BAB 4

IMPLEMENTASI DAN PERANCANGAN SISTEM

4.1. **Implementasi**

Tahap implementasi dilakukan setelah analisis dan perancangan sistem selesai dilakukan, selanjutnya akan diimplementasikan kedalam bahasa pemograman dan diuji kedalam aplikasi. Aplikasi yang telah dibangun akan diimplementasikan untuk mengetahui system dapat berjalan sesuai tujuan atau tidak. Tujuan dari implementasi adalah untuk menerapkan perancangan yang telah dilakukan terhadap sistem, sehingga pemain dapat memberikan masukan untuk perkembangan sistem yang telah dibangun pada *game scrabble* dalam menerapkan algoritma genetika.

4.1.1. Implementasi Perangkat Keras

Spesifikasi perangkat keras *(hardware)* yang digunakan dalam implementasi algoritma genetika pada *game scrabble* dapat dilihat pada tabel 4.1

Tabel 4.1 Spesifikasi perangkat keras

Komputer	Spesifikasi Perangkat Keras
Prosesor	Prosesor 1,8 Ghz
Resolusi Layar	1024 x 768 *pixel*
Memori	Memori 2 Giga
Hardisk	*Hardisk* 100 Gb
Keyboard dan *touchpad*	Standar

4.1.2. Implementasi Perangkat Lunak

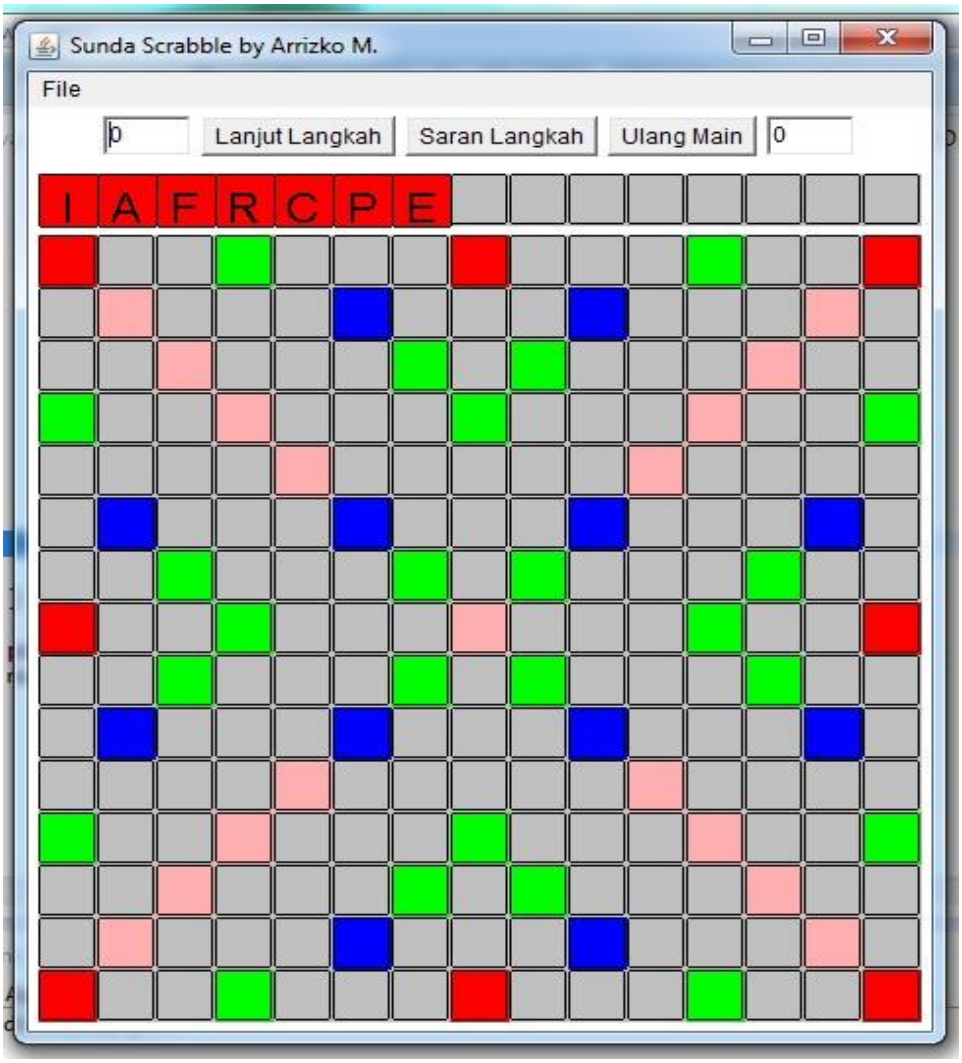
Spesifikasi perangkat lunak *(software)* yang digunakan dalam implementasi algoritma genetika pada *game scrabble* dapat dilihat pada tabel 4.2.

Tabel 4.2 Spesifikasi perangkat lunak

Spesifikasi Perangkat Lunak
Windows 7
eclipse

4.1.3. Implementasi Antarmuka

Algoritma yang telah dirancang pada sistem, diimplementasikan kedalam program dengan antarmuka sebagai berikut :

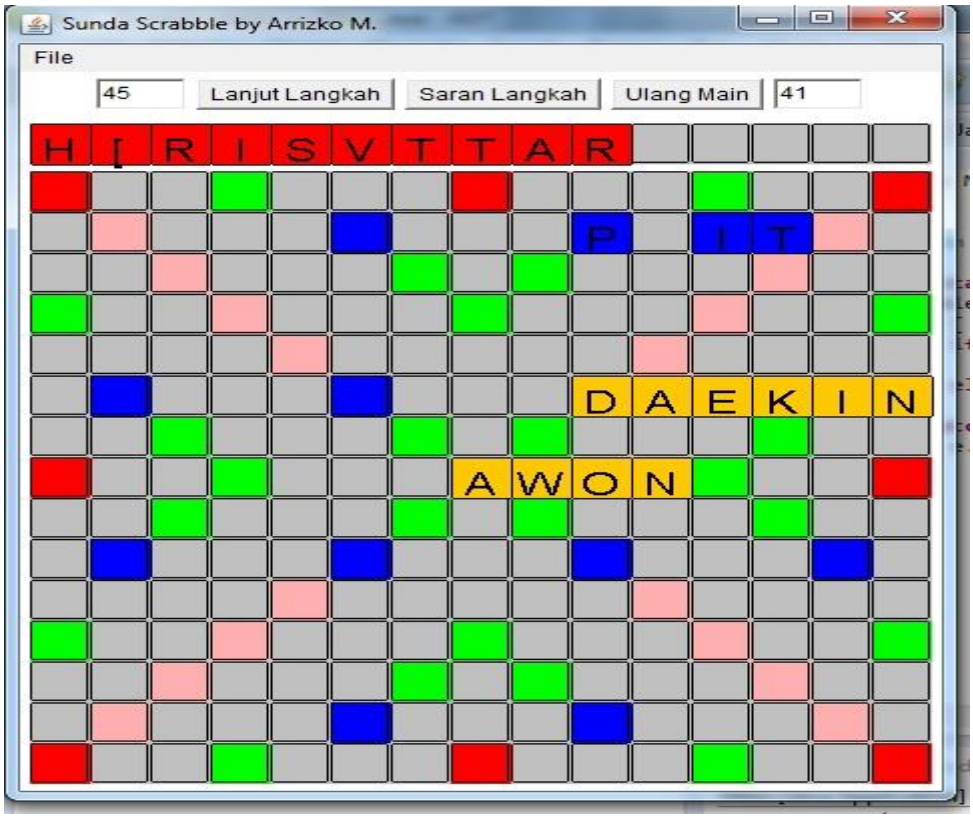


Gambar 4.1 Tampilan antarmuka

Pada tampilan ini, saat menjalankan aplikasi sudah menampilkan papan permainan dengan sistem *random* dan menampilkan tombol-tombol utama untuk memainkan *game scrabble* ini diantaranya terdapat tombol lanjut langkah yang berfungsi untuk mengecek kata yang akan di susun apakah memenuhi kriteria yang dicapai, tombol saran langkah berfungsi untuk menyarankan huruf-huruf yang ada dalam kotak huruf secara ulang disoal permainan secara *random,* serta tombol ulang main berfungsi untuk mengembalikan huruf-huruf ke menu utama untuk memulai atau me-*restart* permainan. Hasil dari tombol lanjut langkah akan menghasilkan huruf di mana komputer langsung mengecek dengan sistem kerja dari algoritma genetika yang dihasilkan oleh sistem. Terdapat kondisi-kondisi dalam memainkan permainan *scrabble* yang dibuat, yaitu :

4.1.4. Implementasi tampilan pencarian algoritma genetika

Tampilan ini merupakan tampilan hasil pencarian kata menggunakan algoritma genetika berdasarkan pola kata yang di cari, dimana dari hasil tersebut di hasilkan keberadaan kata pada papan beserta hasil nilai pencarian menggunakan algoritma genetika. Bisa di lihat pada gambar 4.2



Gambar 4.2 hasil dari pencarian algoritma genetika

4.2. **Pengujian Sistem**

Pengujian sistem dilakukan untuk menemukan kekurangan atau kesalahan pada perangkat lunak yang akan diuji. Pengujian ini bermaksud untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut. Pada pengujian perangkat lunak ini menggunakan metode pengujian *white box* dan *black box.* Pengujian dengan metode *white box* adalah pengujian dengan melihat kedalam modul untuk meneliti kode-kode program yang ada, dan menganalisis apakah terdapat kesalahan atau tidak. Pengujian dengan menggunakan metode *black box* berfokus pada persyaratan fungsional perangkat lunak.

4.2.1. Pengujian White Box

Pengujian *white box* merupakan metode perancangan *test case* yang menggunakan struktur *control* dari perancangan metode dalam mendapatkan *test case.* Adapun metode yang digunakan dalam pengujian *white box* ini adalah metode *basic path.* Metode *basic path* memungkinkan perancang uji kasus untuk memperoleh ukuran kompleksitas logis dari sebuah perancangan prosedral dan menggunakan ukuran ini sebagai sebuah panduan dalam mendefinisikan sekumpulan dasar dari alur eksekusi.

Pengujian *white box* akan digunakan pada algoritma genetika untuk mengetahui kinerja logika berdasarkan *flowchart* yang telah dibuat pada tahap analisis. Pengujian *white box* meliputi beberapa tahap, yaitu :

4.2.1.1.1. Pengujian Algoritma genetika Pada *Game srabble*

Pada pengujian ini, algoritma genetika akan diuji dalam mengoptimalkan huruf pada *game scrabble*. Berikut merupakan pengujian algoritma genetika pada *game scrabble* dengan menggunakan *whitebox.*

1. Mengubah *source code* menjadi *flowchart,* kemudian diubah kedalam *flowgraph.* Berikut *source code* algoritma genetika dalam mengoptimalkan huruf-huruf permainan :

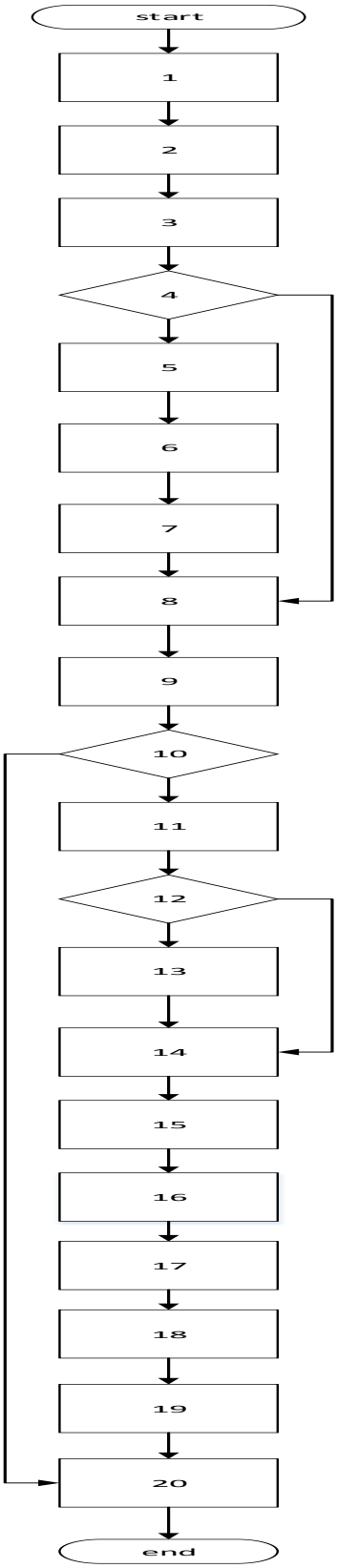
Tabel 4.3 *Source code* algoritma genetika dalam mengoptimalkan huruf

	`public class Genetic {`
	`  /** String containing the CVS revision. Read out via reflection!*/`
1	`  private static final String CVS_REVISION = "$Revision: 1.9 $";`
2	`  public static void main(String[] args) {`

```
    int numEvolutions = 500;
    Configuration gaConf = new DefaultConfiguration();
    gaConf.setPreservFittestIndividual(true);
    gaConf.setKeepPopulationSizeConstant(false);
3   Genotype genotype = null;
    int chromeSize;
    if (args.length > 0) {
      chromeSize = Integer.parseInt(args[0]);
    }
4   else {
      chromeSize = 16;
    }
5   double maxFitness = Math.pow(2.0, (double) chromeSize)
- 1;
6   if (chromeSize > 32) {
      System.err.println("This does not handle " +
                         "Chromosomes greater than 32 bits
in length.");
      System.exit( -1);
    }
    try {
7     IChromosome sampleChromosome = new Chromosome(gaConf,
8         new BooleanGene(gaConf), chromeSize);
9     gaConf.setSampleChromosome(sampleChromosome);
10    gaConf.setPopulationSize(20);
11    gaConf.setFitnessFunction(new MaxFunction());
12    genotype = Genotype.randomInitialGenotype(gaConf);
    }
    catch (InvalidConfigurationException e) {
      e.printStackTrace();
      System.exit( -2);
    }
    int progress = 0;
13  int percentEvolution = numEvolutions / 100;
```

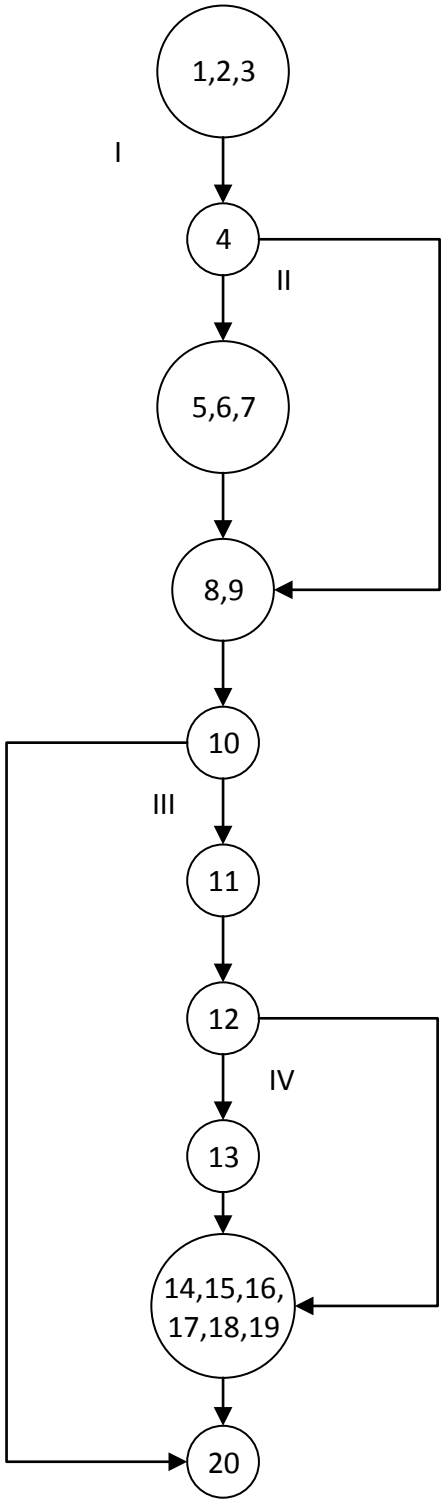
```
14      for (int i = 0; i < numEvolutions; i++) {
15        genotype.evolve();
          // Print progress.
          // ---------------
16        if (percentEvolution > 0 && i % percentEvolution ==
    0) {
            progress++;
17          IChromosome fittest =
18  genotype.getFittestChromosome();
            double fitness = fittest.getFitnessValue();
            System.out.println("Currently fittest Chromosome
19  has fitness " +
                                fitness);
            if (fitness >= maxFitness) {
              break;
            }
        }
      }
      // Print summary.
      // --------------
20    IChromosome fittest = genotype.getFittestChromosome();
      System.out.println("Fittest Chromosome has fitness " +
                          fittest.getFitnessValue());
    }
  }
```

2. Mengubah *source code* menjadi *flowchart*, dapat dilihat pada Gambar 4.2


start
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
end

Gambar 4.2 *Source code* menjadi *flowchart*

3. Mengubah *flowchart* menjadi *flowgraph*, dapat dilihat pada



Gambar 4.4 *Flowchart* menjadi *flowgraph*

4. *Cyclomatic complexity*

Dari diagram alir pada Gambar 4. dapat dihitung *cyclomatic complexity,* yakni :

V(G) = Edge – Node + 2

= 12 - 10 + 2

= 2 + 2

V(G) = 4

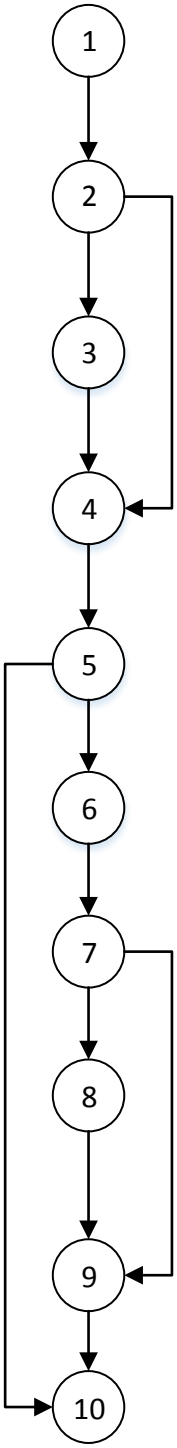
5. *Predicate node*

V(G) = Jumlah *grah matriks* + 1

V(G) = 3 + 1

V(G) = 4

6. Penomoran ulang *flowgraph*, dapat dilihat pada Gambar 4.4.



Gambar 4.5 Penomoran ulang *flowgraph*

7. *Graph matrix*, dapat dilihat pada Tabel 4.4

Tabel 4.4 *Graph matrix*

Node	1	2	3	4	5	6	7	8	9	10	Perhitungan
1		1									0
2			1	1							1
3				1							0
4					1						0
5						1				1	1
6							1				1
7								1			0
8									1		0
9										1	0
10											0
Jumlah (x)											3

V(G) = x + 1

V(G) = 3 + 1

V(G) = 4

4.2.2. Pengujian Black Box

Pengujian *black box* adalah pengujian yang mengatakan benar bila *input* yang diberikan akan menghasilkan *output* yang sesuai dengan spesifikasi program tanpa memperhatikan struktur logika dalam program.

4.2.3. Rencana Pengujian

Rencana pengujian yang akan dilakukan pada aplikasi ini dapat dilihat pada tabel 4.5

Tabel 4.5 Rencana Pengujian

Komponen yang diuji	Skenario pengujian	Jenis pengujian
Menu Utama	Memilih kotak	*Black box*
	Memasukan huruf	*Black box*
	Menekan tombol lanjut langkah	*Black box*

4.2.4. Kasus Dan Hasil Pengujian *(Black Box)*

Kasus dan hasil uji aplikasi yang sudah dibangun dengan menggunakan metode *black box* berdasarkan pengamatan yang dilakukan dan menghasilkan kesimpulan dari pengujian tersebut. Kasus dan hasil pengujian dapat dilihat pada tabel 4.6.

Tabel 4.6 Kasus dan hasil pengujian

Kasus yang diuji	Skenario pengujian	Hasil yang diharapkan	Kesimpulan
Menu Utama	Memilih kotak	*Pointer* dapat menunjuk kotak yang dipilih.	[√] Berhasil [ ] Tidak berhasil
	Memasukan huruf	Sistem dapat meng-*input*-kan huruf yang dimasukan pemain.	[√] Berhasil [ ] Tidak berhasil
	Menekan tombol lanjut	Sistem memberikan jawaban penyelesaianan	[√] Berhasil [ ] Tidak

	langkah	atau menunjukan huruf dengan menampilkan inputan pada kotak yang terjadi dalam pengisian huruf.	berhasil

4.3. **Kesimpulan**

Dari hasil pengujian dapat disimpulkan bahwa algoritma genetika dapat berfungsi dalam menyelesaikan soal huruf-huruf pada *game scrabble*. Algoritma genetika dapat bekerja dengan baik dalam menyusun kata-kata soal permainan maupun dalam menyelesaikan jawaban pada *game scrabble* ini. Dan ketepatan daripada algoritma genetika untuk pencarian langkah optimal dalam menyelesaikan permainan *scrabble*.