlahan tersebut kemudian disimpan kedalam variael lamdamax.

```
1   public void VektorJumlahBobot () {
2         vektorjumbobot = new double[konsistensi.length];
3         lamdamax = 0;
4         for(int i = 0; i < konsistensi.length; i++) {
5              vekjumbobot[i] = konsistensi[i] / average[i]
6               lamdamax += vekjumbobot[i];
7         }
8         lamdamax = lamdamax / konsistensi.length;
9   }
```

Kode Program 5.5 Proses pemjumlahan vektor bobot

Penjelasan :

- Pada baris 2 adalah inisialisasi variable vektorjumbobot
- Pada baris 3 merupakan inisialisasi dari lamdamax dengan nilai default awal yakno 0
- Pada baris 4 menyatakan looping dari batasan kurang dari konsistensi.length
- Pada baris 5 dan 8 digunakan untuk mencari nilai pada lamdamax

Kode Program 5.6 merupakan proses untuk mencari nilai ci. Dimana proses ini dilakukan dengan cara mengurangi nilai variabel lamdamax dengan nilai kriteria (panjang dari *array* konsistensi), kemudian dibagi nilai kriteria yang sudah dikurangi 1 Hasil dari proses ini kemudian disimpan dalam variabel ci.

```
1   public void Ci () {
2         kriteria = konsistensi.length;
3         Ci = (lamdamax – kriteria) / (kriteria – 1 );
4   }
```

Kode Program 5.6 Proses perhitungan nilai ci

Penjelasan :

- Pada baris 2 dan 3 dmerupakan proses untuk mencari nilai ci.

Kode Program 5.7 merupakan proses mencari nilai pada cr. Proses ini dilakukan dengan cara membagi nilai variabel ci dengan tetapan pada indeks kriteria yang sudah dikurangi 1. Hasil pada proses ini kemudian disimpan kedalam variabel cr.

```
1  public void Cr () {
2        cr = ci / ruleRi[Kriteria-1];


3  }
```

Kode Program 5.7 Proses perhitungan nilai cr

Penjeleasan :

- Pada baris 2 dan 3 digunakan untuk mencari nilai cr.

1.2.2 Implementasi Proses TOPSIS

Implementasi proses TOPSIS dimulai dari pengambilan data dari calon penerima bantuan hingga diperoleh urutan prioritas penerima bantuan . Berikut ini adalah pembahasan implementasi proses TOPSIS yang dijelaskan melalui kode program.

Kode Program 5.8 merupakan proses pengambilan data penerima bantuan untuk dimasukkan dalam *array* 2dimensi dengan nama *array* SUM untuk diolah pada tahapan proses selanjutnya. Hal ini dilakukan agar pengolahan pada proses berikutnya tidak mempengaruhi nilai variable data.

```
1  public void data() {
2       SUM = new double[data.isi.length][data.isi[0].length];
3       for (int i = 0; i < data.isi.length; i++) {
4            for (int j = 0; j < data.isi.length; j++) {
5                 SUM[i] += data.isi[i][j];
6  }}}
```

Kode Program 5.8 Proses pengambilan data

Penjelasan :

- Pada baris 2 merupakan inisialisasi variable
- Pada baris 3 dan 5 berisi sebuah looping, dimana looping tersebut berfungsi sebagai pengisian data yang akan diurutkan pada step berikutnya

Kode Program 5.9 merupakan proses perhitungan matriks ternormalisasi dimana setiap kriteria pada data yang digunakan sebagai sampel. Dimana hasil yang didapat pada perhitungan ini akan digunakan untuk mencari Alternatif bobot.

```
1  public double[][] normAlternatif() {
2    normAlternatif = new
     double[data.isi.length][data.isi[0].length];
3    for (int i = 0; i < normAlternatif.length; i++) {
4       for (int j = 0; j < normAlternatif[0].length; j++) {
5          normAlternatif[i][j] = data.isi[i][j] / SUM[j];
6  }}
7  return normAlternatif;
8  }
```

Kode Program 5.9 Proses perhitungan matriks ternormalisasi

Penjelesan :

- Pada baris 2 merupakan inisialisasi variable
- Pada baris 3-4 berisi sebuah looping 2 lapis yang batasannya adalah normAlternatif.length
- Pada baris 5 melakukan perhitungan matriks normalisasi

Kode Program 5.10 merupakjan proses perhitungan dari matriks normalisasi terbobot.

```
1  public double[][] normBobotP() {
2   normBobotP = new
   double[data.isi.length][data.isi[0].length];
3     for (int i = 0; i < normBobotP.length; i++) {
4         for (int j = 0; j < normBobotP[0].length; j++) {
5  normBobotP[i][j] = normAlternatif[i][j] *
   ahp.vekjumbobot[j];
6  }}
7  return normBobotP ;
8  }
```

Kode Program 5.10 Proses perhitungan matriks normalisasi terbobot

Penjelasan :

- Pada baris 2 merupakan inisialisai variable
- Pada baris 3 dan 4 berisi sebuah looping, dimana looping tersebut berisi 2 lapis
- Pada baris 5 berisi rumus perhitungan dimana hasil dari normBobotP adalah proses perhitungan dari normaAlternatif yang dikalikan dengan hasil perhitungan vektor jumlah bobot dengam nama variable ahp.vekjumbobot

Kode Program 5.11 merupakan proses perhitungan untuk mencari solusi Ideal Positif .

```
1  public double[] AP() {
2    AP = new double[data.isi[0].length];
3    for (int i = 0; i < data.isi[0].length; i++) {

4         AP[i] = normBobotP[0][i];
5         for (int j = 1; j < data.isi.length; j++) {
6           AP[i] = Math.max(AP[i], normBobotP[j][i]);
7  }}
```

```
8  return AP ;
9  }
```

Kode Program 5.11 Proses perhitungan matriks soulusi ideal positif

Penjelasan :

- Pada baris 2 merupakan inisialisasi variable
- Pada baris 3 berisi awalan looping
- Pada baris 6 merupakan rumus perhitungan untuk mencari solusi ideal positif

Kode Program 5.12 merupakan proses perhitungan yang digunakan untuk menentukan solusi ideal Negatif.

```
1  public double[] AN() {
2      AN = new double[data.isi[0].length];
3      for (int i = 0; i < data.isi[0].length; i++) {
4        AN[i] = normBobotP[0][i];
5        for (int j = 1; j < data.isi.length; j++) {
6          AN[i] = Math.min(AN[i], normBobotP[j][i]);
7  }}
8  return AN;
9  }
```

Kode Program 5.12 Proses perhitungan matriks solusi ideal negatif

Penjelasan :

- Pada baris 2 merupakan inisialisasi variable
- Pada baris 3 awalan looping
- Pada baris 6 merupakan rumus dan proses perhitungan untuk mencari solusi ideal negatif

Kode Program 5.13 merupakan proses perhitungan yang digunakan untuk mencari jarak alternatif dari solusi ideal positif.

```
1  public double[] IP() {
2  IP = new double[data.isi.length];
3    for (int i = 0; i < data.isi.length; i++) {
4    IP[i] = 0;
5       for (int j = 0; j < data.isi[0].length; j++) {
6       IP[i] += Math.pow(AP[j] - normBobotP[i][j], 2);
        }
7  IP[i] = Math.sqrt( IP[i]);
   }
8   return IP ;
9  }
```

Kode Program 5.13 Proses perhitungan jarak alternatif solusi ideal positif

Penjelasan :

- Baris 2 merupakan inisialisasi variable

- Baris ke 3 adalah proses looping awal
- Baris 7 merupakan rumus yang digunakan untuk proses perhitungan untuk menentukan jarak solusi ideal positif

Kode program 5.14 proses perhitungan yang digunakan untuk menentukan solusi ideal negatif.

```
1  public double[] IN() {
2    IN = new double[data.isi.length];
3     IN[i] = 0;
4       for (int j = 0; j < data.isi[0].length; j++) {
5       IN[i] += Math.pow(AN[j] - normBobotP[i][j], 2);
        }
6    IN[i] = Math.sqrt(IN[i]);
7  }
8   return IN;
9  }
```

Kode Program 5.14 Proses perhitungan jarak soulusi ideal negatif

Penjelasan :

- Baris 2 merupakan inisialisasi variable
- Baris 3 berisi proses dari looping awal
- Aris 6 rumus perhitungan yang digunakan untuk proses mencari jarak soulusi ideal negatif

Kode program 5.15 merupakan proses pembagian nilai matriks jarak solusi ideal negatif dengan negatif. Hasil dari proses ini akan disimpan dalam *array* 2 dimensi dengan nama *array* KedekatanR.

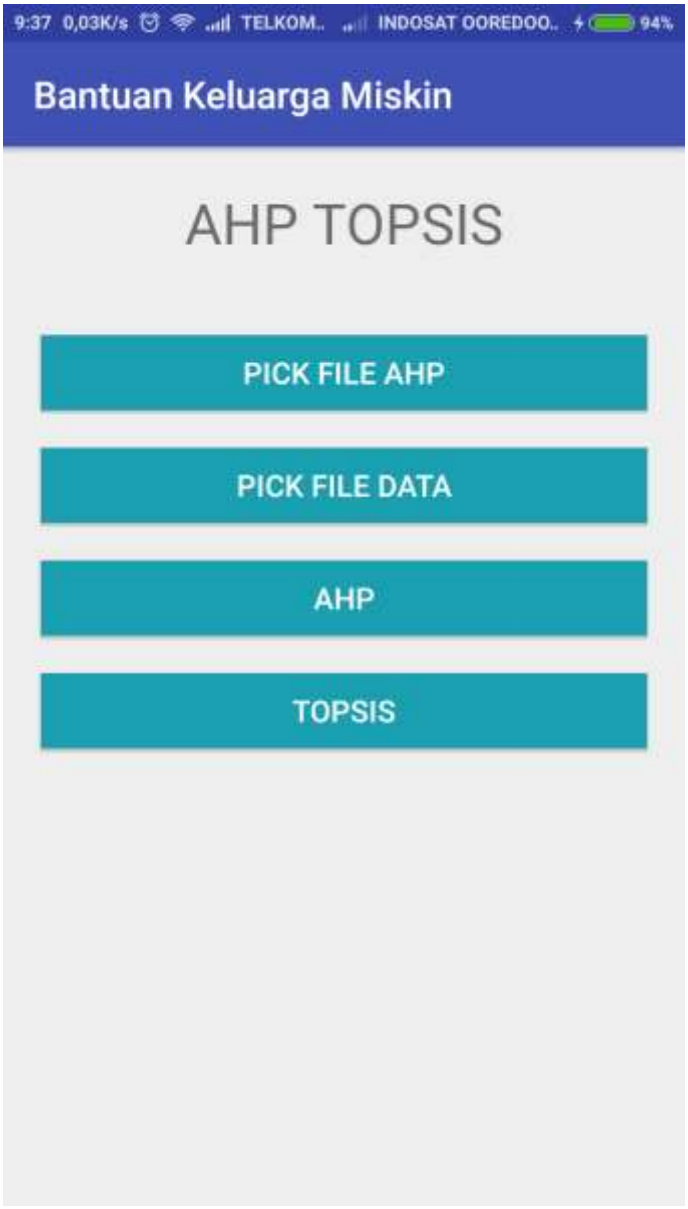
```
1  public double[] KedekatanR() {
2  KedekatanR = new double[data.isi.length];
3  for (int i = 0; i < data.isi.length; i++) {
4  KedekatanR[i] = IN[i] / (IP[i] + IN[i]);
   }
5  return KedekatanR;
6  }
```

Penjelasan :

- Baris 2 merupakan inisialisasi variable
- Baris 3 merupakan proses looping awal
- Baris 4 merupakan rumus dari proses untuk menentukan nilai prefrensi dari alternatif

1.3 Implementasi Antarmuka

Agar user dapat menggunakan sistem, maka diperlukan antarmuka sistem sebagai prantara interaksi. Didalam sistem ini user akan melihat sebuah halaman utama ketika membuka sistem. Tampilan halam utama sistem ditunjukkan pada Gambar 5.1.

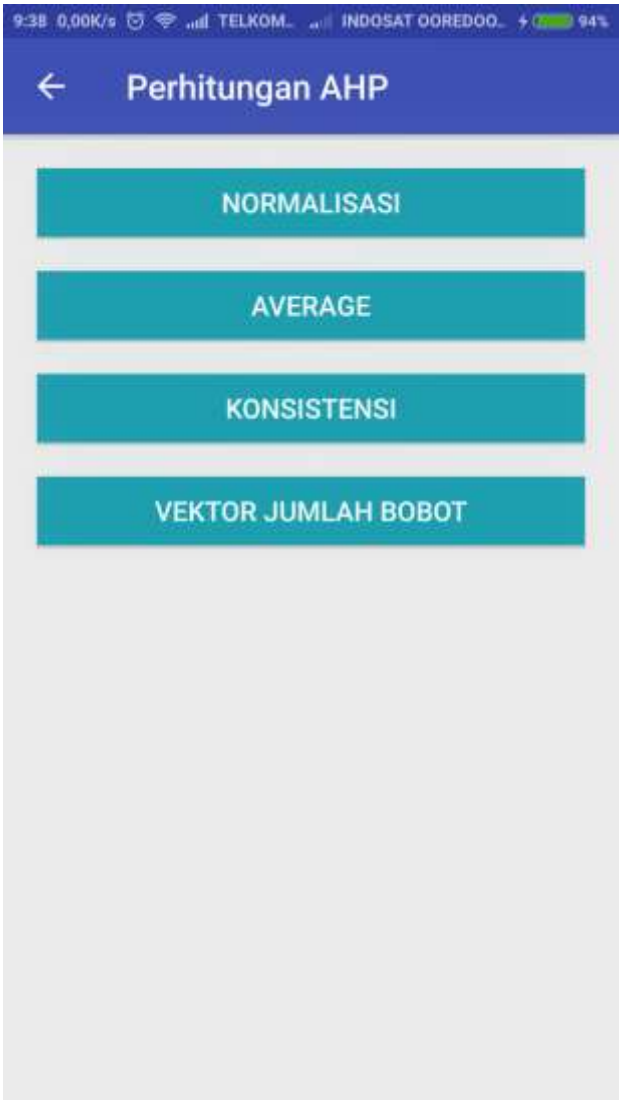


Gambar 5.1 Tampilan Halaman Utama Sistem

Gambar 5.1 menjelaskan bahwa pada halaman utama terdapat 4 menu yang bisa diakses oleh user. Tombol tersebut antara lain :

1. Pick File AHP : digunakan untuk menginputkan data AHP berupa matriks perbandingan berpasangan dengan format txt.
2. Pick File Data : digunakan untuk menginputkan data yang berisi calon penerima bantuan keluarga miskin dengan format txt.

3. AHP : akan menampilkan detail dari proses perhitungan AHP (dapat diakses apabila data AHP dan data calon penerima bantuan berhasil diinputkan). Tampiilan setelah user mengakses menu AHP ditunjukkan pada Gambar 5.2.
4. TOPSIS : merupakan menu yang menampilkan hasil dari perhitungan AHP dan proses perangkingan dengan menggunakan TOPSIS.



Gambar 5.2 Tampilan Menu Perhitungan AHP

Gambar 5.2 menunjukkan tampilan pada saat user memilih menu AHP pada halaman utama sistem. Pada halaman ini, user akan diberikan 4 pilihan menu, antara lain:

1. Nromalisasi : menampilkan matriks perbandingan berpasangan yang telah dinormalisasi. Tampilan dari halaman normalisasi dapat dilihat pada Gambar 5.3 .

Gambar 5.3 Tampilan Pada Menu Normalisasi

2. Averrage : menampilklan hasil rata-rata per baris dari setiap matriks perbandingan berpasangan. Tampilan halaman average dapat dilihat pada Gambar 5.4.

Gambar 5.4 Tampilan Halaman Pada Menu Average

3. Konsistensi : menampilkan nilai dari konsistensi pada perhitungan AHP. Tampilan dari menu konsistensi dapat dilihat pada Gambar 5.5.

Gambar 5.5 Tampilan Halaman Pada Menu Konsistensi

4. Matriks jumlah bobot : pada matriks jumlah bobot merupakan hasil akhir pembobotan menggunakan metode AHP. Pada halaman jumlah bobot, hasil yang akan ditampilkan pada menu ini merupakan bobot dari masing-masing kriteria yang digunakan untuk menentukan calon penerima bantuan keluarga miskin. Tampilan pada matriks jumlah bobot dapat dilihat pada Gambar 5.6.

Gambar 5.6 Tampilan Halaman Pada Menu Vektor Jumlah Bobot

Gambar 5.7 Tampilan Hasil Perhitungan Pada Menu TOPSIS

Gambar 5.7 menujukkan tampilan dimana hasil akhir dari perhitungan TOPSIS berupa perangkingan terhadap data uji yang digunakan sebagai proses perhitungan.